

统计推断在数模转换中的应用

组号 56 孙媛 5140309097 周慧晶 5140309266

摘要：本论文主要针对某种批量生产的电子产品内部模块的非线性输入输出特性之间的函数关系建模与分析。通过数学建模、拟合算法、实验验证等过程，逐步找到为该模块的批量生产，设计一种成本合理的传感特性校准方案。实验通过模拟遗传算法、三次曲线拟合、三次样条插值等方法分别给出该模块输入输出的特征点选取方案和函数关系，最终确定最优方案。

关键词：样本，特征点，曲线拟合，样条差值法，优化采样，遗传算法

1 引言

在生产和研究中，对物理现象的研究可以通过确定其输入输出的关系来确定其性质，这就有必要提出一种解决方案来确定这二者的关系。但是实际研究生产中发现，我们很难找到一个能够精准描述二者关系的函数，这就要求我们运用实际测得的数据，通过曲线拟合来找到一个最符合输入与输出关系的函数。为了保证准确找到最优解，我们就需要运用统计推断的知识找到合适的特征点，对这些特征点进行拟合，计算得到的曲线与实际输出的残差，残差最小的解围最优解。本次统计推断的报告是通过研究电子产品输入的电压信号 X 与被监测的物理量 Y 的关系，分析大量的数据得出最优解，来说明统计推断在获得输入输出关系时的作用。

2 评价标准的构建

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

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

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。

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 $q=12$ 。

3) 某一样本个体的定标成本

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

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。

4) 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。

3 统计算法理论基础

在本试验中我们选取模拟遗传算法来取点，提高取点效率，降低定标成本。

1) 遗传算法

遗传算法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）演化而来的随机化仿生搜索方法。遗传算法将每一个可能的解看做一个生物个体，将若干生命个体组成种群。算法中需要对解进行编码，以模仿生物的基因染色体。遗传算法的流程图如图 1 所示。算法开始运行时，遗传算法将随机生成一个含有一定数量个体的初始种群。之后，遗传算法将计算种群中的个体的适应度，然后将模拟自然选择过程，下一步模拟生物交配繁殖，对两个基因代码进行交叉，产生两个新个体。然后再模拟基因突变，最终产生新一代的种群。重复本段所讲述过程直到满足终止条件即可。

2) 遗传算法的基本步骤

(1) 在一定编码方案下，随机产生一个初始种群；

(2) 用相应的解码方法，将编码后的个体转换成问题空间的决策变量，并求得个体的适应值；

(3) 按照个体适应值的大小，从种群中选出适应值较大的一些个体构成交配池；

(4) 由交叉和变异这两个遗传算子对交配池中的个体进行操作，并形成新一代的种群；

(5) 反复执行步骤 ，直至满足收敛判据为止。

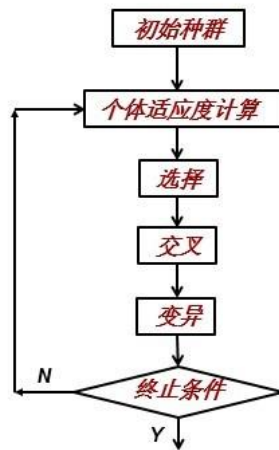


图 1 遗传算法流程图

3) 遗传算法特点分析

与传统的优化算法相比，遗传算法具有如下优点：

- a) 不是从单个点，而是从多个点构成的群体开始搜索；
- b) 在搜索最优解过程中，只需要由目标函数值转换得来的适应值信息，而不需要导数等其它辅助信息；
- c) 搜索过程不易陷入局部最优解。

4 拟合方法基本原理

在选择拟合方法是，我们通过不同的拟合方法，有选择性的随机选点来计算定标成本，初步确定较好的拟合方法。

1) 三次样条插值

三次样条插值法一种非线性插值法，它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。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（边界点的导数为 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）。

