参考他人报告或代码的申明

统计推断课程，2015 年秋季学期第 09 组，成员刘栋 学号 5140309415，卢冰聪 学号 5140309406，在报告编写过程中，以下方面参考了往届报告，现列表说明：

主要参考项目	说明
代码方面	《统计推断在数模模数转换中的应用》，何晗晓，2014 年秋季学期，组号 53 参考了该组的算法思想，并做了改进。

除了以上注明的参考内容，和报告中列出的引用文献注解，本报告其他部分都不包含任何其他个人或集体已经发表或撰写过的作品成果。

统计推断在数模转换系统中的应用

组号：09　　姓名：刘栋(组长)　学号:5140309415　姓名：卢冰聪　学号:5140309406

摘要：本课题要求为某电子产品内部传感器模块的批量生产设计一种最优化的传感特性校准（定标工序）方案。实际工程中，若对每个样品的 51 组数据进行完全拟合，效率低成本高。因此选取少量数据点，通过三次样条插值法、三次多项式插值和三次 Hermite 插值法进行拟合，选点采用遗传算法和模拟退火算法，最终通过一系列的比较选取最合适的方法并用启发式搜索方法得到最优解。

关键词：三次样条插值法，三次 Hermite 插值法，三次多项式插值，遗传算法，模拟退火算法

Application of Statistical Inference in AD&DA Inverting System

Abstract: This project is aimed to find the optimized method of calibration of sensing characteristics ,which is the important part of the batch of production progress of the internal sensor module of some electronic product. In the engineering practice, it's inefficient and expensive to fitting all 51 sets of samples completely. As a result, we use cubic spline interpolation, cubic polynomials interpolation and cubic Hermite interpolation as homologous fitting method, and use genetic algorithm and simulated annealing algorithm to choose the best combination of pots. Eventually we can get optimized solution by comparing all elements and choosing best way with heuristic search algorithm.

Key words: cubic spline interpolation, cubic polynomials interpolation, cubic Hermite interpolation , genetic algorithm, simulated annealing algorithm

1. 引言

现代实际工程系统中，大都带有如本课题所研究的即将大量投入生产的电子产品的内部传感器模块一样的测量设备，这些设备在生产中都要经历定标和校准过程。而实际定标中常常会遇到特性材料的工作特性非线性变化以及材料的一致性不够好，不能用同一条近似曲线拟合的情况。定标时若有大量的数据，那么拟合出来的曲线准确率会比较高，但同时带来的高额的附加成本。因此通过测定尽量少的数据，拟合推断出本成最低的最有传感器定标特性就具有很重要的意义。通过前期我们对多个样本数据的曲线特性观察，发现

（1）曲线可大致分成三段，在每段中曲线变化较均匀，但中间一段的斜率比另外两端小。

（2）曲线形态近似但两两相异，每段中变化虽较均匀但并不是完全线性，会有一定弯曲。

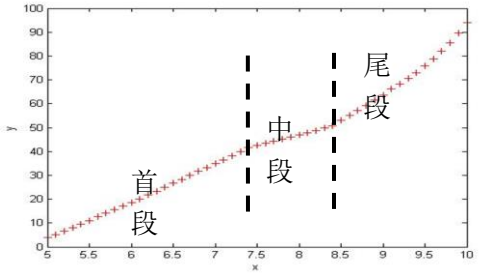

图 1-1　传感部件特性图示[1]

2. 基本求解过程

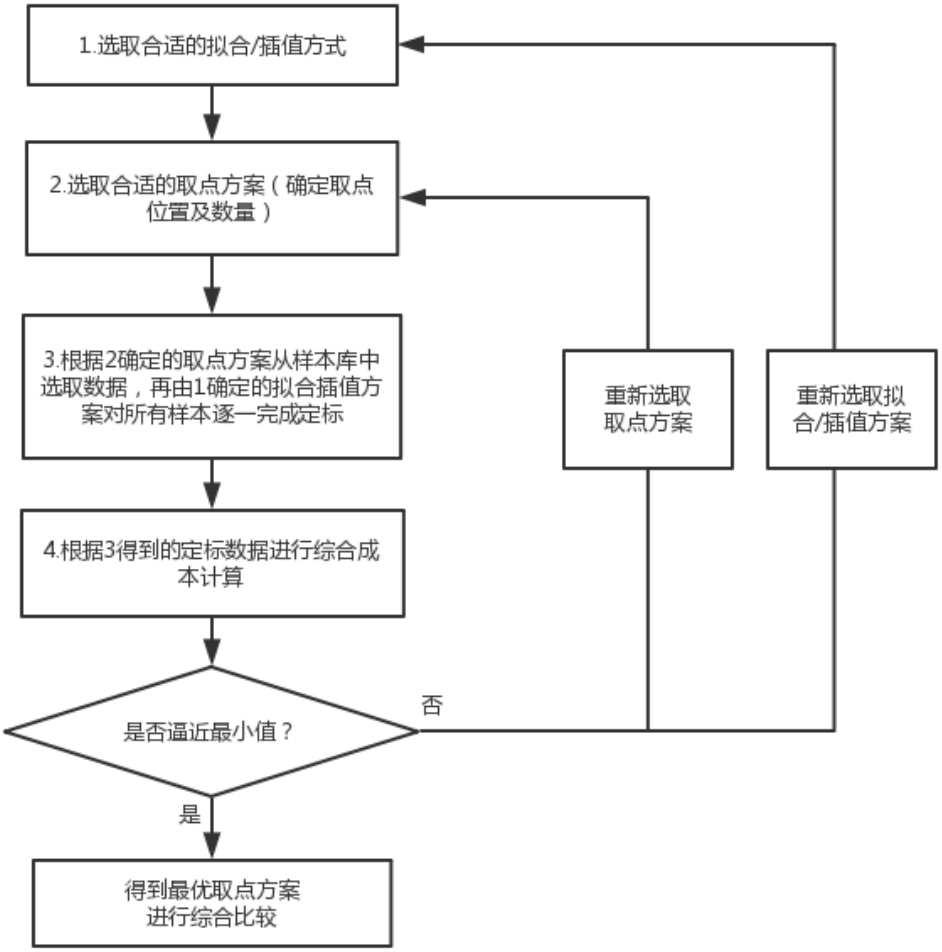

图 2-1　问题求解过程流程图

3. 拟合-插值方法介绍与分析

3.1 数据插值

3.1.1 插值定义[1]

