统计推断在数模转换系统中的应用

组号： 21　　刘长风 5120309107　　黄奕斐 5130309269

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：三次样条插值，MATLAB，遗传算法,埃尔米特插值，曲线拟合

Application of Statistical Inference in DA Inverting System

Team :NO.21　　Liu Chang-feng 5120309107　　Huang Yi-fei 5130309269

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 : Three fitting, MATLAB, Genetic algorithm, Hermite interpolation, curve fitting

1 引言

随着科技的发展，传感器得到广泛得应用，要在确保质量的前提下大批量生产传感器，就需要对样品进行单点测定。我们研究的问题寻找校准工序的优化方案。通过预先抽样研究多个样品的曲线特性分析，发现样品的曲线有如下特点：

（1） 个体产品的特性曲线形态相似但两两相异。

（2） 特性曲线都是单调递增的。

（3） 每条特性曲线大致可以分为首（左），中，尾（右）三段，三段都不是完全线性的，有一定的弯曲度（随机）。

（4） 中段的斜率小于首段和尾端的斜率，且中段的起点位置和长度都带有随机性。

所以我们需要选择一种拟合（或插值）方法，并且选择一种取点方案，依据从样本库获取的单点测定数值，按照选择的拟合方法进行拟合并逐一完成定标，根据定标结果，结合原始数据测算出成本记录，最后方案对比将成本最低的方案作为最终结果。

考虑到拟合效果和运行时间，我们最终选定三次样条插值函数作为拟合函数。同时为了能够高效而准确得筛选出测试点，我们选用了遗传算法，最终进行了多次优化来试图找到最优解。

2 拟合或插值方法选取探究

我们从所给的 469 组数据中随机抽取部分数据做出其曲线图，发现大多数数据都有近似的曲线性状。考虑到，我们将先依据这些数据确定一种较好的方案实现对未知器件可以通过取少量的点来实现拟合。并建立如下的评分规则[1]：

（1） 公式单点定标误差成本按照式（2-1）评估

（2） 单点测定成本即实施一次单点测定的成本以符号 q 记。本课题指定 q=12。

（3） 某一样本的定标成本按照公式（2-2）评估。

（4） 按式（2-3）计算评估校准方案的总体成本，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。

依据这样的评分规则研究每一组数据，我们将得到的曲线点与其原数据值比较得到分数，去所有组得分的平均值作为本组解的得分。

$$s_{i,j}=\begin{cases}0 & \text{if } |\hat{y}_{i,j}-y_{i,j}|\le 0.5\\ 0.5 & \text{if } 0.5<|\hat{y}_{i,j}-y_{i,j}|\le 1\\ 1.5 & \text{if } 1<|\hat{y}_{i,j}-y_{i,j}|\le 2\\ 6 & \text{if } 2<|\hat{y}_{i,j}-y_{i,j}|\le 3\\ 12 & \text{if } 3<|\hat{y}_{i,j}-y_{i,j}|\le 5\\ 25 & \text{if } |\hat{y}_{i,j}-y_{i,j}|>5\end{cases} \qquad (2\text{-}1)$$

$$S_i=\sum_{j=1}^{51}s_{i,j}+q\cdot n_i \qquad (2\text{-}2)$$

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。

$$C=\frac{1}{M}\sum_{i=1}^{M}S_i \qquad (2\text{-}3)$$

2.1 多项式拟合

2.1.1 概述

拟合就是寻找一条光滑的曲线，最不失真得表现测量数据，反过来说，对测量的实验数据，要对其进行公式化处理，也是一种计算方法，构造一个函数关系。Matlab中有一个函数命令 polyfit即可以实现该功能，其命令格式为：

P=polyfit(x,y,n) （2-4）

式中：x,y为已知数据，n为拟合多项式的阶次，p为返回所得多项式的系数向量。

2.1.2 数据分析

在观察了大量数据之后，我们发现曲线图可大致分为首，中，尾三段，都不是完全线性的，有一定的弯曲度并且中段的斜率小于首段和尾端的斜率，且中段的起点位置和长度都带有随机性。如图2-1和2-2分别显示了第100组和第290组数据点的图像。

