统计推断在数模转换系统中的应用

第 64 组：汤顺堃 5140309058 杨祎威 5140309375

摘要：本课程设计旨在探究某输入输出具有非线性关系的传感器的校准方案。现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。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。我们运用统计推断的方法，获得电子产品的性能参数特性曲线上若干关键点并进行拟合（定标）最终结合数学表达试的运算得出一种成本合理的传感特性校准方案。由于实验提供的数据量较为庞大，若使用穷举搜索的方式我们显然无法胜任这项工作。因而，我们决定将启发式搜索运用到其中。这里，我们使用的是遗传算法。

关键词：数模转换系统，多项式拟合，三次样条插值，Hermite 插值，附近探测法，统计推断，遗传算法。

ABSTRACT: This course is designed to explore an input/output with the nonlinear relationship between sensor calibration scheme. There is a certain type of electronic products existing in trial production of electronic products, whose internal has a module, and its function is to monitor a physical quantity related to the external environment. The monitoring module input and output characteristics of the sensor unit is obviously nonlinear. We use the method of statistical inference, electronic products on the performance parameters of the characteristic curve of some key points and fitting (scaling) eventually a cost is obtained based on the mathematical expression of arithmetic reasonable sensing characteristic calibration scheme. Because of the relatively large amount of experimental data, if we use exhaustive search way,we obviously cannot fit for the job. Therefore, we decided to apply heuristic search. Here, we use the genetic algorithm.

KEYWORDS:conversion system, polynomial fitting, cubic spline interpolation,Hermite interpolation,nearing soundex,statistical inference,genetic algorithm

1. 引言

统计推断根据带随机性的观测数据（样本）以及问题的条件和假定（模型），而对未知事物做出的，以概率形式表述的推断。统计推断问题常表述为如下形式：所研究的问题有一个确定的总体，其总体分布未知或部分未知，通过从该总体中抽取的样本（观测数据）做出与未知分布有关的某种结论。

在工程实践中，我们常借助统计推断的理念，对变量间的函数关系进行分析。通常选取能代表变量之间关系的特征点，经过多种拟合方式进行拟合，从而求解出能够代表海量变量之间函数关系的最优化方式。

2. 课题概述

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。

3. 模型

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。

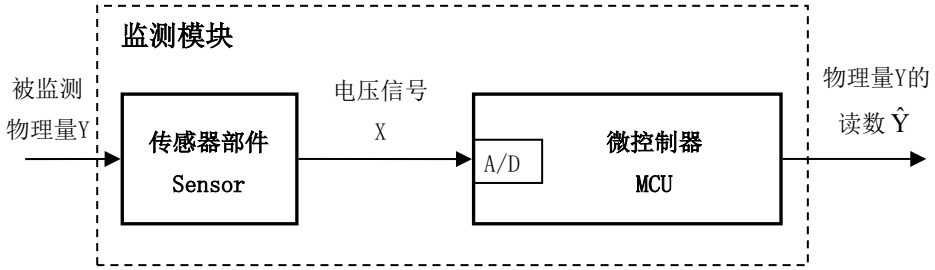


图 1 监测模块组成框图

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）。

所谓传感特性校准，就是针对某一特定传感部件个体，通过有限次测定，估计其 Y 值与 X 值间一一对应的特性关系的过程。数学上可认为是确定适用于该个体的估测函数 $\hat{y}=f(x)$ 的过程，其中 x 是 X 的取值，$\hat{y}$ 是对应 Y 的估测值。

考虑实际工程中该监测模块的应用需求，同时为便于在本课题中开展讨论，我们将问题限于 X 为离散取值的情况，规定

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

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

3.1 传感部件特性

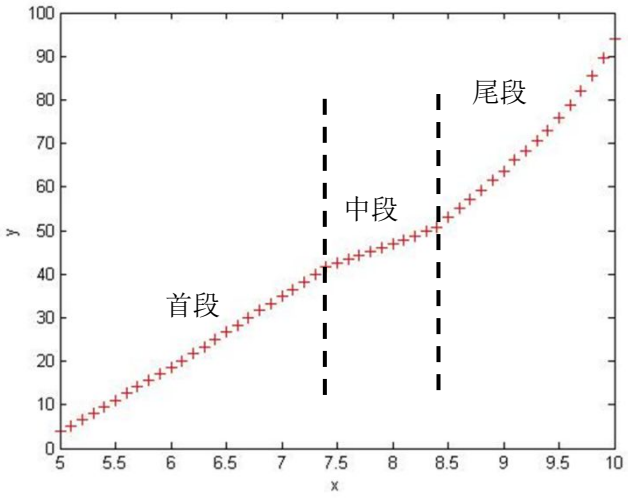


图 2 传感特性图示

一个传感部件个体的输入输出特性大致如图 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。

为进一步说明情况，图 3 对比展示了四个不同样品个体的特性曲线图示。

3.2 标准样本数据库

前期已经通过试验性小批量生产，制造了一批传感部件样品，并通过实验测定了每个样品的特性数值。这可以作为本课题的统计学研究样本。数据被绘制成表格，称为本课题的“标准样本数据库”。

该表格以 CSV 格式制作为电子文件。表格中奇数行存放的取值，偶数行存放对应的取值。第 2i – 1 行存放第 i 个样本的 X 数值，第 2i 行相应列存放对应的实测 Y 数值。

