统计推断在数模转换系统中的应用

组号：58　周思宇 5140109060，沈冠廷 5140309473

摘要：本文是《统计推断在数模转换系统中的应用》的课程小论文终稿。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遗传算法在 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，每组样本观测点 51 个。但在实际过程中，如果对每个样品均密集选点测定，则将消耗大量的时间和经济。因此我们选择减少测定次数以提高效率。本文阐述了一种通过统计推断获得电子器件的性能参数特性曲线若干关键点并拟合出特性曲线的一种方案。

关键词：统计推断，MATLAB，建模拟合，遗传算法

ABSTRACT: This report is the final report for the course ‘Application of Statistical Inference in DA Inverting System’. 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 samples are provided by the teacher, each group sample observation point 51. However, in engineering practice, it will spare lots of time and money to choose numerous sample to measure. This paper describes a scheme to obtain a characteristic curve of parameters of an electronic element by fitting several key points of the curve statistical inference from statistical inference.

Key words: statistical inference, MATLAB, modeling fitting, GA

一、　引言

在我们的课程实验以及工程实践中，我们通常面对的对象是一个由多个变量组成的系统。所以在这样的情况下，当我们需要寻找两个变量间的关系时，尽管通过一些理论原理计算得到这两个变量的函数关系，但往往得到的关系都是不太精确的。所以在工程中，人们通常采取对多种不同曲线进行拟合的方法，算出多种不同方案，对各检测方案进行评估，最终选出最优的方案。

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]。

1.2 设计思路的考虑

在这个任务中我们主要考虑到测定点和估算点两个方面的问题：

(1)测定点方面：数量越多则测定成本越高；

(2)估算点方面：数量越多则误差损失越高。

由于此问题所备选的组合方案数量巨大，难以使用计算机采取穷举的方式进行求解，故经过讨论，我们采用遗传算法。

二、 数学模型

2.1 数据来源

监测模块的组成框图如图 1。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 $\hat{Y}$ 作为 Y 的读数（监测模块对 Y 的估测值）。

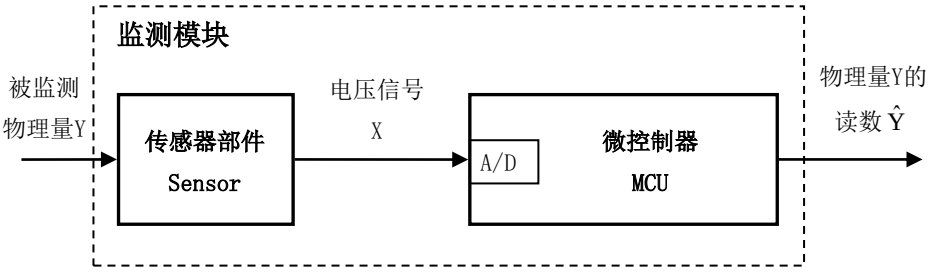

图 1 监测模块组成框图

2.2 传感部件特性

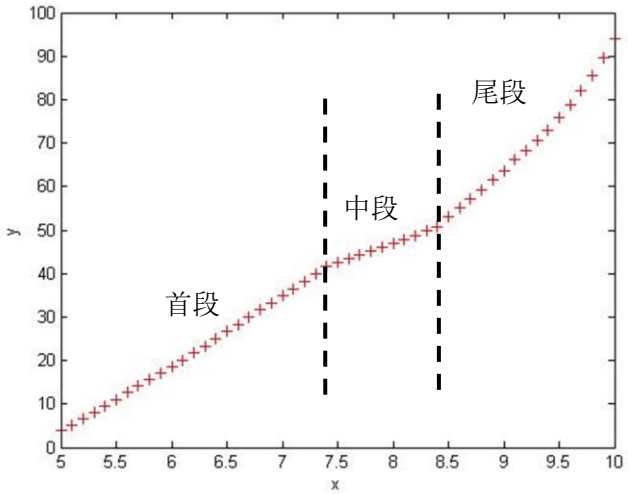

图 2 传感特性图示

[1]上海交大电子系. 统计推断在数模转换系统中的应用课程讲义

对此样品的特性曲线进行大致的定性分析，可以得到以下特点：

(1) 此特性曲线整体单调递增；

(2) X 的取值在[5.0,10.0]之间，Y 的取值在[0,100]的区间内；

(3) 曲线大致可以分为三段，每段都不是完全线性的，都有一定的弯曲度，具有随机性差异；

(4) 中段的斜率明显小于首段和尾段的斜率，而且中段的起点位置和长度都无法精确确定。

所以我们对特性曲线分为三段，分别进行拟合。由此中段的起始位置需要确定，我们观察得到三段的斜率有明显差异，故我们计算样本值任意两点间的斜率并进行大小比较，其中有两点前后斜率变化最大的即为中段的起始点，则以此两点分界，将特性曲线分为三段，分别进行拟合。

