统计推断在数模转换系统中的应用

组号 53 姓名 罗文杰 学号 5140309332 姓名 姜伟鑫 学号 5140309330

摘要：本课题探究对于输入输出特性呈明显的非线性关系的传感器部件进行传感特性校准的问题，采用遗传算法为主要工具。考虑到拟合效果，探究中综合采用了样条插值和Hermit插值拟合，通过遗传算法的选择、交叉、变异，一代代选出相应的最优个体。最终得到课题所需的最优解，即最佳校准方案。

关键词：遗传算法 样条插值 Heimite插值 最优解

Statistic Application On The DA Converting System

Abstract : The research project focus on the sensor component which has non-linear input and output characters. We will correct the characters. The main tool is the genetic algorithm. We use the Spline Interpolation and Heimite Interpolation, and through the selection, intersection and heteromorphosis, we finally obtain the optimal solution.

Key words: genetic algorithm, Spline Interpolation, Heimite Interpolation, Optimal solution

引言

某工业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。寻找校准方案时，取得数据点越多，曲线拟合得就越精确，但成本越高。取得数据点越少，成本越低，但是曲线拟合的精确性也越低。所以为了既兼顾成本和精确度，就必须限定取的点的数量并且在曲线最具有代表性的地方取这些有限的点。 本课题中，三次样条插值和 Hermit 插值拟合是一种比较好的拟合选择；而数据点的选择，可以采用以遗传算法等优化算法。

