统计推断在数模转换系统中的应用

组号：02　　王一童 5140309426，李欲永 5140309408

摘要：数模转换中的校准定标问题的数学建模是工程上的难题之一。求解校准定标问题的方法之一是结合差值法以及遗传算法，求解问题的近似最优解。本科题求解的是给定数据模型的校准定标方法。我们首先分析了数据模型的特点，再通过对对比不同插值及搜索算法得出的解的时间效率以及定标成本，最终，设计出了在计算时间上相对较短，计算结果上成本较小的定标算法。

关键字：校准定标，数模转换，启发式搜索，遗传算法，插值

ABSTRACT: Calibration of Analog-to-digital convertors is one focus in the field of engineering. One of the solutions to the calibration problem is by using interpolation and Genetic Algorithm. The model of the problem we discussed has been preset. By analyzing the features of the model and comparing different ways of interpolation and search algorithm, we could finally carry out the method of calibration.

Key word: calibration, Analog-to-digital convention, Heuristics Search, Genetic Algorithm, interpolation

统计推断课程，2015年秋季学期第02组，成员王一童学号5140309426, 李欲永学号5140309408, 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》，孙煜，2014年秋季学期，组号50 在该组报告附录提供的程序代码基础上，进行了较多修改。
算法描述方面，包含流程图	《统计推断在数模转换系统中的应用》，张哲迪，2014年秋季学期，组号10 参考了该组报告的算法描述文字，引用了其流程图。
结果统计和分析	《统计推断在数模转换系统中的应用》，程政瑜，2014年秋季学期，组号41 部分参考了该组报告的统计方法和列表形式。

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。

本课题来源于工业中数模转换中的校准定标问题。

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。

1.1 实验模型与实验数据[1]

本科题中进行校准定标的实际问题模型是监测模块（如图1-1）。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）。

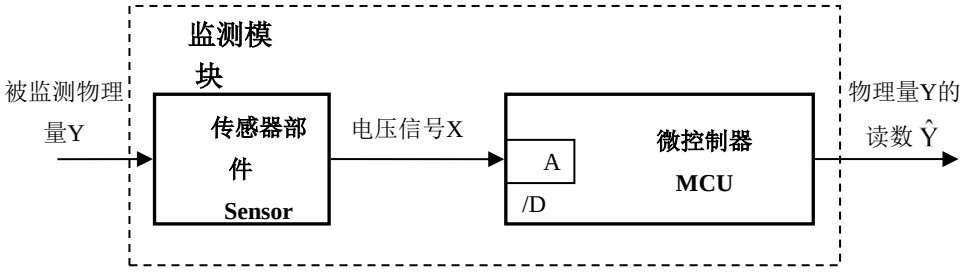

图1-1 检测模块组成框图[1]

校准定标，针对这个问题中，就是通过有限次测定，估计不同传感部件其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:

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

本课题的原始数据是400组离散的电压信号X及其对应的被监测物理量Y。其数据如图1-2所示。

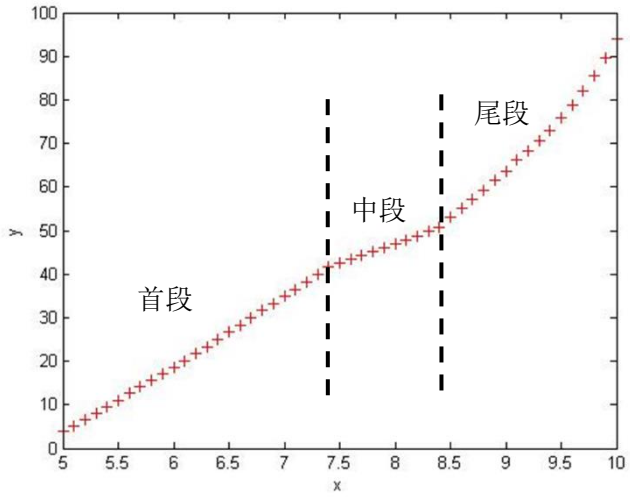

图 1-2 传感特性图示

每组数据都有以下主要特征：

(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。

为评估和比较校准方案，还特制定特定的成本计算规则，规则内容详见《“统计推断”课程设计的要求V2.2》[1]。

1.2 插值

要解决本课题问题的数学模型，首先要解决如何获得样本的估测函数$y = \hat{f}(x)$的问题。

推算估值函数$y = \hat{f}(x)$的已知量是由若干个花费一定测定成本测得的观测点。由有限个不同点做出近似替代函数的这类问题，在工程上，统称为插值问题，插值问题指的是：

