统计推断在数模转换系统中的应用

组号 10 姓名 彭珅晖 学号 5140309304，姓名 张凌峰 5140309298

摘要：本文以某型产品内部的检测模块为研究对象，在确保测量精度的前提下，运用统计推 断方法，以 MATLAB 为工具，通过选取原始数据库中 469 组密集选点作为实验数据进行分析， 来寻找校准工序的优化方案，同时尽量降低测定成本。

关键词：统计推断 多项式拟合 样条函数插值法 Hermite 插值法 遗传算法

The Application of Statistical Inference in the Digital-analog Conversion System Group

Group No.010 Peng Shenhui 5140309304 Zhang Lingfeng 5140309298

Abstract: The report is based on a certain type of products within the detection module as the research object.On the premise of assuring the accuracy of measurement, we use statistical inference method, based on the MATLAB tool.We choose 469 pairs of intensive study in the original database as experimental data analysis, to find out the optimized plan for the calibration process and at the same time to reduce the measurement cost as much as possible.

Key words: Statistical Inference , Polynomial fitting , Spline interpolation , Cubic pchip interpolation , Genetic Algorithm

1 引言

本课题来源于工业中校准定标问题。在工业生产中，我们往往需要通过传感器测量相关的参数，但是传感器使用种种原理将带测量转换为直接可观 测的量的过程中往往要经过一系列的非电信号到电信号 转换与机械传动，因此被检测物理量与直接测出量之间绝 大数时候并不呈线性，当非线性带来的误差不能被接受的 时候就需要重新定标。然而对于器件一致性差，样本容量大的情况，传统的 密集选点法并不能高效地完成校准定标的工作，并且在测量中将付出极大的成本，这就要求 我们寻求更为优化的方法完成校准定标的工作。

2 课题内容目标与解决思路

2.1 课题内容

本课题研究的内容为传感器部件检测对象 Y 与传感部件的输出电压 X 之间的函数关系，由于二者之间并不存在确定的函数关系，因而可以通过研究大量的数据找到适当的你和函数。由于样品存在个体差异，因而对单个样本的研究不能代表整体，在对整体研究时，如果用所有测试点进行函数拟合，运算量大成本高。因而，该课题的研究内容为找出所有样本共同的数据特征点，使得用这些点拟合的函数与样本数据的误差尽可能小，以至于工程上可以接受[1]。

2.2 课程目标

单点定标误差成本：

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

单点测定成本：q=12 ;

某一样本个体的定标成本：

$$S_i=\sum_{j=1}^{51}s_{i,j}+12N_i \quad (2)$$

校准方案总体成本：

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

总体成本较低的方案，工程上可认为是较优方案。

2. 3 解决思路

本课程的总体思路如图（2-3-1）所示。本组的思路基于课程要求，并且进行

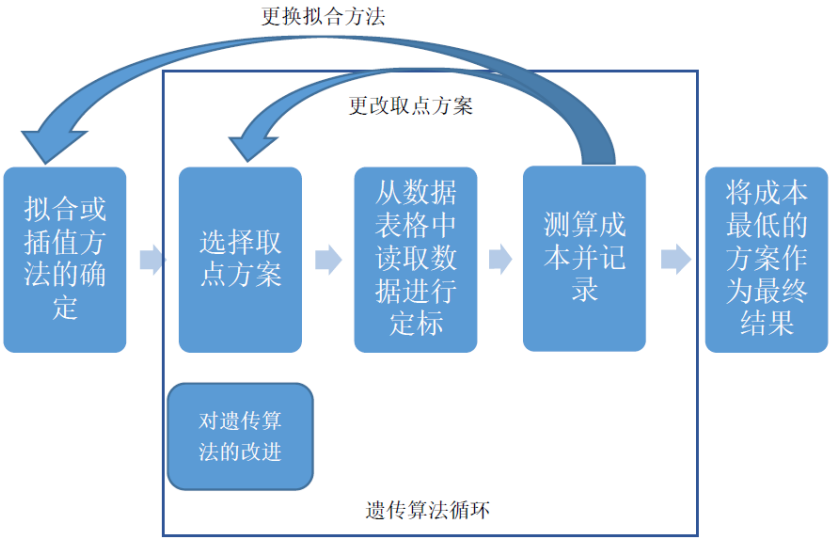

图 2-3-1 思路流程

了适当的改进，具体方案如下：通过老师的讲解与课下的资料查阅，本组决定在三次多项式拟合，三次样条函数插值，三次 Hermite 插值中选择拟合插值方法。通过遗传算法的迭代不