1.1 样品特性分析

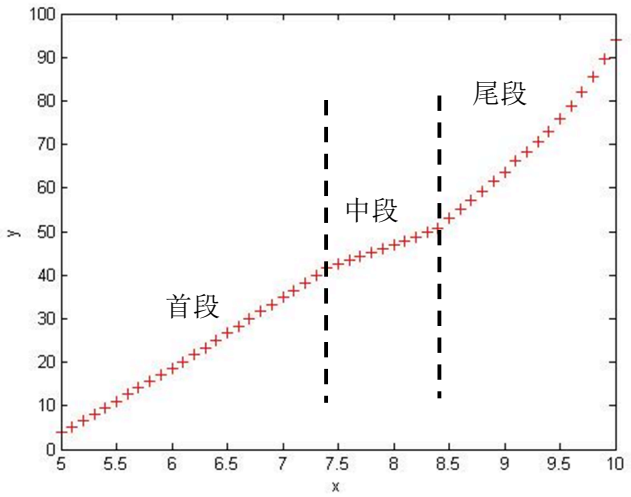


图 1 传感特性图示

一个传感部件个体的输入输出特性大致如图 1 所示，有以下主要特征：

（1）Y 取值随 X 取值的增大而单调递增；

（2）X 取值在[5.0,10.0]区间内，Y 取值在[0,100]区间内；

（3）不同个体的特性曲线形态相似但两两相异；

（4）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；

（5）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；

（6）不同个体的中段起点位置、终点位置有随机性差异。

1.2 模型评判标准

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

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

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=12。

- 某一样本个体的定标成本

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

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。

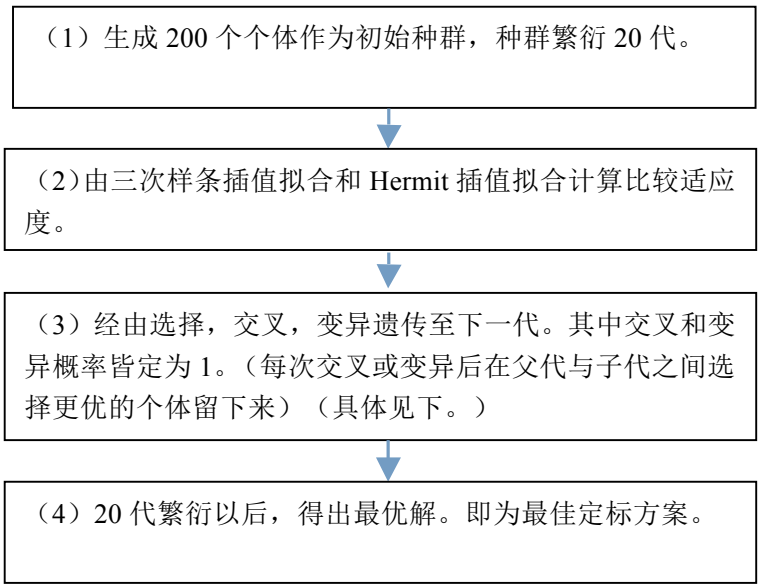
- 校准方案总成本

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。

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

总成本较低的校准方案，认定为较优方案

2 基于遗传算法和多项式拟合&&插值拟合的求解流程图



3. 特征点的选择——遗传算法

实验的目标是对给定的成本函数（目标函数），找到其最优解。对于取点方案最优化解的搜索，本实验采用遗传算法的最优化解（特征点）的搜索方法。

3.1 遗传算法原理

遗传算法（Genetic Algorithm）是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法。

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），但也可以用其他表示方法。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），通过交叉和变异产生新的生命种群，该种群在算法的下一次迭代中成为当前种群。

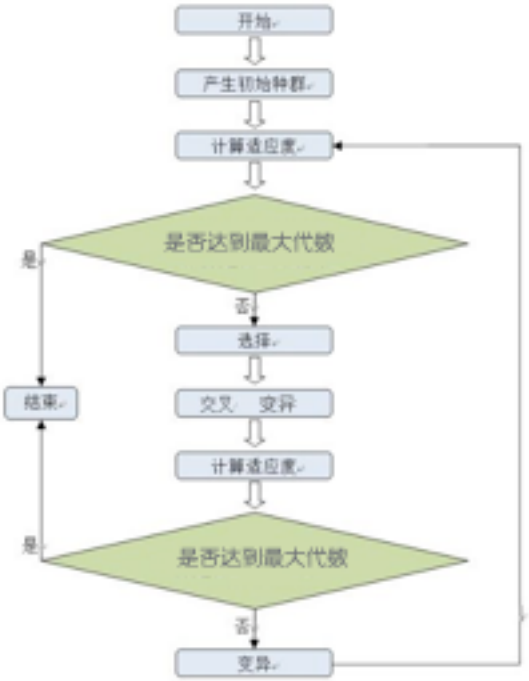


图 2 遗传算法图解

3.2 遗传算法的运算实现过程

遗传算法的基本运算过程如图 2，相关阐释以及具体到本课题中的实现方法简略如下：

（1）初始化：设置进化代数计数器 t=0，设置最大进化代数 T，随机生成 M 个个体作为初始群体 P(0)。

实现：任选一样本个体，由 51 个数据点生成的 51 位二进制编码中。取点记为 1，否则记为 0。种群规模初步设定为 200.（原本种群规模为 30，繁衍 50 代，后经老师指导改为了种群规模 200，繁衍 20 代，我们发现得到了更优秀的解。）

（2）个体评价：计算群体 P(t)中各个个体的适应度。

实现：本课题中适应度函数为成本平方的倒数：

$$C = \frac{1}{M}\sum_{i=1}^{M} S_i \quad \text{（所有样本个体的定标成本的统计平均）} \qquad (3\text{-}1)$$

其中

$$S_i = \sum_{j=1}^{51} s_{i,j} + q \cdot n_i \quad \text{（个体定标成本）} \qquad (3\text{-}2)$$

（3）选择运算:将选择算子作用于群体。选择的目的是把优化的个体直接遗传到下一代或通过配对交叉产生新的个体再遗传到下一代。选择操作是建立在群体中个体的适应度评估基础上的。

实现： 将各个个体的适应度除以种群中最大的适应度（最优解），作为个体遗传至下一代的概率。同时，为尽量减少概率不确定性的影响，强制将适应度最大的两个体直接遗传至下一代。即在选择子代时，将一次选择强制取成最大的两个个体，剩下的再按概率获取（按概率获取的实现是对每个个体对应一个累积概率，通过产生随机数的方式来选取父代）。（可以利用一个 flag 加以实现）