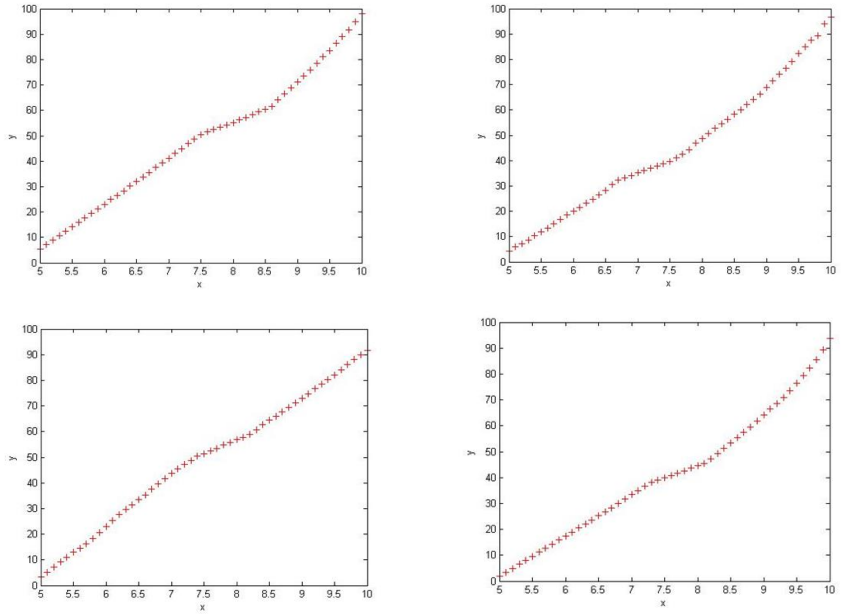

图 3 四个不同样本个体特性图示对比

2.4 成本计算

- 单点定标误差成本

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

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

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

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。

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

总成本较低的校准方案，认定为较优方案。

三、 方案设计

3.1 遗传算法

3.1.1 简介

遗传算法（Genetic Algorithm）是一类借鉴生物界的进化规律（适者生存，优胜劣汰遗传机制）演化而来的随机化搜索方法。它是由美国的 J.Holland 教授 1975 年首先提出，其主要特点是直接对结构对象进行操作，不存在求导和函数连续性的限定；具有内在的隐并行性和更好的全局寻优能力；采用概率化的寻优方法，能自动获取和指导优化的搜索空间，自适

应地调整搜索方向，不需要确定的规则[2]。

3.1.2 基本概念

种群：遗传算法从样本子空间或通过上一代的选择得到的一组随机产生的解空 间。

染色体：种群中的每个个体，每个染色体都是问题的一个解。染色体是一串符 号，比如一个二进制字符串。

遗传：即染色体在后续迭代中不断进化。

适值：每一代中用来测量染色体好坏的值。

后代：生成的后一代染色体。

3.2 求“最优解”步骤及参数设置

a)初始化种群：设置进化代数计数器 t=0，设置最大进化代数 T=200，随机生成 100 个个体（种群）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。

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。观察点设置为 7 个观察点。

这里需要强调一下，在最初我们并没有确定具体观察点个数，而根据后期多次的拟合尝试发现，7 个点能够达到预期效果；若选取 6 个点或是更少，我们发现拟合的准确度降低，影响实验结果；而若选取 8 个点或是更多，则造成不必要的工序累赘。故选取 7 个观察点为宜。

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。交叉概率设置为 90%。

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。变异概率设置为 10%。

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=T,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。