插值通常有两种含义。一是指曲线必须通过若干已知离散数据点的一种拟合；二是指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。本课程中主要指第二种含义。

接下来我们将分析几种常用插值方法的优缺点。

3.1.2 最邻近插值[2]

在 MATLAB 中，对应参数 method 的选项为 nearest，实现方法是在已知数据点附近设置插值点，对插值点的数据进行四舍五入，超出范围的数据点返回 NaN。

我们对此方法得到的拟合结果与其他方法进行了比较，nearest 方法得到的曲线光滑性方面最差，是水平直线的连接，数据是不连续的，因此误差比较大，不利于得到最优解。

3.1.3 三次多项式插值[2]

在 MATLAB 中，对应参数 method 的选项为 cubic，通过设插值多项式函数

$$P_n(x)=a_0+a_1x+a_2x^2+\ldots+a_nx^n \qquad (3\text{-}1)$$

将各节点的函数值代入多项式里，得到等式，进而得到一个关于多项式里系数的线性方程组，解此线性方程组，便得到所要求的插值多项式。简单来说就是通过一条曲线不断逼近。对此方法得到的图像分析，其曲线的光滑性比较好。

3.1.4 三次样条插值[2]

在 MATLAB 中，对应参数 method 的选项为 spline。三次样条插值的原理是：对于 n+1 个给定点的数据集{x_i}，我们可以用 n 段三次多项式在数据点之间构建一个三次样条。如果

$$S(x)=\begin{cases}S_0(x), x\in[x_0,x_1]\\ S_1(x), x\in[x_1,x_2]\\ \ldots\\ S_{n-1}(x), x\in[x_{n-1},x_n]\end{cases} \qquad (3\text{-}2)$$

表示对函数 f 进行插值的样条函数，那么需要：

- 插值特性，$S(x_i)=f(x_i)$
- 样条相互连接，$S_{i-1}(x_i) = S_i(x_i)$，$i=1,\ldots,n-1$

两次连续可导，$S'_{i-1}(x_i) = S'_i(x_i)$以及 $S''_{i-1}(x_i) = S''_i(x_i)$, $i=1,\ldots,n-1$.

由于每个三次多项式需要四个条件才能确定曲线形状，所以对于组成 S 的 n 个三次多项式来说，所以需要 $4n$ 个条件才能确定这些多项式。由于其一阶二阶导数在节点处均连续，使得数据点之间的变化更加平缓，从而增加了真实性。并且可以样条插值可以利用低阶多项式来得到较小的插值误差，避免出现高阶多项式所出现的“龙格”现象。

对此方法得到的图像分析，其曲线的光滑性较好。

3.1.5 Hermite 插值[3]

埃尔米特插值是另一类插值问题，这类插值在给定的节点处，不但要求插值多项式的函数值与被插函数的函数值相同。同时还要求在节点处，插值多项式的一阶直至指定阶的导数值，也与被插函数的相应阶导数值相等，这样的插值称为 Hermite 插值。

Hermite 插值在不同的节点，提出的差值条件个数可以不同，若在某节点 x_i，要求插值函数多项式的函数值，一阶导数值，直至 m1-1 阶导数值均与被插函数的函数值相同及相应的导数值相等。我们称 x_i 为 m_i 重插值点节,因此，Hermite 插值应给出两组数，一组为插值点{x_i}n_i=0 节点，另一组为相应的重数标号{m_i}ni=0。

$$\sum_{i=0}^{n} m_i = N+1 \qquad (3\text{-}3)$$

若等式（3-3）成立，这就说明了给出的插值条件有 N+1 个，为了保证插值多项式的存在唯一性，这时的 Hermite 插值多项式应在 P_n 上求得，于是可作如下定义:

f 为 [a,b]上充分光滑函数，对给定的插值定节{x_i}$^n_{i=0}$，及相应的重数标号{m_i}$^n_{i=0}$，有等式（3-3）成立时，若有$H(x)\in P_n$满足

$$H^l(x_i) = f(x_i), l = 0,1,\ldots,m_{i-1}; i = 0,1,\ldots,n. \quad (3\text{-}4)$$

则称 $H(x)$ 为 $f(x)$ 关于节点$\{x_i\}_{i=0}^n$及重数标号$\{m_i\}_{i=0}^n$的 Hermite 插值多项式。

对此方法得到的图像分析，其曲线的光滑性较好。

以下是三种不同插值方式所得拟合图像。

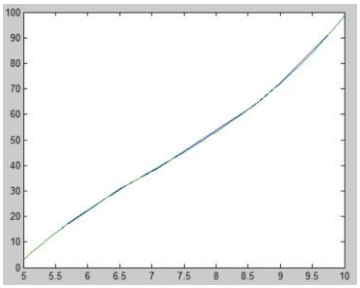
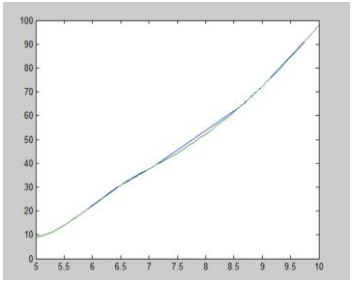
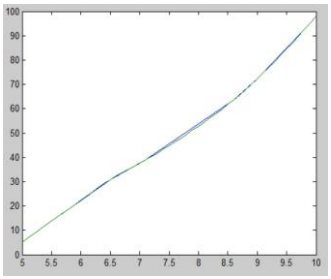

图 3-1 三次多项式插值拟合　图 3-2 三次样条插值拟合　图 3-3 三次 Hermite 插值拟合

3.2 曲线拟合

由于测量得到的原始数据带有一定的噪声测量数据，单纯根据插值方法来使用误差较大，因此可以结合曲线拟合的方法，寻求平滑曲线表现两个函数变量之间的关系和变化趋势。在 MATLAB 中曲线拟合方法使用最小方差函数来进行多项式拟合。