（4）交叉运算：将交叉算子作用于群体。遗传算法中起核心作用的就是交叉算子。

实现：以 p=1 的概率进行交叉。随机选择两个个体，随机选择两个个体中的同一位置，将该位置后的序列进行交换。

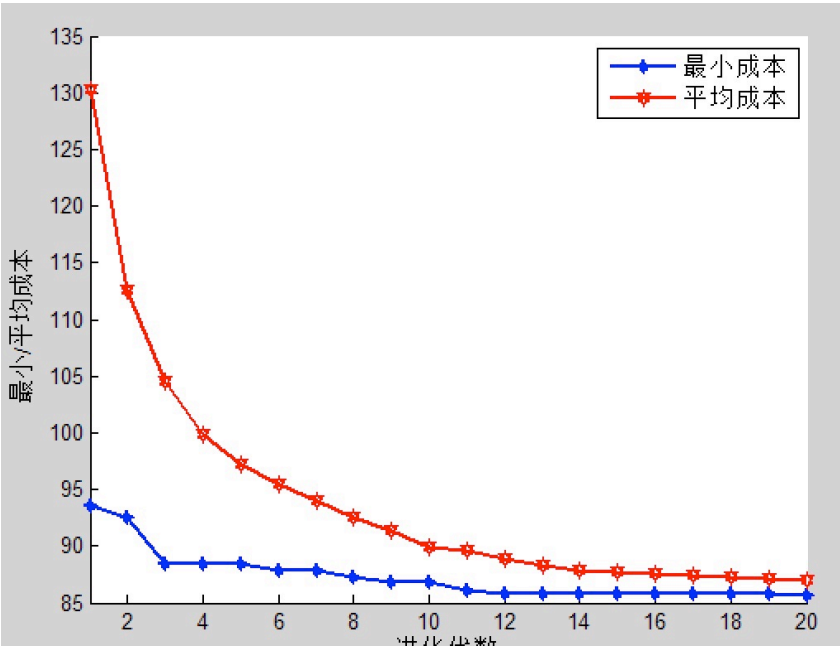
（5）变异运算：将变异算子作用于群体。即是对群体中的个体串的某些基因座上的基因值作变动。

实现：以 p=1 的概率进行变异。生成随机数 r∈{0,51}。变异方法有替换，缺失两种。将 r 对应编码位置上的 0 换成 1（或 1 换成 0）。缺失则依次补上，由 0 补上第 51 位。

群体 P(t)经过选择、交叉、变异运算之后得到下一代群体 P(t+1)。

注意：对于交叉和变异，为防止交叉（变异）将本已最优的个体破坏，会将交叉（变异）前后的个体进行比较， 选择适应度高的作为交叉（变异）结果。因此我们可以选取交叉变异的概率为 1 来提高种群进化的效率而不用担心最终种群的曲线波动巨大不稳定。

（6）终止条件判断:若 t=T,则以进化过程中所得到的具有最大适应度个体作为最优解输出，终止计算。此时我们获得平均成本与最低成本的图像，通过图像的稳定程度可以判断图像是否成熟。（如果最终平均成本没趋于稳定且和最小成本差异很大的话，则需要扩大种群规模或者增加代数再次进行模拟。）



如上图在 16 代以后，明显趋于稳定，说明得到的解是相对的最优解。

（算法设计之初，我们组计划初始种群个体定为 30，繁衍 50 代。得出最低成本超过 90。在

老师的建议下，我们将个体数扩大至 200 代，同时在保证繁衍次数足够的前提下，改为繁衍 20 代。结果证明，16 代以后即趋于稳定，可看出繁衍 20 代是合理的。）

4. 函数拟合的选择

拟合就是寻找一条光滑的曲线，最不失真得表现测量数据。本课题中，最初考虑采用三次多项式拟合和样条插值拟合，但后发现效果不甚佳，因此最终决定采用三次样条插值拟合和 Hermit 插值拟合。

4.1 三次样条插值拟合

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。由于样条插值可以使用低阶多项式样条实现较小的插值误差，这样可以避免了使用高阶多项式所出现的 Runge 现象。

在 MATLAB 中，我们可以使用 Spline 函数进行三次样条拟合。
假设在[a,b]上测量有 n-1 个点，在[a,b]上划分。△:a=x0<x1<….<xn-1<xn=b，则三次样条插值法所得的函数 S(x)具有如下特征[1]：

