统计推断在模数、数模转换系统中的应用研究报告

第05组 刘学成 5130309724 江枫 5130309640

摘要：本课题运用统计推断的方法为监测模块的批量生产设计一种成本合理的传感特性校准方案。综合运用遗传算法、模拟退火算法、随机数函数算法进行取点方案的选择，最终提出了一种折衷的传感特性校准方案。
关键词：特征点、曲线拟合、样条插值、模拟退火、遗传算法、随机数函数算法

1 课题内容与目标

1.1 课题内容的提出

本课题研究的内容为传感器部件监测对象Y与传感部件的输出电压信号X之间的函数关系，由于二者之间并不存在确定的函数关系，因而可以通过研究大量的数据寻找适当的拟合函数。由于样本存在个体差异，因而对单个样本的研究不能代表整体。在对整体研究时，如果用所有测试点进行函数拟合，运算量大，成本高。因而，该课题的研究内容为找出所有样本共同的数据特征点，使得用这些点拟合的函数与样本数据的误差尽可能小，以至于工程上可以接受。

1.2 课题目标

1.2.1 概述

课题的目标是找到几个特征点，使得根据这几个特征点所对应数据确定的拟合函数在与实际数据进行比较评价后定标成本尽可能小。相应的符号表示如下：

符号	含义
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	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

表 1: 符号表

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，规定

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

相应的Y估测值记为$\hat{y}_i = f(x_i)$，Y实测值记为y_i，$i = 1, 2, 3, \ldots, 51$。

1.2.2 成本计算

为评估和比较不同的校准方案，特制定一下成本计算规则：

- 单点定标误差成本

$$s_{i,j} = \begin{cases} 0, & \text{if } \mid \hat{y}_{i,j} - y_{i,j} \mid \leq 0.5 \\ 0.5, & \text{if } \mid \hat{y}_{i,j} - y_{i,j} \mid \leq 1; \\ 1.5, & \text{if } 1 < \mid \hat{y}_{i,j} - y_{i,j} \mid \leq 2; \\ 6, & \text{if } 2 < \mid \hat{y}_{i,j} - y_{i,j} \mid \leq 3; \\ 12, & \text{if } 3 < \mid \hat{y}_{i,j} - y_{i,j} \mid \leq 5; \\ 25, & \text{if } \mid \hat{y}_{i,j} - y_{i,j} \mid > 5. \end{cases} \tag{1}$$

- 单点测定成本 $q = 12$
- 某一样本个体的定标成本

$$S_i = \sum_{j=1}^{51} s_{i,j} + q \cdot n_i \tag{2}$$

- 校准方案总体成本

$$C = \frac{\sum_{i=1}^{M} S_i}{M} \tag{3}$$

总体成本较低的校准方案，工程上可以认定为较优方案。

2 寻找满足条件的拟合函数

2.1 概述

根据Taylor公式

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^n + \frac{f^{n+1}(\xi)}{(n+1)!}(x - x_0)^{n+1}$$

其中ξ 介于x和x_0之间
我们令$x_0 = 0$，有

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(0)}{k!}(x)^n + \frac{f^{n+1}(\xi)}{(n+1)!}(x)^{n+1}$$

当$\frac{f^{n+1}(\xi)}{(n+1)!}(x)^{n+1}$足够小时可以用多项式近似拟合图像曲线足够光滑的函数，因此我们可以考虑用多项式拟合$X - Y$关系。