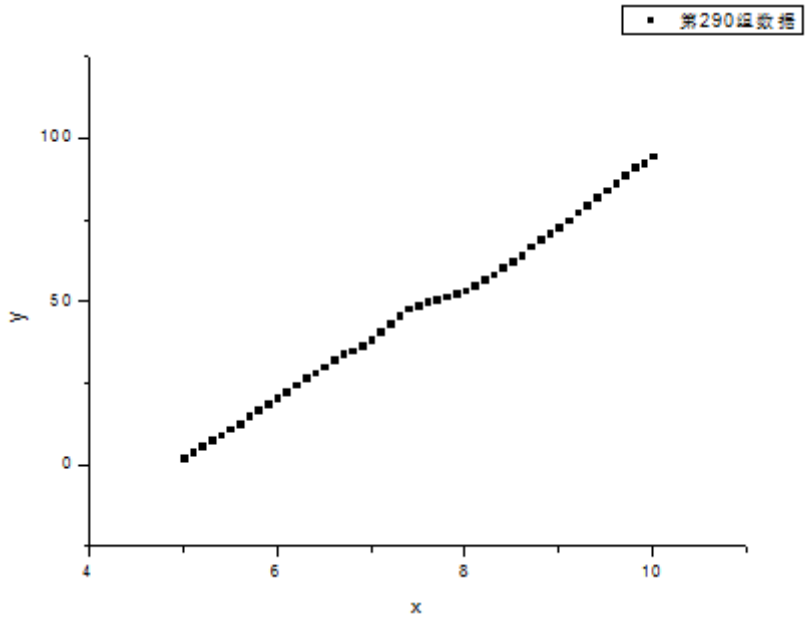

图 2-1：第 290 组数据

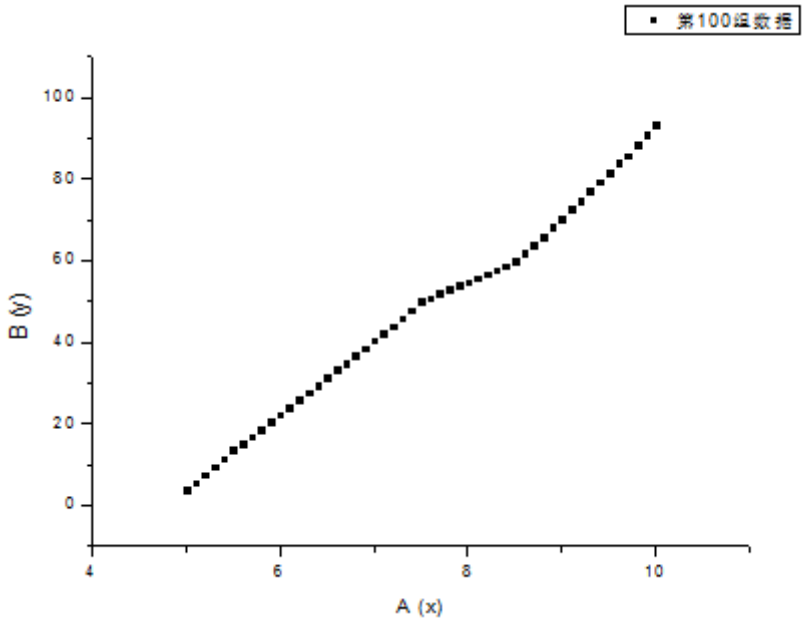

图 2-2 第 100 组数据点的图像

依据第290组和第100组数据做出的曲线图以及对其它各种数据的综合分析，我们认为可以通过多项式拟合的方法来拟合图像。于是，我们建立了四次函数，五次函数模型，通过一个标准的评判规则来对各个模型进行评分，最后将得到最好的模型。

2.2 三次样条插值拟合

2.2.1 插值方法的选择

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。

2.2.2 三次样条插值拟合的概述

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。对于（n+1）个给定的数据集合{xi},我们可以使用 n 段三次多项式在数据点之间构建一个三次样条。用公式（2-3）[2]表示对函数 f 进行插值的样条函数，需要：

（1） 插值特性： $S(x_i)=f(x_i)$

（2） 样条相互连接： $S_{i-1}(x_i)=S_i(x_i)$, i=1,2…n-1

（3） 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,…n

S(x)是分段函数，取值如下：

$$\begin{cases} S_0(x), x\in[x_0, x_1] \\ S_1(x), x\in[x_1, x_2] \\ \cdots \\ S_{n-1}(x), x\in[x_{n-1}, x_n] \end{cases} \quad (2\text{-}5)$$

由于对于每个三次多项式函数需要有四个参数如公式（2-4）的形式，所以对于组成 S 的 n 个三次多项式而言，需要 4n 个条件确定这些多项式。

$$y = ax^3 + bx^2 + cx + d \quad (2\text{-}6)$$

2.3 Hermite 插值函数概述[3]

Hermite 插值是在给定的节点处，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。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 xi，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 xi 为 mi 重插值点节,因此，Hermite 插值应给出两组数，一组为插值点{xi}ni=0 节点，另一组为相应的重数标号{mi}ni=0。

$$\sum_{i=0}^{n} m_i = N + 1 \quad (2\text{-}7)$$

若满足式子（2-7）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，于是给出下面定义：

f 为 $[a,b]$ 上充分光滑函数，对给定的插值定节$\{x_i\}_{i=0}^n$，及相应的重数标号$\{m_i\}_{i=0}^n$，$\sum_{i=0}^{n} m_i = N + 1$ 时，若有 $H(x) \in P_n$ 满足

$$H^l(x_i) = f(x_i)\ , l = 0,1,\ldots,m_{i-1}; i = 0,1,\ldots,n. \quad (2\text{-}8)$$

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。

3 基于遗传算法寻找最优解

3.1 遗传算法（Genetic Algorithm）

3.1.1 遗传算法概述

遗传算法是借鉴生物界的“适者生存，优胜劣汰”原则，模拟基于达尔文生物净化论的自然选择和遗传学原理的生物进化过程的启发式搜素算法。在自然界中，生物通过染色体的遗传、交叉以及变异来保留、产生优秀性状，创造更加适应环境的子代。它的主要特点是：