（1）在每个小区间[xj,xj+1]上是三次多项式。
（2）在每个节点 S(xj)=yj。

（3）S(x)在[a，b]二阶导数连续。
因为 S(x)在[a，b]上二阶导数连续，所以在每个节点，由连续性得 S(xj-0)=S(xj+0); S’(xj-0)=S’(xj+0); S’’(xj-0)=S’’(xj+0)。共 3n-3 个条件；除此之外，由插值可得 n+1 个条件，仍需两个，这两个条件有边值得出：
（1）给定断点 x0,xn 处的一阶导数值
（2）给定断点 x0,xn 处的二阶导数值（特别的 S’’(x0+0)=0, S’’(xn+0)=0 称为自然边值条件）
（3）周期性便捷条件：即以 b-a 为周期，S’’(x0+0)= S’’(xn+0)，S’(x0+0) =S’(xn+0)。

4.2 Hermite 插值

为了保证插值函数能够更好地逼近原函数,不仅要求两者在节点上有相同函数值,而且要求在节点上有相同的导数值。这类插值称为 Hermite 插值。Hetmite 插值法具有精度高,拟合出的曲线光滑程度好;且计算简单易于实现,同时又具有一定的局限性,如果想要部分 修改数值是,并不会影响全局。

Matlab 关于插值法有现成的插值函数可以调用:y=interp1(x,y, i x ,method)
其中,x 和 y 均为数组,在本文附带程序中便是我们选取的 7 个点对应的占空比 D 和输出电压 U;xi 可以是一个数值也可以是一组数据;method 是所采用的插值方法,包括: ’nearest’:最近邻点插值;’linear’:线性插值;’spline’:三次样条函数插值; ’pchip’:分段三次 Hermite 插值。

其中,适合本次曲线拟合的是分段三次 Hermite 插值法(“pchip”)。首先线性差值显然不符合所需要求，故不做讨论。在几次拟合的比较中，较之 spline 插值，我们发现 pchip 插值的运行效率要明显更高，而且得出的成本普遍更低。

下面是几次实验数据。

Hermit

运行次数	取点	所取点	成本
1	5	4 16 26 36 49	85.57
2	5	5 16 25 35 48	85.54
3	6	5 14 23 30 38 49	86.99

三次样条插值

运行次数	取点	所取点	成本
1	7	2 9 22 28 35 44 50	96.9430
2	7	2 9 19 26 33 43 50	95.1147

所以我们确定相对来说更为优化的拟合方法应该是 Hermit 插值法。

5.课程结论

最优解的得到需要最佳的拟合方式和优良的进化效率。根据控制变量法，在同等的取点方式下，对拟合的方式进行选择，发现 Hermit 插值的效率更高，所得最优解的成本可低至 90 以下。另外，为提高进化效率，将遗传算法中交叉、变异概率设为 1，效果显著。终得出的结论优的取点个数应该为 5 个点。经多次测试取点，最终得出的最优解成本为 85.54.取点方案为【5， 16，25，35，48】。

6．鸣谢

感谢袁焱老师和助教老师在课程过程中的帮助，面谈中，助教老师给予了我们宝贵的建议，让我们在设置初始种群和进化代数上作了改进，并指出了我们论文报告中不甚合理之处，感谢！同时这次课题研究的顺利完成也离不开小组各成员的认真探讨，分工合作。