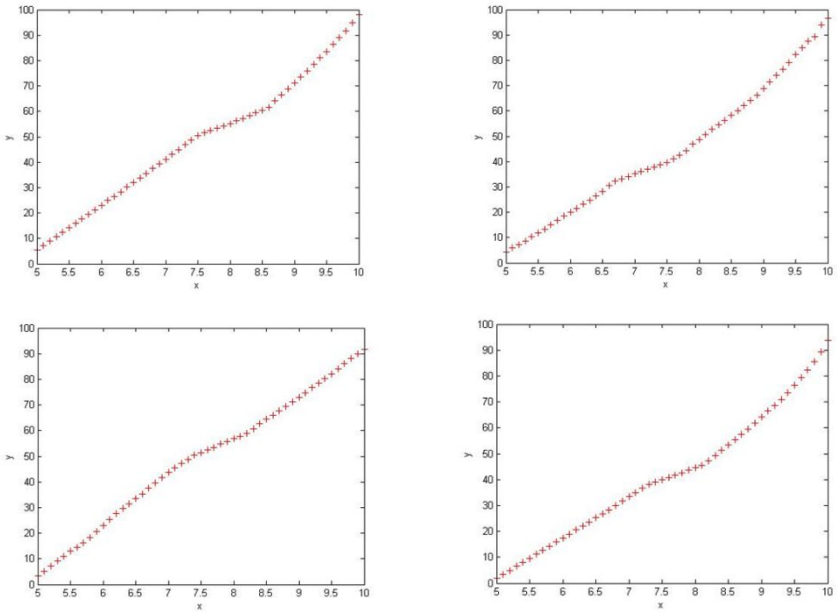


图 3 四个不同样本个体特性图示对比

3.3 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

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

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。

4. 遗传算法

4.1 简介

遗传算法 I 1(GA)是模拟生物在自然环境中的遗传和进化过程而形成的一种自适应全局优化概率搜索算法。由于该算法采用随机选择，对搜索空间无特殊要求，无需求导，具有运算简单、收敛速度快等优点．尤其适用于处理传统搜索方法难以解决的复杂和非线性问题。它是一种全局搜索能力很强而局部搜索能力不足的算法。GA 直接以目标函数作为搜索信息．在一定范围内同时搜索多个点．比传统优化算法寻优效率高．适合处理较复杂的优化问题。如组合优化．生产调度．复杂布局，自动控制，神经网络，图像处理等。研究发现，GA 可以用极快的速度达到最优解的 90%左右．但要得到真正的最优解则要花费很长时间。

4.2 选择算子

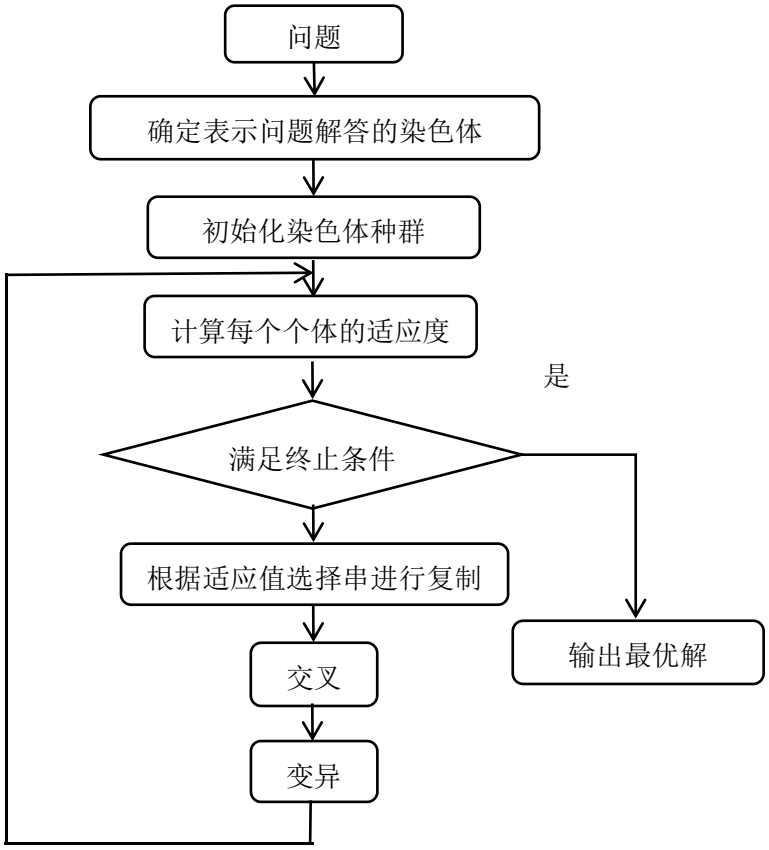
遗传算法中的选择操作就是用来确定如何从父代群体中按某种方法选取哪些个体遗传到下一代群体中的遗传运算，是对自然选择的一种模拟，也是遗传算法中一个重要的算子。

4.3 交叉算子

交叉算子通过模拟自然界生物的杂交过程对个体进行交叉操作，不断产生新个体、增加种群的多样性、扩大寻优范围，从而使得遗传算法具有较强的搜索能力，可以说交叉算子在遗传算法扩展求解空间，并达到全局最优的过程中发挥着至关重要的作用。

4.4 变异算子

变异算子通过变异不断在交叉算子产生的新个体的基础上进行微调、增加种群的多样性、使遗传算法在交叉算子决定的全局搜索能力的基础上还具有一定的局部搜索能力。



5. 多项式拟合

我们决定先用多项式拟合的方法进行尝试，这里我们使用到自定义函数 polyfitn，初始化为默认的 crtbp，变异也是默认的变异，不作优化。

最终我们得到了 5 组结果，分别对应着三、四、五、六、七次多项式。通过观察实验数据我们发现，随着拟合多项式次数 n 的增大，成本并没有如预期那样不断降低，反而呈现了先变小后边大的趋势。我们查阅了相关资料，找出了其中的原因。较低次成本高是因为拟合曲线不能很好贴合样本点，误差大；而较高次多项式出现结果偏差是由于取点过多，造成舍入误差，以至于发生所谓的“龙格现象”。

事实表明，多项式拟合的方法并不能给我们的实验带来太大的帮助，反倒是得到了并不优秀的成本结果。因此，我们将采取其他的拟合方法来帮助我们解决问题。

拟合多项式次数 n	最优解	成本
3	1 15 27 39 51	125.867
4	1 10 23 30 39 51	116.272
5	1 8 16 25 34 44 51	110.752
6	1 8 15 24 30 39 45 51	115.16
7	1 6 12 20 25 31 40 46 51	121.854

6. 方案设计

在我们的实验中，主要运用到三次样条插值拟合、Hermite 插值函数以及 gatbox 工具箱内的基本函数作为我们方案的基本设计要素。

6.1 三次样条插值拟合

在数值分析中，插值过程可以具体使用线性插值，三次样条插值，立方插值等操作。在曲线插值中最常用的是线性插值法，它是估计两个主干点之间数值的最简单，最易行的方法，但由于其不能显示连接主干点间的凸状弧线，因此我们决定使用三次样条插值法进行拟合。

样条插值使用一种特殊分段的分段多项式（称其为“样条”）进行插值的形式。样条插值可以使用低阶多项式样条来实现较小的插值误差，这样就很好得规避了使用高阶多项式所出现的“龙格”现象。

三次样条函数:

定义:函数 S(x)∈C2[a,b] ，且在每个小区间[xj,xj+1]上是三次多项式，其中

a =x0 <x1<...< xn=b 是给定节点，则称 S(x)是节点 x0,x1,...xn 上的三次样条函数。

若在节点 xj 上给定函数值 Yj= f(Xj).(j=0,1,...,n)，并成立

S(xj)=yj(j=0,1,...,n)，则称 S(x)为三次样条插值函数。

6.2Hermite 插值函数

如果不仅已知插值节点处的函数值，而且还掌握插值节点处的导数值(1 阶甚至高阶)；或者说，不仅要求在节点处插值多项式与被插值函数的值相等(Lagrange 条件)，而且还要求相应阶的导数值也要相等，这就是带导数插值，也称为 Hermite(埃尔米特)插值。从几何上看，Hermite 插值意味着插值函数不仅要过被插函数的已知点，而且在这些点上，两者还要“相切”(即导数值相同)，可见这种插值函数与被插函数的“密切”程度比 Lagrange 插值的情况更好，因此，Herimite 插值也称为密切插值。

6.3 利用 gatbox 工具箱制作遗传算法构建顺序参考

（gatbox 基本函数介绍见附录 1）

6.3.1 种群表示和初始化

种群表示和初始化函数有：crtbase，crtbp，crtrp。

GA 工具箱支持二进制、整数和浮点数的基因表示。二进制和整数种群可以使用工具箱中的 crtbp 建立二进制种群。crtbase 是附加的功能，它提供向量描述整数表示。种群的实值可用 crtrp 进行初始化。在二进制代码和实值之间的变换可使用函数 bs2rv，它支持格雷码和对数编码。

6.3.2 适应度计算

适应度函数有：ranking，scaling。

适应度函数用于转换目标函数值，给每一个个体一个非负的价值数。这个工具箱支持 Goldberg 的偏移法(offsetting)和比率法以及贝克的线性评估算法。另外，ranking 函数支持非线性评估。

6.3.3 选择函数

选择函数有：reins，rws，select，sus。

这些函数根据个体的适应度大小在已知种群中选择一定数量的个体，对它的索引返回一个列向量。现在最合适的是轮盘赌选择(即 rws 函数)和随机遍历抽样(即 sus 函数)。高级入口函数 select 为选择程序，特别为多种群的使用提供了一个方便的接口界面。在这种情况下，代沟是必须的，这就是整个种群在每一代中没有被完全复制，reins 能使用均匀的随机数或基于适应度的重新插入。

6.3.4 交叉算子

交叉算子函数有：recdis，recint，reclin，recmut，recombin，xovdp，xovdprs，xovmp，xovsh，xovshrs，xovsp，xovsprs。

交叉是通过给定的概率重组一对个体产生后代。单点交叉、两点交叉和洗牌交叉是由 xovsp、xovdp、xovsh 函数分别完成的。缩小代理交叉函数分别是：xovdprs、xovshrs 和 xovsprs。通用的多点交叉函数是 xovmp，它提供均匀交换的支持。为支持染色体实值表示，离散的、中间的和线性重组分别由函数 recdis、recint、reclin 完成。函数 recmut 提供具有突变特征的线性重组。函数 recombin 是一高级入口函数，对所有交叉操作提供多子群支持入口。

6.3.5 变异算子

变异算子函数有：mut，mutate，mutbga。

二进制和整数变异操作由 mut 完成。实值的变异使用育种机函数 mutbga 是有效的。Mutate 对变异操作提供一个高级接口。

6.3.6 多子群支持

多子群支持函数：migrate。

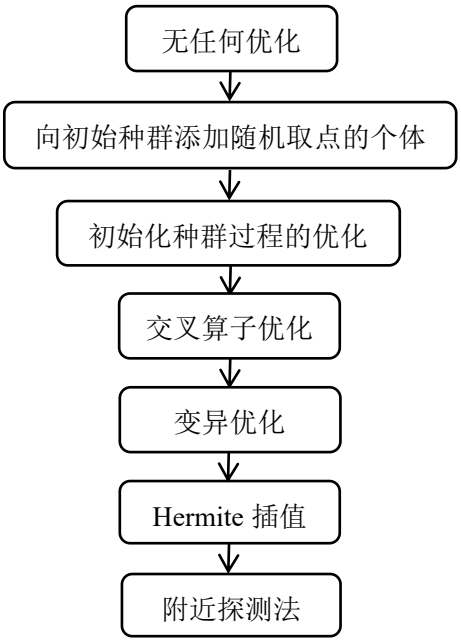
遗传算法工具箱通过高层遗传操作函数 migrate 对多子群提供支持，它的一个功能是在子群中交换个体。一个单一种群通过使用工具箱中函数修改数据结构，使其分为许多子种群，这些子种群被保存在连续的数据单元块中。高层函数如 select 和 reins 可独立地操作子种群，包含在一个数据结构中的每一子种群允许独自向前衍化。基于孤岛或回迁模式，migrate 允许个体在子种群中迁移。

7. 基于遗传算法的优化

7.0 实验参数表及优化流程

实验参数名称	意义
LIND	染色体长度
NIND	种群中染色体（个体）数量
GGAP	代沟即自然选择中存活率
MAX_GEN	最大遗传代数
K	函数 mycrtbp2 初始化种群规定的每个个体取样数
RecOpt	交叉概率
Recmod	交叉算子
MutOpt	变异概率
MAX_MUT	变异概率上限

实验参数名称	意义
MIN_MUT	变异概率下限
MAX_PER	初始化选择基因的概率上限
MIN_PER	初始化选择基因的概率下限
maxRepTime	最大容忍最优解重复次数
MIN_NUM	每个样本的测定点数，低于此值将不能拟合函数



7.1 没有任何优化的情况

使用 gatbx 默认函数参数如下（output1.1,1.2,1.3）拟合使用三次样条插值

Output	最优解	成本
1.1	1 10 23 30 41 49	98.7945
1.2	1 10 19 24 34 44 51	98.87875
1.3	4 10 20 25 30 41 49	99.3

7.2 优化一：向初始种群添加随机取点的个体

相对于第一次默认的情况，经过多次测试，我们发现实际上可能的最优解取的测试点不会很多，于是通过向初始种群中添加 10 个只随机取 5 个采样点的个体，来进行优化。（output2.1,2.2,2.3）使用三次样条插值可以发现最终的得到的最优解成本比默认的情况下降一点，但是却早早收敛，经过分析应该是初始化之后种群早熟，收敛于局部最优。

Output	最优解	成本
2.1	1 9 19 28 35 43 51	97.3342
2.2	1 10 18 26 33 44 51	97.6375
2.3	1 10 23 30 41 49	98.7945