（1） 遗传算法从问题解的串集开始搜索，而不是从单个解开始，覆盖面大，利于全局择优。

（2） 具有内在的隐并性和很好的全局寻优能力，可以同时处理群体中的多个个体，即对

搜索空间中的多个解进行评估，减少了陷入局部最优解的风险，同时使算法本身易于实现并行化。

（3） 应用范围大，采用概率化的寻优方法，能自动获得和指导优化的搜索空间，自适应得调整搜索方向，不需要确定的规则。

（4） 直接对结构对象操作，不存在求导和函数连续性的限定，也不用搜索空间的知识及其他的辅助信息。仅用适应度函数来评价个体，在此基础上进行遗传操作。适应度函数不仅不受连续可微的约束，而且定义域可以任意设定。

（5） 遗传算法在适应度函数选择不当的情况下可能收敛于局部最优而不能达到全局最优

3.1.2 遗传算法与本实验的结合

所以，算法实现时可以随机选取指定个数的产品作为初始种群根据达尔文的“优胜劣汰，适者生存”法则，适应度较强的个体保留的可能性较大。然后在保留下来的个体中，它们的染色体以一定概率进行交叉互换、以一种更小的概率发生变异新，而具有新染色体的新个体再经过“优胜劣汰，适者生存”以及交叉变异等，如此循环往复一定次数，最终留存下较适合整个环境数据（即染色体），即为我们所求的解。

值得注意的是，遗传算法在全局寻优时，只能找到较为优化的解，而不能保证一定是最优解（即最终留存下来的适应环境的染色体有多条）。

3.2 算法实现

由于 MATLAB 高级语言的通用性，对问题用 M 文件编码，与此配对的是 MATLAB 先进的数据分析，可视化工具，特殊目的的应用领域工具箱和展现给使用者具有研究遗传算法可能性的一致环境。本实验我们选择采用英国苏菲尔德大学开发的遗传算法工具箱 gatbx 工具箱。

3.2 .1 Matlab 中 gatbx 库函数介绍[4]

a. 种群表示和初始化函数：crtbase,crtbp,crtp

b. 适应度计算：ranking,scaling

适应度计算函数用于转换目标函数值，给每个个体一个非负的价值数。

c. 选择函数：reins,rws,select,sus

函数据适应度的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。

d 交叉算子：recombin,xovshrs,xovsp,xovsprs.

交叉是通过给定的概率重组一对个体而产生后代。

e. 变异算子：mut,mutate,mutbga

f. 多子群支持：migrate

该函数的功能是在子群中交换个体。

3.3 实验过程

3.3.1 实验参数设置

表3-1 实验参数表

实验参数名称	取值	意义
XNUM	10	基因（取点）数目
MAXGEN	40	最大遗传代数
GGAP	0.9	代沟 即选中的存活率
PM	0.07	变异概率
gen	0	代计数器（用于统计代数）

适应度函数也成为评价函数，适应度函数的设计应该满足单值，连续，非负，最大化，合理，一致，计算量小，通用性等要求。[5]

本实验中我们使用 gatbx 自带的 ranking 函数作为适应度函数，并且使用了 reins 函数，该函数可以使用均匀的随机数或基于适应度的重新插入。

3.3.2 交叉变异的异常处理已经多子群支持

两个染色体交叉或变异后，可能在一个染色体中产生两个或多个相同基因，这样的染色体进行拟合函数时可以回出现异常。我们采用 gatbx 库中的自带函数 recombin 和 mut 函数分别完成交叉变异，多子群支持的操作，并通过循环操作实现。

4 基于遗传算法的算法优化

我们按照上述最基本的想法利用遗传算法的知识和 matlab 中自带的 gatbx 库完成了最基本的寻求最优解的方法。但是我们发现产生的子代个体内部基因会有重复测试点的问题。另外，求得的最优解误差较大而且“早熟”现象严重。

为了解决子代内部基因重复的问题我们对代码优化，首先，我们使用了二进制编码方便选点，并且加入了自定义的循环来使得产生的基因不重复。

由于数据较多，我们不可能采用“暴力穷举”法来得到更优的解。我们认为可以人为得干预初始种群，调整选点的概率，变异概率等来使初始种群的产生更加随机。依据对多次实验结果的观测分析，我们自定义了一种优化方法“附近探测法”来找到最优的解。