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

[2]百度百科 遗传算法词条

[3]唐家德,基于 MATLAB 的非线性曲线拟合[J] 计算机与现代化，2008 年第 6 期

代码附录

main.m

```
clear all;close all;
BitLength=51;
popsize=200;
generationmax=20;
global data;
global tmpp;
data=csvread('20150915dataform.csv');

population=zeros(popsize,51);s=0;
for i=1:popsize
  population(i,:)=RandomProduce();            %产生第一代种群
end

[Fitvalue,accumPro]=fitness(population);       %Fit存放适应度，acc存放概率密度函数
Aftercross=zeros(popsize,BitLength);
Aftermutation=zeros(popsize,BitLength);
PointsBest=zeros(generationmax,BitLength);    %每一代的最优个体
CostBest=zeros(generationmax,1);              %最优个体对应成本
AveCost=zeros(generationmax,1);               %每一代的总成本平均值

generation=1;
while generation<generationmax+1
   disp(generation);
   [maxtmp,postmp]=max(Fitvalue);             %maxtmp为最大适应度，postmp为最大适应度索引
   Fitvalue(postmp)=0;                        %取出最大（找第二大）
   [maxtmp2,postmp2]=max(Fitvalue);           %找到第二大的
   select(1)=postmp;
   select(2)=postmp2;
   flag=false;

   %用以保证待会第一次选择的时候不是随机选择，而是选择的两个最优秀子代
   Fitvalue(postmp)=maxtmp;                   %把最大的放回去
   for j=1:2:popsize     %交配变异，产生子代

      if flag
         select=selection(population,accumPro); %依据acc中的概率选择两个体
      end

      [selectPoints]=CrossOver(population,select);

      cost1=1./((costSum(selectPoints(1,:)).^2));
      cost2=1./((costSum(selectPoints(2,:)).^2));


     if Fitvalue(select(1))>=cost1
         Aftercross(j,:)=population(select(1),:);
         cost1=Fitvalue(select(1));
     else Aftercross(j,:)=selectPoints(1,:);
     end

     if Fitvalue(select(2))>=cost2
         Aftercross(j+1,:)=population(select(2),:);
```

代码附录

```
            cost2=Fitvalue(select(2));
            else Aftercross(j+1,:)=selectPoints(2,:);
        end

        %变异操作
        [AferA]=mutation(Aftercross(j,:));
        [AfterB]=mutation(Aftercross(j+1,:));

        tmp=1./(costSum(AferA).^2);
       if cost1>=tmp
            Aftermutation(j,:)=Aftercross(j,:);
       else
            cost1=tmp;
            Aftermutation(j,:)=AferA;
       end


        tmp=1./(costSum(AfterB).^2);

       if cost2>=tmp
            Aftermutation(j+1,:)=Aftercross(j+1,:);
       else
            cost1=tmp;
            Aftermutation(j+1,:)=AferA;
       end

        Fitvalue(j)=cost1;
        Fitvalue(j+1)=cost2;
        flag=true;
    end

        population=Aftermutation;          %新种群
        %计算新的累计概率
        SumAccum=sum(Fitvalue);
        ProEvery=Fitvalue/SumAccum;        %选择概率
        accumPro(1)=ProEvery(1);
        for i=2:popsize
            accumPro(i)=accumPro(i-1)+ProEvery(i);
        end

        %记录当前代最好的适应度和平均适应度
        [value,pos]=max(Fitvalue);
        CostBest(generation)=1./((value).^0.5);          %把适应度对应的成本转化回来
        disp(CostBest(generation));                      %输出本代最佳个体的成本
        AveCost(generation)=mean(1./((Fitvalue).^0.5));

        xx=population(pos,:);    %当代最佳染色体
        PointsBest(generation,:)=xx;
        generation=generation+1;
end

[Bestvalue,minPos]=min(CostBest);          %所有代结束以后，从每代最佳选最佳
Bestmethod=PointsBest(minPos,:);

Bestpos=find(Bestmethod==1);
```

代码附录

```
disp('最优选择方案为')
disp(Bestpos);

%作图看平均适应度和最大适应度，判断是否成熟
figure(1);
hand1=plot(1:generationmax,CostBest);
set(hand1,'linestyle','-','linewidth',1.8,'marker','*','markersize',6)
hold on;
hand2 = plot(1:generationmax,AveCost); set(hand2,'color','r','linestyle','-','linewidth',
1.8,'marker','h','markersize',6)
xlabel('进化代数');ylabel('最小/平均成本');xlim([1 generationmax]); legend('最小成本','平均成本');
box off; hold off;
```

selection.m

//用以选择父代个体

```
function Individual= RandomProduce()    %个体随机生成函数,并在此过程中限定选中观测点数不小
于 5
s=0;
Individual = zeros(1,51);
while s<4
for i=1:51
   if rand()>0.8                %因为显然取点规模过大不符合要求，所以将取点概率减小为0.2
      Individual(1,i)=1;
   end
end
s=sum(Individual);
end
end
```