在科研中的许多问题都可以用一个未知函数$y = f(x)$表示某两种变量的关系，研究这种关系是往往只能获得$y = f(x)$上有限个不同点x_i的值。由这些已知量做出一个近似替代$f(x)$的函数$\hat{f}(x)$来反映$f(x)$的特性的问题。[2]

插值方法多种多样，主要有代数插值，Newton插值，三次样条插值，Hermite插值。关于插值方法的详细内容可以参考《函数逼近论方法》（莫国端，科学出版社，2003）[2]。。

下面仅做简单介绍[2]：

(1)代数插值：当近似代替$f(x)$的函数$\hat{f}(x_i)$为代数多项式时就称这种插值方式为代数插值。对于一个已知n个点的差值问题，我们可以确定一个或者无穷个次数不超过n-1的多项式使得 $\hat{f}(x_i) = f(x_i)$。本课题中我们用三次多项式插值。

(2)Newton插值:基本公式为

$$N_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0) + \cdots + f[x_0, x_1, \ldots, x_n](x - x_0) \ldots (x_{n-1}) \quad (1\text{-}1)$$

$f[x_0, x_1, \ldots, x_n]$是f(x)关于基点$x_0, x_1, \ldots, x_n$的n阶商差。本课题牛顿插值我们引用的Matlab社区中提供的开源算法[3]。

(3)三次样条插值：对于已知节点的点集{xi},在每一段非边界小$[x_i, x_{i+1}]$上，我们用$\{x_{i-1}, x_i, x_{i+1}, x_{i+2}\}$四个点的拟合函数作为近似替代函数$\hat{f}(x)$的值，在边界区间上上，用相邻区间的函数的延拓作为近似替代函数$\hat{f}(x)$的值。

(4)Hermite插值：对于不仅要求插值函数和原来函数在节点上函数值相等，而且要求相切，对应的导数值也相等，甚至要求高阶导数也相等的插值方法，我们称之为Hermite插值。Hermite插值差值的特点是能反映插原函数在插值点处的变化趋势。

不同插值法的特性不同，插值法的选择直接决定了成本大小，所以找到最适应本课题问题的插值方法是十分重要的。

1.3 寻求解的方法（启发式搜索）[3]

启发式搜索就是在状态空间中的搜索对每一个搜索的位置进行评估，得到最好的位置，再从这个位置进行搜索直到目标。这样可以省略大量无谓的搜索路径，提高了效率。在启发式搜索中，对位置的估价是十分重要的。采用了不同的估价可以有不同的效果。我们先看看估价是如何表示的。

启发中的估价是用估价函数表示的，如：

最佳优先搜索的最广为人知的形式称为A*搜索(发音为“A星搜索”)。它把到达节点的耗散g(n)和从该节点到目标节点的消耗h(n)结合起来对节点进行评价：f(n)=g(n)+h(n)。

因为以g(n)给出了从起始节点到节点n的路径耗散，而h(n)是从节点n到目标节点的最低耗散路径的估计耗散值，因此f(n)=经过节点n的最低耗散解的估计耗散。这样，如果我们想要找到最低耗散解，首先尝试找到g(n)+h(n)值最小的节点是合理的。可以发现这个策略不只

是合理的：倘若启发函数h(n)满足一定的条件，A*搜索既是完备的也是最优的。

如果把A*搜索用于Tree-Search，它的最优性是能够直接分析的。在这种情况下，如果h(n)是一个可采纳启发式--也就是说，倘若h(n)从不会过高估计到达目标的耗散——A*算法是最优的。可采纳启发式天生是最优的，因为他们认为求解问题的耗散是低于实际耗散的。因为g(n)是到达节点n的确切耗散，我们得到一个直接的结论：f(n)永远不会高估经过节点n的解的实际耗散。