断尝试与更改取点方案，并对遗传算法中的取点进行优化，从而得到成本最低的拟合+取点方案。

3 拟合或插值法的选择

3.1 概述

拟合就是用最接近的函数的关系，去表征数据点所反映的输入输出关系。拟合方法选取是否得当直接影响了实验结果是否能够具有良好的适应度。不同的拟合方法有不同优劣。下面初步就三次多项式拟合法和三次样条插值法的实现过程作介绍。

3.2 三次多项式拟合

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 a=polyfit(x,y,m)进行多项式拟合。其中输入参数 x，y 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。

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。

3.3 三次样条函数插值

三次样条插值法对于中间部分的相邻 4 个点采用三次曲线拟合，并在中间两点处做出拟合曲线；对于边界部分的 2 个点，选取与之靠近的三个点采用二次曲线拟合，并在靠近边界的两点处做出拟合曲线。如此操作，在断点处斜率和曲率都将连续。选取 6 个不同的点，通过三次样条插值拟合得到一条拟合曲线，对比曲线中的每一个点与实验测的数据，通过评测函数计算其成本。对比两种拟合方式所得成本，及其所需时间，便可评测出拟合方式的优劣。

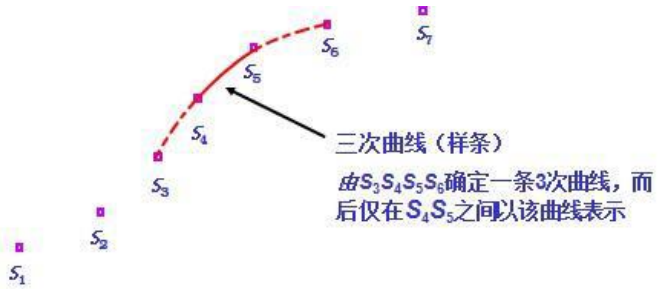

图 3-3-1 样条插值示意图

3.4 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 6 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”）或者是三次样条插值法（“spline”）。pchip 与 spline 调用方式完全相同，为了保证运行结果与时间准确， 且可进行横向对比，我们分别采用分段三次 Hermite 插值法“pchip”和三次样条函数插值 法“spline”计算[2]。

4 基于遗传算法的特征点的选择

4.1 概述

遗传算法是从代表问题可能潜在的解集的一个种群（population）开始的，而一个种群则由经过基因（gene）编码的一定数目的个体(individual)组成。每个个体实际上是染色体(chromosome)带有特征的实体。染色体作为遗传物质的主要载体，即多个基因的集合，其内 部表现（即基因型）是某种基因组合，它决定了个体的形状的外部表现，如黑头发的特征是 由染色体中控制这一特征的某种基因组合决定的。因此，在一开始需要实现从表现型到基因 型的映射即编码工作。由于仿照基因编码的工作很复杂，我们往往进行简化，如二进制编码， 初代种群产生之后，按照适者生存和优胜劣汰的原理，逐代（generation）演化产生出越来越好的近似解，在每一代，根据问题域中个体的适应度（fitness）大小选择（selection） 个体，并借助于自然遗传学的遗传算子（genetic operators）进行组合交叉（crossover） 和变异（mutation），产生出代表新的解集的种群。这个过程将导致种群像自然进化一样的 后生代种群比前代更加适应于环境，末代种群中的最优个体经过解码（decoding），可以作 为问题近似最优解。 遗传算法是具有“生成+检测”的迭代过程的搜索算法。基本流程如图 3 所示。可见遗 传算法是一种群体型操作，该操作以群体中所有个体为对象。选择（selection）、交叉 （crossover）和变异(mutation)是遗传算法的三个主要操作算子。 遗传算法是模仿遗传二字进行运算的。借阅生物界的遗传规律形成的算法。优胜略汰， 适者生存，是一种淘汰型的搜索和进化，生物界的“程序”编译运行着，同时知道下一步该 怎样运行，对哪些单位进行何种的筛选，拥有启发式，自动获取优化的方向和空间。即为程序完善自身运行的过程[3]。