RandomProduce.m

```
function Individual= RandomProduce()    %个体随机生成函数,并在此过程中限定选中观测点数不小
于 5
s=0;
Individual = zeros(1,51);
while s<4
for i=1:51
   if rand()>0.8                %因为显然取点规模过大不符合要求，所以将取点概率减小为0.2
      Individual(1,i)=1;
   end
end
s=sum(Individual);
end
end
```

mutation.m

```
function [AfterMutation] = mutation(snew) %子程序:突变变异操作,z 用于判断是否变异成功
BitLength=size(snew,2);
AfterMutation=zeros(1,51);

while sum(AfterMutation)<4
    chb=round(rand*(BitLength-1))+1;       %在[1,BitLength]范围内随机产生一个变异位
    snew(chb)=1-snew(chb);
    AfterMutation=snew;                    %记录产生了变异的点位
end
```

代码附录

crossover.m

```
function [selectPoints]=CrossOver(population,select) %新种群交叉互换操作

BitLength1=size(population,2);

selectPoints=zeros(2,51);
while sum(selectPoints(1,:))<3 ll sum(selectPoints(2,:))<3      %(防止突变以后出现两个个体选点都很少的情况)
   crosspos=round(rand*(BitLength1-2))+1;                  %在[1,Bitlength-1]范围随机产生一个交叉位
   selectPoints(1,:)=[population(select(2),1:crosspos) population(select(1),crosspos+1:51)];
   selectPoints(2,:)=[population(select(1),1:crosspos) population(select(2),crosspos+1:51)];

end
end
```

fitness.m

```
function [Fitvalue,accumPro]= fitness(population)          %计算适应度函数
popsizenum=size(population,1);
Fitvalue=zeros(1,popsizenum);        %初始化适应度矩阵
accumPro=zeros(1,popsizenum);
for i=1:popsizenum
x=population(i,:);                %在30种不同的选点方案下，计算400个个体的成本
Fitvalue(i)=costSum(x);             %Fitvalue每个元素代表各自的对应的总成本。
end
Fitvalue=1./((Fitvalue).^2);       %将成本与适应度对应起来
fsum=sum(Fitvalue);                %得到适应度的总和
Pperpopulation=Fitvalue/fsum;
accumPro(1)=Pperpopulation(1);     %计算累积概率
for i=2:popsizenum
accumPro(i)=accumPro(i-1)+Pperpopulation(i); %概率分布函数（离散）
end
Fitvalue=Fitvalue'; accumPro=accumPro';
end
```

costsum.m

```
function [ average ] = costSum(points )
%计算一个个体的成本,points 为选点方案
global data;
global tmpp;
current=1;
for i=1:2:799
x=data(i,:);
y=data(i+1,:);
c(current)=costOne(x,y,points);
current=current+1;

end
average=mean(c);
tmpp=average;
end
```

代码附录

costsum.m

```
function [cost] = costOne( x,y,points ) %计算一个样本的成本.x,y 为样本数据,points 为选点方案

pos=find(points==1);
x2=x(pos);
y2=y(pos);

q=interp1(x2,y2,x,'spline');

diffvalue=abs(q-y);
sum=0;
for i=1:length(diffvalue)
if diffvalue(i)>0.4 && diffvalue(i)<=0.6
    sum=sum+0.1;
end
if diffvalue(i)>0.6 && diffvalue(i)<=0.8
    sum=sum+0.7;
end
if diffvalue(i)>0.8 && diffvalue(i)<=1
    sum=sum+0.9;
end
if diffvalue(i)>1 && diffvalue(i)<=2
    sum=sum+1.5;
end
if diffvalue(i)>2 && diffvalue(i)<=3
    sum=sum+6;
end
if diffvalue(i)>3 && diffvalue(i)<=5
    sum=sum+12;
end
if diffvalue(i)>5
    sum=sum+25;
end
end
cost=sum+12*length(pos);

end
```