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 27 组，成员：衣帆 学号 5140309319, 方曦 学号 5140309325, 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》，陈琦，2014 年秋季学期，组号 52 遗传算法的整体实现思路参考 52 组同学代码，代码细节自己实现。 《统计推断在数模模数转换中的应用》，谢昊男，2014 年秋季学期，组号 06 沿用了该组模拟退火算法的部分思路，进行了一些改进，以及经过独立实验修改了设定参数和代码逻辑。
引言	《统计推断在数模模数转换中的应用》，殷柯，2014 年秋季学期，组号 30 参考该组引言，并进行了少量修改。
问题引入与变量声明	《统计推断在数模转换系统中的应用》，王博远，2014 年秋季学期，组号 23 参考该组同学的案例分析与变量的声明。并加入大量自己的分析。

统计推断在数模转换系统中的应用

第 27 组　方曦 5140309325，　衣帆 5140309319

摘要：本文以传感器的批量生产为背景，为其提供了优化的方案。利用遗传算法和模拟退火算法来确定优化的取点方案；对比不同拟合方法的效果，从而选取较为适合的拟合方法。并以降低校准陈本且提高精度为原则，结合实际问题对遗传算法进行了优化，从而得到最优解。

关键词：遗传算法，模拟退火算法，Hermite 插值，MATLAB

Application of Statistical Inference in AD-DA Converting System

ABSTRACT: Based on sensor batch production as the background, to find the optimized plan for the calibration. We uses genetic algorithm and the simulated annealing algorithm to gain the optimized plan; we also use several ways such as polynomial fitting, cubic spline interpolation and Hermite interpolation to fit the function of sample points. On the principle of reducing cost of calibration and improving the accuracy, we optimize the genetic algorithm according to the practical problems, and the optimal solution are obtained.

Key words: genetic algorithm, simulated annealing algorithm, Hermite interpolation, MATLAB

1 引言

在工程实践中，对某些物理量（如温度、压力、光强等）的测量需要用到特殊的测量工具，这些测量工具主要由传感器组成。实际生产中，这些测量工具的输入输出特性会有差异，并且该输入输出特性往往呈现明显的非线性。因此为了得到较精确的结果，一种办法是尽可能多地选取测试点定标，然而这样必然会使定标成本加大。因此需要合理地选择某些点进行定标，由这些点拟合出来的特性曲线近似替代其真实的特性曲线。合理的方案既要考虑尽可能减小拟合与真实值之间的误差，又要减少由定标测试点的个数带来的定标成本。本文就上述问题，探讨了如何选取测定点的数量和位置，以及合理的拟合方法得到输入输出的特性，使得在一定精度要求下，尽可能降低定标耗费的成本。

2 问题引入与变量声明

2.1 问题分析及研究方法

本实验中一共提供了 400 个样本，每个样本中有 51 组数据，x 均为 5.0 到 10.0 间且隔为 0.1 时对应不同的取值，y 在 0 到 100 之间随 x 单调递增，且为非线性曲线关系，大致类似于陡缓陡的三段线性关系。本定标实验目的是通过某种的拟合方式，再通过一定的算法找到最优的取点方案，使得总定标成本最小，其中总定标成本包括误差成本和取点成本，误差成本为对于每一个样本取同样的位置点数，从样本库获取并按选定拟合方式计算该样本所有拟合值，对所有样本逐一完成定标，计算总误差和取点成本，最后对样本个数取平均值。

取点时一方面要使取点数尽可能少来降低取点成本，另一方面拟合要越接近越好来降低误差成本。本文采用遗传算法和模拟退火算法，分别尝试取 5、6、7 个点，分别得到最优取点组合方案。再对比两种不同算法得到的最优取点方案的关系。此外，还采用其他不同拟合方式进行比较分析，得出最终结论。研究方法框图如下：

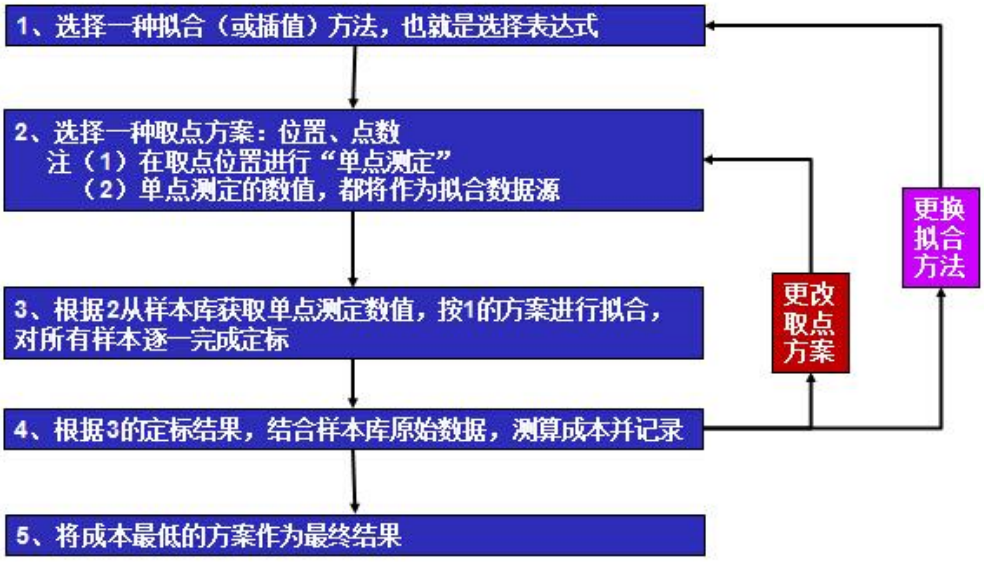

图 2.1.1 研究方法框图[2]