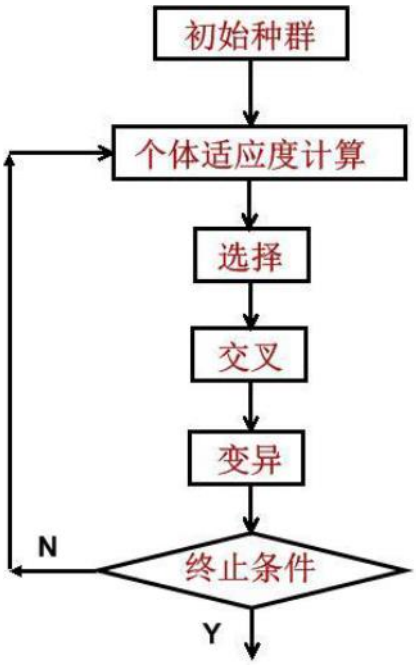

图 4-1-1 遗传算法的基本过程

4.2 遗传算法的实现

4.2.1 遗传编码的实现

假设当前所取得采样点数为 n，则每个个体的都是一个容量为 n 的数组，从 51 个位点中挑选不重复的 n 个，当成基因放入数组中即可，即每个个体的染色体都是一种取点方案，选择优秀的个体就是选择成本最低的取点方案。由于拟合方案最少要求 4 个点，综合各方面因素本组决定采用 6 个样本点的方案。

4.2.2 个体适应度的计算

本次课程内容要求的个体适应度就是每个校准方案的总体成本，在附录中的 cost_check.m 中包含了对取点方案进行成本计算的函数，所得结果即为该取点方案（个体的）适应度，与一般的适应程度不同的是，本实验的成本越低适应度越高，这一点在下面的选择过程中会有所体现。

4.2.3 选择的实现

与常规的遗传算法相同，本组采用轮盘赌算法进行优良个体的选择，即将每一个个体与轮盘上的一块扇形区域相对应，且适应度高的个体所占面积大，并且适应度之比等于面积之比。由于本次实验将总体成本作为适应度的判断依据，即成本越低，适应度越高。所以本实验将 P_i 作为个体 i 在轮盘上的所占比例。

$$\mathrm{Pi} = \frac{2C_{max} - C_i}{\sum 2C_{max} - C_k} \quad （4）$$

其中 为该种群所有个体中最大的成本为第 i 个个体的成本。具体实现代码请见附录 selection.m 部分。

4.2.4 变异的实现

在遗传算法中变异包括交叉与突变两部分。交叉就是在买足交叉概率时选择种群中的两个个体，随机从中间某个位置断开，断点后半部分交换后放入原来的位置。具体实现见 crossover.m。突变的实现比较复杂，先升成初始值为 0 的容量为 51 的数组，将个体取点对应的位置的 0 改为 1，然后在满足突变概率的情况下从这 51 个位点中随机选择一个，如果原来是 0 则改为 1，并且随机将原来的一个 1 改为 0；如果原来是 1 则反之。最后将 1 对应的位点位置数重新记录到该个体中。

5 初步结果

本小组通过上文所叙述的三种拟合插值方法，与遗传算法的代码编写，结合老师所给的样本数据得到如下结果。

分别采用了三次多项式拟合，三次样条函数插值，三次 Hermite 插值方法进行测试，下表是以 6 个取样点，种群的个体数 100，交叉概率 0.5，突变概率 0.05，50 世代的模拟结果。

表 5-1-1　初步结果表

拟合方法	时间（s）	最优解	最优成本
三次多项式拟合	1767.9826	04 16 27 35 42 49	124.7155
三次样条函数插值	405.0117	03 13 23 32 43 49	96.5670
三次 Hermite 插值	383.9858	02 12 20 27 37 50	89.1017

从上表的数据可以看出三次多项式拟合不仅不能很好的找到最优解与最优成本而且耗时远远多于另外两个方法，性能最差；三次样条函数插值耗时不错但最优成本偏高；三次 Hermite 插值用时最短且寻找到的成本最低，性能最好。综合上面的分析，本组认为采用三次 Hermite 插值结合遗传算法找到的最优解为 02 12 20 27 37 50，最优成本为 89.1017。