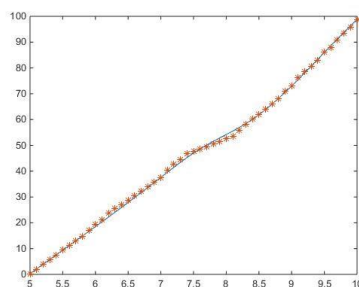


图 2 三次样条插值拟合 函数图像

2) 五次多项式插值

多项式拟合是最容易想到的拟合方法。易知多项式次数越高，拟合的误差越小，但是次数高了之后也容易出现不符合物理规律的现象。本次实验中选取五次多项式来进行拟合，并计算其定标成本。通过直接调用 `polyfit(x, y, n)` 函数来拟合，经过 `matlab` 运算得出的五次多项式的成本。

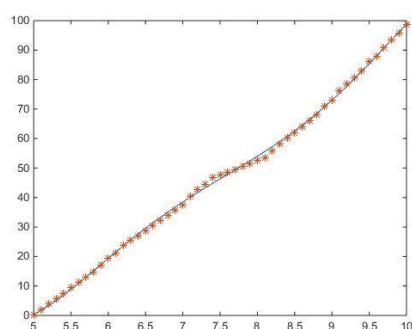


图 3 五次多项式插值 函数图像

3) 分段三次 Hermite 插值

一般的，从各种试验得来的数据总有一定的数量，而利用插值技术能够从有限的数据中获取整体的状态。而 Hermite 插值不仅保证了插值函数与原函数在给定数据点处得拟合，同时保证了在相应点处导数的相同，从而在很大程度上保证了曲线的“光滑性”。因此，通过 `Matlab` 实现 Hermite 插值具有很普遍的意义。

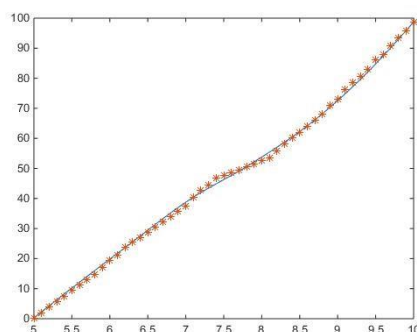


图 4 分段三次 Hermite 插值函数图像

4) 拟合方法的选取

在本试验中，分别采用三次样条插值法、五次多项式插值法和分段三次 Hermite 插值法进行试验，最优实验结果如下：

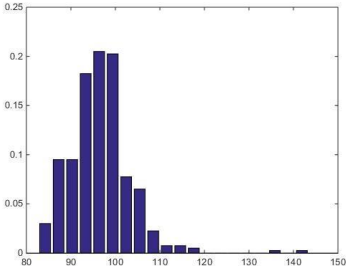
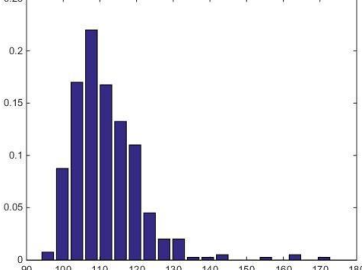
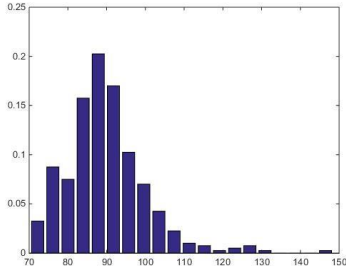
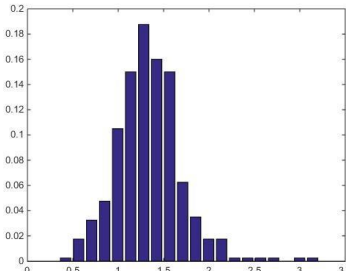
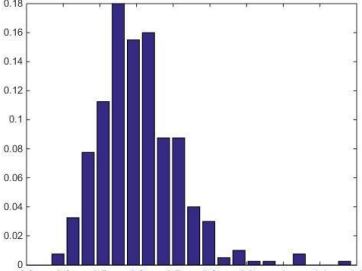
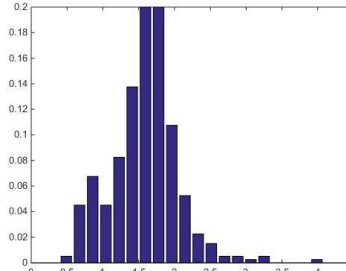
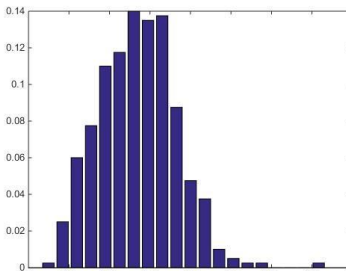
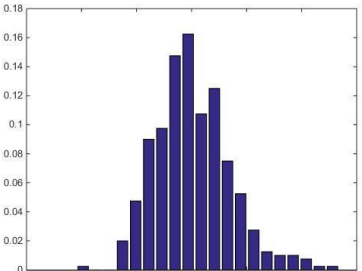
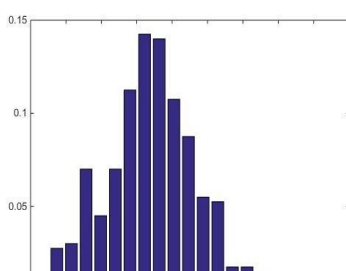
插值方法	三次样条插值法	五次多项式插值	分段三次 Hermite 插值
试验结果	1 8 20 27 34 44 51 总体成本为96.68 1 9 20 27 35 44 51 总体成本为 96.51 1 9 20 28 34 44 51 总体成本为 96.95	1 7 15 27 35 45 51 总体成本为111.75 1 7 15 29 36 46 51 总体成本为115.44 1 6 18 26 34 48 51 总体成本为116.79	1 14 23 31 39 51 总体成本为 89.72 1 15 25 33 43 51 总体成本为 89.77 1 13 22 29 38 51 总体成本为 89.82
成本分布			
最大误差概率分布			
平均误差概率分布			