2.2 变量声明

表 2.2.1 变量声明

X	自变物理量，本题 X∈[5.0,5.1,...10.0]
Y	原始的被监测物理量
M	原始样本总个数
$\hat{Y}$	通过传感器自身的功能测算，即通过计算、拟合方式得到的数据
S(i,j)	误差成本，根据 Y 和 差值的大小进行分段函数确定，俗称“惩罚”
n	取点数目，即单点测定次数
Q	单点测定成本
S	测定的综合成本

3 方案设计

本实验需要选择测定点的组合方案，相当于解一个组合优化问题。而由于组合方案数目庞大，是典型的 NP-hard 问题，因为计算时间过长，目前的计算机尚难以通过暴力穷举的方法寻找出最优的测定点组合方案。因此，我们考虑采用启发式算法进行优化搜索。同时对测定点采用多种拟合方法进行拟合，并对比效果。

3.1 遗传算法（GA）

3.1.1 遗传算法概述[3]

遗传算法（Genetic Algorithms，简称 GA）是一种基于自然选择原理和自然遗传机制的搜索算法，它是模拟自然界中的生命进化机制，在人工系统中实现特定目标的优化。遗传算法的实质是通过群体搜索技术，根据适者生存的原则逐代进化，最终得到最优解或准最优解。

它必须做以下操作：初始群体的产生、求每一个体的适应度、根据适者生存的原则选择优良个体、被选出的优良个体两两配对，通过随机交叉其染色体的基因并随机变异某些染色体的基因后生成下一代群体，按此方法使群体逐代进化，直到满足进化终止条件。

3.1.2 遗传算法在本实验中的实现

（1）基因：认为 5.0—10.0 这 51 个点里面每个点都是一个基因，实验时分别假定基因数为 5,6,7。

（2）选取初始群体：从 5.0—10.0 这 51 个点中随机选取 n（n 为个体的基因数）个不同的点作为一个个体。

（3）适应度函数：对每一个个体解，可以在输入数据 X 的某一段取值区间拟合出一条曲线，然后和真实值比较，可以得到该区间上样本总体的平均成本。平均成本越低，适应度越大。适应度函数的大小反应了个体解的生存几率，适应度函数大的个体解生存几率大，反之相反。

（4）排序：将个体按适应度排序，然后进行下面的操作。

（5）种群选择：在计算种群中每个个体的适应度之后，将适应度最低的十个个体淘汰。然后由初始化函数重新生成十个个体代替他们。

（6）交叉：设交叉概率为 p，每个个体有 p 的概率进行交叉。在根据适应度排序之后，认为前 50%的个体为优势个体，后 50%的个体为非优势个体。为了提高效率，仅进行优势个体之间交叉，非优势个体与优势个体交叉。交叉得到的子代若优于原个体则保留，否则淘汰。

（7）变异：设变异概率为 q，每个个体有去的概率进行变异。变异的基因由随机函数随机产生。变异得到的个体若优于原个体则保留，否则淘汰。

（8）循环终止条件：循环的终止条件为循环的代数，该数值由人工指定。代码见附录。

3.2 模拟退火算法（SA）

3.2.1 模拟退火算法概述

模拟退火算法是用固体退火模拟组合优化问题，将内能 E 模拟为目标函数值 f，温度 T 演化成控制参数 t，即得到解组合优化问题的模拟退火算法：由初始解 i 和控制参数初值 t 开始，对当前解重复“产生新解→计算目标函数差→接受或舍弃”的迭代，并逐步衰减 t 值，算法终止时的当前解即为所得近似最优解。

模拟退火算法可以分解为解空间、目标函数和初始解三部分。

3.2.2 模拟退火算法在本实验中的实现

（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：产生新解 S′

（3）（计算增量Δ t′=C(S′)-C(S)，其中 C(S)为评价函数

（4）若Δ t′<0 则接受 S′作为新的当前解，否则以概率 exp(-Δ t′/T)接受 S′作为新的当前解.

（5）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。

（6）T 逐渐减少，且 T->0，然后转第 2 步

3.3 拟合方法

3.3.1 多项式拟合

多项式曲线拟合在实际中应用广泛，若用最小二乘法拟合，通常利用电脑进行矩阵运算能快捷方便地予以处理,多项式拟合的表达式见图 4.1.1：

$$y = p_0x^0 + p_1x^1 + p_2x^2 + \cdots + p_nx^n$$

图 3.3.1 多项式拟合示意图[1]

通过曲线的变化规律，很容易推断出，采用多项式进行拟合是合理的。但当次数增加时函数拟合速度会变慢，并且多项式拟合的阶次过大时在数值上可能拟合效果较好，但会引起拟合曲线的震荡，拟合出来的曲线与事实相差较大。在本实验中我们只讨论阶次较小、且能够拟合曲线变化规律的 3、4 次多项式拟合法。

（注：MATLAB 中已提供了 polyfit 函数，无需自行实现）

3.3.2 三次样条插值

三次样条插值法对于中间部分的相邻 4 个点采用三次曲线拟合，并在中间两点处做出拟合曲线；对于边界部分的 2 个点，选取与之靠近的三个点采用二次曲线拟合，并在靠近边界的两点处做出拟合曲线。如此操作，在断点处斜率和曲率都将连续。