4. 遗传算法与模拟退火算法介绍与分析

4.1 遗传算法

4.1.1 定义[4]

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。

4.1.2 实现过程

按照遗传算法的实现过程，我们对本实验中遗传算法的应用做出如下设置：

（1）初始化：初步设置初始种群个体数为 100（计算过程中我们发现初始种群个数并不需要选取这么大），每个个体的基因长度为 51，用 0 和 1 两个状态分别代表不选取该点。

（2）选取适应度函数：适应度函数用于评价个体的优劣程度,适应度越大个体越好,反之适应度越小则个体越差。由于本算法需要找到成本的最优解，所以应该将成本函数的最小值变换到找取最大目标的函数值，所以应采用成本函数的倒数的平方作为适应度来进行计算。

（3）选择：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）交叉和变异：交叉运算是产生新个体的主要过程，我们的程序中选择了单点交叉法，首先用用随机数产生交配点的位置，然后两个个体在交配点位置互换部分基因码，形成两个子个体；变异运算是使用基本位进行基因突变，为了避免算法迭代后期出现种群过早收敛，对于二进制的基因码组成的个体种群，实行基因码小几率反转。本课题中需要固定变异点为最后一至两位，以防止变异过大导致最优解的偏离。

（5）终止：当达到判断条件后，即输出最优解，终止计算。

4.1.3 实现流程图

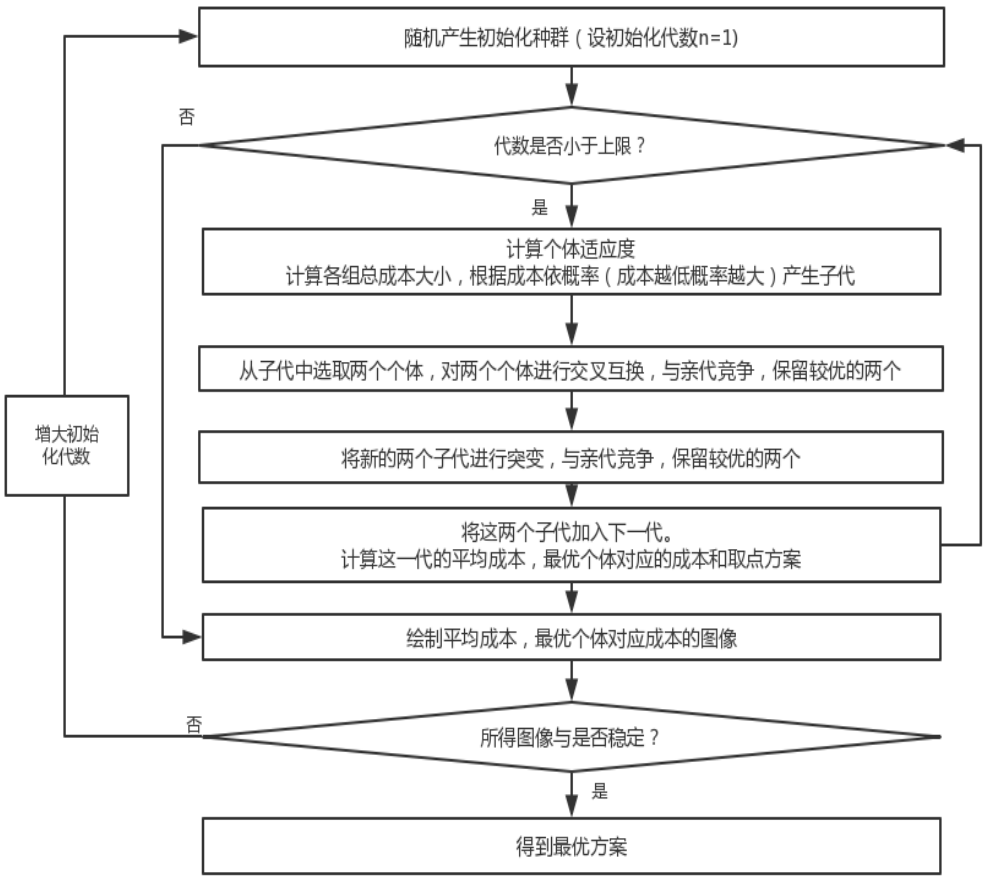

图 4-1　遗传算法求解流程图

4.1.4 特点分析

遗传算法区别于传统优化方法最明显的地方为遗传算法是从问题解的串集开始搜索而不是单个解，并且可以对搜索空间中的多个解进行评估。因此遗传算法覆盖面大，利于全局择优。并且遗传算法相比于传统优化方式，不需要对全部的数据点进行处理，因而大大降低了成本。

4.2 模拟退火算法

4.2.1 定义[5]

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。

4.2.2 实现流程

（1）初始化：仅设立一个个体，其基因长度为 51，每一个基因点是 0 或 1 两个状态代表选取该点进行拟合或不选取该点。

（2）由一个产生函数从当前解产生一个位于解空间的新解.

（3）计算与新解所对应的目标函数即定标成本函数的差。因为目标函数差仅由变换部分产生，所以目标函数差的计算最好按增量计算。

（4）判断新解是否被接受，判断的依据是一个接受准则，最常用的接受准则是 Metropolis 准则：若 $\Delta t'<0$ 则接受 S′作为新的当前解 S，否则以概率 $\exp(-\Delta t'/T)$ 接受 S′作为新的当前解 S。

（5）当新解被确定接受时，用新解代替当前解，这只需将当前解中对应于产生新解时的变换部分予以实现，同时修正目标函数值即可。此时，当前解实现了一次迭代。可在此基础上开始下一轮试验。而当新解被判定为舍弃时，则在原当前解的基础上继续下一轮试验。