启发算法有：蚁群算法，遗传算法、模拟退火算法等。在此次实验中，我们选择了遗传算法来解决问题。遗传算法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。遗传算法是从代表问题可能潜在的 解集的一个种群开始的，而一个种群则由经过基因编码的一定数目的个体组成。每个个体 实际上是染色体带有特征的实体。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。初代种群产生之后，按照适者生存和优胜劣汰的原理，逐代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大小选择个体，并借助于自然遗传学的遗传算子进行组合交叉和变异，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，可以作为问题近似最优解。

优点：质量高，初值鲁棒性强，简单、通用、易实现。

缺点：(1)单一的遗传算法编码不能全面地将优化问题的约束表示出来。考虑约束的一个方法就是对不可行解采用阈值，这样，计算的时间必然增加。(2)遗传算法通常的效率比其他传统的优化方法低。(3)遗传算法容易过早收敛。(4)遗传算法对算法的精度、可行度、计算复杂性等方面，还没有有效的定量分析方法。

下面将阐述我们解决本课程问题的详细步骤与方法。

本课题要求解的寻找最优插值点的问题属于np-hard问题，运算时间随数据量几何爆炸，是无法用一般的枚举或者搜索算法计算的。但是可以遗传算法等现代启发式搜索算法寻找近似最优解[4]。在实验中，我们采用的就是遗传算法。

用遗传算法解决本课题的问题的基本思路是，以插值点的集合作为种群基因，以该插值点集合进行定标的定标成本为该插值点的适应度的参考，并设计适宜的的选择，交叉以及变异算法，通过这样的方式实现遗传算法，最终以遗传算法中种群中的最优个体为该问题的近似最优解。

需要设计出适合于本课题的遗传算法。首先要确定的是不同样本点的定标的成本，而定标成本，取决于插值方式的选择。

2.1 寻求最优化的插值法

对于确定的定标成本算法[1]，插值方法是唯一决定插值点集合和与定标成本值变量的关系的因素。即定标成本与插值点集于插值法满足函数关系：

$$p = f(A, S) \tag{2-1}$$

其中p表示定标成本，A是插值点集，S代表插值方法。以下我们将对提到过的代数插值，Newton插值，三次样条插值以及Hermite插值进行分析讨论。

分析与讨论将的主要方面有：(1)比较相同插值点集在不同插值方式下的定标成本；(2)举特例分析不同插值方式的残差大小及其分布的规律。用作测试的插值点集取自于我们初期测试的种群（见附录6.2），共包含20个点集，为不失一般性，前10个点集为六点集合，后10个位7点集合，完全随机选取。

