统计推断在数模转换系统中的应用

组号:14　　刘宸钰 5140309220 吕飞霄 5140309231

摘要：本文以某型产品内部的检测模块为研究对象，利用统计推断的方法，探究某输入输出具有非线性关系的传感器的校准方案。由于取点方案数量较大，若采取穷举搜索法时间复杂度过高，无可操作性能，所以采用启发式搜索的拟合方案。遗传算法是一种模仿自然界生物遗传机制的启发式搜索方法，在解决优化问题上具有很高的效率，能够得到一个最佳方案的近似解，本文以遗传算法为优化方法，采用多项式曲线拟合以及三次样条拟合的方式来寻求最优方案。

关键词：统计推断，组合优化，曲线拟合，插值，MATLAB

ABSTRACT: We rely on the statistical inference method to explore whether there exists a plan that certain inputs and outputs can be adjusted in a nonlinear fitting. Because of the vast amounts of data and the complexity of time in exhaustion method, we use heuristic search algorithm. Genetic algorithm is a method stimulating the nature that are efficient and can lead to close results. This thesis takes genetic algorithm as optimization, which adopts polynomial curve fitting and three times spline fitting for results.

Key words： Statistical Inference; Genetic Algorithm, Polynomial fitting, Polynomial fitting

1 引言

1.1 问题的提出

监测模块的组成框图如图 1-1。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）。

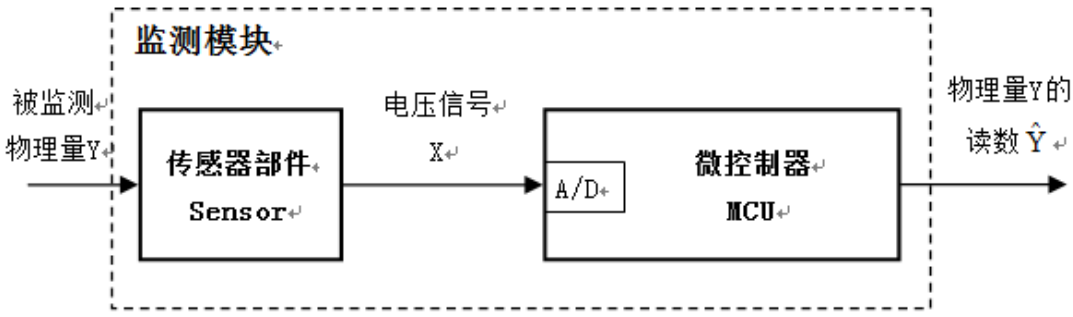


图 1-1 监测模块组成框图

我们可以通过调节传感器可获得不同的输出电压 U，我们一共测量 51 组离散的数据。通过分析输入与输出的一个 X -Y 受控关系，来研究能否得到 X-Y 关系的一个曲线函数表

达式。同时，在保证定标准确的前提下考虑能否减少测量点的个数，可以得到对整个样本空间适用的 X -Y 关系。如果将这个数模转换可调电源系统大规模生产，我们只需要测定几组电压就可以得到整条曲线的 X- Y 关系，达到节约人力物力的目的。

1.2 传感器部件特性

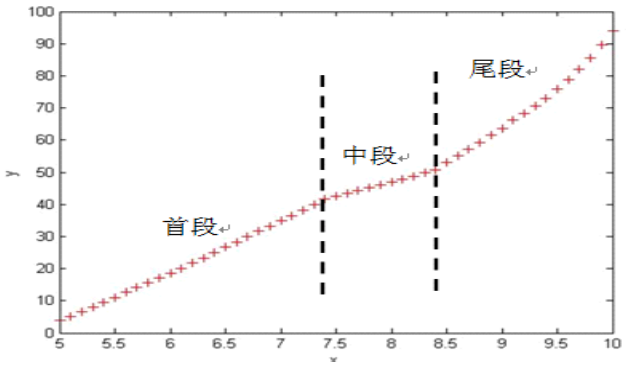


图 1-2 传感特性图示

一个传感部件个体的输入输出特性大致如图 1-2 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]