4.2.3 实现流程图

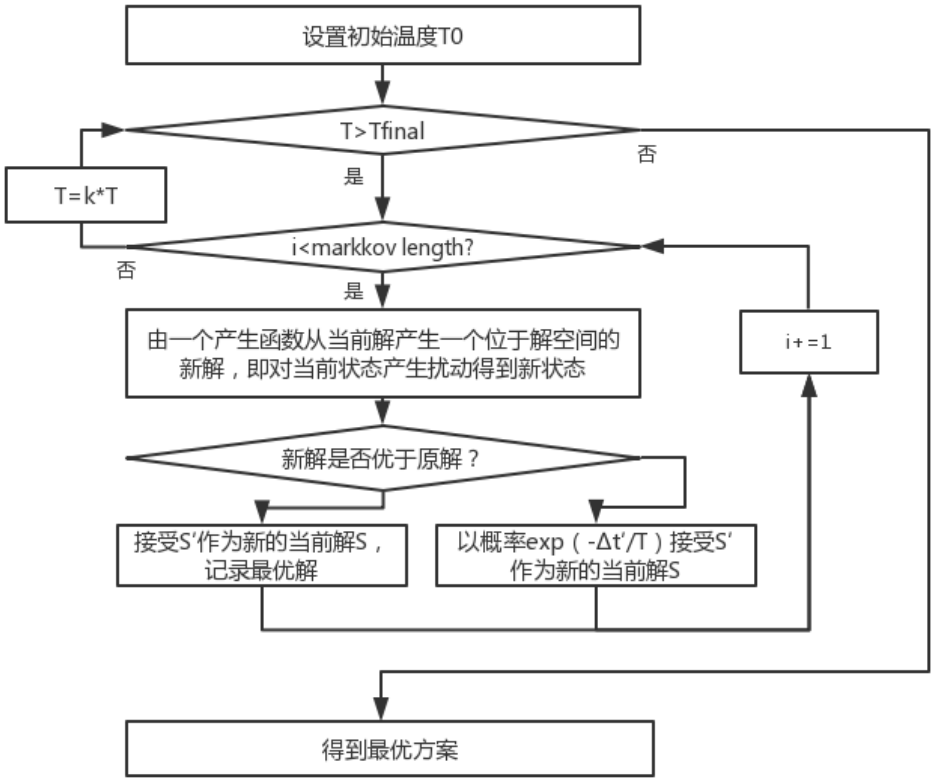

图 4-2　模拟退火算法求解流程图

4.2.4 特点分析

模拟退火算法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。[4]但其参数设置比较困难，可能会导致收敛速度慢、执行时间长的问题。

5. 结果分析

5.1 最终结果

在遗传算法结合三次 Hermite 插值法运算下，得到的最佳取点方案为[4,16,26,35,48]，最优成本为 84.7483。

5.2. 关于三种拟合方式的比较

在上文的介绍中，可知三次多项式插值，三次样条插值与三次 Hermite 插值得到的拟合曲线都比较光滑，但三种插值方式在最优成本以及收敛速度上有着明显差异。下面将就这两个方面加以比较。基于遗传算法（50 代），经比较前期的代码测试，得到三次多项式插值最优取点组合及最优成本。

5.2.1 总体比较结果

表 5-1 三种插值方法最终结果比较

插值方式	取点方案	最优成本	得到最优成本时间

三次 Hermite 插值	[4, 16, 25, 35, 48]	85.1535	113.309s
三次样条插值	[3, 12, 22, 31, 41, 50]	95.2242	74.519s
三次多项式插值	[4, 15, 23, 29, 38, 49]	86.7765	96.131s

5.2.2 三种插值方式成本及运行时间比较

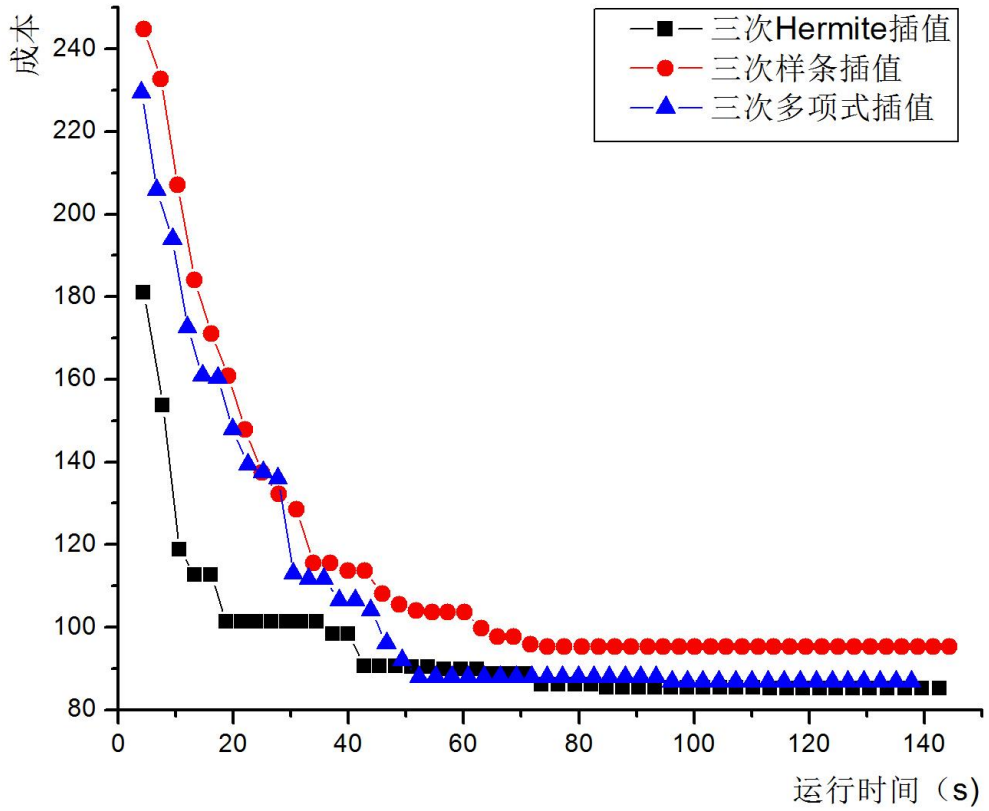

图 5-1　三种插值方法成本随运行时间变化关系

通过图例我们可以得到以下结论:

