统计推断在数模转换系统中的应用

组号：4 王桑田 江文宇

摘要：本报告以实验数据为基础，探讨统计推断在数模转换系统中的应用。本报告的研究对象是某产品中的一个检测模块，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。由于每次的输入输出测定成本较高，因此需要用较少的测量次数得到较高的校准精度。

关键字：插值拟合、粒子群算法、模拟退火算法、遗传算法

1.引言

对于某产品中的一个输入输出呈明显非线性的检测模块，需要找到一种合理的成本校准方案。由于测量成本较高，所以需要用较少的测量次数得到较高的校准精度。寻找这个校准方案的两个核心问题是输入输出曲线拟合方式的选取以及测量数据点的选择。对于输入输出曲线拟合方式，插值拟合是一种比较好的选择；对于数据点的选择，为了尽量快速、准确的选出数据点，可以采用以遗传算法等优化算法。

2.拟合方式的研究

2. 1 多项式拟合

可直接使用matlab中的polyfit()函数进行多项式拟合。我们尝试用三次、四次、五次、六次多项式进行拟合，通过对残差平方和的比较来确定合理的拟合方式。首先选取了6组特征点，分别为：

(1)[1,7,12,24,33,48,51]　(2)[1,6,13,25,36,47,51]

(3)[1,4,14,24,37,49,51]　(4)[1,4,17,29,38,47,51]

(5)[1,5,16,29,38,48,51]　(6)[1,5,14,27,39,45,51]