7.3 优化二：初始化种群过程的优化

出现“早熟”现象的原因可能是初始化种群不够随机，于是我们认为可以进一步更加随机化来产生初始化种群。为了避免早熟，又要能收敛到更好地结果，在初始化种群时自定义了初始化方法 mycrtbp，通过设定初始化选点概率的上下限，达到避免“早熟”的效果。

Output	概率上限	概率下限	最优解	成本
3.1	0.1	0.03	1 11 20 27 35 44 51	97.3512
3.2	0.2	0.03	1 8 20 27 32 44 51	97.4857
3.3	0.4	0.03	1 10 20 30 41 49	96.989
3.4	0.6	0.03	2 10 19 30 40 49	97.52075
3.5	0.8	0.03	4 10 20 30 41 49	97.062
3.6	1	0.03	1 9 20 28 35 45 51	96.9557
3.7	0.4	0.1	4 10 23 30 41 49	97.896
3.8	1	0.1	1 10 19 24 34 44 49	98.636

7.4 优化三：初始化方法的再优化

我们试着尝试另一种初始化方法，因为前面知道实际上可能的最优解取的测试点不会很多，这次初始化的时候直接规定每个个体的取样数 K 一定。 首先几次测试最大遗传代数仍为 50，为了先看看新的初始化方法 mycrtbp2 的效果。

这里，我们改变取样数 K 以观察成本的变化。我们发现随着 K 的增大成本的大小基本保持不变，甚至还有上升。

Output	取样数 K	最优解	成本
4.1	4	4 10 20 30 41 49	97.062
4.2	5	4 10 20 30 41 50	97.1675
4.3	6	1 10 20 27 36 43 51	98.484
4.4	7	1 9 18 26 31 44 51	98.3937

7.5 优化四：交叉算子的优化

在 4 中发现收敛太快，之前染色体重组的时候都是单点交叉 xovsp，尝试两点交叉 xovdp，和通常多点交叉 xovmp。这里，我们控制取样数 K=5 保持不变。

我们很容易发现，较单点交叉而言，两点交叉和多点交叉在结果上要更优些，但跨度并不是很大。

Output	交叉	最优解	成本
4.2	单点交叉 xovsp	3 10 20 30 41 50	96.19025
5.1	两点交叉 xovdp	3 10 21 30 40 49	95.503
5.2	多点交叉 xovmp	2 10 20 30 40 49	96.01275

7.6 优化五：变异的优化

之前的 4-5 用的都是 mycrtbp2 初始化种群，为了测试新增加的避免早熟的逐渐收敛的算法，在此我们用到 crtbp 初始化种群。

另外这次使用自定义变异函数 mymutate，如果前两次的最优解相同就提高变异率

Output	种群中染色体个数 NIND	最优解	成本
6.1	80	4 10 20 30 41 49	97.062
6.2	100	2 10 21 30 40 49	95.486

7.7 优化六：插值方法的探索

在经过反复的尝试之后，我们发现 95 左右是我们很难逾越的瓶颈，但是在我们的能力范围内再也不能有更好的拟合，于是我们决定更换插值方式，采用 crtbp 初始化种群分段三次 Hermite 插值（output7.1）。

令人惊喜的是最优解 cost 达到了 90 以下。注意到最后虽然收敛了但还可能有更优解，所以让突变概率增大试一试(output7.2)。

发现结果是后来很难再有更小的成本，分析原因是后期变异概率太高，脱离原基因趋势，不能再原最优解附近寻找更优解。于是接着尝试将变异概率上限降低，有一定效果（output7.3)。

Output	变异概率	概率上限	概率下限	最优解	成本
7.1	0.02	0.6	0.02	5 17 26 36 48	86.012
7.2	0.1	0.6	0.1	4 17 26 35 49	86.007
7.3	0.02	0.05	0.02	5 17 25 36 49	85.98725

7.8 优化七：更深一步地探索

在上述的数据中我们发现并没有更优的解产生，分析可能是代数太高之后基因多样性缺失，在最开始几代因为总体取点过多随着个体淘汰了大量的基因。所以我们结合现在的最优解取点为 5 个，将这次初始化方法改为 mycrtbp2，初始化种群中每个个体取 K 个点。

第一次尝试以失败告终，收敛太早。于是我们提高变异概率。

第二次尝试同样出现了麻烦，从第 13 代时每个个体的成本可以看出断层，说明变异过度，初始化基因多样性不够，前十个基本一样。

最终我们做出了如下的改进取得了一定的效果：增加初始化样本数，增加遗传代数，使用默认变异函数固定变异概率，减小代沟，改变交叉算子为单交叉，不顾早熟。

Output	最优解	成本
8.1	4 17 26 36 48	85.7215
8.2	4 17 26 36 49	85.69675

7.9 优化八：附近探测法

我们发现找到的各代最优解，测定点对的位置基本一致，或是左右偏差不大。于是，我们思考是否可以在得到一组局部最优解之后，对每个采样点的左右进行探测，以试出一个更优质的解。我们将这个方法定义为“附近探测法”。

附近探测法的实现过程是这样的，我们首先通过随机抽样的思想结合“具有早熟缺陷的遗传算法”找到若干组局部最优解，记录下这些局部最优解中的最优的解，然后我们对这个最优解的每个采样点左右探测两个位置，以期获得所谓的更优解。总之，这是一次值得一试

的尝试.

同样的，我们将得到的更优解再次使用“附近探测法”。

终于我们得到了最优解，如下表所示。

	最优解	成本
采用前	5 17 26 35 48	85.2205
采用后	4 17 26 35 48	84.915
再次采用	4 16 26 35 48	84.74825

7. 实验结论

上述实验结果表明，使用优化后的遗传算法较之传统的遗传算法能够找到一个成本更低的传感特性校准方案。在多项式拟合、三次样条插值、Hermite 插值这三中插值方法中间，我们得出的结论是 Hermite 插值拟合的效果更为优秀。

在实验的过程中，我们虽然走了很多弯路，但最终得到的实验结果还是比较令人满意。

我们对于本课题的结论是，在使用 Hermite 插值函数的前提下，并综合利用附近探测发，使最低成本能够达到 84.74825。最优解为 4 16 26 35 48。

前提	最优解	成本
Hermite 插值，附近探测法	4 16 26 35 48	84.74825

8. 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]李书全，孙雪，孙德辉，边伟鹏 《遗传算法中的交叉算子的述评》天津财经大学，商学院 2012,48(1).

[3]田东平，迟洪钦 《混合遗传算法与模拟退火算法》 上海师范大学数理信息学院 2006，22.