（1） 最优成本：三次 Hermite 插值<三次多项式插值<三次样条插值

（2） 收敛速度：三次 Hermite 插值<三次多项式插值<三次样条插值

结合以上两条结论以及本课题研究目的，我们认为三次 Hermite 插值优于另外两种插值方式，原因如下：

（1）三次 Hermite 插值得到的成本更低。考虑本课题的目的是找到最优化的传感特性校准（定标工序）方案，三次 Hermite 插值法得到的成本明显较低。

（2）三次 Hermite 插值法虽然收敛速度较慢，但仍在可接受的范围内。三种插值方法中三次样条插值收敛最快，但其成本结果很差，所以不能采用。想比于运行时间相近的三次多项式插值法，三次 Hermite 插值法的结果更优，因此采用三次 Hermite 插值法。

5.3. 两种算法的比较。

由于遗传算法与模拟退火算法相比于传统优化方式，都不需要对全部的数据点进行处理，因而大大降低了成本。但是两种算法有着明显的优劣，我们通过代码测试进行了比较。设置遗传算法 100 代、模拟退火算法降温次数 97 次。

5.3.1 总体结果

表 5-2 两种算法最终结果比较

算法种类	选点方案	最优成本	得到最优成本时间
遗传算法	[4, 16, 26, 35, 48]	84.7483	225.436
模拟退火算法	[4, 16, 25, 35, 48]	86.6145	416.600

5.3.2 两种算法成本与运行时间比较

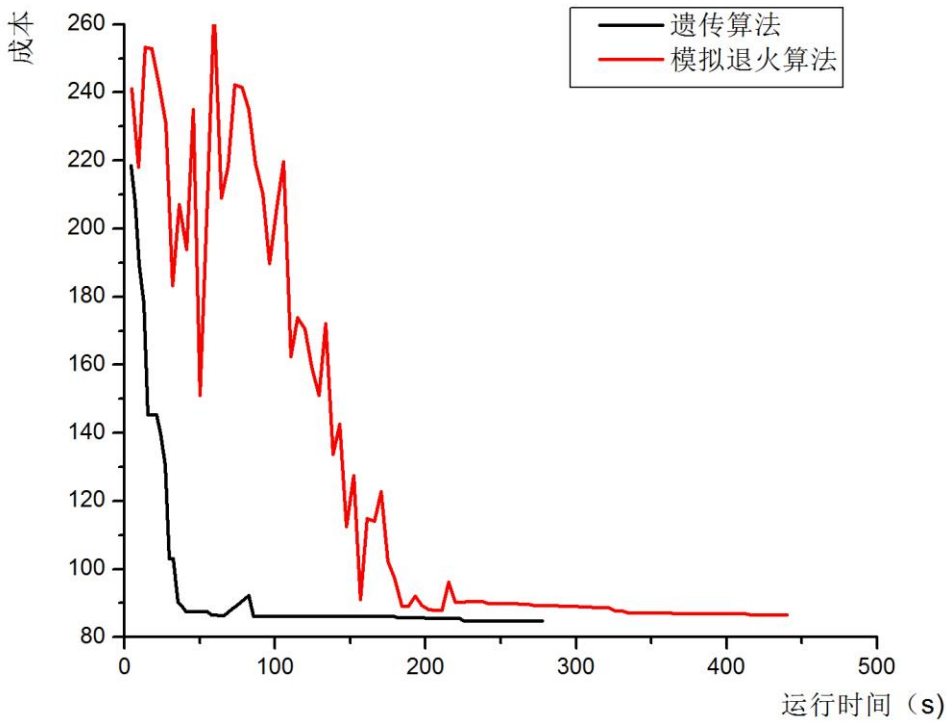

图 5-2　两种算法方法成本随运行时间变化关系

通过图例我们可以得到以下结论:

（1）遗传算法在较短的时间内达到了稳定状态，整体波动较小。但模拟退火算法在计算初期所得结果波动很大，经历了比较长的时间才达到稳定状态。

（2）遗传算法所得成本在不稳定段远远小于模拟退火算法所得成本，在稳定期两者差距逐渐缩小，但就遗传算法运行 100 代、模拟退火算法降温次数 97 次的情况下，遗传算法所得成本仍优于模拟退火算法。

（3）遗传算法最终在 277.509s 时完成运行，而模拟退火算法则在 440.033s 时才结束运行。

结合以上三条结论以及本课题研究目的，我们认为遗传算法总体优于模拟退火算法，原因如下：

（1）遗传算法更加高效稳定。由于模拟退火算法本身的性质，在运算初期必然出现如图所示剧烈的波动。因此在不稳定时期对两种算法成本的比较没有意义，但也明显发映出遗传算法有较快的收敛速度和较稳定的数据结果，有利于更快速地得到最优解。

（2）遗传算法能得到更优的解。在两个算法的计算过程中，我们发现最优解的出现并不是稳定的，而是有一定的偶然性。但在我们的多次测试结果中，遗传算法不仅得到了最优解 84.7483，而且在大多数情况下，遗传算法得到的结果要优于模拟退火算法。

（3）遗传算法的运行效率更高。在遗传算法运行 100 代、模拟退火算法降温次数 97 次这样小规模的测试情况下，遗传算法的运行时间约为模拟退火算法的一半，所以效率更高。

6. 拓展探究

6.1 算法初始参数的探究

经过实验比较，我们发现所给初始参数对于最后得到最优解的效果上有不小的影响。

6.1.1 遗传算法初始参数探究[6]

在遗传算法中，理论上会有：若采用太大的变异率会导致丢失最优解，而采取过小的变异率会导致算法过早的收敛于局部最优解；同时交换概率的选取也会对最后的效果有影响。于是，在保证初始种群大小及其他一些参数不变的情况下，我们选取了以下几个初值比较初始参数对算法的影响，每组参数进行三次实验来保证结果的准确性。