（见图 4.1.2）

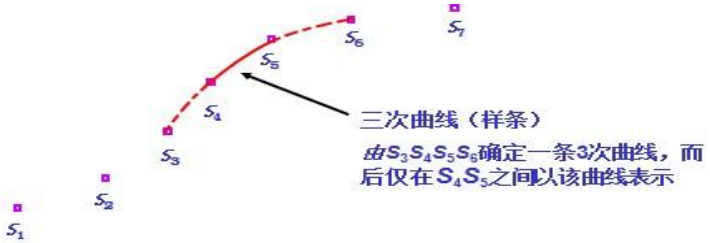

图 3.3.2 样条插值示意图[1]

（注：MATLAB 中已提供了 spline 函数，无需自行实现）

3.3.3 埃尔米特插值（Hermite）

许多实际插值问题中，为使插值函数能更好地和原来的函数重合，不但要求二者在节点上函数值相等，而且还要求相切，对应的导数值也相等，甚至要求高阶导数也相等。这类插值称作切触插值，或埃尔米特(Hermite)插值。

三次 Hermite 插值多项式（n=1）：

```
H3(x)=f(x0)(1+2(x0-x)/(x0-x1))((x-x1)/(x0-x1))^2
  +f(x1)(1+2(x1-x)/(x1-x0))((x-x0)/(x1-x0))^2
    +f'(x0)(x-x0)((x-x1)/(x0-x1))^2
      +f'(x1)(x-x1)((x-x0)/(x1-x0))^2                    （3-1）
```

（注：MATLAB 中也已提供了 pchip 函数，无需自行实现）

4 实验内容

4.1三种拟合方式对比

利用遗传算法，设定初始种群有100个个体，每个个体5个基因，遗传50代，交叉概率1，变异概率0.1。每种方案运行两次，对比三种拟合方法的效果。

表 4.1.1 三种拟合方式效果对比

插值方案	运行结果（取点）					成本
三次样条插值	3	14	26	39	49	105.7755
	3	9	29	43	48	109.4950
三次 Hermite 插值	4	17	27	36	48	**85.4860**
	4	16	26	35	48	**84.7482**

续表 4.1.1

插值方案		运行结果（取点）					成本
三次多项式插值	三次多项式	4	16	27	39	48	115.3660
	四次多项式	2	12	25	39	49	118.0360

分析实验数据可以发现：相同插值方法两次运行计算成本的结果相近，三次 Hermite 插值为三者中最优的插值方案，得到的取点方案成本最低，运行时间也很短。

4.2取点数分别为5,6,7时的结果对比

由4.1可知，Hermite插值的取得的最好，故采用Hermite插值。利用遗传算法，设定初始种群有100个个体，遗传30代，交叉概率1，变异概率0.1。分别取5,6,7个点每种取点个数运行两次，将得到的结果对比。

表 4.2.1 不同取点数的运行结果对比

取点个数	运行结果（取点）	运行结果（成本）
五个点	4 17 27 36 48	**85.4860**
	4 16 26 35 48	**84.7482**
六个点	3 12 23 30 40 48	87.6158
	3 15 23 30 38 49	86.7888
七个点	2 12 20 26 33 39 49	92.6290
	3 11 19 25 31 39 49	92.7620

分析实验数据可以发现:同样的取点数目多次运行结果相近，选取5个样本点进行三次Hermite插值时，算得的成本最低，其次是6个点，选7个点成本则偏大。因而我们选用5个点进行三次Hermite插值，可以获得较优的插值拟合结果。

4.3遗传算法最优解与模拟退火验证

由4.1和4.2可得，当拟合方法为Hermite插值，取点数为5时的成本最小。此时的最优结果如下：

表 4.3.1 遗传算法最优解

取点方案1	**4**	**16**	**26**	**35**	**48**
成本	**84.7482**				
取点方案 2	**3**	**15**	**26**	**36**	**49**
成本	**84.7482**				

将该取点方案得到的拟合曲线和表中第一组数据用MATLAB画出。（如图4.3.1）

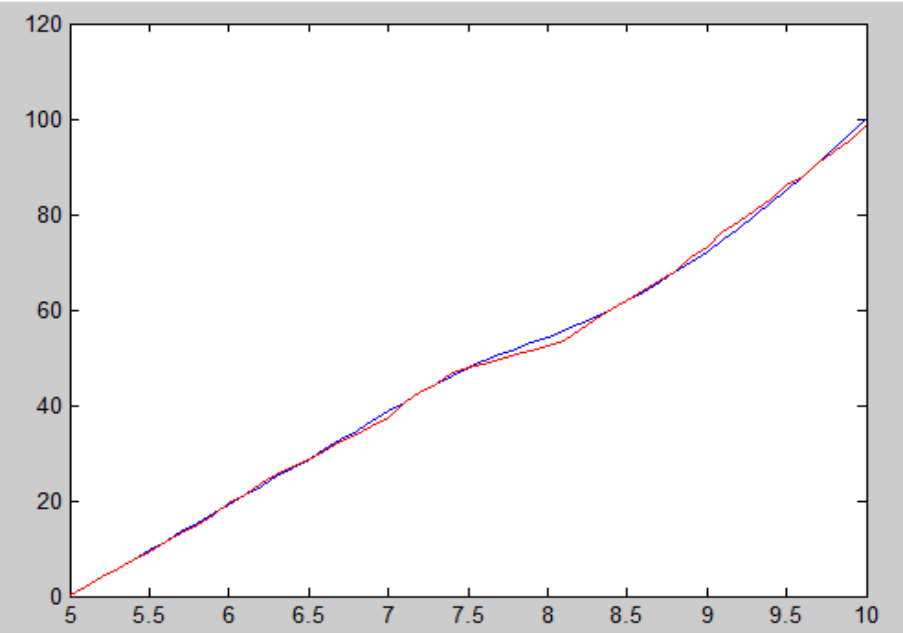

图 4.3.1 拟合曲线与实际曲线对比

图中红色曲线为由第一组数据画出的实测曲线，蓝色为由Hermite插值计算上述最优取点方案1得到的拟合曲线。从图中可以看出二者十分接近。

然后利用模拟退火算法寻找最优取点组合：经多次实验，我们决定把初始温度设定为100，终止温度设定为 0.001，降温比例为 97%。此种设定可以得到较好的运行结果。并且采用 Hermite 插值，取点数为 5 时的成本最小。多次运行得到的最优结果如下：