四、 运行结果

1、 三次样条插值运行结果

【2，6，18，29，33，44，51】

成本为 116.7375

2、 三次多项式拟合运行结果

[2] 百度百科 遗传算法

【2，3，11，12，14，19，47】

成本为 129.9350

五、 方法评价及总结

（1） 实验过程中我们发现，模拟遗传算法对降低定标成本有一定效果，但是很大程度上牺牲了效率，算法运行时间大大增加。因为模拟遗传算法在选取测量点中变异和重组是比较随机的，故每次运行的结果可能出现差异，随着迭代次数的增加，标定成本趋近于一定值。

（2） 插值法优于多项式拟合的原因：由于三次样条插值是要将已知点拟合到函数上，而三次多项式拟合则是选取一条到已知点的误差最小的曲线，因此，三次样条插值对于已知观察点的拟合优于三次多项式拟合，而 且由于三次多项式拟合倾向于与已知点在最小二乘法下最小，因此容易使得观察点比较靠近时适应度更大，从而陷入局部最优解,因此我们在使用三次多项式拟合时使用的变异概率较大。

（3） 实践中发现适当增加突变频率会增大跳出局部最优解的几率，更易获得更优的解。

六、 致谢

感谢袁焱老师在统计推断课程中的悉心指导以及在遇到困难时对我们的耐心帮助。感谢与我们一同讨论的其他小组，通过小组间的讨论和交流，我们开拓了思路，也找到了解决问题的方法。感谢学长学姐提供的优质的报告，让我们更快更好地理解了遗传算法。也要感谢图书馆为我们提供的 Matlab 相关书籍。谢谢！

附录

【MATLAB 源代码】

一、 使用三次样条插值的遗传算法

```
global pop; %取观察点种群
global y; %样本列表
global x; %样本占空比列表
global len; %样本组数量
global fitTable; %适应度列表
global tmpMin; %当前种群的最高适应度
global minCost; %最高适应度
global minCostPop; %最高适应度对应的观察点
global minCostPos;

popSize=100;
dataSize=7;
crossRate=0.9;
mutateRate=0.01;
Generation=200;
minCost=37*dataSize;

rand('state',sum(100*clock));
sample=csvread('20150915dataform.csv');
len=length(sample)/2;
x=sample(1:2:(len)*2-1,:);
y=sample(2:2:(len)*2,:);

initialize(popSize,dataSize,51);
for G=1:Generation
    minCost
    minCostPop
    select(popSize,dataSize);
    cross(popSize,dataSize,crossRate);
    mutate(popSize,dataSize,mutateRate);
    for i=1:popSize
        fitTable(i)=fit(pop(i,:));
    end
    tmpMin=min(fitTable);
    if tmpMin<minCost
        minCost=tmpMin;
        minCostPos=find(fitTable==tmpMin);
        minCostPop=pop(minCostPos,:);
    end
```

```
    G
end
minCost;
minCostPop;

function initialize(popSize,dataSize,maxNum)
global pop;
global fitTable;
global tmpMin;
global minCost;
global minCostPop;
global minCostPos;

pop1=zeros(popSize,dataSize);
for i=1:popSize
    for j=1:dataSize
        pop1(i,j)=randi(maxNum);
        k=1;
        while(k<=j-1)
            if (pop1(i,j)==pop1(i,k))
                pop1(i,j)=randi(maxNum);
                k=0;
            end
            k=k+1;
        end
    end
end

pop=sort(pop1,2);
fitTable=zeros(popSize,1);
for i=1:popSize
    fitTable(i)=fit(pop(i,:));
end
tmpMin=min(fitTable);
if tmpMin<minCost
    minCost=tmpMin;
    minCostPos=find(fitTable==tmpMin);
    minCostPop=pop(minCostPos,:);
end
end

function[score]=fit(method)
global y;
global x;
```

```
global len;
size=length(method);
sum=0;
yy=zeros(1,size);
xx=zeros(1,size);
for i=1:len
    A=0;
    for j=1:size
        yy(j)=y(i,method(j));
        xx(j)=x(i,method(j));
    end
    y1=interp1(xx,yy,x(i,:),'spline');
