统计推断在数模转换系统中的应用

组号：22 组 巴若珉 5130309710 赵庆浩 5130309709

摘要：本课题的问题来源是为某种产品寻求校准工序的优化方案，通过采样进行曲线拟合。利用已知数据推断变化趋势估算拟合数据大致经历拟合和插值两个步骤。[1]在定标过程中，定标成本与拟合精度互为矛盾关系，为了找到一个平衡点，使总成本最小我们进行了一系列探索。我们首先进行了多种拟合方式的探索，确定采用三次样条插值为拟合方法；然后采用遗传算法计算出最佳的选点方案，最后还进行了小范围穷举、粒子群优化算法等拓展，从而进一步优化。

关键字： 三次样条插值法，拟合，遗传算法

ABSTRACT: The source of this problem is the optimization scheme for the calibration process for a product,and curve fitting is performed by sampling. Estimation of data fitting and interpolation fitting experienced two steps using inference known data change trend. During the calibration, the calibration cost and fitting precision are in contradictory relations.In order to find a balance and to minimize the total cost, we conducted a series of exploration. We first explored various fitting methods, determined using cubic spline interpolation for fitting method; then using genetic algorithm to calculate the optimal selecting scheme.Finally we also try expanding small range exhaustive, particle swarm optimization algorithm, for further optimization.

Key words: cubic spline interpolation method，fitting，genetic algorithm

1、引言

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。现有的测量数据是每组 51 个测量点，共 469 组。其中，每组数据对应于一个元件样本，输入与输出的对应关系也不同，但是大致的函数图形是一致的。因此，我们先随机挑选 2 组数据，在每组数据中以全部数据为样本分别进行 4 次多项式拟合，指数拟合和三次样条插值拟合方式,然后根据拟合曲线的质量评估，选择适当的拟合方法。由于测量本身具有很高的成本，所以不能无限取点来降低误差。所以用选定的拟合方法并依据给定的成本函数，通过遗传算法来确定最优的取点，从而使总成本最小。最后由实验中发现的规律，又进行了很多的探索，用以优化定标方案的选择。[2]

2、基础理论

在本次实验中涉及的理论主要有四次多项式拟合、三次样条插值等拟合方式；以遗传算法为代表的启发式算法；以及课程中介绍的成本计算函数等必要的数学公式。

2.1 三次样条插值法

三次样条插值（简称 Spline 插值）是通过一系列形值点的一条光滑曲线，数学上通过求解三弯矩方程组得出曲线函数组的过程。

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。

实际计算时还需要引入边界条件才能完成计算。边界通常有自然边界（边界点的二阶导为 0），夹持边界（边界点导数给定），非扭结边界（使两端点的三阶导与这两端点的邻近点的三阶导相等）。一般的计算方法书上都没有说明非扭结边界的定义，但数值计算软件如 Matlab 都把非扭结边界条件作为默认的边界条件。

2.2 启发式算法

2.2.1 遗传算法

遗传算法就是模拟进化论的优胜劣汰的规则，并加有概率性质的随机遗传的算法。算法中产生个体采用产生 1 至 53 的 7 个随机数的方法，而从父代向子代的遗传主要分三种方式：1、选择（selection），选择父代中优秀的个体直接传递给下一代，即保留父代。2、交叉（crossover），以两个父代的基因为基础，进行交叉重组，生成下一代，好的基因会有更大的概率传给子辈。3、变异（mutation），每个父代个体都有一定的概率发生随机的基因突变。[3]

评价基因好坏的依据就是适应度。在遗传算法的步骤中起着突出作用的其实是变异。正因为有了变异，算法才存在在全局范围内寻找到最优解的可能性。但是遗传算法需要通过反复试验，才有可能给出最优解。

2.2.2 粒子群算法

粒子群算法中，每个优化问题的潜在解都是搜索空间中的一个粒子。所有的粒子都有一个由被优化的函数（q）决定的适值，每个粒子还有一个速度决定它们飞翔的方向和距离。然后粒子们就追随当前的最优粒子在解空间中搜索。

粒子群算法首先初始化一群粒子，这些粒子是随机产生的，每一个粒子相当于一组解（粒子可以是任意维数的，即解是多重的、离散的）。然后通过迭代找到最优解。在每一次迭代中，粒子通过跟踪两个极值来更新自己；第一个就是粒子本身所找到的最优解，这个解称为个体极值；另一个极值是整个种群目前找到的最优解，这个极值是全局极值。