(1)相同插值点集在不同插值方式下的定标成本

测试数据点的定标成本如图2-1所示。

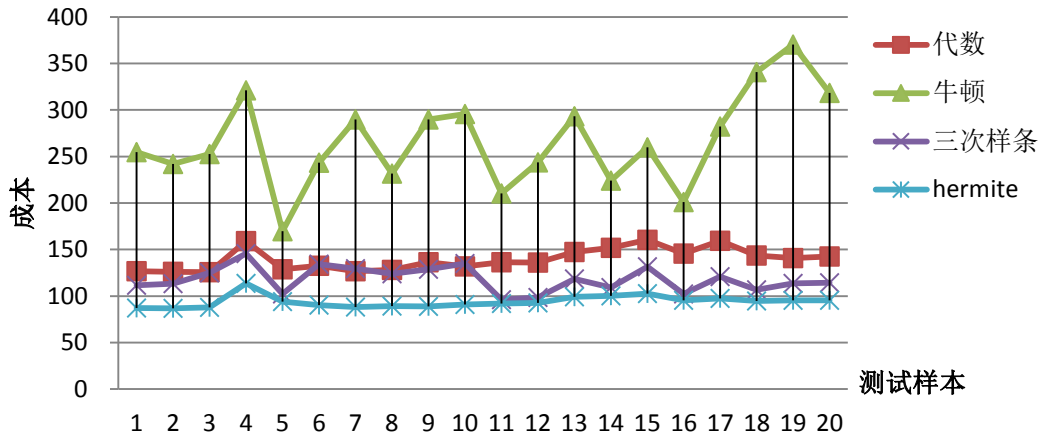

图2-1 不同插值法的成本

图2-1可以初步表明，在本课题中，Hermite插值相对其他插值方式具有较大优势。

(2)不同插值方式的残差大小及其分布的规律。

先定义第i个样本的第j个点的残差：

$$r_{i,\ j} = \hat{y}_{i,j} - y_{i,j} \tag{2-2}$$

其中$\hat{y}_{i,j}$表示要插值点的估计值，$y_{i,j}$表示要插值点的真实值。

我们以第一个测试点集，第一个样本为对象，分析出的不同插值方式的残差大小及其分布的规律如图2-2所示。

分析图2-2，可以得出：

(1)代数插值法可以在样本数据较为平滑的区域可以很好的拟合，但在数据中段样本的拐点处以及样本边界拟合效果很差。

(2)牛顿插值法能保证过每一个插值点，但在插值点之间的残差值较其他插值方法很大，尤其是在边界处的残差值。而且牛顿插值法运算复杂较高，不利于本课题大量数据处理。

(3)三次样条插值叶能保证过每一个插值点，可以部分避免代数插值在数据的拐点残差值过大的问题，但残差值在边界以及部分点两侧依然明显。

(4)Hermite插值较其他插值法优势明显，能较为平滑的通过插值点，产生的残差值不会出现其他插值法出现的大幅度震动的问题，并且在边界处的拟合效果最好。这很有可能是由Hermite插值法能模拟插值点处导数值的性质导致的。

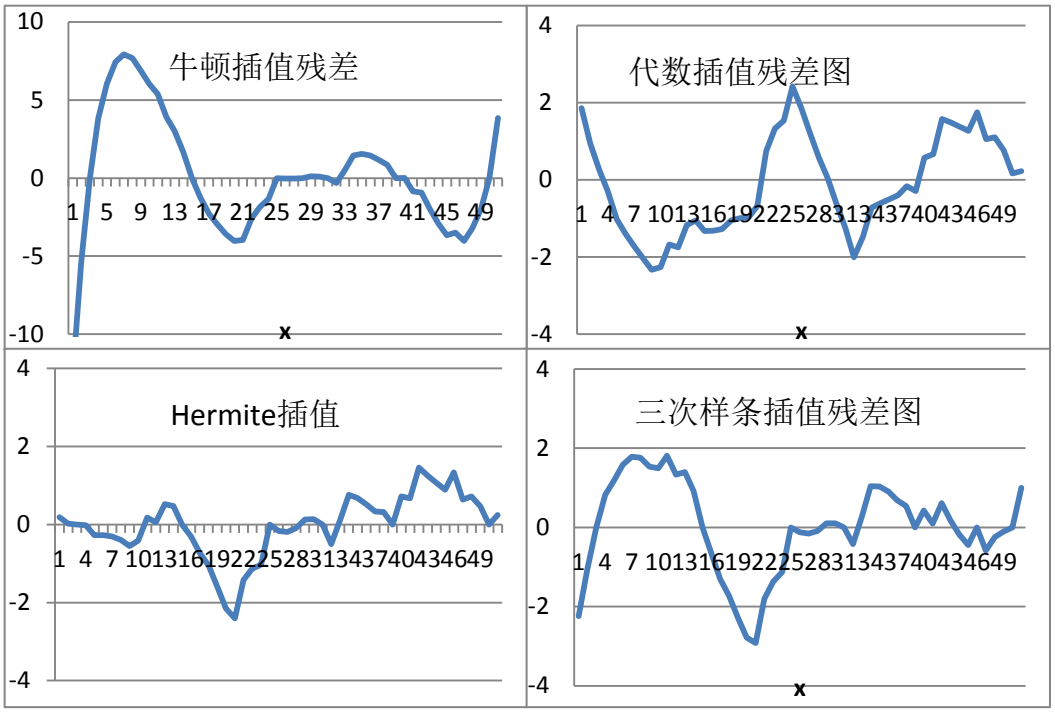

图2-2 不同插值法的残差分布

综上所述，通过实验验证，我们发现，对于本课题提供的样本数据较为适合Hermite插值法来进行插值分析。所以我们将使用Hermite插值法来计算定标成本。

在确定了定标成本之后，我们就可以进行下一步工作，遗传算法的设计与实现。

2.2 设计并实现遗传算法

2.2.1 算法实现[4][5]