[4]蒋腾旭，谢枫《遗传算法中防止早熟收敛的几种措施》九江职业大学计算机系 2006，12.

[5]http://wenku.baidu.com/link?url=awAUiChOkT16t1xRS1T2iDWSMcyzoZCfzPpUc2aw2jqse7Ctwa6_pv_p_MIKwr1UtpE5nP7UbM9khSUJ4IVAxGrJtPTLorbJEnhKD_obVOW《英国 Sheffield 遗传算法工具箱（gatbox）基本函数介绍与编程方法》.

[6]http://wenku.baidu.com/link?url=cg_YZPL-PfhDOta48C-FxVoOT_eSpKqfH_ppl4WzraxsW7oPBpwwN2nNWGmwzqeyceUuVHDdepEMbzdgY-KpdzI-ScaacQXiAxiyLSiYZlS《基于 Matlab 的遗传算法程序设计及优化问题求解》.

[7]http://baike.baidu.com/link?url=klqPcLIH9H-ODshm69ZZxNia-GQL-yfMsfS5uYELEsY4BSCWSZo_pLJ4v_cDPmjXubPeZWYPxo2Ox01jDqBAAa 百度文库-《遗传算法》.

[8]统计推断-第 21 组（刘长风）课程设计报告.pdf.

附录 1（gatbox 基本函数介绍）

1.函数 bs2rv

phen=bs2rv(chrom,fieldD)根据译码矩阵 fieldD 将二进制串矩阵 chorm 转换成实值向量，返回矩阵包含对应的种群表现型。

fieldD=[len;lb;ub;code;scale;lbin;ubin]

len：指明 chorm 中每个变量（基因）的二进制编码的长度，满足 sum(len)==size(chorm,2)

lb 和 ub：分别指明每个变量使用的上下边界
code：指明每个变量的编码方式，1 为标准二进制编码，0 为格雷编码
scale：指明每个变量的是否使用对数或算术刻度，1 是对数刻度，0 为算术刻度
lbin 和 ubin：指明表示范围总是否包含每个边界，1 表示包含，0 不包含
比如：fieldD=[5 6 9 7 %len
0 -5 -1 9 %lb
10 0 10 20 %ub
0 1 1 1 %code
0 1 0 1 %scale
1 1 1 0 %lbin
0 1 0 0] %ubin
表示 chrom 的每个个体（染色体）的编码规则为：
共有 4 个变量（即每个染色体有 4 个基因），
第 1 个基因体是使用 5 位标准二进制编码，上下边界为[0 10]，只包含上边界的算术刻度
第 2 个基因体是使用 6 位格雷二进制编码，上下边界为[-5 0]，包含上下边界的对数刻度

2.crtbase
basevec=crtbase(lind,base)产生向量的元素对应染色体每个基因的基数（每个变量的范围）
lind：每个染色体的基因(变量)个数，sum(lind)==变量个数，且 length(lind)=length(base)
base：对应每个基因的基数
比如：
>>basevec=crtbase([2 3 4 2],[10 20 15 8])
结果为:
>>basevec=[10 10 20 20 20 15 15 15 15 8 8]
上面表示创建有 2+3+4+2=11 个基因(变量)的染色体(个体)
前两的变量的取值为 0-9，接着三个 0-19，还有四个 0-14，最后两个 0-7
由上面可以看出 basevec 其实可使用 matlab 的其它简单命令创建，因此很少用到它，一般使用 rep。再强调一下，简单一点说 basevec 其实就是每个变量的基数（取值范围）

3.crtbp
[newchrom,newlind,newbasevec]=crtbp(nind,lind,basevec)创建离散随机初始种群，经常和 crtbase 一起使用
basevec：其实就是上面创建的 basevec 向量
nind：种群中染色体(个体)的数量
lind：变量的个数，即 lind=length(basevec)
注意在 basevec 给出时,lind 可以省略，但要是 basevec 没有给出，则 lind 必须给，此时默认创建取值为 0-1 的种群
比如：
>>basevec=crtbase([3,2],[8,5])
>>chrom1=crtbp(3,5) %创建一个有 3 个个体，5 个变量的初始种群
>>chrom2=crtbp(4,5,basevec) %4*5 的种群，等效 chrom2=crtbp(4, basevec)
结果为：
>>basevec=[8 8 8 5 5]
>>chrom1=[0 1 0 0 0
1 1 0 1 0
1 0 1 1 0]
>>chrom2=[6 5 7 0 4
7 1 3 4 2
3 7 1 0 4
6 6 4 4 3]

----------前三个函数一般只是在离散编码中使用

4.crtrp

chrom=crtrp(nind,fieldDR)创建由任意个体组成的实值随机原始种群

nind：种群中个体数量

fieldDR：是一个 2*nvar 的二维矩阵，表示每个变量的取值上下限

比如：

创建一个含有 3 个个体，4 个变量的随机种群

```
>>fieldDR=[-100  -50  -30  -20
            100    50   30  20];
>>chrom=crtrp(3,fieldDR)
>>chrom=[40.23  -17.17  28.00  15.38
          82.26    13.25  18.52    -9.09
         -90.20  -13.25    7.89    9.25]
```

5.migrate

[chrom,objv]=migrate(chrom,subpop,migopt,objv)完成当前种群的子种群间的迁移，并返回迁移后的种群

subpop：子种群的数量

migopt：migopt(1)个体迁移率，默认 0.2

migopt(2)迁移方式，0 为均匀迁移，1 为适应度的迁移，默认 0

migopt(3)迁移种群结构，0 为完全网状，1 为临近结构，2 为环状，默认 0

objv：为目标函数值列向量，当 migopt(2)=1 是必须给出

6.mutate

newchorm=mutate(mutfun,oldchrom,fieldDR,mutopt,sunbpop)执行种群变异

mutfun：变异函数

fieldDR：变量的取值范围,实值型时为 2*nvar 的二维矩阵，离散型时为 1*nvar 向量

mutopt：mutopt(1)变异率，实值型默认 1/nvar，离散型默认 0.7/lind

mutopt(2)压缩变异的范围，取值范围为[0 1]，默认 1(不可压缩)，注意当为离散型时不需要给出

7.ranking

finnv=ranking(objv,rfun,subpop)按照个体的目标值 objv 由小到大的顺序对它们进行排序，并返回一包含个体适应度值的 fitnv 的列向量

rfun：rfun(2)指定排序方法，0 为线性，1 为非线性，默认 0

rfun(1)指定压差，当 rfun(2)=0 时，取值为[1 2]，默认为 2，当 rfun(2)=1 时，取值为[1 length(objv)-2]