2.3 成本计算函数

为评估和比较不同的校准方案，特制定以下成本计算规则。

（1）单点定标误差成本[4]

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

单点定标误差的成本按式（2-1）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。

（2）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。

（3）某一样本个体的定标成本

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

对样本 i 总的定标成本按式（2-2）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。

（4）校准方案总体成本

按式（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。

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

总体成本较低的校准方案，认定为较优方案。

3、拟合方法选择

3.1 多种拟合方式比较

我们发现，每个元件的输入输出对应函数图象都是形状相仿的，因此我们随机抽取多组数据分别用四次多项式拟合、指数拟合、对数拟合等拟合方法，得到 X 与 Y 的对应关系，并将其与实测数据进行比较。下面选取一组数据展示探究过程。

第一组数据：

表 3.1 数据样本

5	5.1	5.2	5.3	5.4	5.5	5.6	5.7	5.8
5.33	7.03	8.92	10.74	12.45	14.23	16.05	17.77	19.58
5.9	6	6.1	6.2	6.3	6.4	6.5	6.6	6.7
21.31	23.07	24.87	26.59	28.34	30.18	31.92	33.78	35.64
6.8	6.9	7	7.1	7.2	7.3	7.4	7.5	7.6
37.5	39.31	41.14	42.99	44.91	46.79	48.63	50.53	51.47
7.7	7.8	7.9	8	8.1	8.2	8.3	8.4	8.5
52.41	53.27	54.12	55.12	56.23	57.27	58.34	59.41	60.51
8.6	8.7	8.8	8.9	9	9.1	9.2	9.3	9.4
61.67	64.04	66.38	68.72	71.14	73.62	76	78.46	81
9.5	9.6	9.7	9.8	9.9	10			
83.59	86.43	88.97	91.76	94.92	98.05			

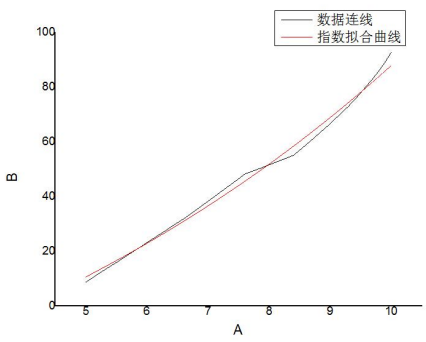

图 3.1（1）指数拟合曲线

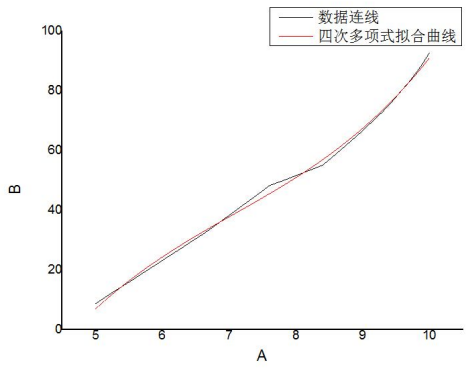

图 3.1（2）四次多项式拟合曲线

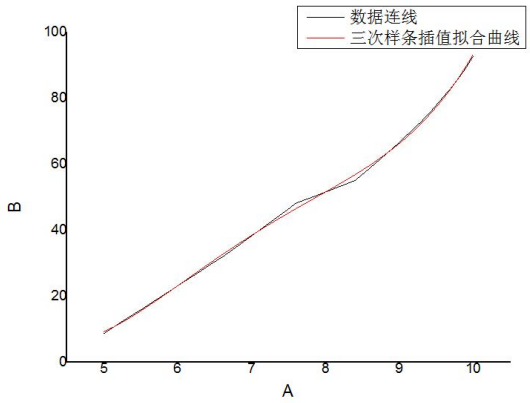

图 3.1（3）三次样条插值拟合曲线

通过 origin 进行线性拟合，由图中数据可以看出，三次样条插值与真实曲线的贴合度最好，所以暂时选取三次样条插值拟合方式为拟合方法。为了进一步验证我们的猜想，我们做出了拟合函数值与真实值误差的分布图：