(1)建初始状态:初始种群是从解中随机选择出来的，将这些解比喻为染色体或基因，该种群被称为第一代，这和符号人工智能系统的情况不一样，在那里，问题的初始状态已经给定了。

(2)评估适应度:对每一个解(染色体)指定一个适应度的值，根据问题求解的实际接近程度来指定(以便逼近求解问题的答案)。不要把这些“解”与问题的“答案”混为一谈，可以把它理解成为要得到答案，系统可能需要利用的那些特性。

(3)繁殖:繁殖(包括子代突变)带有较高适应度值的那些染色体更可能产生后代(后代产生后也将发生突变)。后代是父母的产物，他们由来自父母的基因结合而成，这个过程被称为“杂交”。

(4)下一代如果新的一代包含一个解，能产生一个充分接近或等于期望答案的输出，那么问题就已经解决了。如果情况并非如此，新的一代将重复他们父母所进行的繁衍过程，一代一代演化下去，直到达到期望的解为止。

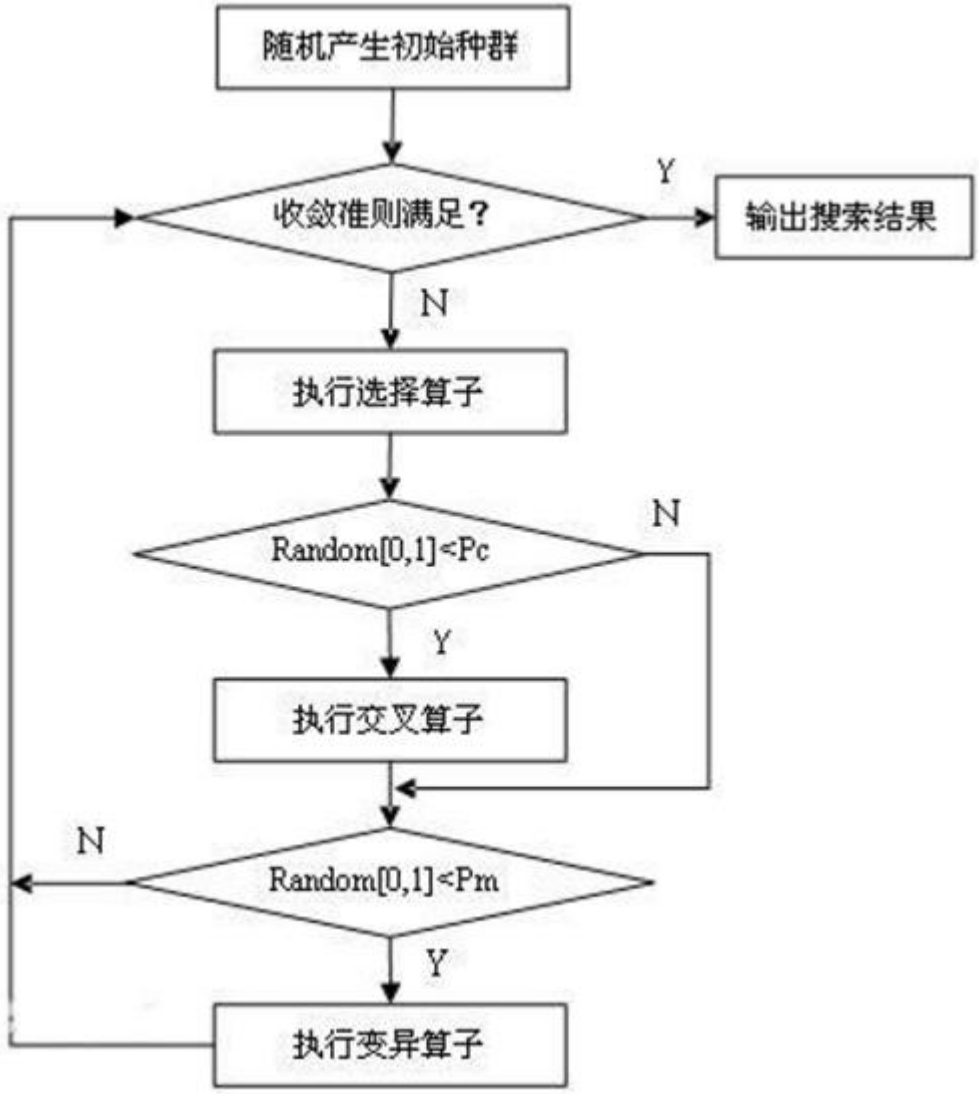

图2-3遗传算法流程图

2.2.2 算法详解

(1)初始化：设置进化代数计数器 g=1，设置最大进化代数 maxGeneration，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g=maxGeneration,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。