2.2 对于样本数据的观察

我们用MATLAB画出样本数据对应的$X - Y$图像如下：

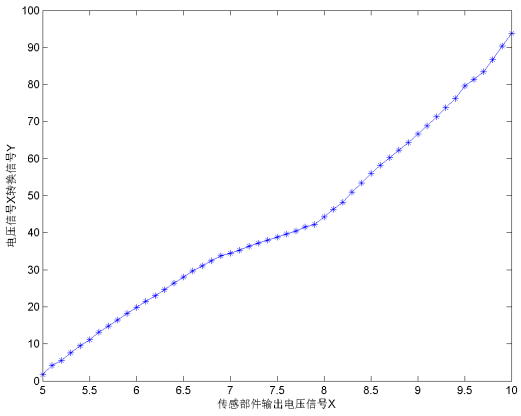

图 1: 样本149

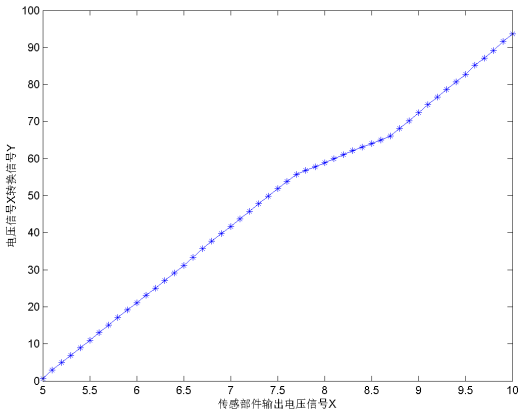

图 2: 样本235

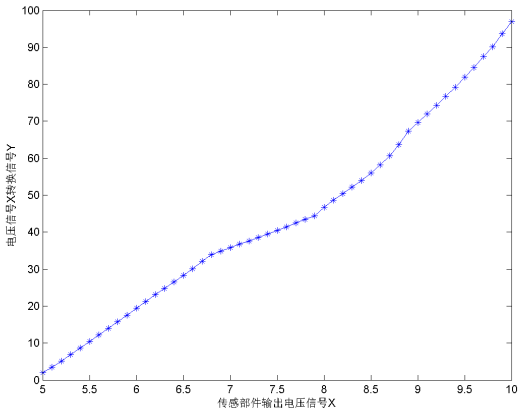

图 3: 样本369

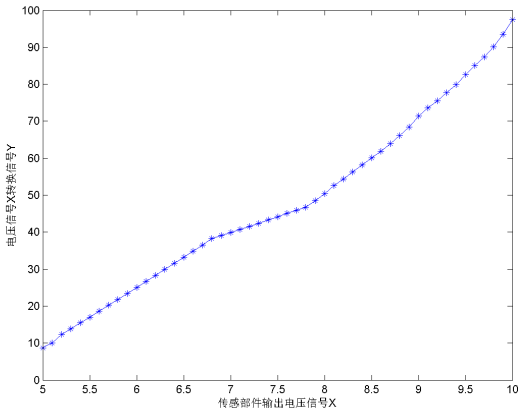

图 4: 样本457

观察发现所有样本数据具有一些共同点：

（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]内，Y的取值在[0,100]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。

2.3 单一多项式拟合

首先我们考虑最简单的方式，即用单个的多项式方程对整条曲线进行拟合，结果如下：

多项式次数	残差平方和均值	相关系数R^2的均值	MATLAB警告的频率
2	201.3720	0.9934	0
3	55.3639	0.9982	0
4	35.3335	0.9989	0
5	23.6225	0.9992	0
6	16.3129	0.9995	1

表 2: 单一多项式拟合

分析表2数据可知，在MATLAB容许范围内，一个五次多项式拟合效果最好。

2.4 分段多项式拟合

2.4.1 区间分段点的选取

实际上，我们已经在对X-Y图像大致观察中发现曲线存在比较明显的分段现象，现在我们进行数值分析以求尽可能精确地确定分界点的位置。

鉴于三段曲线斜率上会有明显的差异，我们可以引入观测点左右差分之差作为分析指标。

定义 2.1. 样本观测点 x_i 的左差分

$$\Delta'_{left}(x_i,d)=\frac{y_{i-d}-y_i}{x_{i-d}-x_i}, i=1+d,2+d,3+d,\ldots,51; d=1,2,3\ldots \tag{4}$$

定义 2.2. 样本观测点 x_i 的右差分

$$\Delta'_{right}(x_i,d)=\frac{y_{i+d}-y_i}{x_{i+d}-x_i}, i=1,2,3,\ldots,51-d; d=1,2,3\ldots \tag{5}$$

定义 2.3. 样本观测点 x_i 的差分差

$$\Delta\Delta'(x_i,d)=\Delta'_{right}(x_i,d)-\Delta'_{left}(x_i,d), i=1+d,2+d,3+d,\ldots,51-d; d=1,2,3\ldots \tag{6}$$