2 选取最优拟合方式

2.1 概述

在本次试验选取的点中，由于数据庞大，我们无法用一个精确的关系式来描述这个函数，但我们可以通过数学建模理论推算的过程利用一定的算法来拟和出一定的曲线，并通过比较不同方案的优劣程度，选取出最优的方案，从而使问题大大简化。课题最终目标是找到七个特征点，使得根据这七个点拟合的数据按照提供的评价函数计算后，所有样本的平均分不低于 94 分。下图为定标准确度评价函数，其中下标 i 代表样本序号，下标 j 代表观测点序号，a 为数据点得分：

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

公式（2-1）

对某一样本 i 的定标准确度评分：

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

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。

$$C = \frac{1}{M} \sum_{i=1}^{M} S_i \qquad \text{公式（2-3）}$$

2.2 选取特征点

想得到更为精确的拟合结果，需要选择更优的拟合方式。在实验中，我们采用多项式拟合和差值拟合，我们直接使用 MATLAB 提供的拟合函数进行拟合，舍去了一些多余的不必要的点。再将数据大致分为若干段，在每一段中，采取随机数的方法各选取一个数字作为特征点下标，使用此组下标作为所有样本数据拟合所用特征点下标。

2.3 多项式拟合分析

多项式拟合是通过多项式$f(x) = a_0 + \sum_{i=1}^{n} a_i x^i$来拟合曲线，使得残差平方和最小的方式，伪代码为：

（1） 获取一组数据（需判断此组数据未经使用）的 9 个特征点，且按照 3 个不同的分段分别放在三个数组中；

（2） 通过 matlab 多项式拟合函数 p=polyfit（x,y,k）得到三段 k 次拟合曲线的系数 ai；

（3） 通过函数 yy1=polyval（p,x1）获取全部 51 组数据的拟合值，再求评价分数；

（4） 令 m=2、3、4、5、6，重复上述过程。

2.4 插值拟合分析

用 matlab 也可以实现插值拟合，这次实验中我们使用 interp1(x0,y0,xi,‘spline’)来实现拟合。代码直接运用 spline 函数，伪代码如下；

（1） 产生一组数据中 7 个随机特征点；

（2） 通过 matlab 函数 yi =(x,y,x1 ,‘spline’)获取全部 51 组数据的拟合值，再求评价分数（去掉每组的第一个和最后一个数据）；

（3） 处理所有数据，最后得出平均评价分数。[2]

2.5 结果

表2-1 拟合预估初步结果表

	平均运行时间/s	拟合组数	计算所得成本
二次多项式	0.34	51	541.1
三次多项式	0.45	51	145.7
四次多项式	0.60	51	139.7
五次多项式	0.61	51	143.8
六次多项式	0.81	51	184.6

结果分析：

（1）以上结果并不能够完全反映每种拟合方式的优劣，尤其对于三段多项式拟合，由于选取点个数的限制，使其产生的误差较大，但也不能说明这种方法完全不好。如果能增加每段取点个数很可能会得出较好的结果，但考虑到后面内容对于取点个数的限制，所以这个

方法并不适用于本次统计推断中。

（2）通过与多项式拟合结果的比较，三次样条插值拟合的方式相对于多项式拟合在精确度方面有了明显的提升。但是感觉MATLAB自带的三次样条插值的spline()函数在成本计算所得结果仍可通过改进来得到提升，因此在接下来的拟合中，将采用遗传算法进行优化。

3 遗传算法

3.1 原理概述

在选取最优七点组合的过程中，我们使用的是遗传算法（GA，Genetic Algorithm），它是一种“放生算法”，原理是把一个“生物个体”对应到一个可能的解答，需要把解答改写（编码）成一维数组形式（对生物基因染色体的仿生），计算适应度函数（每个个体“适者生存”的程度），然后模拟自然选择的过程，适者生存，不适者淘汰。其实现过程还包括了模拟生物的交配繁殖、基因突变、世代更替。这样一代一代地进化，最后就会收敛到最适应环境的一个“生物个体”上，它就是问题的最优解。

