统计推断在模数、数模转换系统中的应用

组号：61　姓名：甘文耀　学号：5140309377　姓名：王建刚　学号：5140309373

摘要：本文以传感器批量生产为背景，寻找校准的优化方案。在确保精度的前提下，运用统计方法结合 MATLAB 对样本中数据进行分析，采用遗传算法找到了最优选点方案，并通过不断优化初始化种群和变异概率，大大丰富了初始种群的多样性，得到了优化后的解决方案，并且通过黑箱测试评价取点和拟合方案，得到了最优解。

关键词：MATLAB，遗传算法,定标

Application of Statistical Inference in DA Inverting System

Team :NO.61　Gan Wen-yao 5140309377　Wang Jian-gang 5140309373

ABSTRACT: Based on sensor batch production as the background, to find the optimized plan for the calibration. In ensuring the precision of the premise, using the statistical method combined with MATLAB to analyze samples data, using genetic algorithm to find the most optimal solution, and through continuous optimization of the initialization population and mutation probability, greatly enriched the diversity of initial population, the optimized solution, and through the black box test evaluation points and fitting scheme, the optimal solution are obtained.

Key words : MATLAB, Genetic algorithm, calibration.

1. 引言

本课题研究的内容为被监测物理量 Y 与 X 之间的函数关系，由于二者不存在确定的函数关系，只能通过研究大量的数据寻找适当的拟合函数。因为样本存在个体的差异，因此对单个样本的研究不能代表整体。而在对整体研究时，如果用所有的点进行函数拟合，运算量太大，几乎难以实现，而且成本太高。因此，需要通过该课题的研究，寻求一个最优化方法，在成本较低的前提下尽量减小误差。

2. 研究课题：

假定有某型投入批量试生产的电子产品，其内部有一个模块，功能是监测某项与外部环境有关的物理量（可能是温度、压力、光强等）。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。[1]

3. 课题分析：

经过仔细分析之后，我们发现了该课题设计的主要矛盾即尽可能提高准确性与尽可能降低测定成本之间的矛盾，具体来说，要想更加准确地对该类传感器定标，就需要对每一个样品测定尽可能多的数据（以 51 组为上限），但随着测定数据的组数增多，成本显然会增加，因此，我们需要解决的主要问题就是，权衡两者的关系，在不影响准确性的情况下，尽量降

低成本。很明显，在样品数量庞大的情况下，根据当前的计算机技术，对每一个样品测定 51 组数据是几乎不可能实现的，而且，这也是不必要的。我们可以通过合理的测定点选取和合理的启发式搜索算法，在测定点数量完全可以接受的情况下，较为合理地模拟出产品的整体性质。

4. 研究目的：

4.1 降低定标成本：

尽量减少测定点的数量，根据已测定点数据，推断未测定点大致数据。

4.2 方法：

（1）线性插值；

（2）非线性插值。

4.3 步骤：

（1）拟合：靠数学方法，使用连续函数（也就是曲线）或者更加密集的离散方程尽量逼近（即最小二乘意义上的差别最小化）这些已知离散数据点集。

（2）插值：利用拟合得到的函数曲线在某区间内的值，作为原本数值未知的点的近似取值。[2]