在此基础上，我们对所有样本点的所有观测点进行差分差的数值计算，并均值化后作图如下：

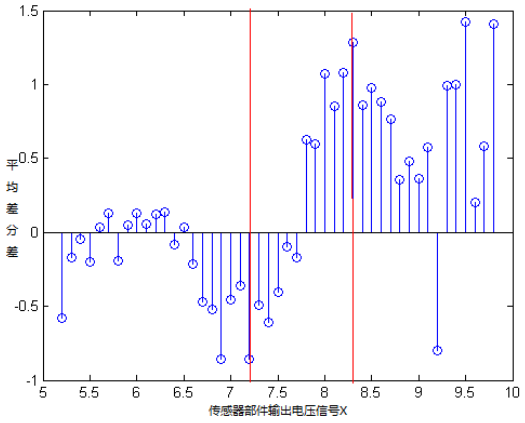

图 5: $d=1$

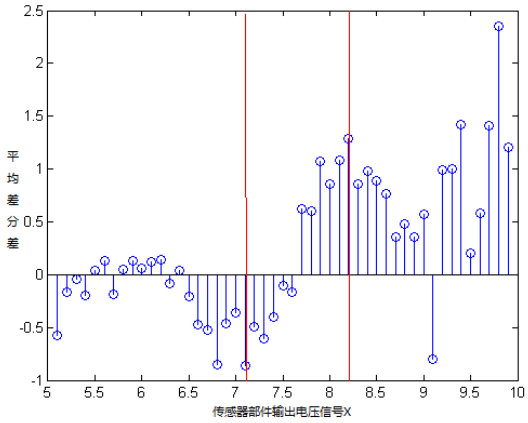

图 6: $d=2$

由于分界点处差分差绝对值应该是最大的，在许多个样本的累计平均作用下，效果会更加明显，因此可在上图中以红线标识出分界点位置。

2.4.2 基于最小二乘法的多项式拟合

通过对大量数据的观察和分析我们发现，数据在中间一段的线性很好，两端线性较差，因而对两端进行三次曲线拟合，对中间一段进行二次曲线拟合，如下表所示：

区间	电压X范围	对应的采样点序号	多项式曲线类型
A	[5.0,7.2]	$1,2,3,\ldots,23$	三次多项式
B	[7.2,8.2]	$23,24,25,\ldots,33$	二次多项式
C	[8.2,10.0]	$33,34,35,\ldots,51$	三次多项式

表 3: 区间分段拟合

区间A：

$$Y_i=a_1X_i^3+a_2X_i^2+a_3X_i+c_1+\varepsilon_i, i=1,2,3,\ldots,23 \tag{7}$$

区间B：

$$Y_j=a_4X_j^2+a_5X_j+a_6+\varepsilon_j, j=23,24,25,\ldots,33 \tag{8}$$

区间C：

$$Y_k=a_7X_k^3+a_8X_k^2+a_9X_k+c_2+\varepsilon_k, k=33,34,35,\ldots,51 \tag{9}$$

以上各式中 $\varepsilon_h\sim N(0,\sigma^2)$ 且相互独立，其中 $h=1,2,3,\ldots,51$
区间A、B衔接点连续：

$$a_1x_{23}^3+a_2x_{23}^2+a_3x_{23}+c_1=a_4x_{23}^2+a_5x_{23}+a_6 \tag{10}$$

区间B、C衔接点连续：

$$a_4x_{33}^2+a_5x_{33}+a_6=a_7x_{33}^3+a_8x_{33}^2+a_9x_{33}+c_2 \tag{11}$$

设误差平方和

$$
\begin{aligned}
Q &= \sum_{h=1}^{51} \varepsilon_h^2 \\
&= \sum_{i=1}^{23} (Y_i - (a_1 X_i^3 + a_2 X_i^2 + a_3 X_i + c_1))^2 + \sum_{j=24}^{32} (Y_j - (a_4 X_j^2 + a_5 X_j + a_6))^2 + \\
&\quad \sum_{k=33}^{51} (Y_k - (a_7 X_k^3 + a_8 X_k^2 + a_9 X_k + c_2))^2
\end{aligned}
$$