3.2 原理实现

3.2.1 遗传算法的一般步骤

（1）初始化：随机生成 M 个个体作为初始群体 P。

（2）个体评价：计算群体 P 中各个个体 P_i 的适应度。

（3）选择运算：选择操作是建立在群体中个体的适应度评估基础上的。选择的目的是生存概率比较高的个体遗传到下一代，而这种遗传是完全遵从概率的。

（4）交叉运算：把两个通过选择算子遗传下来的父代个体的部分结构加以替换重组而生成新个体，这部分是模拟了生物学的基因重组。

（5）变异运算：对群体中的个体串的某些基因座上的基因值作变动。

（6）终止条件判断:若得到的满足终止条件个体，把其作为最终解输出，终止计算。如不中止，则将变异之后的新群体带回到步骤（2），实现循环。

3.2.2 遗传算法的 Matlab 实现

3.2.2.1 初始群体的生成

初始化群体即为初始选择的七点组合，我们选择了初始化的七点组合一共100 个，这些数均由matlab 程序按照要求随机得出，存入P 中。我们认为使用这些数组可以减少在程序中重新生成100 组数据所耗费的时间，而且这些数组是具有普适性和代表性的。

3.2.2.2 适应性值评估检测

个体适应度，即用上述分段三次Hermite 插值法对个体进行拟合，带入938 组原始数据，所得评价函数的平均值。用cost 表示适应度总成本。

3.2.2.3 选择

选择的目的是为了从当前群体个选出优良的个体，使它们有机会作为父代为下一代繁殖子孙。遗传算法通过选择过程体现这一思想，进行选择的原则是适应性强的个体为下一代贡献一个或多个后代的概率大。选择实现了达尔文的适者生存原则。

3.2.2.4 交叉

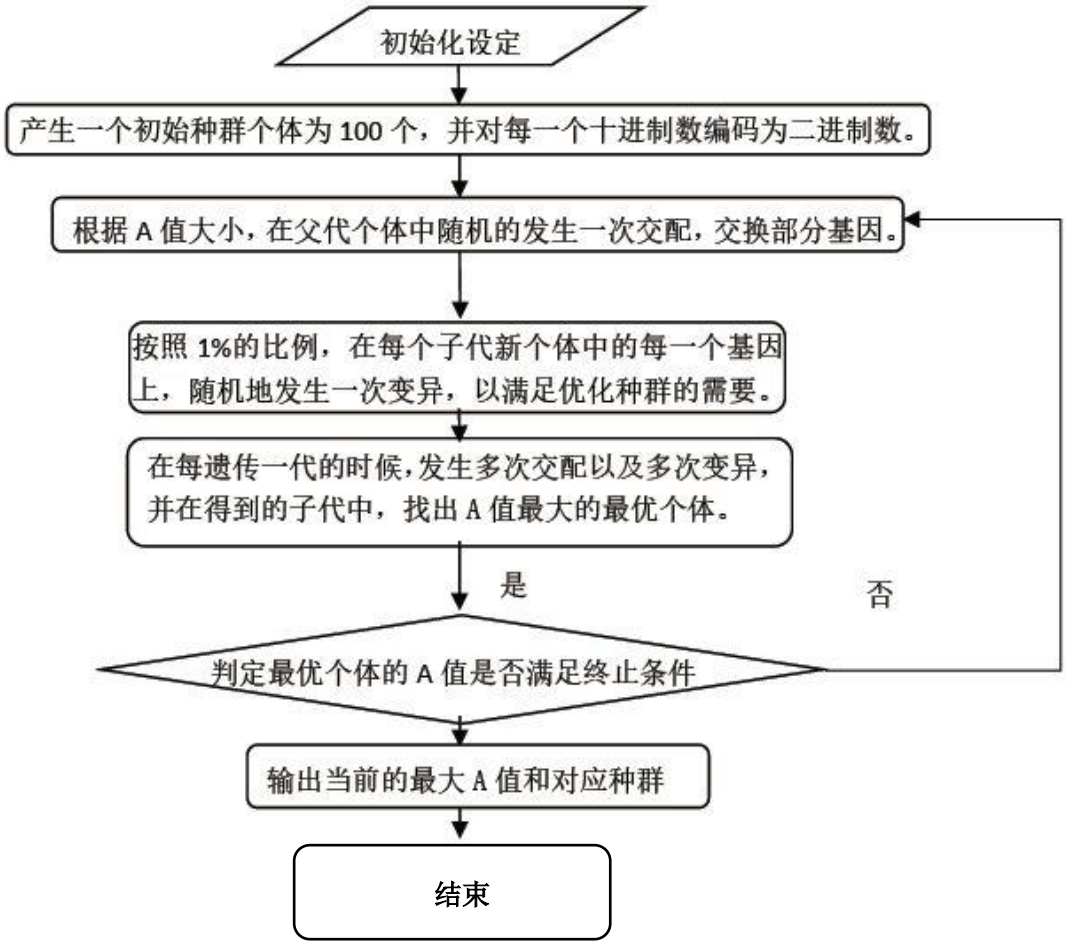
交叉操作是遗传算法中最主要的遗传操作。通过交叉操作可以得到新一代个体，新个体组合了其父辈个体的特性。交叉体现了信息交换的思想。在交叉过程中可能会出现基因重复现象，将其认为基因缺陷，直接淘汰，重新交换，直到每一个染色体内部没有重复的