    yxTable=abs(y1-y(i,:));
    for j=1:51
        yx=yxTable(j);
        if yx<=1
            A=A+0.5;
        elseif yx<=2
            A=A+1.5;
        elseif yx<=3
            A=A+6;
        elseif yx<=5
            A=A+12;
        elseif yx>5
            A=A+25;
        end
    end
    A=A+84;
    sum=sum+A;
end
score=sum/len;
end

function[]=select(popSize,dataSize)
global pop
global fitTable;
fitnessSum(1)=1.0/fitTable(1);
for i=2:popSize
    fitnessSum(i)=fitnessSum(i-1)+1.0/fitTable(i);
end
popNew=zeros(popSize,dataSize);
for i=1:popSize
    r=rand()*fitnessSum(popSize);
    left=1;
```

```
    right=popSize;
    mid=round((left+right)/2);
    while 1
        if r>fitnessSum(mid)
            left=mid;
        else
            if r<fitnessSum(mid)
                right=mid;
            else
                popNew(i,:)=pop(mid,:);
                break;
            end
        end
        mid=round((left+right)/2);
        if(mid==left)||(mid==right)
            popNew(i,:)=pop(right,:);
            break;
        end
    end
end
pop=popNew;
end

function[]=cross(popSize,dataSize,crossRate)
          global pop;
          for i=1:2:popSize
              r=rand;
              if r>crossRate
                  continue;
              end
              p=randi([2,dataSize]);
              tmp=pop(i,p:dataSize);
              pop(i,p:dataSize)=pop(i+1,p:dataSize);
              pop(i+1,p:dataSize)=tmp
              checkPop(i,dataSize);
              checkPop(i+1,dataSize);
          end
       end

function mutate(popSize,dataSize,mutateRate)
    global pop;
    for i=1:popSize
        r=rand();
        if r<mutateRate
```

```
            r=randi(dataSize);
            if rand()>0.5
                x=1;
            else
                x=-1;
            end
            tmp=pop(i,r)+x;
            if (r==1)
                if (tmp>=1)&&(pop(i,r+1)~=tmp)
                    pop(i,r)=tmp;
                end
            elseif (r==dataSize)
                if(tmp<=51)&&pop(i,r-1)~=tmp
                    pop(i,r)=tmp;
                end
            else
                if(pop(i,r-1)~=tmp)&&(pop(i,r+1)~=tmp)
                    pop(i,r)=tmp;
                end
            end
        end
    end
    end

function checkPop(n,dataSize)
    global pop;
    pop(n,:)=sort(pop(n,:));
    flag=0;
    for j=2:dataSize
        if pop(n,j-1)==pop(n,j)
            pop(n,j)=randi(51);
            flag=1;
        end
    end
    if flag
        checkPop(n,dataSize)
    end
    end
```