表 6-1 四组初始参数实验结果

交换概率	变异概率	实验 1	实验 2	实验 3
0.8	0.1	85.1535	84.7483	84.7483
0.7	0.1	86.8090	85.4860	84.9150
0.7	0.05	85.4860	86.6925	85.5960
0.8	0.05	85.1462	85.6310	85.1535

由上述结果看，交换概率以及变异概率对于最后的结果会产生一定的影响。在保证交换概率不变时，变异概率对实验结果会有影响，上述实验反映出了当变异概率为 0.1 时算法的效果较为优异，但同时我们又尝试更大的值与更小的值，发现结果都不尽如人意，可见变异概率的选取也较为重要；在保证变异概率不变时，交换概率也会有一定影响，从实验中看出 0.8 的交换概率较为优异，而采取比上述更小的交换概率则会导致算法效果大打折扣，可见交换概率的选择要稍微大些，这样才能保证种群的多样性。因此，我们可以得出交换概率选取 0.8，变异概率 0.1 选取可达到最好的效果。

从上述分析来看，对遗传算法进行合理的参数设置可以使解的质量有一定提高。

6.1.2 模拟退火算法初始参数探究

在模拟退火算法中初温的设置以及退火速度的设置对于都对于最后的效果有的影响

在试验中我们的初温设置选取 150，200，250；退火速度采用 0.1，0.5，0.9，所得结果如下：

表 6-2 五组初始参数实验结果

初温 T	退火速度 K	运行结果	运行时间
150	0.9	89.2033	430.773
200	0.9	85.1462	437.772
250	0.9	86.6145	416.600
250	0.5	88.3375	70.419
250	0.1	89.7978	23.040

由上述结果看出，初温的选取对于最后的结果有影响。初始温度过低，则容易陷入局部最优解；初始温度较高时，则较容易找到最优解，然而温度过高也不利于最后的求解，因此在采用模拟退火算法的时候要对初温进行多次尝试，选取效果最优的值，本实验中选取的初温为 200 多时则可以达到较好的效果。

当然，退火速度对于结果有明显的影响。一般而言，同一温度下“充分退火”很有必要，但考虑到算法的切实可行性，温度控制的参数 K 应选取接近 1 的较大值；否则 k 值过小时则会导致过早退火从而陷入局部最优解。从本实验来看，K 值选取 0.9 的时候较为合适，此时

较为容易得到最优解情况。

6.2 对遗传算法适应度函数的进一步探究

适应度函数对概率的选取上的不同对最后的收敛结果有一定的影响。适应度函数阶数越高则个体间的差异则会被更为放大，在此情况下收敛速度大大加快。然而，由于赌轮选择会使得下一代中的多样性丧失，从而更易导致局部最优解的出现，不利于产生最后的结果。

在适应度函数选择上，我们有直接取倒数、取平方的倒数以及取三次方的倒数这三种选择。我们通过实验得到了三种适应度函数下各自的运行结果及运行时间

表 6-3 三种适应度函数实验结果

方式	运行结果	运行时间
直接取到数	86.6187	220.118
取平方的倒数	84.7483	225.436
取三次方的倒数	85.4860	266.514

由上表可知，取平方的倒数在运行结果上明显优于另外两种方式，并且运行时间大大优于结果与之相近的取三次方的倒数的方法。从理论上分析，二阶可以适度放大个体间差异，使误差小的个体生存概率大大增加，又尽量避免了陷入局部最优解，在一定程度上保证了基因的多样性，因此是最理想的适应度函数。

7. 评价

对于本课题，我们认为我们完成了以下几方面比较有意义的探究：

（1）较好的达到了实验目的。我们通过横向比较三种插值方式以及两种算法的成本与收敛速度，找到了最适合本实验的解决方案，即采用遗传算法结合三次 Hermite 插值，最终得到了比较理想的成本。

（2）对于算法本身的探讨。遗传算法与模拟退火算法都是非常经典的计算模型，但对于不同的研究背景要对其进行合理的改进。于是我们在得到了最终结果的基础上对两种算法进行了更深入的探究，即在 6 中呈现出的对初始参数与适应度函数的探讨。这一部分也是我们对于统计推断在数模模数转换中的应用更深入的理解，用理论解决实际，用实际优化理论。

但由于时间有限以及对部分理论知识掌握有限，我们许多地方有待改进：

（1） 算法中仍有许多方面可以改进，比如可以采取不同的交换方式如均匀杂交等。

（2） 时间所限，还可以尝试其他一些启发式搜索方法进行比较。

（3） 同时在最后的探究过程中，采用了一般大量情况的结果，而更为科学的方法应该对其进行定量。

8. 参考文献

[1]袁焱. “统计推断”课程设计的要求 V2.2 2015-9-22

[2]周建兴、常春藤等. MATLAB 从入门到精通.人民邮电出版社

[3] 维基百科. 埃尔米特插值. https://zh.wikipedia.org/zh-cn/埃尔米特插值

[4]百度百科. 遗传算法_百度百科.
http://baike.baidu.com/link?url=UH-Cp9mdIcFGI9WRdSq5J_bpbw8zT2ljZuqxXLO1ZOWMdEtRx0JHQyog9Rk5MWp_10xvBot9r6582cacES5cbq

[5]百度百科. 模拟退火算法_百度百科.
http://baike.baidu.com/link?url=_9iZyLHUsw6VPAlYv6mbbKz_uv2Q1lr5asYNibAXBOxyLCFu02MrevYzBq-QzJ2x2Uv0mCp9R4cRQSjLj2zXcK

[6]王慧、丁海军、李峰磊. 改进遗传算法在函数优化问题中的应用.

