统计推断在数模转化系统中的应用

组号：48 姓名：李世通 学号：5140309136 姓名：刘默然 学号：5140309126

摘要：根据本课程规定，本文以传感器批量生产为背景，寻找成本合理的传感器特性校准的优化方案。

关键字：三次样条插值、三次拟合、matlab、遗传算法

1 引言

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）。

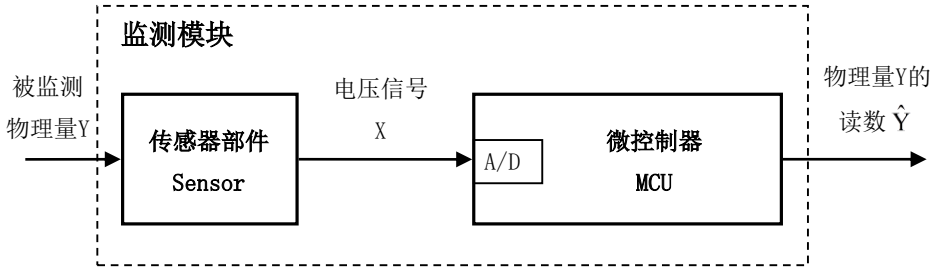

图 1 监控模块的组成

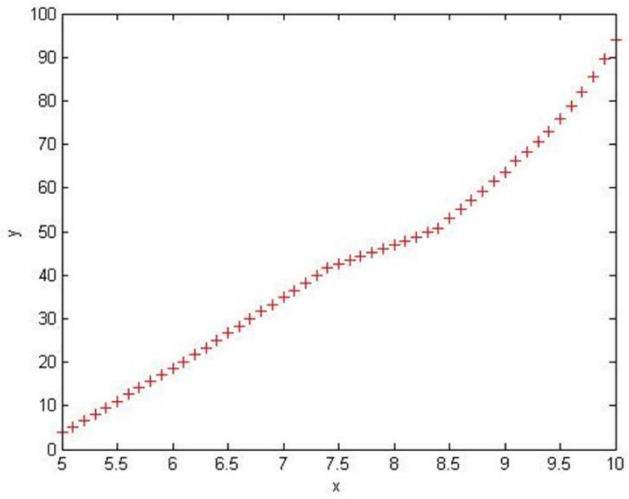

图 2 传感部件个体的输入输出特性

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。[1]

为了得到最优方案，我们需要选择一种拟合或者插值方式与一种取点方式，依据从样本库获取的单点测定数值，按照选择的拟合方法进行拟合并逐一完成定标，根据定标结果，结合原始数据测算出成本记录，最后方案对比将成本最低的方案作为最终结果。

我们分别考虑了三次拟合函数和三次样条插值函数，并选用遗传算法，来找到最优解。

2 基础理论

从上面分析我们可以简单得出：在本课题中，被检测到的物理量 Y 与输出 X 并不具有线性关系，而且不同组的数据之间数据特性的一致性并不强，我们考虑三次拟合方法和三次样条插值方法。

2.1 三次拟合方法的概述

此方法为拟合的常用简单方法，即设一在区间[a,b]上的 n 阶多项式函数

$$\mathrm{P(x)}=\sum_{k=0}^{n} a_k x^k$$

其中 a=x0<x1<„s21<xn=b 是数据点，则它为区间[a,b]上对数据点拟合的多项式。由定义可看出，它有 n+1 个待定系数。确定这些系数 ak 使

$$\mathrm{D}=\sum_{k=1}^{n}[y_k-P(x_k)]\text{^}2$$

最小。这时就得到误差平方拟合多项式。可以把它认为是推断函数。同样，我们利用 matlab 中已有的函数

$$\mathrm{a}=ployfit(x_0,y_0,m)$$

进行多项式拟合。其中输入参数 x0，y0 为要拟合的数据，m 为拟合多项式的次数，输出参数 a 为拟合多项式系数。多项式在 x 处的值 y 可用 matlab 中的 y=polyval(a,x)函数进行计算。[2]

多项式拟合的次数在一定范围内越高，方差等参数越小，拟合曲线与实际测量点的相关性越高，拟合程度越好。但次数的增高导致算法运行时间的延长，效率的降低，而且当次数超过一定范围时，甚至适得其反。例如用 5，6 次函数拟合，运行时间是很难让人接受的，所得结果误差反而越来越大，可见不一定次数越高，拟合曲线就越接近实际曲线。反复测试后，3 次曲线拟合的效果最好。

2.2 三次样条插值的概述

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)∈C2[a,b] ，且在每个小区间[xj,xj+1]上是三次多项式，其中a =x0 <x1<...<xn=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xj 上给定函数值Yj=f(Xj).(j=0,1,...,n)，并成立S(xj)=yj(j=0,1,...,n)，则称S(x)为三次样条插值函数。