4.1 算法优化过程流程图

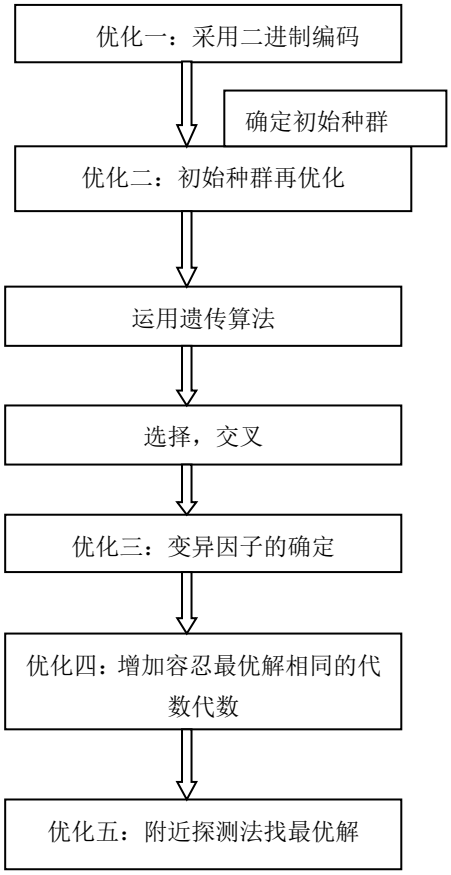

图 4-1 算法优化过程流程图

4.2 优化过程中的参数解释

表4-1 实验参数表

实验参数名称	取值区间或取值区间	意义
NIND	500 或 300	个体数目
MAXGEN	100 或 150 或 400	最大遗传代数
GGAP	0.9	代沟即自然选择中存活率
PM	0.07 或 0.02	当前变异概率
MAXPM	0.5 或 0.97	变异概率上限
MINPM	0.02	变异概率下限
STEPPM	0.05 或 0.2	变异概率改变的步长
MINONES	2 到 6	每个个体中要求的最小测定点数
MAXCHANCE	0.2 到 0.25	单个个体每个基因位产生 1 的期望最大值
MINCHANCE	0.05 到 0.1	单个个体每个基因位产生 1 的期望最小值

4.3 优化一：二进制编码

4.3.1.二进制编码方法简介

它由二进制符号 0 和 1 所组成的二值符号集。每个个体是一个二进制符号串。根据模式理论，采用二进制编码算法处理的模式最多，几乎任何问题都可以用二进制编码来表达。它有以下一些优点：[6]

1) 编码、解码操作简单易行
2) 选择，交叉、变异等遗传操作便于实现
3) 符合最小字符集编码原则
4) 利用模式定理对算法进行理论分析。

4.3.2 实现过程

我们用二进制编码每个个体的基因即每个个体各个位置上的数据，如果选取该位置的数据，那么该位置的基因取 1，如果不取该位置上的基因，那么该位置上的基因取 0，这样就可以方便解决选点个数和点的位置的确定两个问题。

另外我们察觉到初始化种群可能存在选点不够随机，甚至重复选点的问题，于是，我们我们决定采用自己指定任意指定一个固定的概率来初始化种群。

值得注意的是，为了避免重复选点，不同的拟合曲线我们有需要点数的最小值，比如，对于三次样条插值法我们需要至少 2 个点，三次曲线我们需要至少 4 个样本点等，为了保证实验中选点个数可以大于所需要点数的最小值，我们使用变量 MINONES 来记录需要点数的最下值，可以通过循环直到产生需要的相异点的个数之后停止，来实现选点个数大于等于确定曲线所需要的最小点的个数。经过多次尝试，我们发现当指定概率为 0.15 时可以产生一个最优解，结果表 4-2：

表4-2二进制编码后的结果

最优解	成本	产生最优解的代数
3 12 22 31 43 50	92.8774	28 出现了“早熟”

此次优化得到的最优解所需的成本较小，而且实现了子代个体基因不相同

4.4 优化二：对初始化种群过程的优化

从优化一的结果可以看出，虽然得到的解已经比较优秀了，但是“早熟”现象明显。

我们进一步思考，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。

我们使用一个概率范围让每个个体产生 1 的期望个数（即测定点个数）平均落在这个这个概率范围内来初始化种群。这样我们让每个个体上基因的产生概率相同，但不同个体之间的概率不同，这样大大丰富了初始种群。

得到如下实验结果：

表4-3 优化初始化种群后的结果

最优解	成本	产生最优解的代数
2 9 20 26 33 43 50	93.8561	22 “早熟”现象未得到改善

对比优化一的结果可以发现：成本反而增加了，虽然收敛速度大大加快，但是由于收敛过快，收敛于局部最优。

4.5 优化三：变异因子的确定

根据优化二后的结果，我们认为陷入局部最优的原因可能是由于变异的概率过小使得无法跳出局部最优，于是我们对变异因子的确定进行优化。

我们首先尝试了改变变异概率的上限，改进后“早熟”现象得到了有效控制，但是成本却增加了。

我们分析原因可能是因为变异上限改动过大，恰好错过了最优解。于是我们细化对变异因子产生的过程，每一次实验当产生的最优解与上一次相同的时候，增加变异因子，一旦不同，把变异因子减去一个值（变异因子有上下界）。对不同的拟合函数都进行了上述操作。

4.5.1 实验结果

表4-4 修改变异上限后的实验对比

实验次数	拟合函数	控制变量	最优解	成本	产生最优解的代数
第一次	五次函数	MAXPM=0.5;	2 8 20 31 43 49	107.1087	第 9 代
第二次	五次函数	MAXPM=0.97;	3 11 19 28 37 44 50	108.0608	第 50 代
第三次	四次函数	MAXPM=0.5;	2 12 24 31 41 49	108.5885	第 2 代
第四次	四次函数	MAXPM=0.97;	3 10 21 30 41 49	105.2857	第 69 代

4.5.2 实验对比与分析

1. 第一次实验与第二次的区别仅仅在于第二次修改了变异概率的上限由 0.9 增加至 0.97。第二次实验中，第 4 代的成本为 113.3369 这个最优解一直持续到第 42 代；第 50 代早熟然后就不再改变。说明即使到最后变异概率变成 0.97 接近于 1，也无法避免早熟的问题