附录

Matlab 程序

1、 遗传算法

main.m

```
clear all;
close all;
global data;
%popnum=100;                                    %每代种群大小
popnum=30;
Genelength = 51;                                %基因位数
Generationfinal=100;                            %进化最大代数
data=csvread('20150915dataform.csv');
population=zeros(popnum,51);
t1=clock;                                       %第一次记录时间
for i=1:popnum
    population(i,:)=RandomProduce();
end
[Fitvalue,accumPro]=fitness(population);

PointsBest=zeros(Generationfinal,Genelength);   %初始化矩阵--最优个体矩阵
CostBest=zeros(Generationfinal,1);              %初始化矩阵--最优个体的成本
AveCost=zeros(Generationfinal,1);               %初始化矩阵--所有群体的成本平均
值

AfterCross=zeros(popnum,Genelength);            %初始化矩阵--每次交换后矩阵
AfterMutation=zeros(popnum,Genelength);         %初始化矩阵--每次变异后矩阵

Generation=1;
ProCrossOver=0.8;                               %交换概率
while Generation<Generationfinal+1;
    judge=false;                                %判断标识
    disp(Generation);                           %显示当前代数
    [maxcurrent,postcurrent]=max(Fitvalue);
    Fitvalue(postcurrent)=0;
    [maxcurrent2,postcurrent2]=max(Fitvalue);

    select(1)=postcurrent;
    select(2)=postcurrent2;

    Fitvalue(postcurrent)=maxcurrent;
    for j=1:2:popnum
        if judge
            select=selection(population,accumPro);          %选择操
```

```
作
        end

[selectPoints,judge1,judge2]=CrossOver(population,select,ProCrossOver);     %交
换操作
    if judge1==0    %判断两个个体是否交换成功
        cost1=Fitvalue(select(1));
    else cost1=1./((costSum(selectPoints(1,:)).^2));
    end
    if judge2==0
        cost2=Fitvalue(select(2));
    else cost2=1./((costSum(selectPoints(2,:)).^2));
    end
    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),:);
            cost2=Fitvalue(select(2));
        else AfterCross(j+1,:)=selectPoints(2,:);
        end
        [IndividualA,judge3]=mutation(AfterCross(j,:));  %变异操作
        [IndividualB,judge4]=mutation(AfterCross(j+1,:));
        if judge3==1                            %判断两个个体是否变异成功
            tmp=1./((costSum(IndividualA)).^2);
        else tmp=cost1;
        end
        if cost1>=tmp
            AfterMutation(j,:)=AfterCross(j,:);
        else
            cost1=tmp;
            AfterMutation(j,:)=IndividualA;
        end
        if judge4==1
            tmp=1./((costSum(IndividualB)).^2);
        else tmp=cost2;
        end
        if cost2>=tmp
            AfterMutation(j+1,:)=AfterCross(j+1,:);
        else
            cost2=tmp;
            AfterMutation(j+1,:)=IndividualB;
```

```
        end
        Fitvalue(j)=cost1;
        Fitvalue(j+1)=cost2;
        judge=true;
    end
    k=0.1;   %变异概率
    ProCrossOver=ProCrossOver*k;
    population=AfterMutation;          %产生新种群
    SumAccumCost=sum(Fitvalue);                %计算新种群的积累概率
    ProEvery=Fitvalue/SumAccumCost;             %计算选择概率
    accumPro(1)=ProEvery(1);                     %计算累积概率
    for i=2:popnum
        accumPro(i)=accumPro(i-1)+ProEvery(i);
    end

    [value,pos]=max(Fitvalue);   %记录当前代最好的适应度和平均适应度

    CostBest(Generation)=1./((value).^0.5);
    disp(CostBest(Generation));
    AveCost(Generation)=mean(1./((Fitvalue).^0.5));

    bestselectpoint=population(pos,:);                %记录当前代的最佳染色体个
体
    PointsBest(Generation,:)=bestselectpoint;
    Generation=Generation+1;
    t2=clock;                    %第二次记录时间
    etime(t2,t1);
    disp(['循环运行时间:',num2str(etime(t2,t1))]);        %计算当前程序运行的时
间
end

[Bestvalue,minPos]=min(CostBest);        %最佳成本
Bestsolve=PointsBest(minPos,:);

%绘制平均适应度和最大适应度的曲线。
figure(1)
hand1=plot(1:Generationfinal,CostBest);
set(hand1,'linestyle','-','linewidth',1.8,'marker','*','markersize',6)
hold on;
hand2 = plot(1:Generationfinal,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 Generationfinal]);
legend('最小成本','平均成本');
```

```
box off; hold off;
```

fitness.m

```
function [Fitvalue,accumPro]= fitness(population) %计算适应度函数
popsize1=size(population,1);
Fitvalue=zeros(1,popsize1);
accumPro=zeros(1,popsize1);
for i=1:popsize1
    x=population(i,:);
    Fitvalue(i)=costSum(x);
    %disp(i);
end
Fitvalue=1./((Fitvalue).^2);          %转化成合适的值 平方
%Fitvalue=1./((Fitvalue).^3);            %转化成合适的值 三次方
%Fitvalue=1./((Fitvalue).^1);
summ=sum(Fitvalue);              %计算选择概率
Pperpopulation=Fitvalue/summ;
accumPro(1)=Pperpopulation(1);         %计算累积概率
for i=2:popsize1
    accumPro(i)=accumPro(i-1)+Pperpopulation(i);
    %disp(cumsump1(i));
end
Fitvalue=Fitvalue';
accumPro=accumPro';
 end
```

RandomProduce.m

```
function Individual= RandomProduce()
% 随机生成个体作为观测点
s=0;
while s<4
%while s<5
    Individual = round(rand(1,51));
    s= sum(Individual);
end

end
```

selection.m

```
function select= selection(population,accumPro)
%新种群选择操作
```

```
select=zeros(2);
for i=1:2
    r=rand;                 %产生随机数
newrand=accumPro-r;
j=1;
while newrand(j)<0
    j=j+1;
end
select(i)=j;              %选中个体的序号
end
end
```