另外还可以 length(objv)=length(rfun)，此时它则包含对每一行的适应度值计算

至于 scaling 一般使用很少，这里就不介绍了

8.recombin

newchrom=recombine(recfun,oldchrom,recopt,subpop)执行基因重组

recfun：变异函数

recopt：重组交叉率

9.reins

[newchrom,newobjvch]=reins(chrom,selch,subpop,insopt,objvch,objvsel)完成插入子代到当前种群，用子代代替父代并返回结构种群，本函数属于选择函数，但是为了恢复种群数量，必须要有，它没有与 select 重复

oldchrom：父代种群

selch：子代种群

insopt：insopt(1)指明选择方法，0 为均匀选择，1 为基于适应度的选择

insopt(2)重插入的比率，在[0 1]，默认 1

objvch：是 chrom 的目标值（基于适应度的选择，该参数必须有）

objvsel：是 selch 中个体目标值（基于适应度的选择，该参数必须有）

10.select
selch=select(selfun,chrom,fitnv,ggap,subpop)从种群 chrom 中选择优良个体
selfun：选择函数
fitnv：列向量，chrom 的个体适应度值
ggap：代沟率，默认 1.0

附录 2（程序代码）

%mymain.m 文件

```
function mymain( )
%主函数版本 3
%这里是主体的遗传算法

%设定参数
LIND = 51;       %染色体长度
GGAP=0.9;       %代沟
NIND = 50;       %种群中染色体（个体）数量
MAX_GEN = 80;       %最大遗传代数
K = 6;       %函数 mycrtbp2 初始化种群规定的每个个体取样数
RecOpt = 0.7;       %当前交叉概率
RecMod ='xovsp';       %交叉算子，用于设定交叉函数 recombin 的方式
MutOpt = 0.02;       %当前变异概率
MAX_MUT = 0.1;       %变异概率上限，mymutate 函数的参数
MIN_MUT = 0.02;       %变异概率下限，mymutate 函数的参数
MAX_PER = 0.4;       %初始化选择基因的概率上限，mycrtbp 函数参数
MIN_PER = 0.03;       %初始化选择基因的概率下限，mycrtbp 函数参数
maxRepTime = MAX_GEN/2;       %最大容忍最优解重复次数
MIN_NUM = 3;       %每个样本的测定点数，低于此值将不能拟合函数
%比如三次多项式函数至少需要测定 4 个点，三次样条插值至少需要测定 2 个点,
%分段三次 Hermite 插值至少需要 3 个点

genCounter = 0;       %进化代数计数器
repTime = 0;       %最优解重复次数
bestChroms = [];       %存储所有代的最优个体
bestCosts = [];       %存储所有代的最优成本，与最优个体顺序对应

%文件输入输出
data =csvread('20150915dataform.csv');       %读入 csv 文件
fp = fopen('output.txt','wt');       %输出文件

tic;       %计时开始，用于命令行输出时间查看
t1=clock;       %计时开始的参数，用于文件输出计算时间

%以下是功能部分
%创建初始种群
%Chroms = mycrtbp2(NIND,LIND,K,MIN_NUM);
%自定义创建初始种群函数第二个版本，染色体总体为 Chorms，另有以下两个版本
%Chroms = crtbp(NIND, LIND);       %gatbx 中原创建初始种群函数
```

```
Chroms = mycrtbp(NIND,LIND,MAX_PER,MIN_PER,MIN_NUM);%自定义创建初始种群函数
%以下是为了使初始化的 0 尽量多一点，向初始种群中添加 10 个只随采样 5 个点的个体
%Chroms = crtbp(NIND-10, LIND);
%little = mycrtbp2(10,LIND,5);
%Chroms = cat(1,Chroms,little);

%遗传循环开始
while genCounter < MAX_GEN

if(repTime>maxRepTime)       %监测如果重复代数太多，重新初始化种群
        %重新初始化种群可以和最开始一样，有多种方法
        Chroms = crtbp(NIND, LIND);
        %Chroms = mycrtbp(NIND,LIND,MAX_PER,MIN_PER,MIN_NUM);    %另一种方法
        %重设相关参数
        repTime = 0;
        genCounter = 0;
        MutOpt = MIN_MUT;
end      %监测如果重复代数结束

%检测是否变异后子种群中个体都取了不小于 MIN_NUM 个点
for ii = 1:NIND
    while sum(Chroms(ii,:))<MIN_NUM
        Chroms(ii,randi(51)) = 1;%随机增加取样点，否则后期拟合不了
    end
end

%计算种群适应度部分
    objV = zeros(NIND,1);       %初始化目标值
    for ii = 1:NIND
        objV(ii) = cost(Chroms(ii,:),data);       %计算成本目标值
    end
fitnV = ranking(objV);       %适应度计算

%选择个体进行重组变异再插入
selCh = select('rws',Chroms,fitnV,GGAP);
%select 为 gatbx 函数，用于选择本代子群，'rws'代表方法为轮盘赌

selCh = recombin(RecMod,selCh,RecOpt);
%重组，方式由 RecMod 给出，可以选择单点交换、两点交换、多点交换等方式

%变异部分
[selCh,MutOpt,repTime]=mymutate(MAX_MUT,MIN_MUT,selCh,MutOpt,bestChroms,genCounter,MAX_GEN,repTime);
%自定义变异函数，此函数中更新最优解重复次数，并且根据前两次最优解是否重复改变
%当前的变异概率并返回
    %selCh = mutate('mut',selCh,NaN,MutOpt);       %gatbx 原变异函数
    for ii = 1:size(selCh,1)     %检测是否变异后子种群中个体都取了不小于 MIN_NUM
```

```
个点
        while sum(selCh(ii,:))<MIN_NUM
            selCh(ii,randi(51)) = 1;       %随机增加取样点，否则后期拟合不了
        end
end

%计算子群适应度部分
    selObjV = zeros(size(selCh,1),1);%初始化目标值
    for ii = 1:size(selCh,1);
        selObjV(ii) = cost(selCh(ii,:),data);%成本目标值
end

%将子群插入到原种群
[Chroms,objV] = reins(Chroms,selCh,1,1,objV,selObjV);
%完成插入子代到当前种群，用子代代替父代并返回结构种群
%插入时有原种群和子群的目标值作为参考

%选择新种群中最优个体
[val,index] = min(objV);
%val 为当代最优解的 cost 值，index 为其在种群的序号

%一个指数型的变异概率调整步骤，确保收敛
if (genCounter > MAX_GEN/2 && MutOpt > MIN_MUT)
        MutOpt = MutOpt*(1-exp(genCounter/MAX_GEN/2-1.3));
    end

%disp(Chroms(index,:));       %debug 用，命令行显示当代最优解的染色体组成
    disp(val);     %debug 用，命令行显示当代最优解的成本
fprintf(fp, '%d\t', val);       %向文本中输出当代最优解的成本

%完成主要操作，代数增加
genCounter = genCounter + 1;

%debug 用，此处设置断点可以每隔 10 代停下来一次查看相关参数值，并没有实际意义
    if (rem(genCounter,10)==0)
        genCounter=genCounter;
end

%处理当代最优解
    bestChroms = cat(1,bestChroms,Chroms(index,:));       %记录最优个体基因
    bestCosts = cat(1,bestCosts,val);       %记录最优个体成本

%最优解重复次数过多后，使最优解在附近变化
if (repTime>=maxRepTime/2)
        Chroms(index,:) = sway(Chroms(index,:));
end

%以下为记录和观察数据
    disp('代数：');
    fprintf(fp, '%s\t', '代数：');
    disp(genCounter);       %命令行显示遗传代数
```

```
    disp(MutOpt);            %命令行显示当前变异概率
fprintf(fp, '%d\n',genCounter); %文本中记录遗传代数

end%遗传循环结束

%处理当代最优解
[minCost,minGen] = min(bestCosts);      %分别为历代中最低成本和出现代数次序
bestChrom = bestChroms(minGen,:);
toc;      %计时结束，命令行将显示所用时间
t2=clock;      %计时结束的参数，用于文件输出计算时间
t12 = etime(t2,t1);     %计算所用时间

%显示记录最优解
disp('最优解：');
fprintf(fp, '%s', '最优解：');
disp(bestChrom);              %命令行显示最优解
fprintf(fp, '%g\t', bestChrom);      %文本记录最优解

%显示记录最优成本
disp('Cost：');
fprintf(fp, '%s', 'Cost：');
disp(minCost);   %命令行显示最优成本
fprintf(fp, '%g\n', minCost);      %文本记录最优成本

%文本记录时间
fprintf(fp, '%s\t', '时间');
fprintf(fp, '%g\n', t12);

%关闭文本
fclose(fp);

end%结束 mymain 函数
```