表 4.3.1 模拟退火算法最优解

取点方案	4	16	26	35	48
成本	84.7482				

可见两种方法得到的取点方案与成本完全一致，这也能够在一定程度上说明这个解接近或者就是最优解。

4.4结果分析

4.4.1结果分析

在遗传算法和Hermite插值，取5个点的情况下，设定初始种群有100个个体，遗传20代，交叉概率1，变异概率0.1。进行多次实验，将得到的结果列在下表。

表4.4.1遗传算法结果

取点方案 1	**4**	**16**	**26**	**35**	**48**	**成本**	**84.7482**
取点方案 2	2	16	26	35	48	成本	85.798
取点方案 3	**3**	**15**	**26**	**36**	**49**	**成本**	**84.7482**
取点方案 4	4	16	26	35	49	成本	85.1438

分析结果可知，最后得到的成本基本上都在85附近，但是实验时偶尔会出现取点方案相差很大的情况。

在采用遗传算法与Hermite插值，取5个点的情况下，比较的每一代的最小成本值与平均成本值的收敛快慢情况。（如图4.4.1）

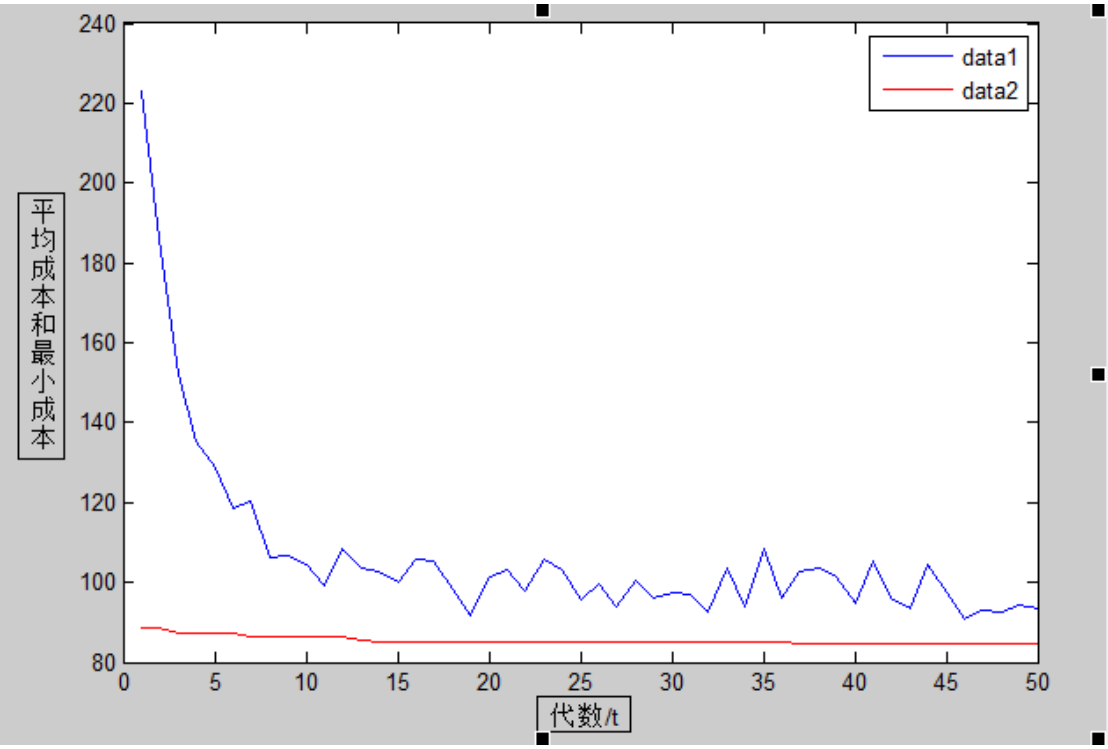

图 4.4.1 遗传算法收敛情况

蓝色为平均成本，红色为最小成本由图可看出最小成本单调减小，平均成本整体在减小。但是由于种群选择函数在淘汰后 10 个个体之后，随机产生了 10 个移民个体加入种群，移民的适应度并没有保证，所以导致平均成本没有单调地减小，而是出现了波动。