6 思考与拓展

6.1 取样点个数的选择

通过对成本函数的分析本组发现：总成本主要由两部分组成，定标成本与误差成本。如

果取样点较少则定标成本较低，但由于可供拟合的点减少会导致拟合图像的误差交大则会提高样本的误差成本；如果取样点较多，虽然图像拟合的较好但会带来很大的定标成本。所以找到这两个成本的平衡点十分重要。由于拟合方法最少需要 4 个样本点，本组便选择 4，5，6，7，8 个样本点进行测试。

下图采用了三种拟合插值方法，种群的个体数 100，交叉概率 0.5，突变概率 0.05，50 世代的模拟结果。

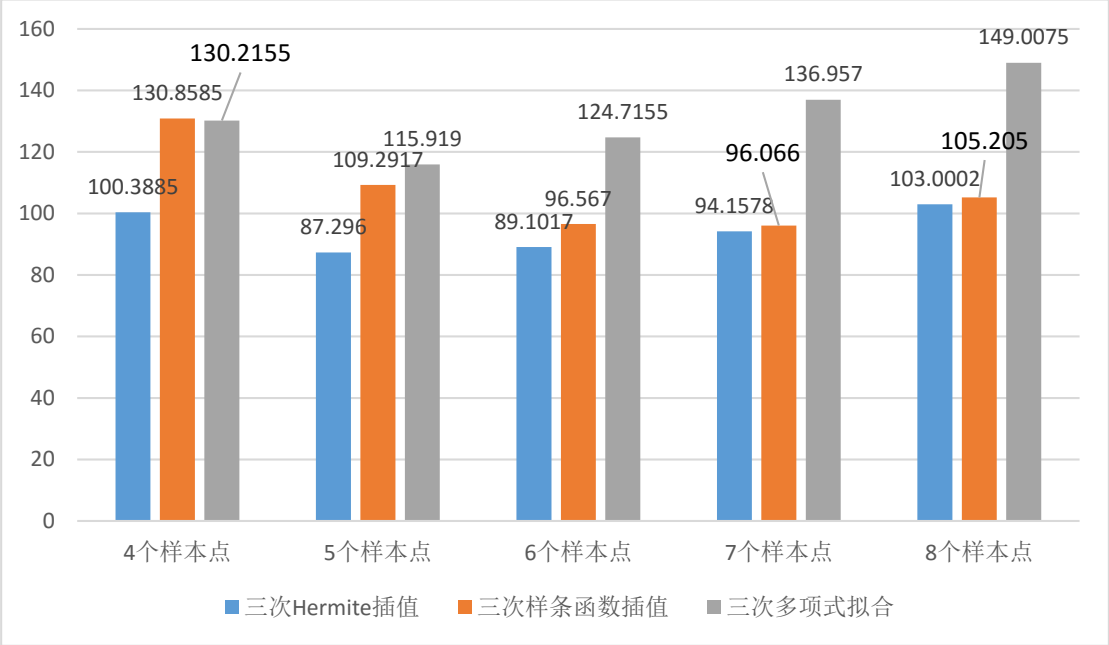

图 6-1-1 样本点个数与成本的关系

从上图可以看出三次 Hermite 插值时取 5 个样本点成本最低；三次样条函数插值去 6 或 7 个样本点成本最低；

6.2 突变率与交换率的选择

在多次的模拟实验中本组发现在两次实验中结果可能会收敛于一大一小两个结果，经过分析后我们认为：收敛于大的解是收敛到了局部最优解。为了跳出局部最优解我们尝试提高突变率与交换率。实验结果如下表。

以种群的个体数 100，5 个样本点，50 世代的模拟结果，采用分段三次 Hermite 插值

.

表 6-2-1 改变变异率测试表

交换率 p0 突变率 pm	p0= 0.5 pm= 0.05	p0=0. 5 pm=0. 25	p0=0. 97 pm=0. 05	p0=0. 97 pm=0. 25
最优成本	87.296	86. 6642	87. 7850	86. 6142

通过上表可以看出提高交换率的效果不明显，但提高突变率可以一定程度上跳出局部最优解，降低成本。

6. 3 对选择函数的优化

通过多次的实验运行本组发现如果采取常规的轮盘赌算法最后结果的收敛速度很慢，且耗费了大量时间，本组对其的分析如下：通过随机生成的适应度界限与随机选择的个体的适应度进行比较达标则放入下一代种群，直到种群已满的选择方法最大程度的保证了随机性，但适应度高的个体进入下一代的概率并没有很大的提高，体现在优良个体的选择要耗费大量的代数，即收敛慢。本组经过思考后将每一代最优的个体先插入到种群的开头，中间，结尾三个位置，再进行一般的选择；这样便有效提高了优良个体在下一代的占比，提高了收