4.4 模型创建：

为了对本课题展开有效讨论，需建立一个数学模型，对问题的某些方面进行必要的描述和限定。

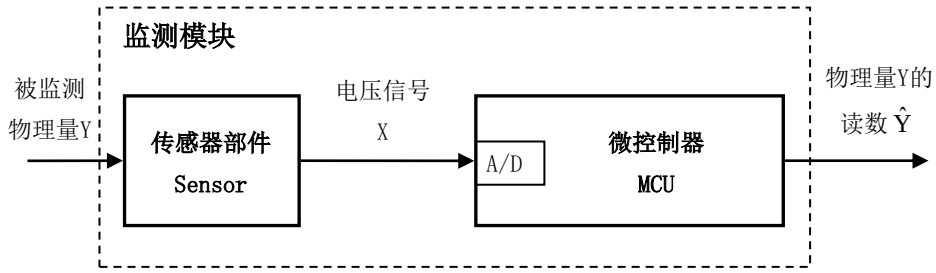


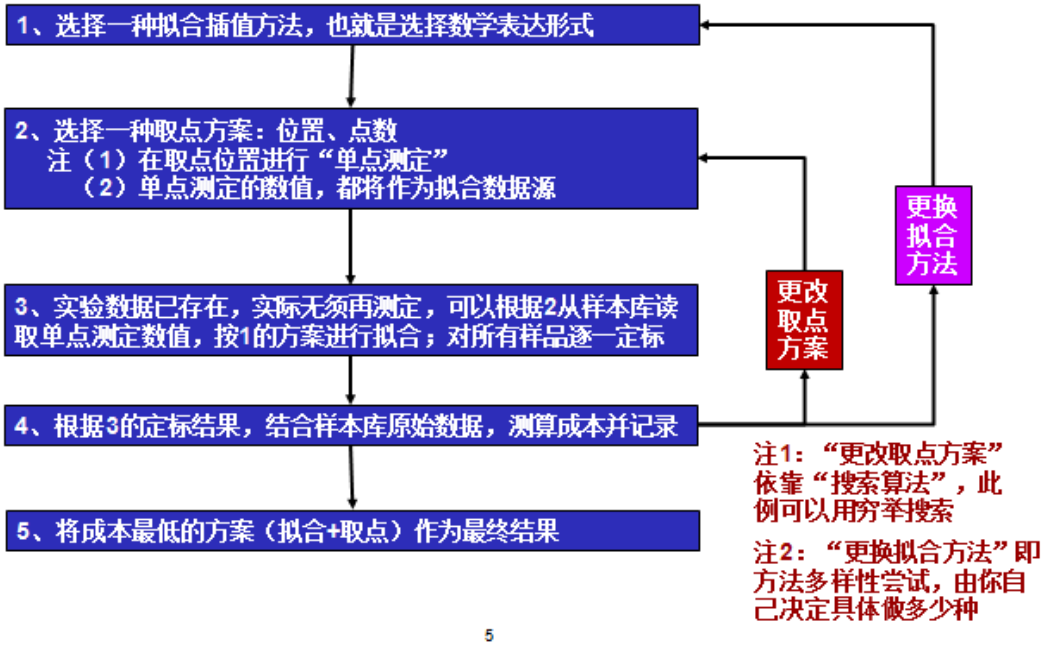
图 1 监测模块组成框图

监测模块的组成框图如图 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）。

其中 $X \in \{x_1, x_2, x_3, ..., x_{50}, x_{51}\} = \{5.0, 5.1, 5.2, ..., 9.9, 10.0\}$

相应的 Y 估测值记为 $\hat{y}_i = f(x_i)$， Y 实测值记为 y_i， $i = 1, 2, 3, ..., 50, 51$。[1]

4.5 求解思路：



[3]

5. 拟合方法探究：

5.1 三次多项式拟合：

(1)使用三次多项式对选取七个的特征点组合即可能解进行拟合。样本为 S={S1，S2……S7}。

设三次多项式为 y=axˆ3+bxˆ2+cx+d

5.2 三次样条插值拟合：

三次样条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。

使用三次样条拟合对一个可能解，即七个特征解组合 S={S1，S2……S7}进行拟合。

6. 算法选取：

6.1 穷举法：

根据课题 5 的部分条件确定答案的大致范围，并在此范围内对所有可能的情况逐一验证，直到全部情况验证完毕。若某个情况验证符合题目的全部条件，则为本问题的一个解；若全部情况验证后都不符合题目的全部条件，则本题无解。[4]但对本课题来说，由于数据过多，计算机需要完成大约 30 亿次拟合，才能找出最优解，是难以完成的，因此否定了该方法，重新寻找更加高效的算法。

6.2 遗传算法：

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，是通过模拟自然进化过程搜索最优解的方法。它从问题解的串集开始搜索，同时处理群体中的多个个体，并且仅用适应度函数来评估个体，采用概率的变迁规则来指导搜索