第三次实验与第四次实验的区别在于将变异上限由 0.5 增加至 0.97。第四次实验中

第 2 代也得到了成本为 108.5885 的局部最优解，但是随着变异概率的逐渐增加，到第 26 代找到了成本为 107.9094 的最优解。在经过若干代保持不变后，第 45 代找到了成本 106.7612 的最优解。同样经过若干代不变后，在第 69 代找到了 105.2857 的最优解（3　10　21　30　41　49），也即最终结果可以看出，提高变异上限，“早熟现象”得到了较有效得控制，但是还是没有解决“早熟”。

2. 第四次实验与第二次实验对比，差别在与第四次实验采用的是四次曲线作为拟合函数而第二次实验采用的是五次函数作为拟合曲线。不难发现，用四次函数作为拟合函数要比五次函数作为拟合函数，拟合的效果好。

4.6 优化四：增加容忍最优解相同的代数上限

前面的实验可以看出，如果“早熟”现象严重，会导致在收敛于局部最优解后，即使运行遗传算法，也无法找到更好的解，反而浪费了计算时间。我们想到可以增加一个参数表征容忍最优解相同的代数上限。一旦若干代产生的最优解相同，那么我们就舍弃这次实验，并记录下最优解的结果，重新初始化种群，开始新一轮的计算。

同时我们注意到，如果容忍的最大相同最优解代数太小，遗传算法本来在若干代相同解之后会产生更优的解，但是由于重新初始化种群没有发挥最大作用。于是我们把忍耐最优解相同的代数增加，来确保发挥遗传算法的作用。

4.6.1 实验结果

针对上述问题，我们对随机遍历抽样法进行了改进得到了如下结果：

表4-5 随机遍历抽样法实验结果

实验条件	最优解	成本
优化前	1 8 19 27 34 43 50	97.2889
优化后	1 8 19 27 34 43 50	94.7964

4.7 优化五：选点的再次优化

另外，我们发现找到的各代最优解的位数相同，测定点对的位置基本一致，最多只相差几位。从中得到启发，对方案进行再次优化。

在得到一组局部最优解后（通过若干代后最优解是否不变来判断），可以对其每个元素进行左右移动操作，试出一个更加优秀的解。我们将这种自定义的做法成为“附近探测法”。

4.7.1 附近探测法的具体实现

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。

之后，对这个最优解，实现“附近探测”最优解的操作，对于最优解中的每个点分别执行向左或向右移动，至多移动两位，不妨假设最优解中有 6 个数据点，这样共执行 5^6 次操作，大概相当于 300 个个体 50 代的运算量，是可以接受。通过这样的尝试可以得到一个更加优秀的解。

4.7.2 附近探测法实验结果及分析

通过之前的对比分析，三次样条插值函数能够较好得实现拟合，于是我们选择三次样条插值函数进行实验，用三次样条插值做一个优化前后的对比，得到结果如下表所示

表4-6 采用附近探测法与未采用附近探测法实验结果对比

实验条件	最优解	成本
三次样条插值函数优化前	1 8 19 27 34 43 50	94.7964
三次样条插值函数优化后	2 9 20 26 33 43 50	93.8561

观察上表的结果可以发现，“早熟”现象得到了较好得控制，而且得到的解与之前几种方案的到的解相比，得到了优化。

4.8 应用优化缩短寻找最优解的时间

依据上述的几种优化，我们试图降低寻找最优解的时间，尽量在最短时间中得到最优解，于是我们应用优化再次实验得到如下结果：

表4-7优化的应用

实验条件	实验参数	最优解	成本	运行时间
遗传算法加附近探测法	30 个体 10 代	3 11 22 31 43 50	93.029	1397s（24min）

4.9 应用遗传算法尝试更多的插值函数

我们之前使用三次样条插值函数拟合曲线，经过多次优化后发现结果得到了优化，因为 matlab 中含有许多子代的插值函数，于是我们决定尝试更多的插值函数来优化结果。

经过多次尝试，我们发现使用分段三次 Hermite 插值多项式拟合的结果最优。

4.9.1 Hermite 函数插值

我们利用 matlab 强大的功能，调用了 pchip 函数[7]，分三段 Hermite 插值，得到如下结果：

表4-8 分段三次Hermite插值拟合

实验参数	最优解	成本	时间
50 个个体 10 代遗传	7 16 27 37 49	86.7676	398s

对本次实验得到的最优解执行一次“附近探测”操作得到结果如下：

表4-9 对分段三次Hermite插值最优解附近探测

探测起点	最优解	成本
7 16 27 37 49	5 16 26 35 48	83.0405

对执行过一次“附近探测”后找到的最优解，我们又执行了一次“附近探测”，在其附近寻找更加优化的解，得到如下结果：

表4-10 第二次附近探测

探测起点	最优解	成本
5 16 26 35 48	4 16 26 35 48	82.7655

可见，插值函数的改变对于解的优化起到了至关重要的作用，而在此基础上我们“附近探测”，进行局部再优化，使得找到了最优解。

5. 实验结论

以上的实验结果表明，使用优化后的遗传算法可以在较短的时间内得到成本较低的传感特性校准方案。仅使用传统的遗传算法会遇到收敛过快、收敛于局部最优解等问题。但是我们基于传统的遗传算法，提出了多种改进策略，在这些改进策略的共同作用下，不仅能够找到更加优秀的方案，而且大大降低了寻找优方案的计算时间，得到了很好的实验结果。

本课题的结论，在希望总体误差误差最小的情况下，可以使用分段三次 Hermite 插值来完成传感特性的校准，见表 5-1

表5-1 实验结果

实验条件	最优解	成本
分段三次 Hermite 插值	4 16 26 35 48	82.7655

6. 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 [EB/OL].ftp://202.120.39.248.