2.3 拟合方法选择的讨论

我们小组在实际matlab运行过程中，发现使用三次拟合函数拟合用时比使用三次样条插值的时间普遍短小。但是三次拟合方法无法保证拟合精度平均对结果影响较大。我们根据两种方法运行出来的最终成本比较，三次样条插值的成本

跟优，所以选择使用三次样条插值来拟合（插值）。

3 成本计算[3]

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. **单点定标误差成本**

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

单点定标误差的成本按式上式计算，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。

2. **单点测定成本**

实施一次单点测定的成本以符号q记。本课题指定的q=12。

3. **某一样本个体的定标成本**

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

对样本i总的定标成本按上式计算，式中n_i表示对该样本个体定标过程中的单点测定次数。

4. **标准方案总成本**

按下式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。

$$C=\frac{1}{M}\sum_{i=1}^{M}S_i$$

总体成本较低的校准方案，认定为较优方案。

4 选点方式：

4.1 遗传算法的概述

遗传算法的实现过程实际上就像自然界的进化过程那样。首先寻找一种对问题潜在解进行“数字化”编码的方案。然后用随机数初始化一个种群，种群里面的个体就是这些数字化的编码。接下来，通过适当的解码过程之后，用适应性函数对每一个基因个体作一次适应度评估。用选择函数按照某种规定择优

选择。让个体基因交叉变异。然后产生子代。遗传算法并不保证你能获得问题的最优解，但是使用遗传算法的最大优点在于不必去了解和操心如何去“找”最优解，只要简单的“否定”一些表现不好的个体就行了。

下图是遗传算法的流程示意图。

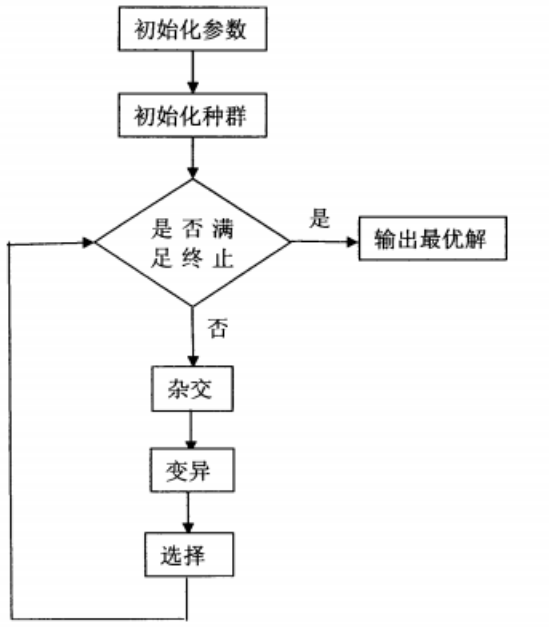

图 3 遗传算法流程示意图

4.2 遗传算法在该实验中的应用与实现（解释）

1.初始条件的设定

种群大小 pop 设定为 100；交叉概率设定为 0.85；突变概率设定为 0.001；进化代数为 100 代。

2.生成初始种群

利用 matlab 中的 rand 函数对矩阵生成随机数，表示产生 100 个个体，作为初始种群。

3.适应度函数的选取

本次的样本是 M 个数据。对于这 M 个点中的每一个点，通过三次插值拟合或三次多项式拟合，之后根据得到的曲线上 x 对应的 y 值对其这一点的残差；接下来对其他点进行相同的操作。最后求所有样本点的残差的平方和，将其取相反数作为种群个体的适应度函数。

4.种群遗传部分

根据轮盘赌法来模拟自然选择过程；

通过交叉互换和基因突变产生一个新种群。

5.终止条件

不断重复进行下一代的繁衍，直到进行到设定的最终进化代数为止。

5 运行结果

由于算法本身的特性，在实际运行中，经常会收敛于一个次优解，而非最优解。因

此，需要多次运行，才能得到相对准确的结果。

我们在多次运行、记录后，得到了一组目前已知的最优解：

1 10 20 28 33 43 51

总成本 97.659250

6 参考文献

[1][3] “统计推断”课程设计的要求 V2.2.doc

[2]百度百科上对三次拟合的相关解释

附录

1.主函数

main.m

```
data=csvread('20150915dataform.csv');
pop=100;    %初始种群个体数，设置为 100
pc=0.85;    %交叉概率（范围为 0.4~0.99）
pm=0.001;   %突变概率（范围 0.0001~0.1）
n=100;      %进化代数
y=zeros(400,51);
y(1:400,:)=data(2:2:800,:);
gene=geneinit(pop);
for g=1:n
    display(g);
    cost=adapt(gene,y,pop);
    gene=select(gene,cost,pop);
    gene=generate(gene,pop,pc);
    gene=mutate(gene,pop,pm);
    display(find(gene(1,:)==1));
end
xx=find(gene(1,:)==1);
assess(xx,y);
```