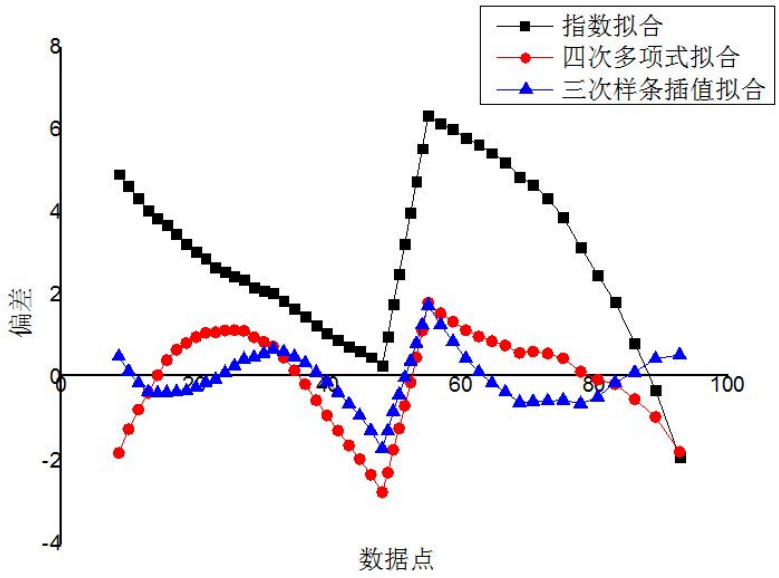

图 3.1（4）误差分析图

从散点图中可见，三次样条插值偏差更靠近横向轴（即偏差更接近 0），四次多项式拟合居其次，指数拟合偏差最大。综合比较可知，应选择三次样条插值法对数据进行拟合。沿用以上所取的数据点组合，我们随机选取了 50 组数据进行拟合，依照准确度评价方法进行评价，三次样条插值的偏差最小。由此可知，三次样条插值法的准确度可以保证。

3.2 分段拟合的尝试

由于 51 个数据具有明显的分界点，基于此，我们提出了先求出数据的平均分界点；并以此为依据，在每一段内进行三次多项式拟合，已得到更高的贴合度，进一步减小误差。所以我们组额外探索了从 origin 图像中可以看出斜率明显变化的拐点如图：

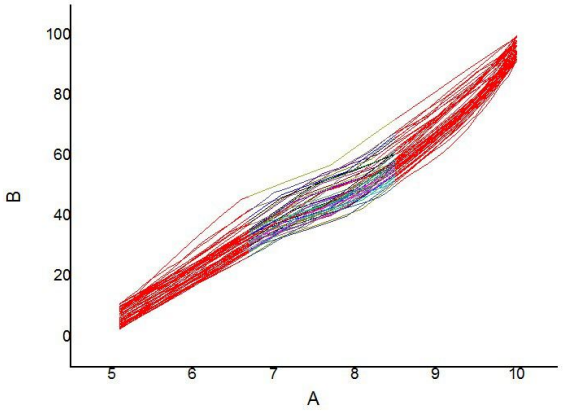

图 3.2 50 组数据的分段图像

从图像中我们很明显能够看到问题所在，即中段的失真率非常高。所以我们又进行了分段函数拟合，即在每一段进行不同的方式拟合。由于每一段的线性程度很高，故假定每一段为线性曲线，然后选取 20 组数据，先选取一些一定在三段各段上的点，然后拟合出三条直线，求出两个交点，然后对样本的交点的 x 值求平均，视为端点。

经过计算可得，交点的平均值为 7.317771272 和 8.470376999，所以认分三段的曲线分界点为 x=7.317771272 及 x=8.470376999。前后两段用三次函数模拟，中间一段由于较为接近线性函数所以用一次函数模拟，之后再利用遗传算法分别求出每段的特征点。通过这种方法得出的成本为 124.578，由于成本过高故此种方法仅作为尝试，没有得出更精确的结论。

另外这一尝试在我们初始的遗传算法中有所应用，在初始的遗传算法中我们固定选点数为 6，而初始化 200 个种群时，我们设计在每一段段中分别任意选取两个点；这样的初始种群要由于在 51 个点中随机选取 6 个点的初始化方法，可以加速收敛。但是后来我们认为固定选点数有一定的局限性，优化了遗传算法，也随之放弃了这一初始化方法。

4、选点方案探究

4.1 遗传算法

4.1.1 初步的遗传算法

由于初期经验不足，我们组独立完成的遗传算法代码能够完成种群优化，但是存在诸多问题：

（1）固定选点个数，使变异范围大大降低，无法真正搜索到最优的解。

（2）设置初始种群个数为 50，种群数过少，不满足要求。

（3）变异率过大，我们将变异率设置为 5%，不满足设置为 1%-0.1%的要求。

（4）使用过多 for 循环，使程序运行速度过慢。

4.1.2 改进的遗传算法

改进的遗传算法主要对代码的 for 循环进行了改进，将其改进为矩阵形式，运算方法也改进为矩阵运算，这样可以大大加快运行速度；并对初始化进行改进，不再限定选点个数，从而能够通过遗传算法确定最优的六个点，尽管结果相同，但可信度加强；进行参数调整，加大种群数量，减小变异率，增大迭代次数。