达到最小，等价于求函数 $Q(a_1, a_2, a_3, \ldots, a_9)$ 的最小值点。
在函数$Q(a_1, a_2, a_3, \ldots, a_9)$的最小值点处应该满足

$$\frac{\partial Q}{\partial a_i} = 0, i = 1, 2, 3, \ldots, 9 \tag{12}$$

由此可导出以下9个方程：

$$a_1 \sum_{i=1}^{23} X_i^4 + a_2 \sum_{i=1}^{23} X_i^3 + a_3 \sum_{i=1}^{23} X_i^2 + c_1 \sum_{i=1}^{23} X_i = \sum_{i=1}^{23} X_i Y_i \tag{13}$$

$$a_1 \sum_{i=1}^{23} X_i^5 + a_2 \sum_{i=1}^{23} X_i^4 + a_3 \sum_{i=1}^{23} X_i^3 + c_1 \sum_{i=1}^{23} X_i^2 = \sum_{i=1}^{23} X_i^2 Y_i \tag{14}$$

$$a_1 \sum_{i=1}^{23} X_i^6 + a_2 \sum_{i=1}^{23} X_i^5 + a_3 \sum_{i=1}^{23} X_i^4 + c_1 \sum_{i=1}^{23} X_i^3 = \sum_{i=1}^{23} X_i^3 Y_i \tag{15}$$

$$a_4 \sum_{j=24}^{32} X_j^2 + a_5 \sum_{j=24}^{32} X_j + a_6 \sum_{j=24}^{32} 1 = \sum_{j=24}^{32} Y_j \tag{16}$$

$$a_4 \sum_{j=24}^{32} X_j^3 + a_5 \sum_{j=24}^{32} X_j^2 + a_6 \sum_{j=24}^{32} X_j = \sum_{j=24}^{32} X_j Y_j \tag{17}$$

$$a_4 \sum_{j=24}^{32} X_j^4 + a_5 \sum_{j=24}^{32} X_j^3 + a_6 \sum_{j=24}^{32} X_j^2 = \sum_{j=24}^{32} X_j^2 Y_j \tag{18}$$

$$a_7 \sum_{k=33}^{51} X_k^4 + a_8 \sum_{k=33}^{51} X_k^3 + a_9 \sum_{k=33}^{51} X_k^2 + c_2 \sum_{k=33}^{51} X_k = \sum_{k=33}^{51} X_k Y_k \tag{19}$$

$$a_7 \sum_{k=33}^{51} X_k^5 + a_8 \sum_{k=33}^{51} X_k^4 + a_9 \sum_{k=33}^{51} X_k^3 + c_2 \sum_{k=33}^{51} X_k^2 = \sum_{k=33}^{51} X_k^2 Y_k \tag{20}$$

$$a_7 \sum_{k=33}^{51} X_k^6 + a_8 \sum_{k=33}^{51} X_k^5 + a_9 \sum_{k=33}^{51} X_k^4 + c_2 \sum_{k=33}^{51} X_k^3 = \sum_{k=33}^{51} X_k^3 Y_k \tag{21}$$

c_1, c_2满足

$$c_1 = -a_1 x_{23}^3 + (a_4 - a_2) x_{23}^2 + (a_5 - a_3) x_{23} + a_6 \tag{22}$$

$$c_2 = -a_7 x_{33}^3 + (a_4 - a_8) x_{33}^2 + (a_5 - a_9) x_{33} + a_6 \tag{23}$$

由此，我们可依次求出 $a_1, a_2, a_3, \ldots, a_9, c_1, c_2$ ，因而我们可以计算出拟合参数中所有的值。

2.5 基于其他函数拟合

2.5.1 概论

以上我们主要针对多项式函数进行了一系列的拟合尝试和理论分析论证，其实还有一些其他的函数可能更适合于这种曲线拟合，下面我们针对一些进行试验。

2.5.2 拟合试验

- Gaussian函数

$$g(x) = ae^{-\frac{(x-b)^2}{2c^2}}$$

- Fourier函数

$$F_k(x) = a_0 + \sum_{i=1}^{k} a_i \cos(i\omega x) + \sum_{i=1}^{k} b_i \sin(i\omega x); k = 1, 2, 3 \ldots$$

- 指数函数

$$E_1(x) = ae^{bx}$$
$$E_2(x) = ae^{bx} + ce^{dx}$$

- S型曲线（Gompertz，Logistic，Usher）

$$G(x) = Le^{-be^{-kx}} + C$$
$$L(x) = \frac{L}{1 + be^{-kx}} + C$$
$$U(x) = \frac{L}{(1 + be^{-kx})^{\frac{1}{a}}}$$

MATLAB拟合结果如下：

拟合函数	误差R^2
$g(x)$	0.9811
$F_1(x)$	0.9914
$F_2(x)$	0.9991
$F_3(x)$	0.9997
$E_1(x)$	0.9608
$E_2(x)$	0.9956
$G(x)$	0.1268
$L(x)$	4.4e-16
$U(x)$	8.3e-8
$splineinterp$	1.0000

