统计推断在数模转换系统中的应用

组号 31　刘牲泰 5130309486　王涌壮 5130309488

摘要：为监测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。

关键词：建模拟合　遗传算法　三次插值拟合　三次多项式拟合

1 引言

在工程实践与课程实验中通常面对的对象是一个由多个变量组成的系统，当我们需要寻找两个变量间的关系时，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的曲线拟合，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 课题内容的提出

本次课题的研究内容是关于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 课题方案的设计

选取部分数据点进行测量，并用测量出的数据值推算其余点的测量值，在测量与推算的过程中会有测定成本以及估算误差产生的成本，我们的目的便是寻找到某一个测量方案，使得我们在以后的成本计算中取得成本的最小值。

1.3 数据来源

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

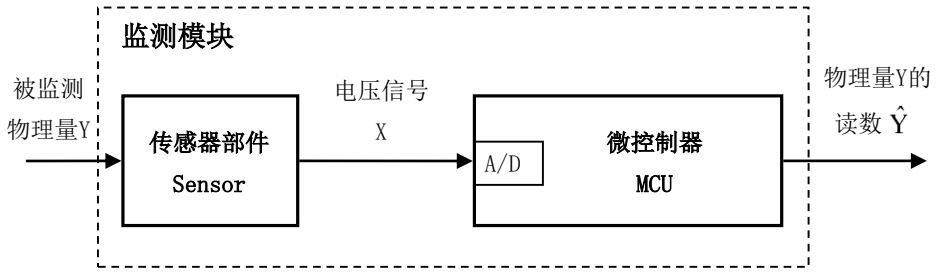


图 1 监测模块组成框图

1.4 校准

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，$i=1,2,3,...,50,51$。

1.5 传感部件特性

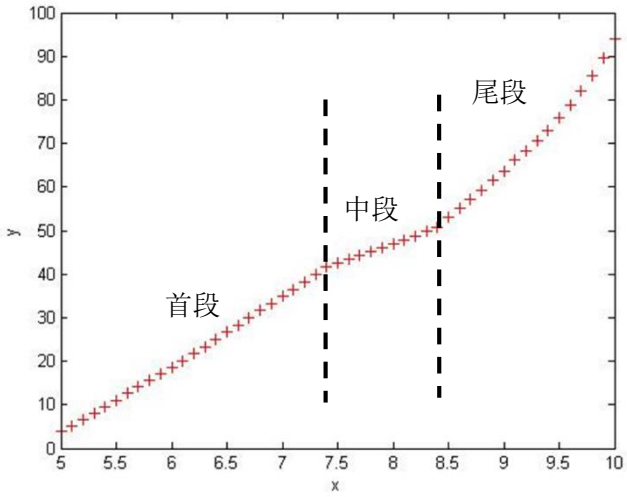


图 2 传感特性图示

一个传感部件个体的输入输出特性大致如图 2 所示，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。

为进一步说明情况，图 3 对比展示了四个不同样品个体的特性曲线图示。

1.6 标准样本数据库

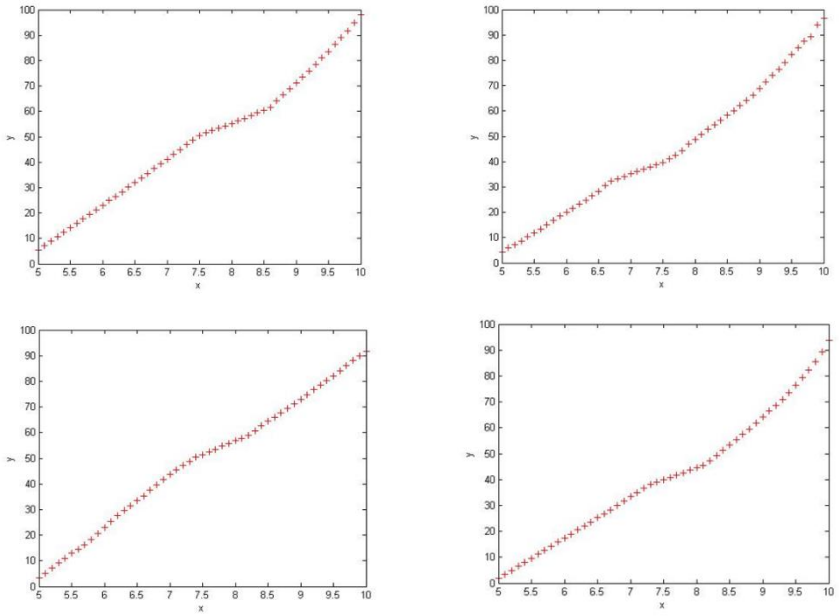
前期已经通过试验性小批量生产，制造了一批传感部件样品，并通过实验测定了每个样品的特性数值。这可以作为本课题的统计学研究样本。数据被绘制成表格，称为本课题的“标准样本数据库”。