（1）参数设置

pop_size=200（种群大小应大于 100）

chromo_size=51（染色体长度）

generation_size=200（迭代次数）

cross_rate=0.7（交叉概率）

mutate_rate=0.005（变异概率在 1%-0.1%之间）

（2）算法流程

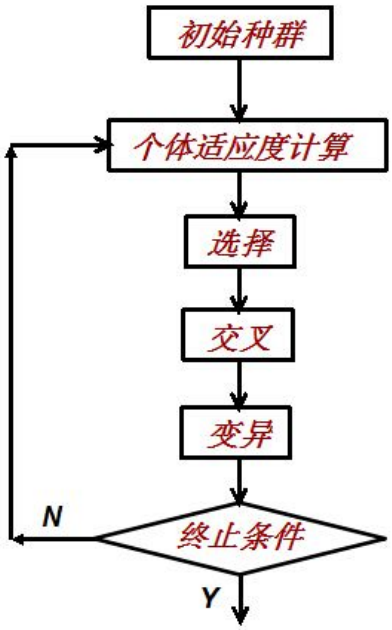

图 4.2 遗传算法流程图

（3）运行结果

本方法运行结果如下

6 points at:

2 11 22 31 41 50

with score 94.1440

4.2 遗传算法结合遍历运算的尝试

通过前几代的遗传算法我们观察到几组特征点的分布如下：

3 12 20 28 44 47 94.3596

3 10 25 32 43 51 93.9370

1 12 24 31 46 50 94.9995

即选点在某几个数范围内摆动，而且摆动范围很小。虽然我们通过遗传算法解决了 NP hard 问题，但是只是接近最优解。由于此时这时变化范围很小，所以我们尝试用遍历的方法找出最优解，在以下六个范围内进行遍历，共 8100 种：

1-3，10-12，20-25，28-32，41-46，47-51

得到最优解为 92.8774，特征点是：3 12 22 31 43 50。

由于单纯的遍历是无法完成的任务，而如果单纯的用遗传算法又无法精确的找到最优解。所以我们组最后认为遗传算法加上小范围的遍历是确定选点方案的最佳方式：先通过遗传算法找出几组解，然后可以发现每一个点的数值都在某个值为中心的范围内摆动，在范围内进行遍历，即可得到最佳的选点方案。[5]

5、自主探究

5.1 减少为 5 个观测点的尝试

我们组尝试能否减少一个观测点，如果用我们改进后的遗传算法，每次得出的解都为 6 个参数，我们用限定选点个数的遗传算法计算五个参数的成本（仍采用三次样条插值的拟合方式）得出的结果在 100-105 之间，不是很理想。

所以我们变换了一种思路，即选点方案选定为 3 12 22 31 43 50，但是我们只测 3 12 22 31 43 这 5 个点，希望能找到一个 f（x1, x2, x3, x4, x5)进行第六个点值的预测。我们观察到第五个点和第六个点存在一点的线性相关关系。