这六组特征点并不是最后的最优选点方案，仅用于对三次、四次、五次、六次多项式拟合的优劣。

多项式拟合的matlab程序见附录1。

图1、2、3、4是采用三次、四次、五次、六次多项式拟合画出的残差平方和的图像。

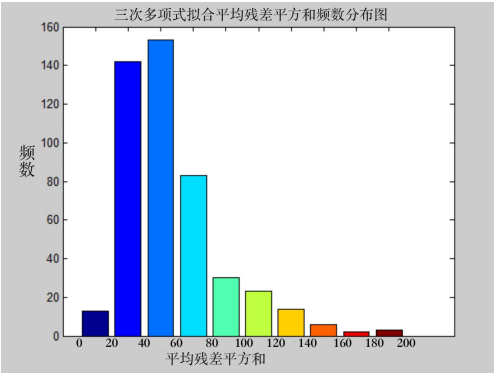

图 1

三次多项式拟合平均残差平方和频数分布图

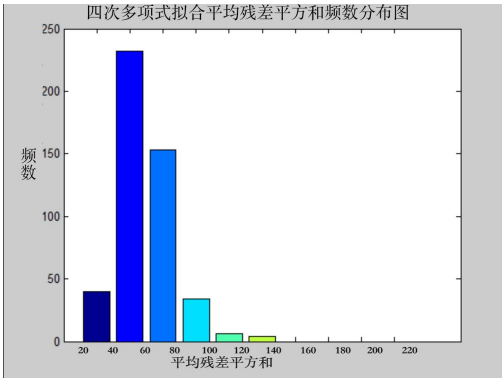

图 2

四次多项式拟合平均残差平方和频数分布图

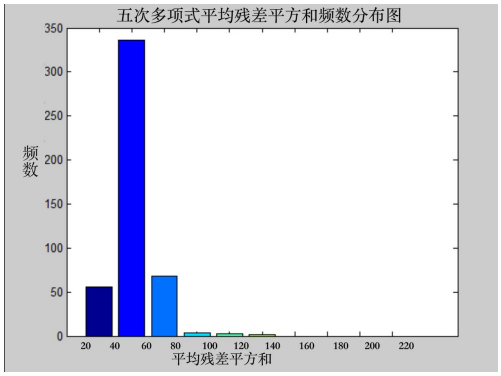

图 3

五次多项式拟合平均残差平方和频数分布图

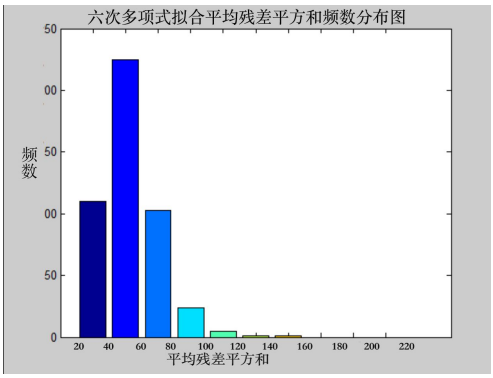

图 2

六次多项式拟合平均残差平方和频数分布图

最后计算出 469 组有效数据的平均残差平方和如下：

多项式次数	平均残差平方和
三次拟合	67.4482
四次拟合	48.9954
五次拟合	40.8937
六次拟合	42.7854

2. 2 插值拟合

插值是在离散数据的基础上补插连续函数，使得这条连续曲线通过全部给定的离散数据点。Matlab 中的插值函数为 interp1()，可以采用线性插值，三次插值，三次样条插值，这三种插值要求数据必须是线性的，本报告采用数据符合这个特点。

多项式拟合的 matlab 程序见附录 2。

为了与多项式拟合比较优劣，选取与多项式拟合所用相同的特征点，分别为：

(1)[1,7,12,24,33,48,51]　(2)[1,6,13,25,36,47,51]

(3)[1,4,14,24,37,49,51]　(4)[1,4,17,29,38,47,51]

(5)[1,5,16,29,38,48,51]　(6)[1,5,14,27,39,45,51]

使用interp1()函数，计算出469组有效数据的平均残差平方和为：

插值方式	平均残差平方和
多项式插值	28.2494
三次样条插值插条	28.7222

可见，用三次样条插值插条和多项式插值所得残差平方和于三次、四次、五次、六次多项式拟合比较，发现三次样条插值插条和多项式插值明显优于三次、四次、五次、六次多项式拟合，因此，决定采用插值。三次样条插值和多项式插值所得残差非常接近，可选用三次样条插值。

3.特征点的选取

3.1 遗传算法

遗传算法是计算数学中用于解决最佳化的搜索算法，是进化算法的一种。进化算法最初是借鉴了进化生物学中的一些现象而发展起来的，这些现象包括遗传、突变、自然选择以及杂交等。遗传算法通常实现方式为一种计算机模拟。对于一个最优化问题，一定数量的候选解（称为个体）的抽象表示（称为染色体）的种群向更好的解进化。传统上，解用二进制表示（即 0 和 1 的串）。进化从完全随机个体的种群开始，之后一代一代发生。在每一代中，整个种群的适应度被评价，从当前种群中随机地选择多个个体（基于它们的适应度），通过自然选择和突变产生新的生命种群，该种群在算法的下一次迭代中成为当前种群。

在本次定标的操作中，我们主要实现了对定标点和个数的选择，而拟合的方法以三次样条插值为主，以 100 个个体为一个种群，每一个个体有 51 个基因。以下是遗传算法的主要思想及实现方式：

（1）编码：使用二进制编码，随机产生一个初始种群。本次编码为长度为 51 的基因链，其中 0 代表不选取此点，以代表选取此点作为定标点。

（2）生成初始群体：种群规模表示每一代种群中所含个体数目。随机产生 N 个初始个体，N 个个体组成一个初始群体，N 表示种群规模的大小。当 N 取值较小时，可提高遗传算法的运算速度，但却降低种群的多样性，容易引起遗传算法早熟，出现假收敛；而 N 当取值较大时，又会使得遗传算法效率降低。遗传算法以该群体作为初始迭代点；

（3）适应度检测：根据实际标准计算个体的适应度，评判个体的优劣，即该个体所代表的可行解的优劣。其中适应度即为所求的成本的函数，在程序中单独存在；

（4）选择：从当前群体中选择优良（适应度高的）个体，使它们有机会被选中进入下一次迭代过程，舍弃适应度低的个体。本例中采用轮盘赌的选择方法，即个体被选择的几率与其适应度值大小成正比； 而由于本次试验要求成本越小越好，所以采取倒数的方法令其为适应度。

（5）交叉：遗传操作，根据设置的交叉概率对交配池中个体进行基因交叉操作，形成新一代的种群，新一代中间个体的信息来自父辈个体，体现了信息交换的原则。交叉概率控制着交叉操作的频率，由于交叉操作是遗传算法中产生新个体的主要方法，所以交叉概率通常应取较大值；但若过大的话，又可能破坏群体的优良模式。一般取 0.4 到 0.99。