%mycrtbp.m 文件

```
%自定义种群生成函数
function [Chroms] = mycrtbp(NIND,LIND,MAX_PER,MIN_PER,MIN_NUM)
%版本 1，依据给定概率区间生成种群
Chroms = zeros(NIND, LIND);%初始化
for ii = 1:NIND
    per = rand()*(MAX_PER-MIN_PER)+MIN_PER;     %在最大最小值之间随机一个取
点概率
    for jj = 1:LIND
        if(Chroms(ii,jj-1) == 1)
            if (rand() < per/3)    %per/3 为了减少 1 的产生
                Chroms(ii,jj)=1;
            end
        else
            if (rand() < per)
                Chroms(ii,jj)=1;
            end
        end
```

```
    end
end
%让每条染色体选取基因个数不小于 MIN_NUM，否则不能拟合函数
for ii = 1:NIND
    while sum(Chroms(ii,:))<MIN_NUM
        Chroms(ii,randi(51)) = 1;
    end
end
end
```

%mycrtbp2.m 文件

```
function [Chroms] = mycrtbp2(NIND,LIND,K)
%版本 2，依据每个个体取点个数生成种群
Chroms = zeros(NIND, LIND);%初始化
for ii = 1:NIND
     index = unidrnd(51,1,randi([K-2,K+2]));
     %随机产生将要被变为 1 的基因位置，randi([K-2,K+2]产生 K-2 到 K+2 的随机整数
     %另一个版本如下
     %index = unidrnd(51,1,K);
     Chroms(ii,index) = 1;
end
%让每条染色体选取基因个数不小于 MIN_NUM，否则不能拟合函数
for ii = 1:NIND
    while sum(Chroms(ii,:))<MIN_NUM
        Chroms(ii,randi(51)) = 1;
    end
end
end
```

%mycrtbp3.m 文件

```
function [Chroms,NIND,LIND] = mycrtbp3()
%用于附近探测的种群生成函数，并返回新生成的种群大小
Chroms=[
4    10   20   30   41   49;
4    10   20   30   41   50;
4    12   23   30   41   49;
    ];     %此处填入已知较优取样点，不能太多以免生成种群过大
newChroms = [];
tmp = zeros(1,51);
for ii = 1:size(Chroms,1)
    tmp(1,Chroms(ii,:))=1;
    tmpC = sway2(tmp);      %使用第二种附近探测函数
    newChroms = cat(1,newChroms,tmp,tmpC);
    tmp = zeros(1,51);
end
Chroms = newChroms;
[NIND,LIND] = size(Chroms);
end
```

%mymutate.m 文件

```
function [selCh,MutOpt,repTime] =
mymutate(MAX_MUT,MIN_MUT,selCh,MutOpt,bestChroms,genCounter,MAX_GEN,repTi
```

```
me)
    %此函数中更新最优解重复次数
    %并且根据前两次最优解是否重复改变当前的变异概率并返回
    if(genCounter>1)
        judge                                                                    =
sum(bestChroms(size(bestChroms,1),:)==bestChroms(size(bestChroms,1)-1,:));
        %judge=51 表示两次的最佳染色体基因相同
        if (judge == 51)
            repTime = repTime+1;    %记录重复次数
            if (MutOpt<MAX_MUT && MutOpt>= MIN_MUT)
                MutOpt = MutOpt*(1.5-exp(genCounter/MAX_GEN-1)*0.5);
                %变异概率减小，步幅减小
            end
        else
            repTime = 0;      %清零重复次数
            MutOpt = MIN_MUT;   %重置变异概率
        end
    end

    selCh = mutate('mut',selCh,NaN,MutOpt);      %按照新的突变概率进行突变
end
```

%sway.m 文件

```
function [Chrom] = sway(Chrom)
%附近探测法，随机选取某个取样点偏离原位置
index_temp = logical(Chrom);    %逻辑化染色体
x=[1:51];
x_optimal=x(1,index_temp);    %选出取点位置
index = randi(size(x_optimal,2));      %随机决定一个将要改变的位置
if (rand()>0.5 && x_optimal(1,index) >2)
    x_optimal(1,index) = x_optimal(1,index)-1;      %向左移动一位
else if (x_optimal(1,index) <49)
        x_optimal(1,index) = x_optimal(1,index)+1;      %向右移动一位
    end
end

%按照新的取点位置产生二进制染色体
Chrom = zeros(1,51);
Chrom(1,x_optimal)=1;
end
```