图 3 四个不同样本个体特性图示对比

1.7 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



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

- 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=20。

- 某一样本个体的定标成本

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

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。

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

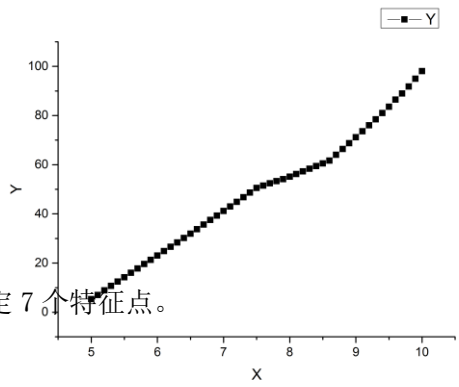
总体成本较低的校准方案，认定为较优方案。

2．方案设计

2.1 数据观察

通过观察样本数据，我们可以发现 x 的增大，图像有较为良好的线性分段平滑度，但有明显的分段跳变。

图 4 某个样本的特性图



2.2 特征点选取

在 51 个数据点中确定 7 个特征点。

2.3 遗传算法

遗传算法就是模拟进化论的优胜劣汰的规则，并加有概率性质的随机遗传的算法。算法中从父代向子代的遗传主要分三种方式：1，选择（select），选择父代中优秀的个体直接传递给下一代，即保留父代。2，交叉（cross），以两个父代的基因为基础，进行交叉重组，生成下一代，好的基因会有更大的概率传给子辈。3，变异（mutate），每个父代个体都有一定的概率发生随机的基因突变。

评价基因好坏的依据就是适应度。在遗传算法的步骤中起着突出作用的其实是变异。正因为有了变异，算法才存在在全局范围内寻找到最优解的可能性。但是遗传算法需要通过反复试验，才有可能给出最优解。

2.3.1 遗传算法下使用三次样条插值程序细节及运行结果

（1）参数设置

popSize=100;种群大小设置为 100

dataSize=7;观察点设置为 7 个观察点

crossRate=0.9;交叉概率设置为 90%

mutateRate=0.1;变异概率设置为 10%

Generation=200;最大代数设置为 200

（2）适应度的确定

使用成本的倒数作为适应度（select）

（3）交叉采用相邻两个染色体之间进行单点交叉。

（4）在运行之处由于变异概率设置的较高，导致偶尔会出现最佳个体难以稳定的状况，最后将其调低。

（5）得到的运行结果

[2, 9, 20, 26, 34, 43, 50]

成本为 93.8998

2.3.2 遗传算法下使用三次多项式拟合程序细节及运行结果

（1）参数设置

popSize = 100; 种群大小设置为 100

chromoSize = 7; 观察点设置为 7 个观察点

generationSize = 200; 最大代数设置为 200

crossRate = 0.9; 交叉概率设置为 90%

mutateRate = 0.1; 变异概率设置为 10%

（2）适应度的确定

使用（25+1）*7-误差成本作为适应度，25 为最大误差成本,7 为观察点的数量。

（3）交叉同样采取相邻染色体进行单点交叉。

（4）变异概率设置为 10%

（5）得到的运行结果

[2, 7, 9, 10, 14, 23, 47]

成本为 131.7932

3.方法评价及总结

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。

附：

实现代码：

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;               %种群大小

dataSize=7;                %观察观察点

crossRate=0.9;             %交叉概率

mutateRate=0.01;           %变异概率

Generation=200;           %最大子代数

minCost=37*dataSize;       %最小成本初始化



rand('state',sum(100*clock));

sample=xlsread('20141010dataform.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;


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

%popSize: 输入种群大小

%chromoSize: 输入染色体长度

%generationSize: 输入迭代次数

%crossRate: 交叉概率

%mutateRate: 变异概率

```
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 fitAverge;

global fitness;

global fitSum;

global sample;


sample = xlsread('20141010dataform.csv',1);

initilize(popSize,chromoSize)

flag = 1;

%pop

for g=1:generationSize

    fit(popSize,chromoSize);

    disp "new"

    g

    %bestGeneration

    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
%global popSize
%global chromoSize
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
```

```
    function fit(popSize, chromoSize)
    global pop
    global sample
    global g
    global fitness
    global bestPop
    global bestParam
    global bestFitness
    global bestGeneration
    sample = xlsread('20141010dataform.csv',1);
    for i=1:popSize
        param(i,:) = polyfit(sample(2*round(i*469/popSize)-1,
pop(i,:)),sample(2*round(i*469/popSize),pop(i,:)),3);
        %param(i,:) = polyfit(sample(2*i-1, pop(i,:)),sample(2*i,pop(i,:)),3);
```

```
end
fitSum=0;
for i=1:popSize
    fitness(i)=469*26*chromoSize;
    for j=1:chromoSize
        for k=1:469
        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)/469;
    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
%for i=1:popSize
    %fitness(i) = fitness(i)/fitSum;
%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;
            %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
%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



function cross( popSize, chromoSize )
global pop
global pop2
%global chromoSize
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

        %crossPos = round(rand * chromoSize * 6);
        %if or (crossPos == 0, crossPos == 1)
            %continue;
        %end
        %for j=crossPos:chromoSize
            %temp = pop2(i,j);
            %pop2(i,j) = pop2(i+1,j);
            %pop2(i+1,j) = temp;
        %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);
%pop
```