方向，具有自组织、自适应、自学习性和动态适应性。

算法实现过程：

（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)；

（2） 个体评价：计算群体 P(t)中各个个体的适应度；

（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；

（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；

（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)；

（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。[4]

6.3 模拟退火算法：

模拟退火算法来源于固体退火原理，是一种基于概率的算法，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。

模拟退火算法从某一较高初温出发，伴随温度参数的不断下降,结合概率突跳特性在解空间中随机寻找目标函数的全局最优解，即在局部最优解能概率性地跳出并最终趋于全局最优。模拟退火算法是通过赋予搜索过程一种时变且最终趋于零的概率突跳性，从而可有效避免陷入局部极小并最终趋于全局最优的串行结构的优化算法，理论上算法具有概率的全局优化性能。

算法实现过程：

（1）初始化：初始温度 T（充分大），初始解状态 S(是算法迭代的起点)，每个 T 值的迭代次数 L；

（2）对 k=1，……，L 做第（3）至第（6）步；

（3）产生新解 S′；

（4）计算增量 Δt′=C(S′)-C(S)，其中 C(S)为评价函数；

（5）若 Δt′<0 则接受 S′作为新的当前解，否则以概率 exp(-Δt′/T)接受 S′作为新的当前解；

（6）如果满足终止条件则输出当前解作为最优解，结束程序，

终止条件通常取为连续若干个新解都没有被接受时终止算法；

（7）T 逐渐减少，且 T->0，然后转第 2 步。[4]

与遗传算法比较：

利用 MATLAB 编程时代码相对简单，计算时间较短，但在实际执行时，具有一定的偶然性。

7. 解决问题：

我们最终确定从每组 51 个数据中选取 7 个点进行拟合并计算适应度，综合考虑效率、准确性等因素，我们最终选择了遗传算法。

7.1 遗传算法实现方法：

（1）生成初始种群

设变量 xyposition 为 400*7 的取点矩阵即为初始种群，初始值为 0。循环 400 次，每次随机生成乱序的 1~50 的数，将前 7 个放入当前行。

（2）计算适应度

拟合每行选取的点，计算出每行即每个个体的成本，用 1 除以成本代表生存能力，将每个个体的生存能力除以每行适应能力总和。为方便后面程序进行，从大到小排序适应度矩阵以及对应的取点矩阵。

（3）进行选择

每个个体选择时根据适应度从大到小依次进行判断，每次取 0~1 随机数，如果当前适应度大于或等于随机数，则选取该个体生存，跳出循环，否则对剩下的个体进行适应度的计算并且排序判断，直到选到一个个体为止。一共选出新的 400 个个体。

（4）染色体互换

根据染色体互换概率计算出互换次数，再随机找出染色体互换位置和两个发生染色体互换的个体。每次互换后寻找是否产生新的最小个体成本。

（5）基因突变

根据基因突变的概率判断每个个体的每条染色体是否发生突变，如果突变，则使该染色体上随机突变为一个与该染色体不同的染色体。然后对发生突变的个体寻找是否产生新的最小个体成本。

（6）找出最低成本

多次循环后选择成本最低的方案。

7.2 成本计算：

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

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

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。

- 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。[1]

8. 问题分析及运算结果

8.1 算法改进：

问题 1：两个个体互换染色体后若有重复的染色体则再重新寻找位置和个体互换，但是当程序运行一段时间后，一个种群中相同或者相似的个体非常多，因此很难找到两个个体互换染色体后各自不出现重复的染色体，循环将很难跳出，甚至陷入死循环，程序运行时间也会非常长。

解决办法：染色体互换后，对于有重复出现的染色体的个体，将互换了的染色体后移一个位置，直到该染色体与其他染色体不重复为止。这样就解决了难以跳出循环的问题。

问题 2：最开始生成的选取染色体的初始位置是从 1-51 中随机选取 7 个数，很容易造成取点过于密集，从而使得选取的数据很难估计全局的数据，从而增大误差。

解决办法：若每个个体选取 n 条染色体，将 1-51 大致均匀分成 n 份，每次生成初始选取位置时从这 n 份中各随机选取 1 条染色体，从而使得初始选取的位置比较均匀。