表 1 不同拟合方法试验结果

经过分析比较我们可以得出结论是，三次 Hermite 插值比五次多项式拟合和三次样条插值总体上误差更小，更加稳定精确，因此在本次课题中，我们选择三次 Hermite 插值方法进行拟合。

5 统计推断算法的实现

Matlab 遗传算法的实现中，需要确定一些初始化的参数，如编码串长度、种群大小、交

叉和变异概率，为保证算法的运行效率和群体的多样性，一般取种群数目为 20~100，交叉是产生新个体的基本方法，概率一般取 0.4~0.99，变异概率也是新个体产生重要参数，一般取 0.0001~0.1，我们选取交叉概率 pc=0.9，突变概率 pm=0.01，进化代数为 100，遗传算法的实现共由一个运行程序和八个函数构成，各功能如下表所示。

函数	功能
mainFun()	作为程序运行的主函数，原始数据的设定和处理。
geneinit(pop_size)	遗传算法矩阵的初始化，数据库样品数量为 pop_size
search(in, s, l, r)	找出 s 向量下标从 l 到 r（元素在 l 到 r 之间从小到大排列）中小于 in 最大项的下标
generate(gene, pop_size, pc)	完成遗传算法中的交换。父母矩阵为 gene，数据库样品数量为 pop_size，交换的概率为 pc
cost(gene, y, pop_size)	计算成本，选定观测点的矩阵为 gene，所有数据观测点的矩阵为 y，数据样品库数量为 pop_size
errorcost(dy)	计算模拟得到函数与真实值偏差的成本。记录模拟值与真实值偏差的矩阵为 dy
select(gene, cost, pop_size)	遗传算法中的选择，选择观测点的矩阵为 gene，成本向量 cost，数据样品库 pop_size
assess(in, y)	在屏幕上打印出模拟的函数，记录选取得测试点的矩阵为 in，所有样品数据的矩阵为 y

表 2 遗传算法

6 结论

在本次试验中，我们共运行了九次遗传算法，其中最优秀的定标方案为：在希望总体误差最小的情况下，可选取 {1, 14, 23, 31, 39, 51} 观察点作为事前观察点，再利用三次 Hermite 插值法进行定标，可以使定标的平均成本达到 89.72。