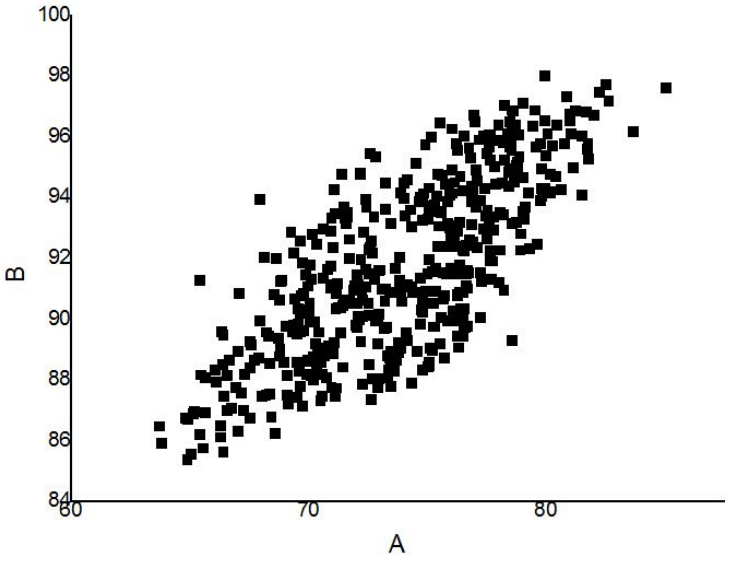

图 5.1 第五个点与第六个点关系图

随后我们对预测函数进行了一系列的尝试，得到了一个较好的预测函数。

$$X_6=0.64931(X_1+X_2+X_3+X_4+X_5)+39.12963 \qquad (5\text{-}1)$$

但是它的运行结果并不是很理想，总成本仍然维持在 100 左右，没有进一步降低成本。我们认为原因主要有没有找到更加合理的函数关系；或者我们认为第六个点和前五个点本身不存在精确的函数关系，所以无法确定。

```
function y1 = mycurvefitting_baruomin( x_premea,y0_premea )
x=zeros(1,6);
y=zeros(1,6);
for i=1:5
    x(1,i)=x_premea(i);
    y(1,i)=y0_premea(i);
end
x(1,6)=9.9;
y(1,6)=0.64931*y(1,5)+39.12963;

x=[5.0:0.1:10.0];
y1=interp1(x_premea,y0_premea,x,'spline');
end
```

5.2 粒子群算法

为了比较不同启发式算法的优点和缺点，我们又用粒子群算法进行了优化选点，然后通

过对两者进行了比较。

5.2.1 粒子群算法流程

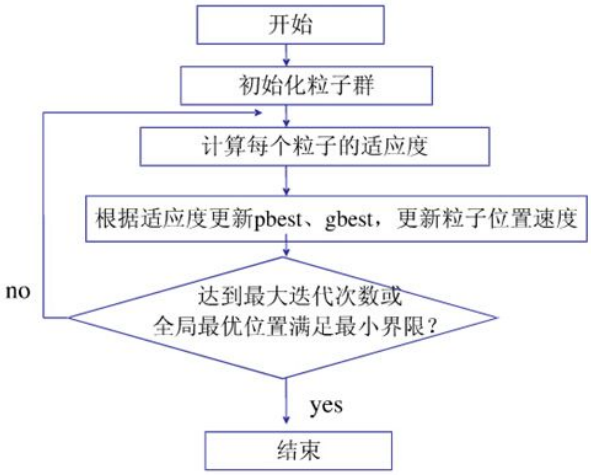

图 5.2 粒子群算法流程[6]

5.2.2 程序设计

程序按照 5.2 的流程图设计：

(1) randint 函数随机产生一个 51×10 的初始特征点矩阵（粒子个数为 10）

(2) 计算每组特征点下的 469 组数据的平均得分，作为“粒子”的适应度，值越低，代表“粒子”适应度越好。

(3) 找到每组特征点生成的局部最优解和所有特征点生成的全局最优解。

(4) 更新特征点矩阵。

(5) 返回(2)继续执行，迭代次数 300 次。

(6) 输出最佳特征点和对应的平均得分。

5.2.3 遗传算法和粒子群算法的比较

我们在编程实现时，两个算法跑出来的结果相近，均为 93-94 左右，但是粒子群算法实现起来要更加简单，而且收敛速度要更快。遗传算法的实现需要许多参数,如交叉率和变异率,并且这些参数的选择严重影响解的品质,而目前这些参数的选择大部分是依靠经验。同遗传算法比较，粒子群的优势在于简单容易实现并且没有许多参数需要调整。

6、结论

通过对所有 469 组数据进行初步的研究以及结合 MATLAB 的编程模拟，得出以下结论：

1、在选择的拟合方法中，三次样本插值拟合最为准确，达到了课程设计的要求。

2、在确定选点方案方面，我们采取启发式算法和小范围遍历相结合的方法，同时发挥两者的优势，能够得到更加优化的方案。

3、我们最终的选点方案为[3 12 22 31 43 50],平均成本为 92.8774。

7、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]《DS 证据理论在雷达体制识别中的应用》 王 勇 毕大平

[3]《遗传算法中交叉和变异概率选择的自适应方法及作用机理》 陈长征 王楠
[4] ftp://202.120.39.248/ 统计推断课件
[5] 杨建. 蒙特卡罗法评定检测不确定度中相关随机变量的 MATLAB 实现[J]. 计测技术,2012,32(4):51-54
[6]俞启泰. 论 Usher、Logistic 和 Gompertz 三种增长曲线的使用价值[J]. 新疆石油地质. (11)2001,22(2):136-141.