8.2 算法参数的选取：

(1)种群中个体数目

种群中个体数目越多就可以获得更多的样本从而获得更多的数据，但是种群数目太多则会严重影响程序运行时间，经过多次尝试，当种群中个体数目为 30 时比较合适。

(2)循环代数

表 1　循环代数测试表

选点5						
交叉概率	0.5	0.5	0.5	0.5	0.5	0.5
突变概率	0.01	0.01	0.01	0.1	0.1	0.1
个体数目	30	30	30	30	30	30
循环代数	10	20	30	10	20	30
最小成本	87.326	86.354	86.7403	89.337	86.6805	86.862
方案	5 17 25 35 50	1 16 27 35 50	6 16 25 35 48	5 14 23 35 49	5 18 28 37 48	6 16 26 36 48
选点6						
交叉概率	0.5	0.5	0.5	0.5	0.5	0.5
突变概率	0.01	0.01	0.01	0.1	0.1	0.1
个体数目	30	30	30	30	30	30
循环代数	10	20	30	10	20	30
最小成本	88.7257	87.8835	87.4897	88.8267	88.7257	87.917
方案	4 13 23 31 41 49	3 14 24 30 37 49	2 14 24 31 39 49	3 14 23 31 38 47	4 13 23 31 41 49	6 15 24 31 39 49

如表 1 所示，当循环代数为 10 时的成本普遍比 20 和 30 时的成本高，而循环代数为 20 和 30 的成本相差不大，因此选取循环代数为 30.

(3)交叉互换概率、单个基因突变概率和染色体选择数目的选取

先尝试一些数据，发现交叉互换概率在 0.5、0.05 中选取，基因突变概率在 0.01、0.1 中选取比较合适，染色体选择的测试数目为 4、5、6、7.

表 2 交叉互换、基因突变概率和染色体选择数目测试表

选点4				
交叉概率	0.5	0.5	0.05	0.05
突变概率	0.01	0.1	0.01	0.1
个体数目	30	30	30	30
循环代数	30	30	30	30
最小成本	101.6873	105.9428	104.0645	103.6218
方案	6 20 34 50	4 18 33 47	5 19 33 46	5 21 33 45
选点5				
交叉概率	0.5	0.5	0.05	0.05
突变概率	0.01	0.1	0.01	0.1
个体数目	30	30	30	30
循环代数	30	30	30	30
最小成本	85.7228	86.862	86.7403	86.6805
方案	5 17 27 35 48	6 16 26 36 48	6 16 25 35 48	5 18 28 37 48
选点6				
交叉概率	0.5	0.5	0.05	0.05
突变概率	0.01	0.1	0.01	0.1
个体数目	30	30	30	30
循环代数	30	30	30	30
最小成本	87.4897	87.917	88.508	87.8835
方案	2 14 24 31 39 49	6 15 24 31 39 49	4 14 24 32 38 48	3 14 24 30 37 49
选点7				
交叉概率	0.5	0.5	0.05	0.05
突变概率	0.01	0.1	0.01	0.1
个体数目	30	30	30	30
循环代数	30	30	30	30
最小成本	93.8177	94.4192	93.6205	94.4192
方案	2 11 19 26 33 42 51	6 12 22 27 34 41 50	1 12 19 28 34 43 50	6 12 22 27 34 41 50