7 参考文献

[1]袁焱

“统计推断”课程设计的要求 V2.2 2015-9-22

[2]上海交大电子工程系

统计推断在数模转换系统中的应用课程讲义

[3]刘国华、包宏、李文超

用 MATLAB 实现遗传算法程序

附：遗传算法代码和三种拟合方法代码：

```
(1)    function []=mainFun()

theData=csvread('20150915dataform.csv');

pop_size=100;

pc=0.9;

pm=0.01;

generation_size=100;

y=zeros(400,51);

y(1:400,:)=theData(2:2:800,:);

gene=geneinit(pop_size);

for g=1:generation_size

    display(g);

    cost1=cost(gene,y,pop_size);

    gene=select(gene,cost1,pop_size);

    gene=generate(gene,pop_size,pc);

    gene=mutate(gene,pop_size,pm);

    display(find(gene(1,:)==1));

end

xx=find(gene(1,:)==1);

assess(xx,y);

end
```

```
(2)    function out = geneinit(pop_size)

out=round(rand(pop_size,51)-0.2);

out(:,1)=1;

out(:,51)=1;

end
```

```

(3)    function out = generate(gene,pop_size,pc)

for i=2:floor(pop_size/2+1)

    out=gene;

    t=rand();

    if t<=pc

        out(i,:)=gene(pop_size-i+2,:);

        out(pop_size-i+2,:)=gene(i,:);

    end

end

end

```

```

(4)    function out = mutate(gene,pop_size,pm)

out=gene;

for i=2:pop_size;

    for j=2:50;

        t=rand();

        if t<=pm

            out(i,j)=~out(i,j);

        end

    end

end

end

```

```

(5)    function [out] = search(in,s,l,r)

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
end

```

```

(6) function out = select(gene,cost0,pop_size)

out=zeros(100,51);

cost1=max(cost0)-cost0;

s0=sum(cost1);

s=zeros(pop_size+1);

s(1)=0;

s(pop_size+1)=1;

s(2:pop_size)=sum(cost1(1:pop_size-1))/s0;

for i=2:pop_size

    t=rand();

    j=search(t,s,1,pop_size+1);

    out(i,:)=gene(j,:);

end

```

```

sort0=[1:pop_size]',cost0];

sort0=sortrows(sort0,2);

out(1,:)=gene(sort0(1,1),:);

end

```

```

(7) function out= cost(gene,y,pop_size)

out=zeros(pop_size,1);

x=5:0.1:10;

for i=1:pop_size

    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)/469;

end

min(out);

mean(out);

end

```

```

function out= cost(gene,y,pop_size)

out=zeros(pop_size,1);

x=5:0.1:10;

for i=1:pop_size

    c=sum(gene(i,:)==1);

    pos=find(gene(i,:)==1);

```

```

        xx=5+(pos-1)*0.1;

        yy=y(:,pos);

        y1=pchip(xx,yy,x);

        dy=y1-y;

        out(i)=12*c+errorcost(dy)/469;

    end

    min(out);

    mean(out);

end

mycurvefitting

function y1 = mycurvefitting( x_premea,y0_premea )

x=[5.0:0.1:10.0];

y1=interp1(x_premea,y0_premea,x,'pchip');

end

function out= cost(gene,y,pop_size)

out=zeros(pop_size,1);

x=5:0.1:10;

for i=1:pop_size

    c=sum(gene(i,:)==1);

    pos=find(gene(i,:)==1);

    xx=5+(pos-1)*0.1;

    for j=1:400

        yy=y(j,pos);

        a=polyfit(xx,yy,5);

        y1(j,:)=polyval(a,x);

```

```

        end

        dy=y1-y;

        out(i)=12*c+errorcost(dy)/469;

    end

    min(out);

    mean(out);

end


mycurvefitting

function y1 = mycurvefitting( x_premea,y0_premea )

x=[5.0:0.1:10.0];

a=polyfit(x_premea,y0_premea,5);

y1= polyval(a,x);

end


(8)    function [out]=errorcost(dy)

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

```

```

(9)    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

```