2.遗传算法

geneinit.m

```
function out = geneinit(pop)
%产生随机初始种群
out=round(rand(pop,51)-0.2);
out(:,1)=1;
out(:,51)=1;
end
```

select.m

```
function out = select(gene,cost,pop)
%模拟自然选择过程
```

```
out=zeros(pop,51);
cost0=max(cost)-cost;
s0=sum(cost0);
s=zeros(pop+1);
s(1)=0;
s(pop+1)=1;
s(2:pop)=sum(cost0(1:pop-1))/s0;
for i=2:pop
    t=rand();
    j=search(t,s,1,pop+1);
    out(i,:)=gene(j,:);

end
sort0=[[1:pop]',cost];
sort0=sortrows(sort0,2);
out(1,:)=gene(sort0(1,1),:);
end
```

search.m

```
function [out] = search(in,s,l,r)
% 二分法查找
mid=floor((l+r)/2);
if in<=s(mid)
    if in>s(mid-1)
        out=mid-1;
    else
        out=search(in,s,l,mid);
    end
else
    if in<=s(mid+1)
        out=mid;
    else
        out=search(in,s,mid,r);
    end
end
end
```

generate.m

```
function out = generate(gene,pop,pc)
%交叉保留
for i=2:floor(pop/2+1)
    out=gene;
    mid=floor(rand()*50)+1;
    t=rand();
```

```
    if t<=pc
        out(i,1:mid)=gene(pop-i+2,1:mid);
        out(pop-i+2,1:mid)=gene(i,1:mid);
        out(i,mid+1:51)=gene(pop-i+2,mid+1:51);
        out(pop-i+2,mid+1:51)=gene(i,mid+1:51);
    end
end
End
```

mutate.m

```
function out = mutate(gene,pop,pm)
%变异
out=gene;
for i=2:pop;
    for j=2:50;
        t=rand();
        if t<=pm
            out(i,j)=~out(i,j);
        end
    end
end
end
```

3.成本计算

errorcost.m

```
function [out] = errorcost(dy)
%单个个体成本误差计算
t=abs(dy);
t0=sum(sum(t<=0.4));
t1=sum(sum(t<=0.6))-t0;
t2=sum(sum(t<=0.8))-t0-t1;
t3=sum(sum(t<=1))-t0-t1-t2;
t4=sum(sum(t<=2))-t0-t1-t2-t3;
t5=sum(sum(t<=3))-t0-t1-t2-t3-t4;
t6=sum(sum(t<=5))-t0-t1-t2-t3-t4-t5;
t7=sum(sum(t>5));
out=0.1*t1+0.7*t2+0.9*t3+1.5*t4+6*t5+12*t6+25*t7;
end
```

adapt.m

```
function out = adapt(gene,y,pop)
%计算每个个体的平均成本
```

```
out=zeros(pop,1);
x=5:0.1:10;
for i=1:pop
    c=sum(gene(i,:)==1);
    pos=find(gene(i,:)==1);
    xx=5+(pos-1)*0.1;
    yy=y(:,pos);
    f=spline(xx,yy);
    dy=ppval(f,x)-y;
    out(i)=12*c+errorcost(dy)/400;
end
min(out)
mean(out)
end
```

assess.m

```
%将选点位置与最终结果写入 answer.txt
function [out] =assess(in,y)
out=length(in)*12;
x=5:0.1:10;
xx=5+(in-1)*0.1;
yy=y(:,in);
f=spline(xx,yy);
dy=ppval(f,x)-y;
out=out+errorcost(dy)/400;
s=sum(dy.^2);
fid=fopen('answer.txt','a');
fprintf(fid,'position: [ ');
fprintf(fid,'%2d ',in);
fprintf(fid,']  mean_cost: %7f\n\n',out,s);
fclose(fid);
end
```

4.测试部分

mycurvefitting.m

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

test_ur_answer.m

```
%%%%%%% 答案检验程序 2015-11-04 %%%%%%%%
```

```
my_answer=[ 1,10,20,28,33,43,51 ];
my_answer_n=size(my_answer,2);

% 标准样本原始数据读入
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;


% 定标计算
index_temp=logical(my_answer_gene);
x_optimal=x(:,index_temp);
y0_optimal=y0(:,index_temp);
for j=1:nsample
    % 请把你的定标计算方法写入函数 mycurvefitting
    y1(j,:)=mycurvefitting(x_optimal(j,:),y0_optimal(j,:));
end

% 成本计算
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;
```

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