[2] 维基百科.样条插值-维基百科，自由的百科全书 [M/OL].[2014-10-26].http://zh.wikipedia.org/wiki/样条插值.

[3] 维基百科.埃尔米特插值-维基百科，自由的百科全书 http://zh.wikipedia.org/wiki/%E5%9F%83%E5%B0%94%E7%B1%B3%E7%89%B9%E6%8F%92%E5%80%BC.

[4] 百度文库 .Matlab 遗传算法工具箱及其应用 . http://wenku.baidu.com/link?url=2FN2TqDehOe6VoNThkIic8O9Ldmvk5NF8_psLPT-6lld4yiFhZO2oy3NCH7eSzIqjsOlOgKlhYiMazslo8_8UNSOBFnufFxe4FP1xjlAtFC&qq-pf-to=pcqq.c2c.

[5] 朱鳌鑫. 遗传算法的适应度函数研究. 系统工程与电子技术, TP301.6 1998 年 11 期 .

[6] 杨建军 丁玉成 赵万华. 遗传算法编码方案比较. 计算机应用研究, 2010 年第一期 .

[7] 百度文库 .Matlab 中插值函数汇总和使用说明 http://wenku.baidu.com/link?url=OMBRi9HrEtMS4oiQtr5YUFCu-9i71bQMVNHlONgcP2q3ahTFhuUbgJ0NM_FDmTfiVm1yTTcfzORgeDjoAzfaVlNA-03yrOWNM_Vn9-Y6vzm&qq-pf-to=pcqq.c2c

7. 附录：程序代码文本清单

注：为简化篇幅，本清单中未包含 gatbx 工具箱中的函数，读者既可以查看本报告附带的.m 文件，也可以从网上搜索下载 gatbx 工具箱。

1.

```
function binarygo_anti_precocity3()
%主函数
%与 binarygo_anti_precocity 的区别，修改了变异上限等变异概率参数
%与 1 比较改进了：1.过程中通过重新初始化种群得到若干个局部最优解；2.对最优的局部最优解进行左右摆动找到最最优的解

tic; %按秒表程序开头

XNUM=51; %基因数目

NIND=50; %300 个体数目%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
MAXGEN=10;    %400 最大遗传代数%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
GGAP=0.9; %代沟  自然选择中存活率
PM=0.02; %0.02 当前变异概率
MAXPM=0.97;   %变异概率上限
MINPM=0.02;   %0.02 变异概率下限
STEPPM=0.02; %0.2 变异概率改变的步长，太大导致不收敛，没有意义
MINONES=2; %每个个体中要求的最小测定点数。2->三次样条差值；4->三次函数拟合；5->四次函数拟合；6->五次函数拟合
MAXCHANCE=0.10;%单个个体每个基因位产生 1 的期望最大值
MINCHANCE=0.03;%单个个体每个基因位产生 1 的期望最小值

rng(10);%取相同的随机数种子，可以重现模拟过程

%产生初始种群
Chrom   = initialCreate( NIND,XNUM,MAXCHANCE,MINCHANCE,MINONES );

gen=0;%进化代数计数器
rep=0;%记录最优解重复的次数

StoreBindex=[];%存储所有代的最优解，二进制形式
StoreCost=[];%存储所有代的最优解的 cost

while gen<MAXGEN
    %个体适应度计算
    ObjV=zeros(NIND,1);%cost 矩阵
    for i=1:NIND
        ObjV(i)=test_my_answer(Chrom(i,:));
```

```
end
%自己写的：FitnV=ones(NIND,1)./ObjV;%cost 越小，适应度越高，取倒数
FitnV=ranking(ObjV); %分配适应度值  ranking 里面越小代表越好
%个体适应度计算完毕

%选择
SelCh=select('sus',Chrom,FitnV,GGAP);  %可以将 sus 换成 rws（轮盘赌选择算法）
%选择完成

%重组
SelCh=recombin('xovsp',SelCh,0.7);
%重组完成

%变异
%SelCh=mut(SelCh); 原来的变异函数
SelCh=mut_new(SelCh,PM);  %使用自定义的变异函数
%变异完成

%检验并保证每个个体的测定点数（1 基因个数）是否大于 MINONES
[SelChN,SelChC]=size(SelCh);
for i=1:SelChN
    while sum(SelCh(i,:))<MINONES
    %保证至少出现 n 个 1，且每个个体一定为 1 的位置不同
    SelCh(i,randi(51))=1; %randi（51） 产生 1-51 之间的随机整数（均匀分布）
    end
end
%检测完毕

%
[r,c]=size(SelCh);
ObjVSel=zeros(r,1);%计算子代的 cost 矩阵
for i=1:r
   ObjVSel(i)=test_my_answer(SelCh(i,:));
end
[Chrom,ObjV]=reins(Chrom,SelCh,1,1,ObjV,ObjVSel);%重新插入子代的新种群

[Y,I]=min(ObjV); %I 为当代最优解的序号
Chrom(I,:);%当代最优解的二进制表示


gen=gen+1 %代计数器增加并输出显示
X=bindex2index(Chrom(I,:))%当代最优解的正常 index，输出显示
ObjV(I)%当代最优解的 cost，输出显示
```

```
    StoreBindex=cat(1,StoreBindex,Chrom(I,:));    %记录当代最优解的二进制表示
    StoreCost=cat(1,StoreCost,ObjV(I)); %记录当代最优解的 cost

    %通过增加变异概率防早熟
    if (gen>1)
        judge=sum(StoreBindex(gen,:)==StoreBindex(gen-1,:));  %judge=51 表示当代最优解和
    上一代最优解一样
        if (judge==51)
            rep=rep+1;
            if (PM<MAXPM) PM=PM+STEPPM;end    %如果与上一代最优解相同，在不大于
    最大变异概率的前提下，增加变异概率
        else
            if (PM>MINPM) PM=PM-STEPPM; end    %如果与上一代最优解不同，在不小于
    最小变异概率的前提下，减小变异概率
            if (PM<MINPM) PM=MINPM;end    %不低于 MINPM，这样 STEPPM 可以增大
        end
    end
    %通过增加变异概率防早熟完毕


    %如果达到 20，重新初始化种群
    if
    (rep>=20)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%    最多 20 代最优解相同可以忍受
        Chrom    =  initialCreate(  NIND,XNUM,MAXCHANCE,MINCHANCE,MINONES  );%重新产
    生初始种群
        rep=0;
        PM=MINPM;
    end

end

[MinCost,MinGen]=min(StoreCost);%MinCost 所有代中最低的 cost，MinGen 最低 cost 的出现
    代数
BBest=StoreBindex(MinGen,:);%二进制的最优解
Best=bindex2index(BBest);    %总体最优解

%  显示遗传算法得到的结果
fprintf('\n 遗传算法得到的最优解：'); Best
fprintf('\n 遗传算法得到的最优解的 cost：'); MinCost
fprintf('\n 遗传算法部分结束，进入最优解的左右摆动寻更优程序：');


%对最优解进行摆动操作找到附近的更优解
```

```
[ Best,MinCost ] = findNearby(Best);


t=toc%程序结尾，看跑了多少秒

% 显示最终结果
fprintf('\n 总体最优解：'); Best
fprintf('\n 总体最优解的 cost：'); MinCost

end
```