基因。

3.2.2.5 变异

由于初始种群是随机生成的，故可省略随机挑选10%子体的步骤，直接选取子体进行变异。采用类似于上一步的等概率方法，让7 个数各有50%的概率变异，变异方法即使其+1，为了防止残疾的下一代，如果变异后的数超出初始设定的区间，则回到区间开始处第一个数。完成变异后的种群可视作下一个循环的初始种群。

3.3 原理图



4 结论

表 4-1 遗传算法和样条插值所得结果

1	95.1316	2	9	20	26	33	43	50
2	95.1316	2	9	20	26	33	43	50
3	95.2423	2	9	20	26	34	43	50
4	95.2824	2	9	20	27	33	43	50
5	95.2824	2	9	20	27	33	43	50

由表4-1 可以看出，本小组采取的分七段取点三次样条插值和遗传算法充分利用了数据的性质，有效的解决了数据量大，耗时长的问题。同时只要遗传代数足够多，选取的亲本够多，则成本还可以更低。但是我们还应看到，在特征点之外的数据无法对结果产生影响，从而导致数据的遗漏。

5 参考文献

[1] 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 [EB/OL].ftp://202.120.39.248。

[2] 许小勇,太勇. 三次样条插值函数的构造与 Matlab 实现 云南民族大学数学与计算机科学学院，云南昆明

附录：

对选取的数据进行曲线拟合以求其始末点

```
function y=steptwo(data)
n=length(data);
m1=zeros(1,51);
m2=zeros(1,51);
for k=2:2:n;
e1=polyfit(data(k-1,1:24),data(k,1:24),1);
e2=polyfit(data(k-1,25:33),data(k,25:33),1);
e3=polyfit(data(k-1,36:51),data(k,36:51),1);
a1=[(e1(1)-e2(1)),(e1(2)-e2(2))];
p1=roots(a1);
p1=(round(10*p1)/10-5)*10;
a2=[(e2(1)-e3(1)),(e2(2)-e3(2))];
p2=roots(a2);
p2=(round(10*p2)/10-5)*10;
m1(1,k/2)=p1;
m2(1,k/2)=p2;
end
m6=1:1:51;
[y1,~]=hist(m1,m6);
bar(m6,y1);title('各点作为中段末尾点的出现次数');xlabel('点数');ylabel('次
数');
m2=m2(:);
[y2,~]=hist(m2,m6);
bar(m6,y2);title('各点作为中段起始点的出现次数');xlabel('点数');ylabel('次
数');
end
```