敛速度。

表 6-3-1 选择函数优化前后对比

	优化前	优化后
最终收敛代数	50	30

6.4 附近探测

经过多次模拟实验我们发现最优解总是为定在某些固定的点附近，我们认为将每一代的最优解的某一个取点左移或右移一位可能得到更优化的结果，并且通过左右移动便可以避免收敛于局部最优解，详细过程通过 near_ detected 函数实现，前后对比的结果如下表。

以种群的个体数 100，交叉概率 0.5，突变概率 0.05，30 世代的模拟结果，三次多项式拟合 5 个样本点，三次样条差值 6 个样本点，三次 Hermite 插值 5 个样本点。

表 6-4-1 附近探测前后对比

拟合方法	时间（s）	最优解	最优成本	优化前最优成本
三次多项式拟合	1268.5952	03 14 25 36 48	114.0915	115.9190
三次样条函数插值	217.3273	03 12 22 31 43 50	94.8983	96.5670
三次 Hermite 插值	205.9089	04 15 25 35 48	85.1462	86.6642

可见附近探测可以有效降低成本。

7 结论

综合上文所有的结果我们小组的结论是：采用三次 Hermite 插值的方法比另外两种方法速度快精确度高，结合优化改进过后的遗传算法可以得到令人满意的结果。

在以种群的个体数 100，交叉概率 0.5，突变概率 0.05，30 世代的模拟，三次 Hermite 插值 5 个样本点得到的最优解为 04 15 25 35 48，最优成本为 85.1462，用时 205.9089s。达到了本课程的要求。

8 参考文献

[1] 袁焱.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] 龚纯王正林，《精通 matlab 最优化计算》，电子工业出版社
[3] 张文修，梁怡，《遗传算法的数学基础》，西安交通大学出版社

9 附录

9.1 老师提供的检验结果代码

本代码中的数据是本小组的最终结果数据。

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

my_answer=[ 04 15 25 35 48 ];%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);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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,'pchip');
```

```
end
```

9.2 本小组编写的 matlab 代码

9.2.1genetic_algorith.m 文件：

```
global p0 pm N
tic;
max_generation=30;
size_of_pop=100;
num_of_sample=5;
p0=0.5; %交换率
pm=0.05; %突变率
N=51; %基因数
pop=zeros(size_of_pop,num_of_sample);
for i =1: size_of_pop
   pop ( i , : )=randperm(51 , num_of_sample) ;     %选择监测点
end
cost_pop=zeros(1,size_of_pop);
for i=1:size_of_pop
    cost_pop(i)=cost_check(pop(i,:));                    %计算成本
end
for generation=1:max_generation                              %进化代数
    selection;                                                %选择
    crossover;                                                %交叉
    mutation;                                                 %突变
    for i=1:size_of_pop
       cost_pop(i)=cost_check(pop(i,:));                 %计算成本
    end
    [minpop,minpop_row]=min(cost_pop);     %找到最大最小值及其个体所在行数
    [maxpop,maxpop_row]=max(cost_pop);
    fprintf('%2d %5.4f %5.4f %5.4f  ：   ' , generation ,mean( cost_pop ),maxpop ,minpop );
                                        %显示代数  平均成本  最高成本  最低成本
    near_detecte;                                    %附近探测
    pop(minpop_row, : )=sort(pop(minpop_row, : ));
    for j =1:num_of_sample
        fprintf ( '%02d ' , pop(minpop_row,j)) ;
    end
    fprintf('%07.4f', minpop)
    fprintf('\n');
    toc ;
end
toc;
```

9.2.2mutation.m 文件：

```
global pm N
for i =1:size_of_pop
      r=rand( ) ;
      if ( r<pm)
         answer_gene=zeros(1,N);
         answer_gene(pop ( i , : ))=1;
         index=randi(N);
         if(ismember( index , pop ( i , : ) ))
         flag=1;
         answer_gene ( index )=0;
      else
         flag=0;
         answer_gene ( index )=1;
      end
      index=randi (N ) ;
      if(flag)
         while ( ismember ( index , pop ( i , : ) ) )
               index=randi (N) ;
          end
          answer_gene ( index )=1;
       else
          while (~ ismember ( index , pop ( i , : ) ) )
                 index=randi (N) ;
           end
       answer_gene ( index )=0;
       end
       point=1:N;
       pop ( i , : )=point(logical(answer_gene));
      end
end
```

9.2.3 selection.m 文件：