2.

```
function X = bindex2index( bindex )
%51 位二进制测定点坐标转换为正常的测定点坐标

    X=[];
    for i=1:1:51
       if (bindex(i)==1)
            X=cat(2,X,i);
       end
    end

end
```

3.

```
function index = explore( index,n)
%对于第 n 位左 2 到右 2 产生 5 种变异

[m,nouse]=size(index);%m 行数

indexlone=index;
indexltwo=index;
indexrone=index;
indexrtwo=index;

if (index(1,n)-1>0)
    indexlone(:,n)=index(:,n)-1;
else
    indexlone=[];
end

if (index(1,n)-2>0)
    indexltwo(:,n)=index(:,n)-2;
else
```

```
        indexltwo=[];
end

if (index(1,n)+1<52)
        indexrone(:,n)=index(:,n)+1;
else
        indexrone=[];
end

if (index(1,n)+2<52)
        indexrtwo(:,n)=index(:,n)+2;
else
        indexrtwo=[];
end

index=cat(1,index,indexlone);
index=cat(1,index,indexltwo);
index=cat(1,index,indexrone);
index=cat(1,index,indexrtwo);

end
```

4.

```
function [ Best,MinCost ] = findNearby( Best )
%对最优解进行摆动操作找到附近的更优解
[m,n]=size(Best);
for i=1:n
        Best=explore(Best,i);
end

Finalcost=[];
[M,N]=size(Best);
BBest=zeros(M,51);
for i=1:M
        %Best(i,:)%%%%%%%%%%%%测试显示使用，正常运行时注释掉
        %index2bindex(Best(i,:)) %%%%%%%%%%%%测试显示使用，正常运行时注释掉
        BBest(i,:)=index2bindex(Best(i,:));
        Finalcost(i,:)=test_my_answer(BBest(i,:));
end

[MinCost,Minindex]=min(Finalcost);
BBest=BBest(Minindex,:);
Best=bindex2index(BBest)
MinCost
```

```
end
```

5.

```
function bindex=index2bindex(index)
%十进制的 index 转成二进制的 index

[m,n]=size(index);
bindex=zeros(1,51);
for i=1:n
    bindex(1,index(1,i))=1;
end

end
```

6.

```
function Chrom    = initialCreate( NIND,XNUM,MAXCHANCE,MINCHANCE,MINONES )
%产生初始种群
%自己写的一定概率方法产生初始种群
Chrom=zeros(NIND,XNUM);   %一代种群
for i=1:NIND
    CHANCE=rand()*(MAXCHANCE-MINCHANCE)+MINCHANCE; %让每个个体测定点的数量期望不同，保证初始种群的丰富性
    for j=1:1:XNUM
        if (rand()<CHANCE)
            Chrom(i,j)=1;
        end
    end
end


%使用 crtbp 函数，产生 0-1base 的种群    CHANCE=0.5  不如自己写的
%Chrom=crtbp(NIND,XNUM);

%让被测定点的个数大于 MINONES
for i=1:NIND
    while sum(Chrom(i,:))<MINONES
    %保证至少出现 n 个 1，且每个个体一定为 1 的位置不同
    Chrom(i,randi(51))=1; %randi（51） 产生 1-51 之间的随机整数（均匀分布）
    end
end
%初始种群产生完毕
```