2.2.3 算法分析

(1)观察点数目的选取：

当观察点数为6时，10代以内迅速收敛，之后减缓，到50代，收敛到86.9820, 之后保持

稳定。

当观察点数为7时，收敛趋势稳定，最终收敛于94.1470，运行结果不如上一种方案。

当观察点数为5时，收敛曲线不稳定，虽然最终收敛到85.9345，但是我们权衡之后舍弃了这一方案。

表2-1 观察点数目

观察点数目	Min*	Avecost*	minScoreMethod*						
5	85.9345	99.3655	3	17	26	35	49		
6	86.9820	104.2859	4	15	24	31	40	50	
7	94.1470	107.4283	4	11	19	26	32	42	51

注：Min代表最低成本，avecost代表种群平均成本，minScoreMethod代表最低成本个体。

(2)选择方法的选取：最开始，我们尝试了精英选择。最开始尝试直接将最优解遗传到下一代而不进行繁殖，运行之后发现困在局部最优解。之后我们选择保留最优解，同时使其进行繁殖，但是操作之后发现，回归的效果并不是很好，所以最终我们还是采取了直接选择的方法，并不处理每一代的最优解，而是由其自然交叉，变异。

(3)变异方法：我们没有按照传统的先交叉再变异的方式。最初我们按照先交叉再变异的方式的方法，100代时结果收敛到87.2110，仍在收敛，但是极为缓慢。当我们换成先变异再交叉的方法，收敛速度明显加快。其具体原因，我们理解为先变异再交叉使得基因的多样性上升，所以更加快速的得出了最优解。针对变异，我们对于变异概率做了很多尝试，最开始由于担心结果困于局部最优解，跳不出，所以我们尝试将变异概率设为0.1，操作之后发现由于变异率过高，虽然20代左右就显现出了收敛的趋势，但是却不很稳定。当将变异概率设为0.05时，收敛较快，且更加稳定，50代左右即趋于稳定。最初我们的变异方法是对选出的点进行+1和-1的操作，后来发现这种操作对于样本改变量太小，还是无法解决困在局部最优解跳不出的问题，之后我们进行改进，采取正态分布的随机办法，即令randn函数的生成值添加到随机一点。这样增大了变异的效果，解决了之前的问题。

表2-2 变异率的影响

变异率	Min	avecost	minScoreMethod					
0．05	87.4192	92.1941	4	15	24	31	39	48
0．1	87.4850	99.8747	3	13	22	29	37	48
0．075	87.0295	96.7246	4	14	23	30	39	50

(4)交叉方法：交叉方法，最初我们只是选择单点交叉，运行之后发现，对于样本的影响不大，基因多样性并没有得到很大的扩展，所以最终收敛结果并不是十分令人满意。所以之后我们又换成了两点交叉，可是效果依然不明显。所以我们改用片段交叉，片段的大小也是完全随机得出，丰富了基因多样性，使得结果更好。同时，我们实验了多组交叉概率得出不同结果，如下表，由下表可得，当取0.8时，最小值最小，同时平均值最小，说明数据更加稳定。

表2-3 交叉概率的影响

交叉概率	Min	avecost	minScoreMethod					
0．7	88.9178	97.5221	4	12	20	27	37	49
0．8	86.8080	97.0950	5	14	22	29	37	48
0．9	86.9877	91.4550	5	15	24	31	38	49

综上所述，利用遗传算法，并使用Hermite插值法进行拟合后，可以找到六个点分别为{5,14,22,29,37,48}使得成本最低，为86.8080，满足评价条件。同时，增加一个点，也可以找到七个点分别为{4,11,19,26,32,42,51}使得成本为94.1470，明显高于前一个选点的成本。因此，六个点为宜。从中选择成本最低的点可以在工程上减少检测成本。