通过表 2 的数据分析可知，当发现交叉互换概率为 0.5，基因突变概率为 0.01 时，成本最小，当染色体选择数目为 5 时，成本最小。

(4)拟合与插值方式

通过多次试验，发现在线性插值、三次样条插值、三次多项式插值中运用三次多项式插值所得的成本最小。

8.3 运行结果

经多多次运行程序，最小成本为 85.7215，最小成本的选点方案为【4 17 26 36 48】。

9. 参考文献

[1]“统计推断”课程设计的要求 V2.2 2015-9-22

[2]统计推断讲座 2_问题的提出和基本求解思路

[3]统计推断讲座 3_问题的求解途径

[4]百度百科

10. 附录

10.1 遗传算法代码

```
function main()
xNum=400;%x 数据行数
eachCost=12;%单点测定成本
chrNum=51;%个体染色体数目
indNum=30;%种群中个体数目
croPro=0.5;%交叉互换概率
metPro=0.01;%单个基因突变概率
loop=30;%循环代数
chooseChrNum=5;%染色体选择数目
allMinCost=99999;%总最小成本
%得到初始取点矩阵,初始尽量均匀取点
xyposition=zeros(indNum,chooseChrNum);
%均匀取 4 点
%for n=1:indNum
%       k=[randperm(13),randperm(13)+13,randperm(13)+26,randperm(12)+39];
%       xyposition(n,:)=[k(1),k(14),k(27),k(40)];
%end
%均匀取 5 点
for n=1:indNum
      k=[randperm(10),randperm(10)+10,randperm(10)+20,randperm(10)+30,randperm(11)+40];
      xyposition(n,:)=[k(1),k(11),k(21),k(31),k(41)];
end
%均匀取 6 点
%for n=1:indNum
%
k=[randperm(9),randperm(8)+9,randperm(8)+17,randperm(8)+25,randperm(9)+33,randperm(9)+4
2];
%       xyposition(n,:)=[k(1),k(10),k(18),k(26),k(34),k(43)];
%end
%均匀取 7 点
%for n=1:indNum
%
k=[randperm(8),randperm(7)+8,randperm(7)+15,randperm(7)+22,randperm(7)+29,randperm(7)+3
```

```
6,randperm(8)+43];
%       xyposition(n,:)=[k(1),k(9),k(16),k(23),k(30),k(37),k(44)];
%end
%读取数据
data=xlsread('data.xlsx');
x=zeros(xNum,chrNum);
y=zeros(xNum,chrNum);
%分离出 xy 矩阵
for n=1:xNum
    x(n,:)=data(2*n-1,:);
    y(n,:)=data(2*n,:);
end
%ceshidata=[4 17 26 36 48];
%ceshicost=getCost(xNum,chrNum,x,y,ceshidata,eachCost,chooseChrNum);
%fprintf('\n 经计算，我测试数据[4 17 26 36 48]的答案对应的总体成本为%5.2f\n',ceshicost);
%开始循环
for loop1=1:loop
cost=zeros(1,indNum);
for n=1:indNum
    cost(1,n)=getCost(xNum,chrNum,x,y,xyposition(n,:),eachCost,chooseChrNum);
end
%获得当前最小成本及其选择方式
[minCost,costIndex]=min(cost);
minCostline=xyposition(costIndex,:);
%获得适应度矩阵
fitness=zeros(1,indNum);
fitnessSum=0;
for p=1:indNum
    cost(1,p)=1/cost(1,p);
    fitnessSum=cost(1,p)+fitnessSum;
end
for q=1:indNum
    fitness(1,q)=cost(1,q)/fitnessSum;
end
%从大到小排序适应度矩阵以及对应的取点矩阵
newxyPosition=zeros(indNum,chooseChrNum);
newfitness=zeros(1,indNum);
col=1;
for n=1:indNum
    [~,maxindex]=max(fitness);
    newfitness(1,col)=fitness(1,maxindex);
    newxyPosition(col,:)=xyposition(maxindex,:);
    fitness(1,maxindex)=-1;
    col=col+1;
```

```
end
%进行选择
nextPosition=newxyPosition;
for n=1:indNum
tmp=newfitness;
for k=1:indNum
    choose=rand();
    if(tmp(1,k)>=choose)
        nextPosition(n,:)=newxyPosition(k,:);
        break;
    else
        sum=0;
        for p=(k+1):indNum
            sum=sum+tmp(1,p);
        end
        for p=(k+1):indNum
            tmp(1,p)=tmp(1,p)/sum;
        end
    end
end
end
%染色体互换
for n=1:round(croPro*indNum)
    position=round(rand()*(chooseChrNum-1)+1);
    chr1=round(rand()*(indNum-1)+1);
    chr2=round(rand()*(indNum-1)+1);
    tmp=nextPosition(chr1,position);
    nextPosition(chr1,position)=nextPosition(chr2,position);
    nextPosition(chr2,position)=tmp;
    while 1%互换后不能有重复的染色体
        test=1;
        for k=1:chooseChrNum
            if k~=position&&nextPosition(chr1,position)==nextPosition(chr1,k)
                %如果有重复，将该互换位置的染色体移动一个位置
                nextPosition(chr1,position)=mod(nextPosition(chr1,position),51)+1;
                test=0;
            end
        end
        if test==1
            break;
        end
    end