4.4.2 存在的问题

对 4.4.1 节的结果进行分析可知，目前的结果存在如下三个问题：

（1）虽然误差成本基本都在 85 附近，但是多次实验后发现出现了取点方案相差很大的情况。即得到的取点方案不稳定，失去了在某个优化解附近找到更优解的机会。

（2）出现了早熟现象，连续很多代的最优解相同，程序陷入局部最优解，此后即便程序在运行也无法找到更好的解，浪费了计算时间。

（3）种群选择函数的缺陷导致平均成本出现振动，并不是严格单调递减。

5 拓展——基于遗传算法的优化

为了解决以上三点问题，我们对算法进行了优化，：

优化一：对变异函数优化，当成本小于一定值的时候，缩小变异范围，解决取点方案不稳定的问题，从而能够在优化解附近找到局部的最优解。

优化二：对选择函数进行优化，对移民的进入加设条件，从而解决平均成本的振动问题，有利于提高算法的效率。

优化三：对主函数进行优化，增加容忍最优解相同的最多代数。从而解决早熟问题。

5.1对变异函数优化

5.1.1算法设计

直接解决取点方案不稳定的问题比较复杂，我们做了一些假设，将问题简化，从而对变异函数优化。

假设一：假定初始种群的个体数足够大时，种群中的个体就已经包含了全部最优秀的基

因。

假设二：在一个优秀解的附近变动，往往可以找到适应度更高的个体。

因此我们可以得到结论：由假设一和二可得，在大种群得到的最优个体附近找到的局部最优解往往可以代表整体最优解。

基于上述结论，我们提出了优化方案：当成本小于 90（90 是由经验得出：成本小于 90 的解已经很好）的时候，缩小变异的范围。而成本大于 90 时，扩大变异范围。

5.1.2 结果展示

在采用遗传算法和 Hermite 插值，取 5 个点的情况下，设定初始种群有 100 个个体，遗传 20 代，交叉概率 1，变异概率 0.1。进行五次实验，将得到的结果列在下表。

表 5.1.1 变异函数优化之后的结果

最后取点方案					成本
4	17	27	36	48	85.4860
5	15	26	35	49	85.1438
4	**16**	**26**	**35**	**48**	**84.7482**
5	14	27	32	47	85.3917
4	**16**	**26**	**35**	**48**	**84.7482**

观察上述结果可知，尽管只遗传 20 代，但与之前相比最终结果较为稳定。说明对变异函数的优化有效果。

5.2对选择函数优化

为了使平均成本的收敛曲线单调减小，我们对移民的进入加设条件：适应度更大的移民才能取代原个体进入种群。

优化后进行实验：在采用遗传算法和 Hermite 插值，取 5 个点的情况下，设定初始种群有 100 个个体，交叉概率 1，变异概率 0.1，遗传 50 代。得到了每一代的最小成本值与平均成本值的收敛快慢情况曲线图。（如图 5.2.1）

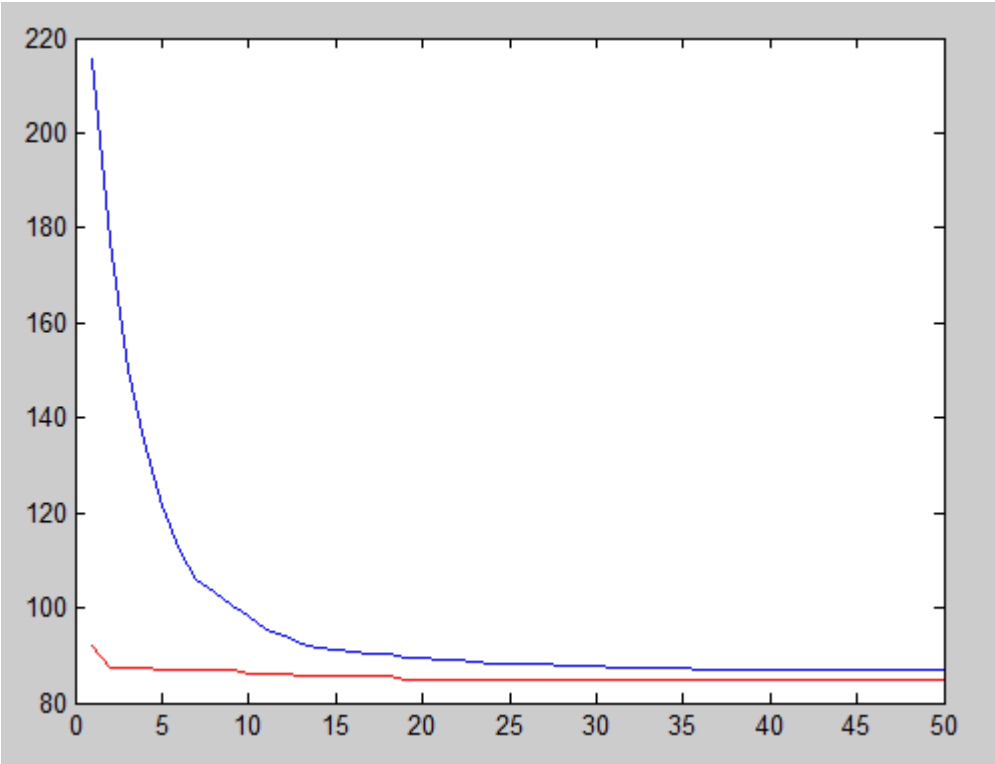

图 5.2.1 最小成本值与平均成本值收敛情况曲线图

图中蓝色为平均成本，红色为最小成本。从图中可以看出平均成本单调减小；由数据可见遗传算法得到最优解的效率也有提高。

5.3对主函数优化

5.3.1算法设计

前面的多次实验可以看出，“早熟”现象严重。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，那么我们就强制修改变异概率为1，同时每次进行种群选择的淘汰数量改为20。

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，而没有发挥最大作用。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。