成本计算：

```
function stepthree(data)
my_answer=[4,14,26,33,43,50];
my_answer_n=size(my_answer,2);
minput=data;
[M,N]=size(minput);
nsample=M/2; npoint=N;
y0=zeros(nsample,npoint);
y1=zeros(nsample,npoint);
d1=0;
d2=0;
d3=0;
for i=1:nsample
y0(i,:)=minput(2*i,:);
```

```
end
for i=1:nsample
k1=2*i-1;
k2=2*i;
m1=[minput(k1,4),minput(k1,14),minput(k1,25)];
n1=[minput(k2,4),minput(k2,14),minput(k2,25)];
m2=[minput(k1,25),minput(k1,33)];
n2=[minput(k2,25),minput(k2,33)];
m3=[minput(k1,33),minput(k1,43),minput(k1,50)];
n3=[minput(k2,33),minput(k2,43),minput(k2,50)];
c1=polyfit(m1,n1,2);
c2=polyfit(m2,n2,1);
c3=polyfit(m3,n3,2);
d1=d1+c1;
d2=d2+c2;
d3=d3+c3;
for j=1:24
y1(i,j)=polyval(c1,j*0.1+4.9);
end
for j=25:33
y1(i,j)=polyval(c2,j*0.1+4.9);
end
for j=34:npoint
y1(i,j)=polyval(c3,j*0.1+4.9);
end
end
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_8-0_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;
d1=d1/400;
d2=d2/400;
d3=d3/400;
```

```
fprintf('\ny=%fx^2+ %fx+ %f\n', d1)
fprintf('\ny=%fx+ %f\n', d2)
fprintf('\ny=%fx^2 +%fx+ %f\n', d3)
fprintf('\n 您的样本成本为%5.2f\n',cost);
end
```

遗传算法代码

主函数testmain

```
function [ P,Costmin ] = testmain()
tic
N=100; T=30; p=0.8; q=0.01; n=7;
data=xlsread('20150915dataform.xlsx');
t=1;
allCost=zeros(1,N);
population=Initialize(N,n);
for i=1:1:N
%计算初始种群每一个的成本并储存
allCost(i)=Costing(population(i,:),data,n);
end
Costmin=allCost(N);
while t<=T
[population,allCost]=Mutate(population,allCost,N,q,data,n); %变异
[allCost,Index]=sort(allCost,'descend');
population=population(Index,:);
[population,allCost]=Crossover(population,allCost,N,p,data,n); %交叉
[allCost,Index]=sort(allCost,'descend');
population=population(Index,:);
[population,allCost]=Select(population,allCost,data,n); %选择
if Costmin>allCost(N)
Costmin=allCost(N);
P=population(N,:);
end
disp('子代数'); %输出适应度最高的子代及其成本
disp(t);
disp(Costmin);
disp(population(N,:));
t=t+1;
end
toc
end
```

初始化函数Initialize

```
function [ population ] = Initialize(N,n)
switch n
```

```
case 7
population=zeros(N,n);
for i=1:1:N
population(i,1)=1+ceil(rand*7);
population(i,2)=8+ceil(rand*7);
population(i,3)=15+ceil(rand*7);
population(i,4)=22+ceil(rand*7);
population(i,5)=29+ceil(rand*7);
population(i,6)=36+ceil(rand*7);
population(i,7)=43+ceil(rand*7);
end
case 6
population=zeros(N,n);
for i=1:1:N
population(i,1)=ceil(rand*6);
population(i,2)=6+ceil(rand*6);
population(i,3)=12+ceil(rand*13);
population(i,4)=25+ceil(rand*13);
population(i,5)=38+ceil(rand*6);
population(i,6)=44+ceil(rand*7);
end
case 5
population=zeros(N,n);
for i=1:1:N
population(i,1)=ceil(rand*8);
population(i,2)=8+ceil(rand*11);
population(i,3)=19+ceil(rand*14);
population(i,4)=33+ceil(rand*10);
population(i,5)=43+ceil(rand*8);
end
case 4
population=zeros(N,n);
for i=1:1:N
population(i,1)=ceil(rand*10);
population(i,2)=10+ceil(rand*13);
population(i,3)=23+ceil(rand*15);
population(i,4)=38+ceil(rand*13);
end
end
end
```