二、 采用三次多项式拟合

```
global popSize;
global chromoSize;
global generationSize;
global crossRate;
```

```
global mutateRate;
popSize = 100;
chromoSize = 7;
generationSize = 200;
crossRate = 1;
mutateRate = 0.1;

global pop;
global pop2;
global popNew;
global g;
global bestGeneration;
global bestFitness;
bestFitness = 0;

global bestPop;
global fitAverage;
global fitness;
global fitSum;
global sample;

sample = xlsread('20150915dataform.csv',1);
initilize(popSize,chromoSize)
flag= 1;
%pop
for g=1:generationSize
    fit(popSize,chromoSize);
    disp "new"
    g
    bestPop
    bestFitness
    bestCost = 37*chromoSize-bestFitness
    if bestCost < (13*chromoSize)
        break
    end
    select(popSize,chromoSize);
    cross(popSize,chromoSize);
    mutate(popSize,chromoSize);
end

function initilize(popSize,chromoSize)
global pop
global pop2
for i=1:popSize
```

```
    for j=1:chromoSize
        temp(i,j) = randi(50);
        for k=1:j-1
            if temp(i,k)==temp(i,j)
                temp(i,j) = randi(50);
                k=1;
            end
        end
    end
end

pop=sort(temp,2);
for i=1:popSize
    for j=1:chromoSize
        tmp1 = pop(i,j);
        for k=1:6
            pop2(i,(j-1)*6+k)=fix(tmp1/(2^(6-k)));
            tmp1 =tmp1-pop2(i,(j-1)*6+k)*(2^(6-k));
        end
    end
end
end

function fit(popSize,chromoSize)
global pop
global sample
global g
global fitness
global bestPop
global bestParam
global bestFitness
global bestGeneration
sample = xlsread('20150915dataform.csv',1);
for i=1:popSize
    param(i,:)      =      polyfit(sample(2*round(i*400/popSize)-1,
pop(i,:)),sample(2*round(i*400/popSize),pop(i,:)),3);
end
fitSum=0;
for i=1:popSize
    fitness(i)=400*26*chromoSize;
    for j=1:chromoSize
    for k=1:400
    x=sample(2*k-1,pop(i,j));
    y=sample(2*k,pop(i,j));
```

```
    xy=param(i,1)*x^3+param(i,2)*x^2+param(i,3)*x+param(i,4);
    if abs(xy-y)>0.5 && abs(xy-y)<=1
        fitness(i) = fitness(i)-0.5;
    end
    if abs(xy-y)>1 && abs(xy-y)<=2
        fitness(i) = fitness(i)-1.5;
    end
    if abs(xy-y)>2 && abs(xy-y)<=3
        fitness(i) = fitness(i)-6;
    end
    if abs(xy-y)>3 && abs(xy-y)<=5
        fitness(i) = fitness(i)-12;
    end
    if abs(xy-y)>5
        fitness(i) = fitness(i)-25;
    end
    end
end
fitness(i) = fitness(i)/400;
if fitness(i) > bestFitness
    bestFitness = fitness(i);
    bestGeneration = g;
    bestParam=param(i,:);
    for k=1:chromoSize
        bestPop(k) = pop(i,k);
    end
end
fitSum = fitSum + fitness(i);
end
end

function select( popSize,chromoSize)
global pop
global popNew
global pop2
global fitness
global fitnessSum
fitnessSum(1)= fitness(1);
for i=2:popSize
    fitnessSum(i)=fitnessSum(i-1)+fitness(i);
end
popNew=zeros(popSize,chromoSize);
for i=1:popSize
    r=rand()*fitnessSum(popSize);
```

```
    left=1;
    right=popSize;
    mid=round((left+right)/2);
    while 1
        if r>fitnessSum(mid)
            left=mid;
        else
            if r<fitnessSum(mid)
                right=mid;
            end
        end
        mid=round((left+right)/2);
        if (mid==left)||(mid==right)
            popNew(i,:)=pop(right,:);
            break;
        end
    end
end
%popNew
for i=1:popSize
    for j=1:chromoSize
        tmp1 = popNew(i,j);
        for k=1:6
            pop2(i,(j-1)*6+k)=fix(tmp1/(2^(6-k)));
            tmp1 = tmp1-pop2(i,(j-1)*6+k)*(2^(6-k));
        end
    end
end
end

function cross( popSize, chromoSize )
global pop
global pop2
global crossRate
for i=1:2:popSize
    if(rand < crossRate)
        crossPos = randi(chromoSize);
        for j=1:chromoSize
            for k=crossPos:6
                tmp = pop2(i,(j-1)*6+k);
                pop2(i,(j-1)*6+k) = pop2(i+1,(j-1)*6+k);
                pop2(i+1,(j-1)*6+k) = tmp;
            end
        end
```

```
    end
end

function mutate( popSize, chromoSize )
global pop2
global pop
global popNew
global mutateRate
for i=1:popSize
    if rand < mutateRate
        mutatePos = round(rand*6);
        if mutatePos == 0
            continue;
        end
        pop2(i,mutatePos) = 1 - pop2(i, mutatePos);
    end
end
for i=1:popSize
    for j=1:chromoSize
        popNew(i,j)=0;
        flag=1;
        while flag==1
            popNew(i,j)=0;
            for k=1:6
                popNew(i,j)=popNew(i,j)+pop2(i,(j-1)*6+k) * 2^(6-k);
            end
            flag=0;
            for k=1:j-1
                if popNew(i,k) == popNew(i,j);
                    flag=1;
                    r=randi(6);
                    pop2(i,(j-1)*6+r) = 1-pop2(i,(j-1)*6+r);
                end
            end
            if (popNew(i,j)>50)||(popNew(i,j)==0)
                flag = 1;
                pop2(i,(j-1)*6+1) = 0;
                pop2(i,(j-1)*6+randi(5)+1) = 1;
            end
        end
    end
end
pop = sort(popNew,2);
end
```