（6）变异：随机选择中间群体中的某个个体，以变异概率大小改变个体某位基因的值。变异为产生新个体提供了机会。变异概率也是影响新个体产生的一个因素，变异概率小，产生新个体少；变异概率太大，又会使遗传算法变成随机搜索。一般取变异概率为 0.0001—0.1。

（7）终止进化代数：本例中规定遗传代数的收敛判据，根据设置好的进化代数，搜索到规定代数后自动停止　本次试验中设为 25 次。

遗传算法实现见附录 3。

运行遗传算法后得到的最佳选点方案为[1，11，19，25，31，44，49]，最低成本为 97.909。

多次运行后发现，每一次运行的结果基本在 15 次左右趋于稳定。由于选取点都包含了首点和末点，较符合实际情况。在观察每次成本和最小成本后，发现每次平均成本下降趋势十分明显，而取点个数稳定在 6 至 7 个点，符合预期的假设。总成本仅能下降到 97 左右，与预想有一定差距。因此，我们尝试了改变拟合方式，在试验了多项式，级数等其他拟合方式后，发现并不能更好的收敛或发现更小的成本。

3.2 模拟退火算法

模拟退火算法来源于固体退火原理，将固体加温至充分高，再让其徐徐冷却，加温时，固体内部粒子随温升变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。

在本次标定中，我们给定 6 个点进行标定，以下是具体思想与实现方法：

（1）在选定范围内随机选出 6 个点作为初始解 S，计算成本，并将成本作为当前最优成本 cost

（2）初始化：初始温度 t=400，当前温度 t_now=400，，初始解状态 S(是算法迭代的起点)

（3）产生新解 S′

（4）计算新解 S′的成本 cost′，并与当前最优成本 cost 比较

（5）如果 cost′<cost，则接受当前解作为最优解，否则，以 1/[10*t_now*(cost-cost′)]概率接受当前解为新解。

（7）进行降温，降温函数为 t_now=t_now*0.999

（8）如果 t_now<10，则退火结束，输出解

模拟退火算法实现见附录 4。

在模拟退火算法中，当新解的成本比最优解的成本大时，需要以一定概率接受新解为最优解，这是防止陷入局部最优解。这里的概率公式的选取很关键，既要保证跳变概率不大，又不能太小（影响算法速度），经过多次试验，我们选取了 1/[10*t_now*(cost-cost′)]。

运行模拟退火后，得到的最优选点为：[1,14,22,29,38,49]；成本为：85.586。多次运行模拟退火算法后，发现一般情况下，最低成本能降到 90 以下，可以认为是比较低的成本。

3.3 粒子群算法

在基于以上算法的同时，我们查资料发现有一种更高效寻找最优解的方法——粒子群算法。

粒子群算法，也称粒子群优化算法（Particle Swarm Optimization），是近年来发展起来的一种新的进化算法。PSO 算法属于进化算法的一种，和遗传算法相似，它也是从随机解出发，通过迭代寻找最优解，它也是通过适应度来评价解的品质，但它比遗传算法规则更为简单，它没有遗传算法的“交叉”(Crossover) 和“变异”(Mutation) 操作，它通过追随当前搜索到的最优值来寻找全局最优。这种算法有实现容易、精度高、收敛快等优点。

PSO 模拟鸟群的捕食行为。即一群鸟在随机搜索食物，所有的鸟都不知道食物在那里，所以最简单有效的就是搜寻目前离食物最近的鸟的周围区域。PSO 初始化为一群随机粒子(随机解)。然后通叠代找到最优解。

在最开始，随机生成一群粒子。由于技术有限，仅假设在 n 维空间内运动，即仅选 n 个点作为特征点（n=5,6,7,8 为宜）。然后，立在开始在自己的周围寻找最优解。在每一次叠代中，粒子通过跟踪两个"极值"来更新自己。第一个就是粒子本身所找到的最优解。这个解叫做个体极值 pBest.另一个极值是整个种群目前找到的最优解。这个极值是全局极值 gBest。在找到这两个最优值时,粒子根据如下的公式来更新自己的速度和新的位置。

v[ ]=w*v[ ] c1*rand()*(pbest[ ]-present[]) c2*rand()*(gbest[]-present[ ]) (a)

present[ ]=persent [ ] *v[] **(b)**

v[]是粒子的速度,w 是惯性权重即粒子保存自身特性的比例,persent[]是当前粒子的位置。c1,c2 是学习因子即粒子跟随当前最优粒子的程度。本次设定 c1=c2=2，w=0.8。然后粒子不断趋于最优解，得到最终结果。