成本计算函数Costing

```
function [ singleCost ] = Costing( P,data,n )
singleCost=0;
```

```
sort(P);
X=data (1,:);
for k=2:2:938
Cost=0;
Y=data (k,:);
Px=zeros(n,1);
for j=1:1:n
Px(j,1)=X(P(j));
Py(j,1)=Y(P(j));
end
Ytheory=interp1(Px,Py,X,'spline');
for i=1:1:51
s=abs(Y(i)-Ytheory(i));
if s<=0.5
c=0;
elseif s<=1
c=0.5;
elseif s<=2
c=1.5;
elseif s<=3
c=6;
elseif s<=5
c=12;
else c=25;
end
Cost=Cost+c;
end
singleCost=singleCost+Cost;
end
singleCost= singleCost/469+12*n;
end
```

变异函数Mutate

```
function [ population,allCost ] =Mutate( population,allCost,N,q,data,n)
for i=1:1:N
if rand<q
while 1
if allCost(i)<(n*12+10)
len=randi([-2,2]);
else
len=randi([-5,5]); %变异步长(1-3)
end
pos=randi(n); %变异位置（1-7）
child=population(i,:);
```

```
child(1,pos)=mod(population(i,pos)+len,49)+2;
if length(unique(child))==n %检查变异后个体是否合法
Cost=Costing(child,data,n);
if allCost(i)>Cost; %变异之后要更新分数列
population(i,:)=child;
allCost(i)=Cost;
end
break;
end
end
end
end
end
```

交叉函数Crossover

```
function [population,allCost] = Crossover( population,allCost,N,p,data,n)
%适应度高的一半内部进行交叉
for i=N:-2:(N/2+2)
sort(population(i,:)); %把个体中的元素重新排列
sort(population(i-1,:));
if rand<p
point=randi(n); %单点交叉
child1=[population(i,1:point),population(i-1,point+1:n)];
child2=[population(i-1,1:point),population(i,point+1:n)];
child1=Check(child1,n); %检查交叉结果是否合法
child2=Check(child2,n);
if (length(unique(child1))==n )&&(length(unique(child2))==n )
Cost1=Costing(child1,data,n); %计算交叉结果的成本
Cost2=Costing(child2,data,n);
if allCost(i)>Cost1 %交叉后的个体成本更低，就取交叉后的个体
population(i,:)=child1;
allCost(i)=Cost1;
end
if allCost(i-1)<Cost2
population(i-1,:)=child2;
allCost(i-1)=Cost2;
end
sort(population(i,:));
sort(population(i-1,:));
end
end
end
%适应度低的与适应度高的进行交叉，下标从N/2到1
for i=N/2:-1:1
```

```
sort(population(i,:)); %把个体中的元素重新排列
if rand<p
point=ceil(rand*4);
child1=[population(i,1:point),population(i+N/2,point+1:n)];%与第i+N/2个元素交叉
child1=Check(child1,n); %检查是否合法
Cost1=Costing(child1,data,n); %计算新得到结果的分数
if allCost(i)>Cost1 %交叉后的个体成本更低，就取交叉后的个体
population(i,:)=child1;
allCost(i)=Cost1;
end
sort(population(i,:));
end
end
end
```

选择函数Select

```
function [ population,allCost] = Select( population,allCost,data,n)%根据适应度高低进行选择
population([1,2,3,4],:)=Initialize(4,n);
allCost(1)= Costing(population(1,:),data,n);
allCost(2)= Costing(population(2,:),data,n);
allCost(3)= Costing(population(3,:),data,n);
allCost(4)= Costing(population(4,:),data,n);
end
```

检查函数Check

```
function [x] = Check(x,n) %检查子代是否有重复
k=1;
sort(x);
new=unique(x);
if length(new)~=n
x=Initialize(1,n);
end
end
```

%%%%%%% 答案检验程序%%%%%%%%

```
my_answer=[ 2,9,20,26,33,43,50 ];%把你的选点组合填写在此
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);
```