8、致谢

首先感谢袁焱老师和课程助教，在上课过程中及之后的面谈时解决了我们很多的疑惑。袁焱老师在面谈时对我们减少到 5 个测试点的探究予以了很大的鼓励和理论上的支持，助教老师给我们的算法提供的修改意见优化了我们程序中的好多环节。在老师的帮助下，不但我们对 MATLAB 的掌握得到了巩固，同时在科学探究的思想或方法上也有了很大提高。

同时也感谢为本篇论文提供许多理论支持的作者。本文引用了数位学者的研究文献，如果没有各位学者的研究成果的帮助和启发，我们将很难完成本篇论文的写作。

最后，限于我们现阶段的知识水平，所写论文难免有不足之处，恳请老师批评和指正！

【附录】

附录 1 词汇表与符号

（1）词汇表

拟合

科学或工程上可以通过实验等方法获得关于某个问题的若干离散数据。依靠数学方法，使用连续函数（也就是曲线）或者更加密集的离散方程尽量逼近（即最小二乘意义上的差别最小化）这些已知离散数据点集，此过程称为拟合。

插值（内插）

通常有两层含义：

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。

搜索算法

利用计算机的高性能运算能力，有目的地检验一个问题之解空间包含的部分或全部可能情况，从而求得问题的优化解的做法。

启发式搜索

在搜索问题解空间时，对当前已搜索的位置进行评估，寻找认为最好的下一步搜索方向，从这个（或这些）方向进行搜索直到目标。启发式搜索可以避免简单穷举式搜索效率低（甚至不可完成）的弊端。

单点测定，单点测定成本

在本课题中，对 X 取某一定值时 Y 的真实取值进行测量确定的过程。完成单点测定要付出一定成本，称为单点测定成本。

单点定标误差成本

系统完成定标后，对物理量 Y 的单点读数（即系统对 Y 的估测值）与该点 Y 实测值之间存在误差，按一定规则把该误差折算为一种成本。

（2）符号

X	传感部件的输出电压
Y	传感部件的输入，即被监测物理量
$\hat{Y}$	监测模块的输出，即监测模块将传感部件输出 X 所转换成的读数（对 Y 的估测值）
$y_{i,j}$	第 i 个样本之第 j 点对应的 Y 实测值（来自标准样本数据库）
$\hat{y}_{i,j}$	第 i 个样本之第 j 点对应的 Y 估测值
$s_{i,j}$	第 i 个样本之第 j 点对应的单点定标误差成本
q	单点测定成本
n_i	对样本 i 定标过程中的单点测定次数
S_i	对样本 i 的定标成本
M	标准样本数据库中的样本总数
C	基于标准样本数据库评价一个校准方案，算得的该方案总体成本

附录 2 优化后的遗传算法 MATLAB 代码

1.1. 拟合函数：三次样条差值函数

```
function y1 = mycurvefitting( x_premea,y0_premea )
x=[5.0:0.1:10.0];
y1=interp1(x_premea,y0_premea,x,'spline');
end
```

1.2. 种群初始化

```
%初始化种群
%pop_size: 种群大小
%chromo_size: 染色体长度

function [    ]=initilize_function(pop_size, chromo_size)
global pop;
        pop = round(rand(pop_size, chromo_size));%用随机数初始化矩阵
```

1.3. 单点交叉

```
%单点交叉操作
%pop_size: 种群大小
%chromo_size: 染色体长度
%cross_rate: 交叉概率

function crossover(pop_size, chromo_size, cross_rate)
global pop;
for i=1:2:pop_size
    if(rand < cross_rate)
        cross_pos = round(rand * chromo_size);
        if or (cross_pos == 0, cross_pos == 1)
            continue;
        end
        for j=cross_pos:chromo_size
            temp = pop(i,j);
            pop(i,j) = pop(i+1,j);
            pop(i+1,j) = temp;
        end
    end
end



clear i;
clear j;
clear temp;
```

```
clear cross_pos;
```

1.4. 单点变异

```
%单点变异操作
%pop_size: 种群大小
%chromo_size: 染色体长度
%cross_rate: 变异概率
function mutation(pop_size, chromo_size, mutate_rate)
global pop;

for i=1:pop_size
    if rand < mutate_rate
        mutate_pos = round(rand*51);
        if mutate_pos == 0
            continue;
        end
        pop(i,mutate_pos) = 1 - pop(i, mutate_pos);
    end
end

clear i;
clear mutate_pos;
```