粒子群算法实现见附录 5。

在本次试验中，由于粒子位置可能超过界线，所以设定了边界条件及越界惩罚。我们认为，既然其有跟随最优解的特性，不如在计算后用其来寻找更优解更加节省时间。对比来说，50 次更新后，以上想法已将成本降到 90 左右，而用随机粒子来迭代仅为 99 左右。最后的最优选点为[1，7，19，28，33，45，51],最低成本为 97.504。

其中，我们发现，粒子群算法并没有一直趋向于某一个方向进行，而是由于边界条件而有一定的随机性，这避免了进入局部最优解的情况。所以，我们认为这种算法是有效的。

4.总结

用不同算法得出的最优选点方案和最优成本如下表：

算法	拟合方式	最优选点方案	最优成本
遗传算法	多项式插值	1，11，19，25，31，44 ，49	97.909
退火算法	多项式插值	1，14，22，29，38，49	85.586
粒子群算法	多项式插值	1,，7，19，28，33，45，51	97.503

其中，三次多项式插值拟合、用退火算法选点，所得结果所得的成本最低。

需要注意的是，以上算出的最低成本不一定就是真正的最低成本。如果要得到最优成本，首先需要遍历每种选点方案，再确定拟合方式，由于总的方案数量庞大，无法在短时间内得出，并且，拟合方式有无穷多种，不能确定到底哪一种是最优的。在时间允许范围内，我们设计了不同算法，比较并确定了拟合方式。得到的最优选点方案为[1，14，22，29，38，49],最低成本为 85.586。

参考文献：

[1]孙祥，《matlab7.0 基础教程》，清华大学出版社

[2]张文修，梁怡，《遗传算法的数学基础》，西安交通大学出版社

[3]龚纯 王正林，《精通 matlab 最优化计算》，电子工业出版社

附录 1：

Polyfit.m

```
data=csvread('20141010dataform.csv',0,0);
err_three=zeros(1,469);
err_four=zeros(1,469);
err_five=zeros(1,469);
err_six=zeros(1,469);
for i=1:469;
   xdata=data(1,:);
   ydata=data([2*i],:);
    n=[1,7,12,24,33,48,51;
    1,6,13,25,36,47,51;
    1,4,14,24,37,49,51;
    1,4,17,29,38,47,51;
    1,5,16,29,38,48,51;
    1,5,14,27,39,45,51];
    difference_three=zeros(1,6);
    difference_four=zeros(1,6);
    difference_five=zeros(1,6);
    difference_six=zeros(1,6);
    for choose =1:6

     fity=ydata(n(choose,:));
     fitx=xdata(n(choose,:));
     fitthree=polyfit(fitx,fity,3);
      fitfour=polyfit(fitx,fity,4);
       fitfive=polyfit(fitx,fity,5);
        fitsix=polyfit(fitx,fity,6);
     infer_y_three=polyval(fitthree,xdata);
      infer_y_four=polyval(fitfour,xdata);
       infer_y_five=polyval(fitfive,xdata);
        infer_y_six=polyval(fitsix,xdata);
        for k=1:51
            difference_three(1,1)=difference_three(1,1)+(infer_y_three(1,k)-ydata(1,k))^2;
             difference_four(1,1)=difference_four(1,1)+(infer_y_four(1,k)-ydata(1,k))^2;
              difference_five(1,1)=difference_five(1,1)+(infer_y_five(1,k)-ydata(1,k))^2;
               difference_six(1,1)=difference_six(1,1)+(infer_y_six(1,k)-ydata(1,k))^2;
        end
    end



average_difference_three=0;
average_difference_four=0;
```

```
average_difference_five=0;
average_difference_six=0;




    for choose=1:6
        average_difference_three=average_difference_three+difference_three(1,choose);
        average_difference_four=average_difference_four+difference_four(1,choose);
        average_difference_five=average_difference_five+difference_five(1,choose);
        average_difference_six=average_difference_six+difference_six(1,choose);
    end



    average_difference_three= average_difference_three/6;

    average_difference_four= average_difference_four/6;

    average_difference_five= average_difference_five/6;

    average_difference_six= average_difference_six/6;

    err_three(1,i)=average_difference_three;

    err_four(1,i)=average_difference_four;

    err_five(1,i)=average_difference_five;

    err_six(1,i)=average_difference_six;
end
figure
stem(err_three);
axis([0,469,0,200]);
xlabel('组号');ylabel('残差平方和');title('三次拟合残差平方和分布');
figure
stem(err_four);
axis([0,469,0,200]);
xlabel('组号');ylabel('残差平方和');title('四次拟合残差平方和分布');
figure
stem(err_five);
axis([0,469,0,200]);
xlabel('组号');ylabel('残差平方和');title('五次拟合残差平方和分布');
figure
stem(err_six);
```

```
axis([0,469,0,200]);
xlabel('组号');ylabel('残差平方和');title('六次拟合残差平方和分布');
averageerr_three=0;
averageerr_four=0;
averageerr_five=0;
averageerr_six=0;
       for i=1:469
           averageerr_three=averageerr_three+err_three(1,i);
           averageerr_four=averageerr_four+err_four(1,i);
           averageerr_five=averageerr_five+err_five(1,i);
           averageerr_six=averageerr_six+err_six(1,i);

       end
       averageerr_three=averageerr_three/469;
       averageerr_four=averageerr_four/469;
       averageerr_five=averageerr_five/469;
      averageerr_six=averageerr_six/469;
      averageerr_three
      averageerr_four
      averageerr_five
      averageerr_six
```