CrossOver.m

```
function  [selectPoints,judge1,judge2]=CrossOver(population,select,possibility)
%新种群交叉互换操作
Genelength1=size(population,2);
judge1=1; %judge 判断用于是否交换
judge2=1;
selectPoints=zeros(2,51);
%单点杂交
while sum(selectPoints(1,:))<5 || sum(selectPoints(2,:))<5
    randomnum=round(rand*(Genelength1-2))+1;                              %在随机产
生一个交叉位
    selectPoints(1,:)=[population(select(1),1:randomnum)
population(select(2),1:51-randomnum)];
    selectPoints(2,:)=[population(select(1),randomnum+1:51)
population(select(2),51-randomnum+1:51)];
end

end
```

mutation.m

```
function [AfterMutation,judge] = mutation( submatrix)
%突变变异操作，judge 用于判断是否变异成功
Genelength=size(submatrix,2);
AfterMutation=zeros(1,51);
pmm=1;
judge=1;
if rand<0.7
    while sum(AfterMutation)<4
        randomnum=round(rand*(Genelength-1))+1;%在[1，Genelength]范围内随机产生
一个变异位
```

```
        submatrix(randomnum)=1-submatrix(randomnum);
        AfterMutation=submatrix;
    end
else
    while sum(AfterMutation)<4
        randomnum=round(rand(1,2)*(Genelength-1))+1; %随机产生一个变异位
        submatrix(randomnum)=1-submatrix(randomnum);
        AfterMutation=submatrix;
end

end
```

costOne.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');%三次样条插值法
q=interp1(x2,y2,x,'pchip');%Hermite 插值法
%q=interp1(x2,y2,x,'cubic'); %三次多项式插值
difference=abs(q-y);  %求其绝对值

sum=0;
for i=1:length(difference)
    if difference(i)>0.4 && difference(i)<=0.6
        sum=sum+0.1;
    end

    if difference(i)>0.6 && difference(i)<=0.8
        sum=sum+0.7;
    end

    if difference(i)>0.8 && difference(i)<=1
        sum=sum+0.9;
    end

    if difference(i)>1 && difference(i)<=2
        sum=sum+1.5;
    end

    if difference(i)>2 && difference(i)<=3
```

```
        sum=sum+6;
    end

    if difference(i)>3 && difference(i)<=5
        sum=sum+12;
    end

    if difference(i)>5
        sum=sum+25;
    end
end
cost=sum+12*length(pos);

end
```

costSum.m

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

2、 模拟退火算法

mian.m

```
k=0.9;   %退火速度
Tinitial=150;%初温
Tfinal=0.01; %末温
global T; global data;
T=Tinitial;
Markov_length=100;%马可夫链长度，任意温度 T 的迭代次数
data=csvread('20150915dataform.csv');
t1=clock;recordTime = zeros(Markov_length);%第一次记录时间

individual=RandomProduce();
```

```
targetreult=costSum(individual);
current=0;
while T>Tfinal  %终止条件
    current=current+1;
    disp(current);
    recordValue(current)=targetreult;
    AveValue1(current)=targetreult;
    recordMethod(current,:)=individual;
    for i=1:Markov_length
        individualNew=flux(individual);
        targetreultNew=costSum(individualNew);
        change=targetreultNew-targetreult;
        if change<=0    % 新解被接受
            individual=individualNew;
            targetreult=targetreultNew;
            if targetreult<recordValue(current)
                recordValue(current)=targetreult;
                recordMethod(current,:)=individual;
            end

        else if rand<exp(-change/T)  %按接受函数概率接受
                individual=individualNew;
                targetreult=targetreultNew;
            end
        end
    end
    disp(recordValue(current));

    T=k*T; %衰减函数
    t2=clock;%第二次记录时间
    etime(t2,t1);
    disp(['循环运行时间：',num2str(etime(t2,t1))]); %循环运行时间
end %重复扰动和接受过程

[BestValue,pos]=min(recordValue);
Bestsolve=recordMethod(pos,:);
Xaxis=0:current-1;
%Xaxis=recordTime;
figure(1)
hand1=plot(Xaxis,recordValue); %次数与成本关系
%hand1=plot(Xaxis,recordTime);%时间与成本
set(hand1,'linestyle','-','linewidth',1.8)
hold on;
xlabel('温度变化次数');ylabel('当前温度最小成本');
```

```
%xlabel('时间');ylabel('当前时间最小成本');
box off; hold off;
```

flux.m

```
function [ individualNew ] = flux(individual)
%产生随机扰动
global T;
individualNew=zeros(1,51);
while sum(individualNew)<5     %随机决定扰动
    if rand<0.5
        pos1=round(rand*50)+1;  %随机决定扰动位置
        individual(pos1)=1-individual(pos1);
        individualNew=individual;
    else if rand<0.9
            pos1=round(rand*50)+1;
            pos2=round(rand*50)+1;
            individual([pos1 pos2])=ones(1,2)-individual([pos1 pos2]);
            individualNew=individual;
        else
            pos=round(rand(1,3)*50)+1;
            individual(pos)=1-individual(pos);
            individualNew=individual;
        end
    end
end
end
```

RandomProduce.m

```
function  Individual= RandomProduce()
%随机生成个体作为观测点
s=0;
while s<4
%while s<5
Individual = round(rand(1,51));
s=sum(Individual);
end

end
```

costOne.m

```
function [ cost] = costOne( x,y,points )
```

```
%计算单个样本的成本
pos=find(points==1);
x2=x(pos);
y2=y(pos);

%q=interp1(x2,y2,x,'spline');%三次样条插值法
q=interp1(x2,y2,x,'pchip');%Hermite 插值法
%q=interp1(x2,y2,x,'cubic'); %三次多项式插值
difference=abs(q-y);

sum=0;
for i=1:length(difference)
    if difference(i)>0.4 && difference(i)<=0.6
        sum=sum+0.1;
    end

    if difference(i)>0.6 && difference(i)<=0.8
        sum=sum+0.7;
    end

    if difference(i)>0.8 && difference(i)<=1
        sum=sum+0.9;
    end

    if difference(i)>1 && difference(i)<=2
        sum=sum+1.5;
    end

    if difference(i)>2 && difference(i)<=3
        sum=sum+6;
    end

    if difference(i)>3 && difference(i)<=5
        sum=sum+12;
    end

    if difference(i)>5
        sum=sum+25;
    end
end
cost=sum+12*length(pos);

end
```

costSum.m

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

3、测试函数：

test_ur_answer.m

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

my_answer=[4,16,26,35,48 ];%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);
```

mycurvefitting.m

```
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
```