1.5. 适应度函数

```
%fitness_calculation 是对样本适应度的计算
%pop_size: 种群样本数
%chromo_size: 染色体的大小

function fitness_calculation(pop_size, chromo_size)
global fitness_value;
global pop;
global data;

for i=1:200
    choose_set(1,:)=selection_set(pop(i,:));%插值点
    for j=1:469
        x=data(1,:);
        y=data(2*j,:);
        x0=[5.0:0.1:10.0];

        fit_para_x(1,:)=x(choose_set);
        fit_para_y(1,:)=y(choose_set);
        y0(1,:)=interp1(fit_para_x(1,:),fit_para_y(1,:),x0,'spline' );
%进行三次线性插值，是我们组的拟合方式
        for k=1:51
        fitness_value(i)=fitness_value(i)+cost(j,k,y0,data);
```

```
            end
            clear y0;
        end
        fitness_value(i)=fitness_value(i)+length(choose_set)*12*469;
        fitness_value(i)=-fitness_value(i)/469;
        clear fit_para_x;
        clear fit_para_y;
        clear choose_set;
    end
```

1.6. 成本函数

```
function [ value ] = cost(x,y,fit_function,data_set)

value=0;

if abs(fit_function(1,y)-data_set(2*x,y))<=0.5
    value=0;
elseif abs(fit_function(1,y)-data_set(2*x,y))<=1
    value=0.5;
elseif abs(fit_function(1,y)-data_set(2*x,y))<=2
    value=1.5;
elseif abs(fit_function(1,y)-data_set(2*x,y))<=3
    value=6;
elseif abs(fit_function(1,y)-data_set(2*x,y))<=5
    value=12;

elseif abs(fit_function(1,y)-data_set(2*x,y))>5
    value=25;
end

End
```

1.7. 选择函数

```
%轮盘赌选择操作
%pop_size: 种群大小
%chromo_size: 染色体长度
%elitism: 是否精英选择

function selection(pop_size, chromo_size)
global pop;
global fitness_table;


for i=1:pop_size
    r = rand * fitness_table(pop_size);%随机生成一个随机数，在 0 和总适应度之间，
fitness_table(pop_size)为总适应度*/
```

```
            first = 1;
            last = pop_size;
            mid = round((last+first)/2);
            idx = -1;
            while (first <= last) && (idx == -1)
                if r > fitness_table(mid)
                    first = mid;
                elseif r < fitness_table(mid)
                    last = mid;
                else
                    idx = mid;
                    break;
                end
                mid = round((last+first)/2);
                if (last - first) == 1
                    idx = last;
                    break;
                end
            end

            for j=1:chromo_size
                pop_new(i,j)=pop(idx,j);
            end
        end

         p = pop_size-1;

        for i=1:p
            for j=1:chromo_size
                pop(i,j) = pop_new(i,j);%若是精英选择，则只将 pop_new 前 pop_size-1 个个体
赋给 pop，最后一个为前代最优个体保留*/
            end
        end

        clear i;
        clear j;
        clear pop_new;
        clear first;
        clear last;
        clear idx;
        clear mid;
        1.8. 成本排序函数
        function rank(pop_size, chromo_size)
        global pop;
```

```
global G;
global fitness_value;
global best_fitness;
global best_individual;
global fitness_table;
global fitness_avg;
global best_generation;
fitvalue=fitness_value';
a=pop;

for i=1:200

    [fitness_value(200+1-i),index]=max(fitvalue);
    pop(200+1-i,:)=a(index,:);
      fitvalue(i,1)=-1000000000;

end
if fitness_value(200)>best_fitness
    best_fitness=fitness_value(200);
    best_generation=G;
    best_individual=pop(200,:);

end
for i=1:200
    fitness_table(i) = 0.;
end
for i=1:200
    if i==1
        fitness_table(i) = fitness_table(i) + fitness_value(i);
    else
        fitness_table(i) = fitness_table(i-1) + fitness_value(i);
    end
end

fitness_avg(G) = fitness_table(200)/200;

clear i;
clear fitvalue;
clear a;
end
```

1.9. 选择结果输出函数

```
function [ output_set ] = selection_set( input )

mark_set=1:51;
```

```
output_set=zeros(1,51);
for i=1:51
    if input(i) ==1
        output_set(i)=mark_set(i);
    else
        output_set(i)=0;
    end

end
output_set(output_set==0)=[];
End
```