附录 2：

```
Insert.m
data=xlsread('20141010dataform.csv');
err_spline=zeros(1,469);

for i=1:469

   datax=data([2*i-1,2*i],:);
   x=datax(1,:);
   y=datax(2,:);
   n=[1,5,12,24,33,48,51;
   1,6,13,25,36,47,51;
   1,4,14,24,37,49,51;
   1,4,17,29,38,47,51;
   1,5,16,29,38,48,51;
   1,5,14,27,39,45,51];
   cubic_difference=zeros(1,6);
   spline_difference=zeros(1,6);
   for choose=1:6
       fitx=x(n(choose,:));
       fity=y(n(choose,:));
```

```
        cubic_yi=interp1(fitx,fity,x,'pchip');
        spline_yi=interp1(fitx,fity,x,'spline');
        for k=1:51
            cubic_difference(1,choose)=cubic_difference(1,choose)+(cubic_yi(1,k)-y(1,k))^2;
            spline_difference(1,choose)=spline_difference(1,choose)+(spline_yi(1,k)-y(1,k))^2;
        end
    end
    cubic_average_difference=0;
    spline_average_difference=0;
    for choose=1:6
        cubic_average_difference=cubic_average_difference+cubic_difference(1,choose);
        spline_average_difference=spline_average_difference+spline_difference(1,choose);
    end
    cubic_average_difference=cubic_average_difference/6;
    spline_average_difference=spline_average_difference/6;

    cubic_dif(1,i)=cubic_average_difference;
    err_spline(1,i)=spline_average_difference;
end
figure
stem(cubic_dif);
axis([0,469,0,200]); xlabel('组号');ylabel('残差平方和');title('三次多项式插值残差平方和分布');
figure
stem(err_spline);
axis([0,469,0,200]); xlabel('组号');ylabel('残差平方和');title('三次样条插值残差平方和分布');
cubic_average=0;
spline_average=0;

for i=1:469;
    cubic_average=cubic_average+cubic_dif(1,i);
    spline_average=spline_average+err_spline(1,i);
end
cubic_average=cubic_average/469;
spline_average=spline_average/469;

spline_average
cubic_average
```