5.3.2结果对比

在采用遗传算法和Hermite插值，取5个点的情况下，设定初始种群有100个个体，遗传50代，交叉概率1，变异概率0.1。

优化前的最小成本随代数收敛图与优化后的最小成本随代数收敛图如下：

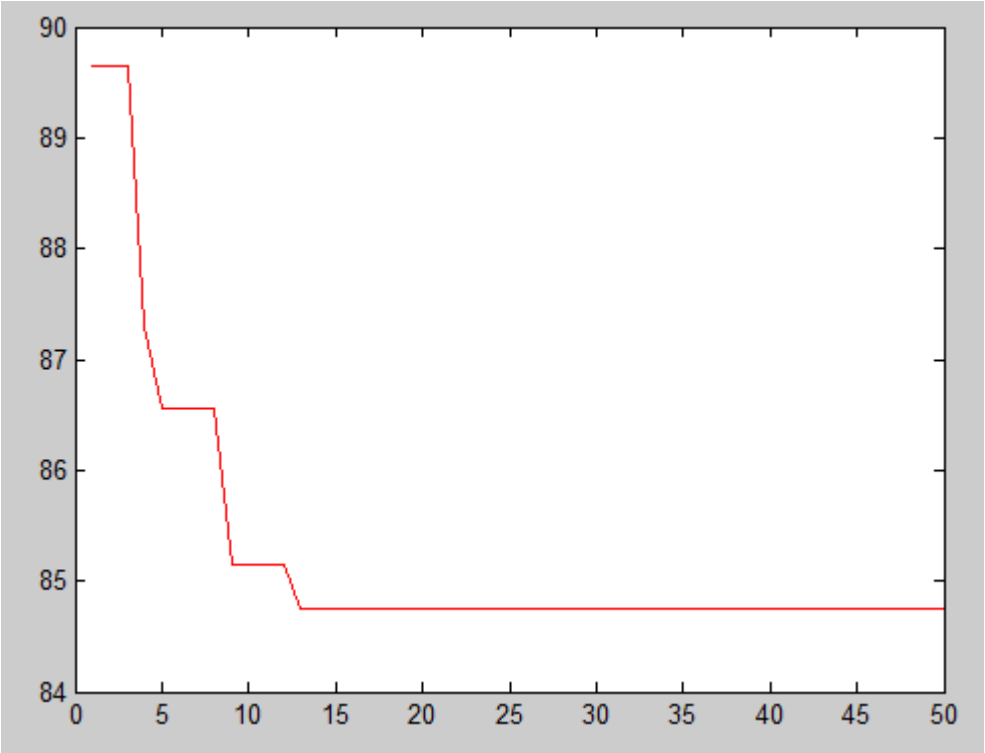

图 5.3.1 优化前的最小成本随代数收敛图

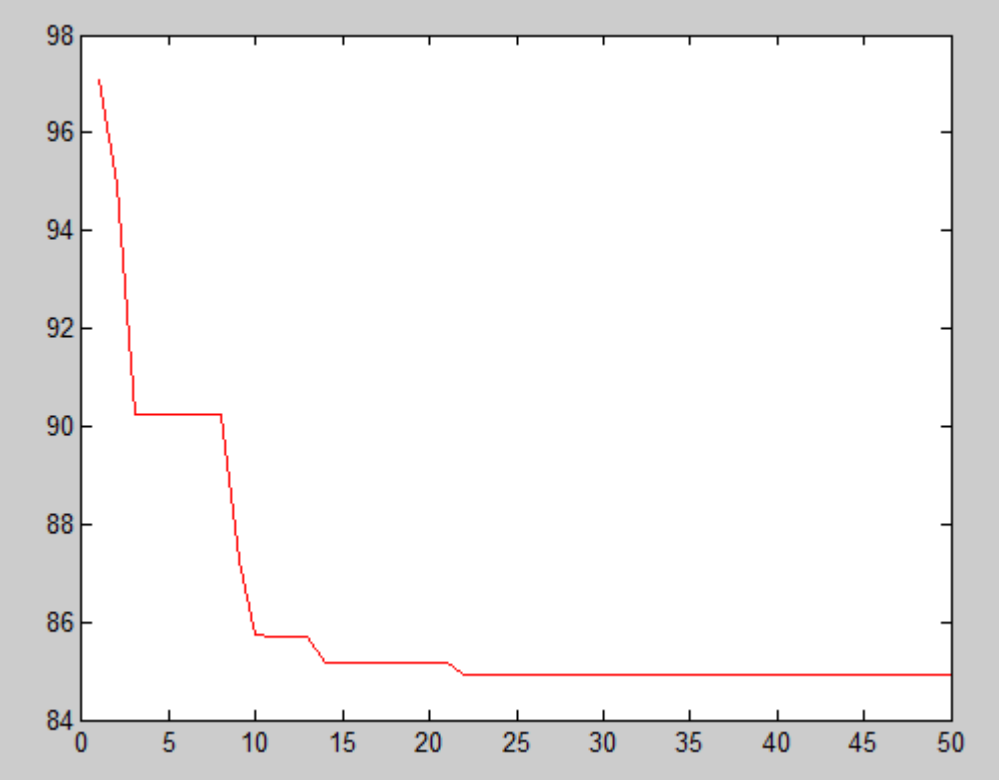

图 5.3.2 优化后的最小成本随代数收敛图

两图对比可知：优化后曲线略微比优化前曲线晚熟。

6 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取点方案结果不稳定、早熟、平均成本收敛曲线不单调等问题。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，在希望总体误差最小的情况下，可以使用三次 Hermite 插值来完成传感特性的校准。经过多次实验，可得到如下四组取点组合都可得到最优解：

表 6.1.1 实验结果

最优取点方案					成本
2	15	25	34	48	84.7482
3	15	26	36	49	84.7482
4	16	26	35	48	84.7482
7	16	26	36	46	84.7482

7 参考文献

[1] 网络资源. http://baike.baidu.com/

[2] 袁焱. 统计推断讲座 3：问题求解的途径

[3] 算法大全[M].http://wenku.baidu.com/link?url=5c-XOHZRWDEiTMHCzV5X_4mkNrP7rGofC-M0YiS3bvVSr0OnSxogGjrzJ5pFi0zpWjUA_TtXGmMJOQc1-nwreHY_5QMZK0jZSAPux9Txtpe