End

7.

```
% MUT.m
%
% This function takes the representation of the current population,
% mutates each element with given probability and returns the resulting
% population.
%
% Syntax: NewChrom = mut(OldChrom,Pm,BaseV)
%
% Input parameters:
%
%    OldChrom - A matrix containing the chromosomes of the
%               current population. Each row corresponds to
%               an individuals string representation.
%
%    Pm     - Mutation probability (scalar). Default value
%               of Pm = 0.7/Lind, where Lind is the chromosome
%               length is assumed if omitted.
%
%    BaseV       - Optional row vector of the same length as the
%               chromosome structure defining the base of the
%               individual elements of the chromosome. Binary
%               representation is assumed if omitted.
%
% Output parameter:
%
%    NewChrom - A Matrix containing a mutated version of
%               OldChrom.
%
% Author: Andrew Chipperfield
% Date: 25-Jan-94
%
% Tested under MATLAB v6 by Alex Shenfield (21-Jan-03)

function NewChrom = mut_new(OldChrom,Pm,BaseV)

% get population size (Nind) and chromosome length (Lind)
[Nind, Lind] = size(OldChrom) ;

% check input parameters
if nargin < 2, Pm = 0.7/Lind ; end %0.7/染色体长度
if isnan(Pm), Pm = 0.7/Lind; end
```

```
if (nargin < 3), BaseV = crtbase(Lind);    end
if (isnan(BaseV)), BaseV = crtbase(Lind);    end
if (isempty(BaseV)), BaseV = crtbase(Lind);    end

if (nargin == 3) && (Lind ~= length(BaseV))
    error('OldChrom and BaseV are incompatible'), end

% create mutation mask matrix
BaseM = BaseV(ones(Nind,1),:) ;

% perform mutation on chromosome structure
NewChrom = rem(OldChrom+(rand(Nind,Lind)<Pm).*ceil(rand(Nind,Lind).*(BaseM-1)),BaseM);
```

8.

```
function y1 = mycurvefitting( x_premea,y0_premea )%测定点的坐标

x=[5.0:0.1:10.0];

y1=interp1(x_premea,y0_premea,x,'pchip');


end
```

9.

```
function y1 = pchip( x_premea,y0_premea )%测定点的坐标
%分段三次 Hermite 插值

x=[5.0:0.1:10.0];


y1=interp1(x_premea,y0_premea,x,'pchip');


end
```

10.

```
function y1 = poly3( x_premea,y0_premea ) %y1 为拟合计算得到的 51 个 y 值;bindex 为 2 进制
    1 行 51 列下标数组
%三次函数拟合
%至少需要四个测定点!

x=5:0.1:10;
y1=zeros(1,51);
```

```
f=polyfit(x_premea,y0_premea,3);
y1=polyval(f,x);

end
```

11.

```
function y1 = poly4( x_premea,y0_premea ) %y1 为拟合计算得到的 51 个 y 值;bindex 为 2 进制
    1 行 51 列下标数组
%四次函数拟合

x=5:0.1:10;
y1=zeros(1,51);

f=polyfit(x_premea,y0_premea,4);
y1=polyval(f,x);

end
```

12.

```
function y1 = poly5( x_premea,y0_premea ) %y1 为拟合计算得到的 51 个 y 值;bindex 为 2 进制
    1 行 51 列下标数组
%五次函数拟合

x=5:0.1:10;
y1=zeros(1,51);

f=polyfit(x_premea,y0_premea,5);
y1=polyval(f,x);

end
```

13.

```
function y1 = poly6( x_premea,y0_premea ) %y1 为拟合计算得到的 51 个 y 值;bindex 为 2 进制
    1 行 51 列下标数组
%六次函数拟合

x=5:0.1:10;
y1=zeros(1,51);

f=polyfit(x_premea,y0_premea,6);
y1=polyval(f,x);

end
```

14.

```
function y1 = sp3( x_premea,y0_premea )%测定点的坐标
%三次样条差值

x=[5.0:0.1:10.0];


y1=interp1(x_premea,y0_premea,x,'spline');


end
```

15.

```
function cost = test_my_answer( bindex )%bindex 2 进制单个体 51 位基因
%自己测试用的带输入的计算一种取点方法的 cost

%二进制下标数组 bindex->选点下标数组 index 的转换
my_answer=[];
for j=1:1:51
    if (bindex(j)==1)
        my_answer=cat(2,my_answer,j);
    end
end

my_answer_n=size(my_answer,2);

% 标准样本原始数据读入
minput=dlmread('20141010dataform.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
    % 修改为对应的拟合函数
    y1(j,:)=pchip(x_optimal(j,:),y0_optimal(j,:));% 记 得 改 对 应 的
    MINONES!!!!!! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

% 成本计算
Q=12;
errabs=abs(y0-y1);
le0_5=(errabs<=0.5);
le1_0=(errabs<=1);
le2_0=(errabs<=2);
le3_0=(errabs<=3);
le5_0=(errabs<=5);
g5_0=(errabs>5);
sij=0.5*(le1_0-le0_5)+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;

end
```