附录3:

```
yichuan.m
clear all;

format long g;
```

```
global npop;
global gene_pop;%50个种群的51个基因
global cost_pop;%每个个体分别成本
global optimal_cost;%最优成本
global optimal_indi; %最,优个体
global x0;
global y0;
global data;
data=csvread('20141010dataform.csv',0,0);
[M,N]=size(data);
nsample=469;          %样本数量
npoint=51;               %个体数量
x=zeros(nsample,npoint);
y0=zeros(nsample,npoint);
for i=1:nsample
x0(1,:)=5.0:0.1:10;           %x的值
y0(1,:)=data(2*i,:);       %y值

end
clear i;

npop=50;         %种群数

 gene_pop=zeros(npop,npoint);

init_pop(npop,51);             %编码


optimal_cost=-1000000;
optimal_indi=zeros(1,51);


cost_pop=zeros(1,npop);
fprintf('   G   mean_cost min_cost\n');

for generation=1:20%代数

    cost_pop=zeros(1,npop);


    fitvalue;      %成本（为负数）
```

```
    sorting ;      %从小到大排序



    opa_selection;       %选择


    opa_crossover;      %交叉

    opa_mutation;       %变异

fprintf('  %2d    %6.1f   %5.1f\n',generation,-mean(cost_pop),-max(cost_pop));  %输出

end
sorting;
-optimal_cost
decode(optimal_indi)


cheng.m
 function [ s ] = cheng(x,y,A,B)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

s=0;

if abs(A(1,y)-B(2*x,y))<=0.5
    s=0;
elseif abs(A(1,y)-B(2*x,y))<=1
    s=0.5;
elseif abs(A(1,y)-B(2*x,y))<=2
    s=1.5;
elseif abs(A(1,y)-B(2*x,y))<=3
    s=6;
elseif abs(A(1,y)-B(2*x,y))<=5
    s=12;

elseif abs(A(1,y)-B(2*x,y))>5
    s=25;
end

end
decode.m
function [ output ] = decode( input )
```

```
ddddd=1:51;
output=zeros(1,51);
for i=1:51
   if input(i) ==1
      output(i)=ddddd(i);
   else
      output(i)=0;
   end

end
output(output==0)=[];
end

fitvalue.m
function [ ] = fitvalue(  )
global gene_pop;
global cost_pop;
global npop;

global data;



for k=1:npop

    my_answer=decode(gene_pop(k,:));
my_answer_n=size(my_answer,2);

% 标准样本原始数据读入
minput=dlmread('20141010dataform.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_5=(errabs<=0.5);
le1_0=(errabs<=1);
le2_0=(errabs<=2);
le3_0=(errabs<=3);
le5_0=(errabs<=5);
g5_0=(errabs>5);
sij=0.5*(le1_0-le0_5)+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_pop(1,k)=sum(si)/nsample;
cost_pop(1,k)=-cost_pop(1,k);


clear s;
clear t;
clear k;

end
end


Init_pop.m
function [  ] = init_pop(npop,npoint)
global gene_pop;

gene_pop=round(rand(npop,npoint));


end


opa_crossover.m
```

```
function [  ] = opa_crossover(  )


global gene_pop;
global npop;


cross_rate=1;

for i=1:2:npop

   if(rand < cross_rate)

      cross_pos = round(rand * 51);

      if or (cross_pos == 0, cross_pos == 1)

         continue;

      end
      for j=cross_pos:51

         temp = gene_pop(i,j);

         gene_pop(i,j) = gene_pop(i+1,j);

         gene_pop(i+1,j) = temp;
      end

   end
   clear i;
   clear j;


end

opa_mumation.m
function [  ] = opa_mutation()
global gene_pop;
global npop;

mutate_rate=0.5;
for i=1:npop
   if rand < mutate_rate
```

```
       mutate_pos = round(rand*51);
       if mutate_pos == 0
           continue;
       end
       gene_pop(i,mutate_pos) = 1 - gene_pop(i, mutate_pos);
   end
end
clear i;



end

opa_selection.m
function [  ] = opa_selection( )
global gene_pop;
global cost_pop;
global npop;
fitness_table=cumsum(cost_pop);
pop_size=npop;
pop_new=zeros(npop,51);



for i=1:npop
   r  =  rand*fitness_table(pop_size);% 随 机 生 成 一 个 随 机 数 ， 在 0 和 总 适 应 度 之 间 ， 因 为
fitness_table(pop_size)为最后一个个体的累积适应度，即为总适应度*/
       first = 1;
           last = pop_size;
           mid = round((last+first)/2);
           idx = -1;
       %下面按照排中法选择个体*/
           while (first <= last) && (idx == -1)
               if r > fitness_table(mid)
                   first = mid;
               elseif r < fitness_table(mid)
                   last = mid;
               else
                   idx = mid;
                   break;
               end
               mid = round((last+first)/2);
               if (last - first) == 1
                   idx = last;
                   break;
```

```
            end
        end

        for j=1:51
            pop_new(i,j)=gene_pop(idx,j);
        end
end
      p = pop_size-1;

for i=1:p
     for j=1:51
         pop_gene(i,j) = pop_new(i,j);%若是精英选择，则只将pop_new前pop_size-1个个体赋给pop，最后一个为前代最优个体保留*/
     end
end

clear i;
clear k
clear p;
clear fitness_table;



end


sorting.m
function [ ] = sorting( )
global gene_pop;
global npop;
global cost_pop;
global optimal_cost;
global optimal_indi;
fitvalue=cost_pop';
pop=gene_pop;

for i=1:npop

   [cost_pop(1,npop+1-i),index]=max(fitvalue);
   gene_pop(npop+1-i,:)=pop(index,:);
     fitvalue(i,1)=-1000000000;

end
 if cost_pop(1,npop)>optimal_cost
```

```
    optimal_cost=cost_pop(1,npop);
    optimal_indi=gene_pop(npop,:);

 end



clear i;
clear fitvalue;
clear pop;



end
```