在不同繁衍代数情况下多次运行代码结果如表3-1所示。

表3-1 同繁衍代数情况下的实验结果

代数	Min	avecost	minScoreMethod					
1	89.8855	220.9400	4	16	26	29	40	49
10	89.1802	106.4158	2	12	22	28	37	47
20	88.8605	95.2112	2	13	21	28	36	50
50	86.9335	90.8168	3	15	24	31	38	49
100	86.8080	90.4550	5	14	22	29	37	48
代数	Min	avecost	minScoreMethod					
1	92.4498	240.1427	2	17	28	34	40	49
10	89.1927	104.1603	5	16	26	30	38	48
20	88.8560	101.4647	2	16	25	30	38	50
50	88.5660	97.2198	5	16	25	30	38	48
100	87.4832	94.6521	3	15	23	30	38	48

由上表可知，由于种群个数为100，使得末代种群最优个体存在很大的波动性，可以看出50代或者100代并不能对于20代有过多的优越性，这是由于种群规模较小，如果在时间花费允许的情况下，种群规模有必要进行更大的扩展，那么将会大大 减少出现成本为100以上的近似最优解，在此限于MATLAB运行时间过长，仅能通过这样的种群规模来大致反映遗传算法的特点及优化的必要性。综上所述，在种群规模仅为100个个体，繁衍代数仅有100代的情况下，通过以上方法已经可以得出近似的最优解。

[1] 袁焱. “统计推断”课程设计的要求V2.2

[2] 莫国端 函数逼近论方法 科学出版社,2003

[3] Muhammad Rafiullah Arain (12 Apr 2005) *Newton's Interpolation* Matlab Central File Exchange [Online] Available: http://cn.mathworks.com/matlabcentral/fileexchange/7405-newton-s-interpolation?s_tid=srchtitle

[4] Michalewicz Zbigniew, Fogel David B,曹宏庆,李艳,董红斌 如何求解问题 现代启发式方法 中国水利水电出版社,2003

[5] 邓方安,周涛,徐扬 软计算方法理论及应用=Theory and application of soft computing method 科学出版社,2008

在此感谢袁焱老师的指导，以及开设统计推断这门非常有趣的课程。

6.1 遗传算法源码

6.1.1 main.m

```
clear;


global pop;                    %取观察点种群
global fitnessTable;       %成本列表
global voltageX;              %样本电压X列表
global outputY;             %样本物理量Y列表
```

```
global len;                    %样本组数量

popSize=100;                   %种群大小
codeSize=6;                    %观察点数量
crossRate=0.8;                 %交叉概率
mutateRate=0.05;               %变异概率
maxGeneration=100;             %最大子代数
minScore=200;                  %最低成本
avecost=0;

rand('state',sum(100*clock));

readData();                    %读取数据
init(popSize,codeSize,51); %初始化种群
%第一轮循环
fitnessTable=zeros(popSize,1);
for i=1:popSize
    fitnessTable(i)=fitness(pop(i,:));    %计算适应度
end
tmpMin=min(fitnessTable)
avecost=mean(fitnessTable)
ansval=tmpMin;

minScoreLocation=find(fitnessTable==tmpMin);%寻找最优解
minScoreMethod=pop(minScoreLocation,:)          %输出最优解
g=1;
while g<=maxGeneration
    g
    selection(popSize,codeSize);                %选择
    mutation(popSize,codeSize,mutateRate);    %变异
    cross(popSize,codeSize,crossRate);          %两点交叉
    for i=1:popSize
        fitnessTable(i)=fitness(pop(i,:));    %计算适应度
    end
    tmpMin=min(fitnessTable);
    avecost=mean(fitnessTable)
    minScoreLocation=find(fitnessTable==tmpMin);%寻找最优解
    minScoreLocation=minScoreLocation(1);
    minScoreMethod=pop(minScoreLocation,:);
    if(ansval>tmpMin)                                %储存最优解
        ansval=tmpMin                                 %输出最优解
        ans=minScoreMethod
    end
    g=g+1;
```

```
end
```