8 附录：程序代码清单

8.1遗传算法

8.1.1主函数

```
function [ points,MinCost ] = genetic_main( N,n,T,p,q )
% N为种群大小，T为迭代次数，n为每个个体基因个数,p为交叉概率，q为变异概率
% points为遗传算法得到的最优取点组合，MinCost为最少成本

tic % 开始计时
a_import_data;

min = zeros(2,T); % min记录每一代的最小值
ave = zeros(2,T); % ave记录每一代的平均值
x = 10; %每次选择的淘汰个体数
lastmin = 0; %记录上一代最小值
counter = 0;

CostArr = zeros(1,N); %CostArr记录种群的适应度，值越小适应度越高
group = genetic_Init(N,n); % 产生初始种群
for i=1:1:N %计算个体成本
    CostArr(i)=CountCost(group(i,:),data,n);
end
% 先将父代按成本进行排序；
[CostArr,Index]=sort(CostArr,'descend');
group=group(Index,:);
MinCost = CostArr(N);

%模拟遗传过程
t = 1;
while t<=T
    % 对父代进行选择
    [group,CostArr] = genetic_Select(group,CostArr,data,n,x);
    if MinCost>CostArr(N)
        MinCost=CostArr(N);
        points=group(N,:);
    end

    % 优势个体交配，非优势与优势交配
```

```
    [group,CostArr] = genetic_Cross(group,CostArr,N,p,data,n);

    % 父代进行变异
    [group,CostArr] = genetic_Mutate(group,CostArr,N,q,data,n);

    % 子代对成本进行降序排序，优势个体在下方；方便下一代进行选择、变异、交配
    [CostArr,Index] = sort(CostArr,'descend');
    group=group(Index,:);

    ave(1,t) = t;
    ave(2,t) = mean(CostArr);
    disp('平均成本：');
    disp(mean(CostArr));

    min(1,t) = t;
    min(2,t) = CostArr(N);
    disp('最小成本：');
    disp(CostArr(N));
    disp(group(N,:));

    disp('new generation!');
    disp(t);
    t = t+1;

    if lastmin==CostArr(N)
        counter = counter+1;
        if counter>=5
            counter = 0;
            x = 20;
            q = 1;
        end
    else
        counter = 0;
    end
    lastmin=CostArr(N);
end

toc % 记录时间

plot(min(1,:),min(2,:),'-r'); %ave(1,:),ave(2,:),

end
```

8.1.2变异函数

```
function[ group,CostArr ] = genetic_Mutate( group,CostArr,N,q,data,n )
% 变异
for i=1:1:N
    if rand<=q
        while 1
            % 在成本小于90时减小变异幅度
            if CostArr(i)<90
                delt=randi([-1,1]);
            else
                delt=randi([-5,5]);
            end
            % 由随机函数产生突变的基因
            MutNum=randi(n);
            child=group(i,:);
            child(1,MutNum)=mod(group(i,MutNum)+delt,49)+2;
            child = sort(child);

            % 判断子代合法性，如果不同的基因座上有不同基因，则跳出
            if length(unique(child))==n
                Cost=CountCost(child,data,n);
                % 子代更优，则取代亲代
                if CostArr(i)>Cost;
                    group(i,:)=child;
                    CostArr(i)=Cost;
                end
                break;
            end
        end
    end
end
end
```

8.1.3初始化函数

```
function [ group ] = genetic_Init( N,n )
% 初始种群中有 N 个个体，每个个体 n 个不同基因

group = zeros(N,n);
for i=1:N
    group(i,:) = sort(randperm(49,n));
end

end
```

8.1.4选择函数

```
function [ group,CostArr] = genetic_Select( group,CostArr,data,n,x)
%适应度低的相当于被移民取代，移民的是原始种群

child = zeros(x,n);
child(1:x,:)=genetic_Init(x,n);

for i = 1:1:x
    tmpcost = CountCost(child(i,:),data,n);
    if tmpcost<CostArr(i)
        group(i,:) = child(i,:);
        CostArr(i) = tmpcost;
    end
end
end
```

8.1.5交叉函数

```
function [group,CostArr] = genetic_Cross( group,CostArr,N,p,data,n)

% 优势个体交叉
for i=N:-2:(N/2+2)
    %为了减少代数，p都取100%
    if rand<=p
        while 1
            % 两个初代产生两个后代
            % exchange表示交配过程中从哪个基因开始交换
            exchange=randi(n);
            child1=[group(i,1:exchange),group(i-1,exchange+1:n)];
            child2=[group(i-1,1:exchange),group(i,exchange+1:n)];
            % 判断子代合法性，若出现不同基因座上有相同基因，重新交叉
            if (length(unique(child1))==n )&&(length(unique(child2))==n )
                Cost1=CountCost(child1,data,n);
                Cost2=CountCost(child2,data,n);
                if CostArr(i-1)>Cost1
                    if CostArr(i)>Cost1
                        group(i,:)=child1;
                        CostArr(i)=Cost1;
                    end
                    group(i-1,:)=child1;
                    CostArr(i-1)=Cost1;
                end
                if CostArr(i-1)>Cost2
                    if CostArr(i)>Cost2
                        group(i,:)=child1;
                        CostArr(i)=Cost2;
```

```
                    end
                    group(i-1,:)=child1;
                    CostArr(i-1)=Cost2;
                end
                break;
            end
        end
    end
end

%适应度低的与适应度高的进行交叉，下标从N/2到1
for i=N/2:-1:1
    if rand<p
        exchange2=randi(n);
        child1=[group(i,1:exchange2),group(i+N/2,exchange2+1:n)];
        if (length(unique(child1))==n )
            Cost1=CountCost(child1,data,n);
            if CostArr(i)>Cost1
                group(i,:)=child1;
                CostArr(i)=Cost1;
            end
        end
    end
end

end
```