附录 4:

```
Tuihuo.m
data=xlsread('20141010dataform.csv');
data_now=zeros(1,5);
xi=(1:51);
t0=400;
t_now=t0;
tf=10;
while
(data_now(1)==data_now(2)||data_now(3)==data_now(2)||data_now(3)==data_now(4)||data_now(4)==
data_now(5))
    data_now=ceil(rand(1,5)*51);
    data_now=sort(data_now);
end
cheng_sum=zeros(1,469);
cheng=0;
    my_answer=data_now;
    my_answer_n=size(my_answer,2);

    % 标准样本原始数据读入
    minput=dlmread('20141010dataform.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

        y1(j,:)=mycurvefitting(x_optimal(j,:),y0_optimal(j,:));
    end

    % 成本计算
    Q=12;
    errabs=abs(y0-y1);
    le0_5=(errabs<=0.5);
    le1_0=(errabs<=1);
    le2_0=(errabs<=2);
    le3_0=(errabs<=3);
    le5_0=(errabs<=5);
    g5_0=(errabs>5);
    sij=0.5*(le1_0-le0_5)+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

    cheng=cost;
cheng_now=cheng
cheng_best=cheng_now;
data_best=data_now;



while t_now>tf
    m=size(data_now,2);
    k=ceil(rand(1,1)*m);

    j=ceil(rand(1,1)*2);
    if j==1
        data_now(1,k)=data_now(1,k)+2;
    else
        data_now(1,k)=data_now(1,k)-2;
    end
```

```
for p=1:m
    if(data_now(p)<=0)
        data_now(p)=1;
    end
    if(data_now(p)>=51)
        data_now(p)=51;
    end
end

data_now=unique(data_now);
data_now=sort(data_now);
while (size(data_now)<6)
    data_now(size(data_now)+1)=ceil(rand(1,1)*51);
end



my_answer=data_now;
my_answer_n=size(my_answer,2);



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
    y1(j,:)=mycurvefitting(x_optimal(j,:),y0_optimal(j,:));
end

% 成本计算
Q=12;
errabs=abs(y0-y1);
le0_5=(errabs<=0.5);
le1_0=(errabs<=1);
le2_0=(errabs<=2);
le3_0=(errabs<=3);
le5_0=(errabs<=5);
g5_0=(errabs>5);
sij=0.5*(le1_0-le0_5)+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;

    cheng=cost;
    cheng_now=cheng;
    if(cheng_now<=cheng_best)
        data_best=data_now
        cheng_best=cheng_now
    elseif(rand(1,1)*t_now*(cheng_now-cheng_best)<0.1)
        data_best=data_now
        cheng_best=cheng_now
    end
    t_now=t_now*0.999;
end

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


mycurvefitting.m
function y1 = mycurvefitting( x_premea,y0_premea )

x=[5.0:0.1:10.0];

% 将你的定标计算方法写成指令代码，以下样式仅供参考
y1=interp1(x_premea,y0_premea,x,'spline');


end
```