6.1.2 readData.m

```
function readData()
origin=xlsread('20150915dataform.csv');
global voltageX;
global outputY;
global len;
len=length(origin)/2;
voltageX=origin(1:2:(len)*2-1,:);
outputY=origin(2:2:(len)*2,:);
end
```

6.1.3 init.m

```
function init(popSize,codeSize,maxNum)
global pop;
pop1=zeros(popSize,codeSize);
for i=1:popSize
    for j=1:codeSize
        pop1(i,j) = randi(maxNum);
        k=1;
        while (k<=j-1)
            if (pop1(i,j) == pop1(i,k))
                pop1(i,j) = randi(maxNum);
                k=0;
            end
            k=k+1;
        end
    end
end
pop=sort(pop1,2);
end
```

6.1.4 fitness.m

```
function [score] = fitness(method)
global voltageX;
global outputY;
my_answer=method;
my_answer_n=size(my_answer,2);
nsample=size(voltageX,1); npoint=size(voltageX,2);
y1=zeros(nsample,npoint);
x=voltageX;
y0=outputY;
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
    y1(j,:)=interp1(x_optimal(j,:),y0_optimal(j,:),[5.0:0.1:10.0],'pchip');
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-le
2_0)+12*(le5_0-le3_0)+25*g5_0;
si=sum(sij,2)+Q*ones(nsample,1)*my_answer_n;
cost=sum(si)/nsample;

score=cost;
```

6.1.5 cross.m

```
function [] = cross(popSize,codeSize,crossRate)
global pop;
for i=1:2:popSize
    r=rand;
    if r>crossRate
        continue;
    end
    p=randi([2,codeSize]);
    q=randi([2,codeSize]);
    A=[p,q];
    A=sort(A,2,'descend');
    tmp=pop(i,A(2):A(1));
    pop(i,A(2):A(1))=pop(i+1,A(2):A(1));
    pop(i+1,A(2):A(1))=tmp;
    checkPop(i,codeSize);
    checkPop(i+1,codeSize);
end
```

```
end
```

6.1.6 checkPop.m

```
function checkPop(n,codeSize)
global pop;
pop(n,:)=sort(pop(n,:));
```

6.1.7 mutation.m

```
function mutation(popSize,codeSize,mutateRate)
global pop;
for i=1:popSize
    r=rand();
    if r<mutateRate
        r=randi(codeSize);
        pop(i,r)=pop(i,r)+int32(randn());
        if(pop(i,r)<1)pop(i,r)=1;
        end
        if(pop(i,r)>51)pop(i,r)=51;
        end
        checkPop(i,codeSize);
    end
end
end
```

6.1.8 selection.m

```
function [] = selection(popSize,codeSize)
global pop;
global fitnessTable;
fitnessSum=zeros(popSize,1);
fitnessSum(1)=1/(fitnessTable(1));
for i=2:popSize
    fitnessSum(i)=fitnessSum(i-1)+1/(fitnessTable(i));
end
popNew=zeros(popSize,codeSize);
for i=1:popSize
    r=rand()*fitnessSum(popSize);
    left=1;
    right=popSize;
    mid=round((left+right)/2);
    while 1
        if r>fitnessSum(mid)
            left=mid;
        else
            if r<fitnessSum(mid)
                right=mid;
            else
                popNew(i,:)=pop(mid,:);
```

```
                    break;
                end
            end
            mid=round((left+right)/2);
            if (mid==left)||(mid==right)
                popNew(i,:)=pop(right,:);
                break;
            end
        end
    end
    pop=popNew;
```

6.2 差值测试点

[3,15,25,31,39,50]；[4,15,23,29,38,49]；[3,13,21,28,36,48]；[3,8,28,29,36,48]；
[3,13,21,28,40,48]；[5,13,23,31,36,48]；[3,14,21,27,36,48]；[4,10,21,28,36,48]
[1,15,21,28,37,49]；[4,15,21,28,37,51].
[3,12,20,26,33,43,50]；[3,12,20,27,33,41,49]；[3,12,16,27,30,38,47]；[3,10,16,27,30,40,47]；
[7,10,16,27,32,40,47]；[2,13,19,28,34,47,50]；[1,15,19,28,33,47,50]；[4,9,20,26,30,38,50]；
[2,10,20,26,30,37,50]；[1,10,20,26,31,38,47].