    while 1%互换后不能有重复的染色体
        test=1;
```

```
            for k=1:chooseChrNum
                if k~=position&&nextPosition(chr2,position)==nextPosition(chr2,k)
                    %如果有重复，将该互换位置的染色体移动一个位置
                    nextPosition(chr2,position)=mod(nextPosition(chr2,position),51)+1;
                    test=0;
                end
            end
            if test==1
                break;
            end
        end
    %每次互换后寻找是否为最小成本
        nextcost=getCost(xNum,chrNum,x,y,nextPosition(chr1,:),eachCost,chooseChrNum);
        if nextcost<minCost
            minCostline=nextPosition(chr1,:);
            minCost=nextcost;
        end
        nextcost=getCost(xNum,chrNum,x,y,nextPosition(chr2,:),eachCost,chooseChrNum);
        if nextcost<minCost
            minCostline=nextPosition(chr2,:);
            minCost=nextcost;
        end
    end
    %基因突变
    for n=1:indNum
        ifchange=0;
        for m=1:chooseChrNum;
            ifmetPro=rand();
            if metPro>=ifmetPro;
                ifchange=1;
                while 1%基因突变后不能有重复的染色体
                    position1=round(rand()*(chrNum-1)+1);
                    test=1;
                    for k=1:chooseChrNum
                        if position1==nextPosition(n,k)
                            test=0;
                        end
                    end
                    if test==1
                        nextPosition(n,m)=position1;
                        break;
                    end
                end
            end
```

```
        end
        if ifchange==1
            %每次个体变异后寻找是否为最小成本
            nextcost=getCost(xNum,chrNum,x,y,nextPosition(n,:),eachCost,chooseChrNum);
            if nextcost<minCost
                minCostline=nextPosition(n,:);
                minCost=nextcost;
            end
        end
    end
    if minCost<allMinCost
        allMinCost=minCost;
        allMinCostline=minCostline;%总最小成本方案
    end
    xyposition=nextPosition;
    end
    %输出最小成本及其方案
    disp('经计算，我测试的最低总体成本为');
    disp(allMinCost);
    disp('经计算，我测试的最低总体成本所对应的选取点为');
    disp(allMinCostline);
    end

    function cost1=getCost(xNum1,chrNum1,x1,y1,xyposition1,eachCost1,chooseChrNum1)
    %获得成本
    xyposition1=sort(xyposition1,2);%排序以便进行三次多项式拟合
    xdata=zeros(1,chooseChrNum1);
    ydata=zeros(1,chooseChrNum1);
    cost1=0;
    for n=1:xNum1
        number=1;
        for m=xyposition1
            xdata(1,number)=x1(n,m);
            ydata(1,number)=y1(n,m);
            number=number+1;
        end
        for k=1:chrNum1
            py=interp1(xdata,ydata,x1(n,k),'PCHIP');%进行三次多项式拟合
            dif=abs(y1(n,k)-py);
            if dif<=0.4
            elseif 0.4<dif&&dif<=0.6
                cost1=cost1+0.1;
            elseif 0.6<dif&&dif<=0.8
                cost1=cost1+0.7;
```

```
            elseif 0.8<dif&&dif<=1
                cost1=cost1+0.9;
            elseif 1<dif&&dif<=2
                cost1=cost1+1.5;
            elseif 2<dif&&dif<=3
                cost1=cost1+6;
            elseif 3<dif&&dif<=5
                cost1=cost1+12;
            else
                cost1=cost1+25;
            end
        end
    end
    cost1=(cost1+eachCost1*chooseChrNum1*xNum1)/xNum1;
    end
```

8.2 测试代码 test_ur_answer

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

my_answer=[4 17 26 36 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
```