%sway2.m 文件

```
function [Chrom] = sway2(Chrom)
%附近探测法第二版，使每个点均偏离原位置，返回多个偏离尝试组成的个体组
index_temp = logical(Chrom);      %逻辑化染色体
x=[1:51];
x_optimal=x(1,index_temp);            %记录选点位置
newOptimal = x_optimal;   %newOptimal 表示当前所选的点位置
%以下存储四种移动
ml1=[];
ml2=[];
mr1=[];
```

```
mr2=[];
for ii = 1:size(x_optimal,2)
    if (x_optimal(1,ii) >1)  %防止超出范围
        newOptimal(1,ii) = x_optimal(1,ii) -1;      %向左移动一位
        ml1=newOptimal;
        newOptimal = x_optimal;
        tmp = zeros(1,51);
        tmp(1,ml1) = 1;
        Chrom = cat(1,Chrom,tmp);
    end
    if (x_optimal(1,ii) >2)
        newOptimal(1,ii) = x_optimal(1,ii) -2;      %向左移动两位
        ml2=newOptimal;
        newOptimal = x_optimal;
        tmp = zeros(1,51);
        tmp(1,ml2) = 1;
        Chrom = cat(1,Chrom,tmp);
    end
    if (x_optimal(1,ii) <50)
        newOptimal(1,ii) = x_optimal(1,ii) +1;     %向右移动一位
        mr1=newOptimal;
        newOptimal = x_optimal;
        tmp = zeros(1,51);
        tmp(1,mr1) = 1;
        Chrom = cat(1,Chrom,tmp);
    end
    if (x_optimal(1,ii) <49)
        newOptimal(1,ii) = x_optimal(1,ii) +2;     %向右移动两位
        mr2=newOptimal;
        newOptimal = x_optimal;
        tmp = zeros(1,51);
        tmp(1,mr2) = 1;
        Chrom = cat(1,Chrom,tmp);
    end
end
end
```

%cost.m 文件

```
function [ avgCost ] = cost( chrom , data)
%此函数用于计算个体对应成本
%chrom 为 1x51 染色体，内含二值化基因序列，表示对应取值点是否被取
%data 为样本数据
index_temp = logical(chrom);%逻辑化染色体基因片段
x = data(1,:);         %选出所有测试点
y0 = data(2:2:800,:);    %选出样本真实值
y1 = zeros(400,51);       %初始化拟合值
%定标计算
x_optimal=x(:,index_temp);   %选出对应测试点值
x_n=size(x_optimal,2);       %计算测试点个数
y0_optimal=y0(:,index_temp);   %选出对应样本真实值
for jj=1:400
    y1(jj,:)=mycurvefitting2(x_optimal,y0_optimal(jj,:));
```

```
        %此为分段三次 Hermite 插值
        %下面是三次样条插值
        %y1(jj,:)=mycurvefitting1(x_optimal,y0_optimal(jj,:));
        %下面是多项式拟合通用函数，最后一个参数指定多项式次数
        %y1(jj,:) = polyfitn( x_optimal,y0_optimal(jj,:),7 );
end
%  成本计算
Q=12;%单点测定成本
errabs=abs(y0-y1);%拟合值与真实值差值的绝对值
%以下是单点定标误差分段计算
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);
%所有 cost 求和
s=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-l
e2_0)+12.*(le5_0-le3_0)+25.*g5_0;
%求平均值，即校准方案总成本
avgCost = sum(sum(s))/size(s,1)+Q*x_n;
%disp(avgCost);     %debug 用，显示每个个体成本
end
```

%mycurvefitting2.m 文件

```
function y1 = mycurvefitting2( x_premea,y0_premea )
% Using cubic pchip fitting.
%分段三次 Hermite 插值
x = [5.0:0.1:10.0];
y1=interp1(x_premea,y0_premea,x,'pchip');
end
```

%mycurvefitting1.m 文件

```
function y1 = mycurvefitting1( x_premea,y0_premea )
%三次样条插值
x=[5.0:0.1:10.0];
y1=interp1(x_premea,y0_premea,x,'spline');
end
```

%polyfitn.m 文件

```
function y1 = polyfitn( x_premea,y0_premea,n )
%n 次多项式拟合
x=[5.0:0.1:10.0];
p=polyfit(x_premea,y0_premea,n);
y1=polyval(p,x);
end
```

test_ur_answer.m

```
%%%%%%%  答案检验程序  2015-12-06 %%%%%%%%%
```

```
my_answer=[ 4     16    26    35    48 ];%把你的选点组合填写在此

%  标准样本原始数据读入
data=csvread('20150915dataform.csv');
[M,N]=size(data);
nsample=M/2; npoint=N;
x = data(1,:);
y0 = data(2:2:M,:);
y1 = zeros(nsample,npoint);

my_answer_gene=zeros(1,npoint);
my_answer_gene(my_answer)=1;
index_temp = logical(my_answer_gene);%逻辑化染色体基因片段

%  定标计算
x_optimal=x(:,index_temp);
x_n=size(x_optimal,2);
y0_optimal=y0(:,index_temp);
for jj=1:400
      %%%%%%记得修改对应的最小取点数 MIN_NUM%%%%%%%%
      %n 次多项式函数至少需要测定 n+1 个点，三次样条插值至少需要测定 2 个点,
      %分段三次 Hermite 插值至少需要 3 个点
      y1(jj,:)=mycurvefitting2(x_optimal,y0_optimal(jj,:));
      %上面是 Hermite 插值函数
      %下面是三次样条插值函数
      %y1(jj,:)=mycurvefitting1(x_optimal,y0_optimal(jj,:));
      %下面是多项式拟合，最后一个参数是多项式次数
      %y1(jj,:) = polyfitn( x_optimal,y0_optimal(jj,:),6 );
end
%  成本计算
Q=12;%单点测定成本
errabs=abs(y0-y1);%拟合值与真实值差值的绝对值
%以下是单点定标误差分段计算
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);
%所有 cost 求和
s=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-l
e2_0)+12.*(le5_0-le3_0)+25.*g5_0;
%求平均值，即校准方案总成本
avgCost = sum(sum(s))/size(s,1)+Q*x_n;
%disp(avgCost);%debug 用
%  显示结果
fprintf('\n 经计算，你的答案对应的总体成本为%5.2f\n',avgCost);
```