```
temp=-cost_pop+2*max(cost_pop);
pop1=temp/sum(temp);
pop2=zeros(1,size_of_pop);
for i=1:size_of_pop
      pop2(i)=sum(pop1(1:i));
end;
size_of_tmp_pop=size_of_pop;
tmp_pop=zeros(size_of_tmp_pop ,num_of_sample ) ;
%pmin=min(cost_pop);
[mincost,mincost_row]=min(cost_pop);    %mincost is useless
tmp_pop ( 1 , : )=pop (mincost_row , : ) ;
```

```
tmp_pop (size_of_pop , : )=pop ( mincost_row , : ) ;
tmp_pop(size_of_pop / 2 , : )=pop (mincost_row , : ) ;
for i =[2:size_of_tmp_pop/2-1 ,( size_of_tmp_pop/2+1):(size_of_tmp_pop-1) ]
    r=rand();
    while(1)
        j=randi(size_of_pop);
       if(pop2( j )>=r )
            break;
        end
    end

     tmp_pop( i , : )=pop ( j , : ) ;
end
pop=tmp_pop;
```

9.2.4 crossover.m 文件：

```
global p0
flag=0;
times=10;
for i=1:times
    for k=1:size_of_pop
        r=rand();
        if(r<p0 && flag==0)
            a=k ;
            flag=1;
        end
        if(r<p0 && flag==1)
            b=k ;
            flag=2;
            break;
        end
    end
    if (flag==2 && a~=b )
       % r=randi([2,num_of_sample] , 1 ) ;
        r=randi(num_of_sample) ;
       tmp=pop ( a , r : num_of_sample ) ;
       pop( a , r : num_of_sample )=pop(b , r:num_of_sample) ;
       pop(b , r : num_of_sample )=tmp ;
    end
end
```

9.2.5 curvefitting.m 文件：

```
function y=curvefitting ( x_premea , y0_premea )
x =[5.0:0.1:10.0 ];
```

```
y=interp1( x_premea ,y0_premea ,x,'pchip' ) ;

% a=polyfit(x_premea, y0_premea,3);
% y=polyval(a,x);
%spline
```

9.2.6 cost_check.m 文件：

```
function cost=cost_check(answer)
size_of_answer=length(answer);
data=csvread( '20150915dataform.csv' ) ;
[M,N]=size(data);
x=zeros (M/2 ,N) ;
y0=zeros (M/2 ,N) ;
y1=zeros (M/2 ,N) ;
for i =1:M/2
    x ( i , : )=data (2* i-1 , : );
    y0 ( i , : )=data (2* i , : ) ;
end
answer_gene=zeros ( 1 ,N) ;
answer_gene(answer )=1;
index_temp=logical (answer_gene ) ;
x_optimal=x ( : , index_temp ) ;
y0_optimal=y0 ( : , index_temp ) ;
for i =1:M/2
    y1 ( i , : )=curvefitting( x_optimal ( i , : ) , y0_optimal ( i , : ) );
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-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 (M/2 ,1) * size_of_answer ;
cost=sum( si ) /(M/2) ;
```

9.2.7 near_detected.m 文件：

```
for place=1:num_of_sample
```

```
        tmp_pop=zeros(1,num_of_sample);
        tmp_pop=pop(minpop_row,:);
        if rand()<0.5
          if    tmp_pop(1,place)<51;
            tmp_pop(1,place)=tmp_pop(1,place)+1;
          end
        else if tmp_pop(1,place)>1;
                  tmp_pop(1,place)=tmp_pop(1,place)-1;
            end
        end
        if length(unique(tmp_pop))==num_of_sample
            cost_tmp_pop=cost_check(tmp_pop(1,:));
            if    cost_tmp_pop<minpop
                  minpop=cost_tmp_pop;
                  pop(minpop_row,:)=tmp_pop(1,:);
                  cost_pop(minpop_row)=minpop;
            end
        end
end
```