8.2 成本计算与拟合函数

8.2.1 成本计算函数

```
function [ cost ] = CountCost( point,data,n )
%  成本计算函数
% point为待计算的取点方法，n为取点个数

%  标准样本原始数据读入
[M,N]=size(data);
x = data(1:2:end,:);
y = data(2:2:end,:);
y1=zeros(M/2,N);

for i=1:1:400
    a = x(i,point);
    b = y(i,point);
    y1(i,:) = b2_pchip(a,b);
```

```
        % 三次样条插值用y1(i,:) = b1_spline(a,b);
        % Hermite插值用y1(i,:) = b2_pchip(a,b);
        % 三次函数拟合用y1(i,:) = b3_poly3(a,b);
        % 四次函数拟合用y1(i,:) = b4_poly4(a,b);
        % 五次函数拟合用y1(i,:) = b5_poly5(a,b);
end

% 计算定标误差成本
errabs=abs(y-y1);
case0_6 = (errabs>0.4 & errabs<=0.6);
case0_8 = (errabs>0.6 & errabs<=0.8);
case1_0 = (errabs>0.8 & errabs<=1);
case2_0 = (errabs>1 & errabs<=2);
case3_0 = (errabs>2 & errabs<=3);
case5_0 = (errabs>3 & errabs<=5);
caseend = (errabs>5);
err_cost = 0.1*case0_6 + 0.7*case0_8 + 0.9*case1_0 + 1.5*case2_0 + 6*case3_0 + 12*case5_0 +
25*caseend;

% 计算总成本
si = sum(err_cost,2);
cost = sum(si)/400 + 12*n;

end
```

8.2.2 三次样条插值函数

```
function y1 = b1_spline( x_det,y_det )
%对测定点进行三次样条插值

x = 5.0:0.1:10;
y1=interp1(x_det,y_det,x,'spline');

end
```

8.2.3 Hermite插值函数

```
function y1 = b2_pchip( x_det,y_det )
%对测定点进行三次Hermite插值

x = 5.0:0.1:10;
y1=interp1(x_det,y_det,x,'pchip');

end
```

8.2.4 三次函数拟合函数

```
function y1 = b3_poly3( x_det,y_det )
%三次函数拟合

x=5:0.1:10;
f=polyfit(x_det,y_det,3);
y1=polyval(f,x);

end
```

8.2.5 四次函数拟合函数

```
function y1 = b4_poly4( x_det,y_det )
%四次函数拟合

x=5:0.1:10;

f=polyfit(x_det,y_det,4);
y1=polyval(f,x);

end
```

8.3模拟退火算法及其插值函数

8.3.1模拟退火算法（选点函数集成了成本计算函数）

```
%%%%%%%   模拟退火（SA）寻找选点组合     %%%%%%% No.27

minput=dlmread('20150915dataform.csv');%读入数据
x=minput(1:2:end,1:end);%取出 X
y=minput(2:2:end,1:end);%取出 Y

all=randperm(51);                    %随机打乱 51 个点
choice=[ 1,11,21,31,41,51 ];%初始化为平均取点

score=0;        %记录优化成本
score_save=0;%保存上一次分数
cost=0;          %当次成本
num=0;            %模拟的次数
choice_min = choice;%记录取点

Tf=0.001;          %终止温度<---
Ts=100;             %初始温度<---

tic;%开始运行记录时间
while Ts>Tf
      num=num+1;
```

```
remain=setdiff(all,choice);     %寻求 all,choice 差集
A=remain(randperm(45));%再次打乱顺序     <----
B=randperm(6);              %                    <----此处修改选点数目
C=choice;                   %当前的新的选点方案
C(B(1))=A(B(1)+1);          %替换其中一个
C=sort(C);                  %%%%%%%随机替换取点方案中的一个位置

%-------------------------------------------------
%%%%%%%%%  成本计算测试函数---start %%%%%%%%%%%%%%%
my_answer=C;
my_answer_n=size(my_answer,2);

[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
    %  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;
    %%%%%%%%  成本计算测试函数---end %%%%%%%%%%%%%
    %-------------------------------------------------


    if num==1 %第一次计算成本
        score=cost;
        score_save=cost;
        choice_min=C;
        choice=C;
    elseif cost<score %花费代价小于上一次
        fprintf('\n 当前最优总成本为%5.2f\n',cost);
        score=cost;
        score_save=cost;
        choice_min=C;
        choice=C;
    elseif rand<exp((score_save-cost)/Ts) %决定点是否变化
        score_save=cost;
        choice=C;
    end
    Ts=Ts*0.97;%降温
end

toc; %记录结束的时间

choice_min    %输出取点方案
fprintf('\n 据此选点方案，总成本为：%5.2f\n',score);
```

8.3.2插值函数

```
function y1 = mycurvefitting( x_premea,y0_premea )

x=[5.0:0.1:10.0];

%  定标计算方法写成指令代码如下：
%y1=interp1(x_premea,y0_premea,x,'spline');          %三次样条插值
y1=interp1(x_premea,y0_premea,x,'pchip');            %三次 Hermite 插值
%y1=polyval(polyfit(x_premea,y0_premea,3),x);    %三次多项式插值（3->n,则为 n 次多项式插
```

值）

end
%%%通过调整注释号“%”，可以改变插值方法