附录 5:

```
liziqun.m
clear all;

data=csvread('20141010dataform.csv');
ng=7;                %粒子维度
num=10;              %粒子个数
bestchoose=zeros(num,ng);

totalbestchoose=zeros(1,num);     %最优选择
min_q=100000000*ones(1,num);
totalmin_q=100000000;             %最优成本
v=zeros(1,ng);
```

```
choose= randi([1,51],num,ng);
choose(1,:)=[1,6,14,28,32,43,51];        %追随最优粒子

   for i=1:num                %防止粒子重复
      tempchoose=unique(choose(i,:));
      while(length(choose(i,:))>length(tempchoose))

          choose(i,:)=[tempchoose,randi([1,51],1,length(choose(i,:))-length(tempchoose))];
         tempchoose=unique(choose(i,:));
      end
   end
   for i=1:num                          %排序以便操作
      choose(i,:)=sort(choose(i,:));
   end



 fprintf(' G     mincost    totalmincost\n');
   for time=1:50                                %开始循环
   tempcheng=zeros(num,469);
   temp_q=zeros(1,num);
      for number=1:469
         xdata=data(1,:);
         ydata=data(2*number,:);
         fitx=xdata(choose);
         fity=ydata(choose);
         for k=1:num
            infery(k,:)=interp1(fitx(k,:),fity(k,:),xdata,'spline');
            for p=1:51
            tempcheng(k,number)=tempcheng(k,number)+cheng(number,p,infery,data);          %
预计个体适应度

            end

            tempcheng(k,number)=tempcheng(k,number)+12*length(choose(num,:));
         end

      end
      for i=1:num
         temp_q(1,i)=sum(tempcheng(i,:),2);                      %总适应度
      end
      [tempmin_q,tempposition]=min(temp_q);
for i=1:num
      if (tempmin_q<totalmin_q)
```

```
        totalmin_q=tempmin_q;
        totalbestchoose=choose(tempposition,:);
    end

end


    for i=1:num                                    %更新每组原始特征点派生的局部最优特征点和对应q
        if(temp_q(i)<min_q(i))
            min_q(i)=temp_q(i);
            for j=1:ng

                bestchoose(i,j)=choose(i,j);
            end

        end
    end


          for i=1:num

v=v+2*rand().*(bestchoose(i,:)-choose(i,:))+1.8*rand().*(totalbestchoose-choose(i,:));
              for t=1:ng
              if v(1,t)>10
                  v(1,t)=10*rand;
              end
              end


              choose(i,:)=choose(i,:)+v;                          %位置更新公式


              for j=1:ng                       %避免特征点越界
                  if(choose(i,j)<1)
                      choose(i,j)=1;
                  end
                  if(choose(i,j)>51)
                      choose(i,j)=51*(1-0.1*rand());
                  end

                  choose=round(choose);
                  if(choose(i,j)==0)
                      choose(i,j)=1;
                  end
```

```
                end

            end

    choose=round(choose);                    %特征点四舍五入，保证为整数；
        for i=1:num                        %避免重复
    tempchoose=unique(choose(i,:));
    while(length(tempchoose)<length(choose(i,:)))
    choose(i,:)=[tempchoose,randint(1,length(choose(i,:))-length(tempchoose),[1,51])];
    tempchoose=unique(choose(i,:));
    end
        end
    for i=1:num                    %特征顶排序，以便后面运算
    choose(i,:)=sort(choose(i,:));
    end

 fprintf('%2d     %5.1f      %5.1f\n  ',time, min(temp_q)/469,totalmin_q/469);

    end
    Tempcheng=zeros(1,469);

    for i=1:469                                %另计算最优成本
        Xdata=data(1,:);
        Ydata=data(2*i,:);
        Fitx=Xdata(totalbestchoose);
        Fity=Ydata(totalbestchoose);
        Infery=interp1(Fitx(1,:),Fity(1,:),Xdata,'spline');
        for p=1:51
           Tempcheng(1,i)=Tempcheng(1,i)+cheng(i,p,Infery,data);
        end
        Tempcheng(1,i)=Tempcheng(1,i)+12*length(totalbestchoose);
    end
    Q=sum(Tempcheng,2);

    totalbestchoose

    Q=Q/469
```

cheng.m

```
function [ s ] = cheng(x,y,A,B)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

s=0;
```

```
if abs(A(1,y)-B(2*x,y))<=0.5
    s=0;
elseif abs(A(1,y)-B(2*x,y))<=1
    s=0.5;
elseif abs(A(1,y)-B(2*x,y))<=2
    s=1.5;
elseif abs(A(1,y)-B(2*x,y))<=3
    s=6;
elseif abs(A(1,y)-B(2*x,y))<=5
    s=12;

elseif abs(A(1,y)-B(2*x,y))>5
    s=25;
end

end
```