1. 10. 遗传算法主函数

```
%遗传算法主函数
%pop_size: 输入种群大小
%chromo_size: 输入染色体长度
%generation_size: 输入迭代次数
%cross_rate: 输入交叉概率
%cross_rate: 输入变异概率
%elitism: 输入是否精英选择
%m: 输出最佳个体
%n: 输出最佳适应度
%p: 输出最佳个体出现代
%q: 输出最佳个体自变量值
function [m,n,p,q] = GeneticAlgorithm(pop_size, chromo_size, generation_size, cross_rate,
mutate_rate, elitism)

global G ; %当前代
global best_cost;%最优成本
global best_fitness;%历代最佳适应值
global fitness_avg;%历代平均适应值矩阵
global best_individual;%历代最佳个体
global best_fitness;%历代最佳适应值
global best_generation;%最佳个体出现代
global pop;%200 个种群的 51 个基因
global fitness_value;%每个个体分别成本
global data;

fitness_avg = zeros(200,1);
pop_size=200;
chromo_size=51;
elitism=true;
cross_rate=0.7;
mutate_rate=0.005;
generation_size=200;
```

```
fitness_size=200;
best_fitness=-300;
best_individual=zeros(1,51);
data=csvread('20141010dataform.csv',0,0);
x=zeros(469,51);
y=zeros(469,51);
for i=1:469;
x(1,:)=5.0:0.1:10;   %x 的值
y(1,:)=data(2*i,:); %y 值
end
clear i;
fitness_value(200)=0;
best_generation = 0;
pop=zeros(200,51);
initilize_function(200, 51);%初始化
best_individual=zeros(1,51);


for G=1:200

    fitness_calculation(pop_size, chromo_size);   %计算适应度
    rank(pop_size, chromo_size);     %对个体按适应度大小进行排序
    selection(pop_size, chromo_size);%选择操作
    crossover(pop_size, chromo_size, cross_rate);%交叉操作
    mutation(pop_size, chromo_size, mutate_rate);%变异操作

n = -mean(fitness_value);%获得每一次的适应度
disp "代数："
G
disp "最优适应度"
n
end

disp "最优适应度"
-best_fitness
m = best_individual;%获得最佳个体
disp "最佳选点方案"
q=selection_set(m)

clear;
```

附录 3 小范围遍历 MATLAB 代码

```
% Author Ba Ruomin ;Zhao Qinghao
% Date 2014-11-30
%Program: This program is mainly used to genetic algorithm to optimize the
%selection of the data

%遗传算法之开头及初始化部分
data=csvread('20141010dataform.csv',0,0);

count_number=zeros(1,49152);
group_matrix=zeros(49152,51);
most_excellent=150;
most_excellent_solution=zeros(1,51);


for h=1:49152; %生成 100 个初始规模%选取 6 个点
    turn1=fix(h/12288);
    turn2=fix((h-12288*turn1)/2048);
    turn3=fix((h-12288*turn1-2048*turn2)/256);
    turn4=fix((h-12288*turn1-2048*turn2-256*turn3)/32);
    turn5=fix((h-12288*turn1-2048*turn2-256*turn3-32*turn4)/4);
    turn6=fix((h-12288*turn1-2048*turn2-256*turn3-32*turn4-8*turn5)/1);
    group_matrix(h,(turn1+1))=1;
    group_matrix(h,(turn2+10))=1;
    group_matrix(h,(turn3+20))=1;
    group_matrix(h,(turn4+28))=1;
    group_matrix(h,(turn5+41))=1;
    group_matrix(h,(turn6+47))=1;
end
disp "完成初始化"

      for h=24577:49152   %每个子代对数据的处理得到适应度
           for k=1:469; %对 469 组数据进行处理
                x=data(2*k-1,:);
                y=data(2*k,:);

                x1=zeros(1,6);
                y1=zeros(1,6);
                j=1;
                for i=1:51;
                     if group_matrix(h,i)==1;
                     x1(j)=x(i);
```

```
                    y1(j)=y(i);
                    j=j+1;
                    end
                end
                x_set=[5.0:0.1:10.0];
                y_function=interp1(x1,y1,x_set,'spline');
                for i=1:51
                delta=y_function(i)-y(i);
                count_number(h)=error_evaluation_function(delta)+count_number(h);
                end
                count_number(h)=count_number(h)+72;
            end
count_number(h)=count_number(h)/469;
if count_number(h)<most_excellent
most_excellent=count_number(h);
for j=1:1:51
most_excellent_solution(1,j)=group_matrix(h,j);
end
end
disp(h)
end


 disp(most_excellent);
 selection_set=zeros(1,6);
 countee=1;
 for jog=1:51
      if most_excellent_solution(1,jog)==1;
           selection_set(1,countee)=jog;
           countee=countee+1;
      end
 end

selection_set
```