表 4: 其他函数拟合

3 基于特征点的插值拟合

3.1 概述

事实上，由于每组数据由51个X-Y数值对构成，若每次都使用51对数值对X-Y关系进行研究，既增加了实际中测量的工作量，也使得通过全部51对被标定的数据得到的拟合曲线失去工程上的实际意义。因而我们希望在这51个数值对中选取若干个特征点进行曲线拟合，使得校准方案总成本最小，因此我们需要解决去多少个点和该怎样取的问题。

3.2 拟合方法选择

样条插值是一种工业设计中常用的、得到平滑曲线的一种插值方法，三次样条又是其中用的较为广泛的一种。并且由表4可知三次样条插值很适合X-Y曲线拟合。因此我们采用三次样条插值拟合作为拟合方法。

3.3 特征点的确定

3.3.1 概述

特征点的确定直接决定了我们最终的拟合方案是否最优。我们首先想到的最简单的方式是用暴力搜索，我们需要排查$C_{51}^{k}(k = 1, 2, 3, \ldots, 50, 51)$ 种情况，时间复杂度相当大，效率过低。因此，我们需要采用其他的启发式搜索算法，这里我们考虑了模拟退火算法、遗传算法和随机数函数算法。

k	6	7	8	15	20	25
C_{51}^{k}	1.8009e+7	1.1577e+8	6.3676e+8	3.1887e+12	7.7535e+13	2.4796e+14

表 5: 暴力搜索时间复杂度

3.3.2 基于模拟退火算法

模拟退火算法是一种通用概率演算法，用来在一个大的搜寻空间内找寻最优解，Metropolis准则保证了这一点。

结合本课题，模拟退火主要流程如下：

（1）初始化：设置初始温度（充分大） $T_0 = 100$ ，温度衰减系数 $\lambda = 0.95$ ，终止温度设定为 $T_t = 1$ ，每个 T 值的迭代次数 $L = 5$ ，初始解为特征值串 $S = s_1 s_2 s_3 \dots s_n$ ，串长为 n ；

（2）对 $k = 1, 2, \dots, L$ 做步骤（3）（4）（5）；

（3）用随机算子产生新特征值串 $S' = s'_1 s'_2 s'_3 \dots s'_n$ ：从集合 $\{1, 2, 3, \dots, n\}$ 中随机选取一个数 k ，然后原特征值串中除 $s_k \neq s'_k$ 外，都满足 $s_i = s'_i; i \in \{1, 2, 3, \dots, n\} - \{k\}$ ，然后 $s_k = s_k + 1$ ，如果存在 $i \in \{1, 2, 3, \dots, n\} - k$ 使得 $s_k = s_i$ 则重复 $s_k = s_k + 1$ (if $s_k = 51$,then $s_k + 1 = 1$)，直到满足条件为止；

（4）计算增量 $\Delta C = C(S') - C(S)$ ，其中 $C(S)$ 为评价函数。接受新解 S' 当且仅当 $\Delta C < 0$ 或随机数 $r < e^{-\frac{\Delta C}{T}}$ ，如果满足终止条件则输出当前解作为最优解，结束程序；

（5）当 $k = L$ 时，$T = \lambda \cdot T$ ，转步骤（2）。

事实上这种方法试验表明不太理想，主要是由于接受新解的条件 $r < e^{-\frac{\Delta C}{T}}$ 引起，导致最佳取点方式容易被丢弃，于是可以稍微改进一点：将此条件去掉，仅当 $\Delta C < 0$ 时接受新解，新解的产生不依赖于已存在的最优解。用MATLAB实验如下：

取点个数	特征值组合	校准方案总体成本	实验耗时/s
4	{17,49,38,5}	129.69	163
5	{8,49,2,21,36}	113.20	163
6	{11,50,3,21,31,38}	101.91	168
7	{2,45,49,15,37,29,9}	105.16	173
8	{20,2,49,26,40,29,9,24}	108.33	170
9	{30,16,23,44,1,8,31,37,49}	113.79	167
10	{31,18,24,45,2,8,33,40,51,28}	123.83	168

表 6: 退火算法寻找最佳取点组合

3.3.3 基于遗传算法

遗传算法是模仿自然界生物进化机制发展起来的随机全局搜索和优化方法，它借鉴了达尔文的进化论和孟德尔的遗传学说。其本质是一种高效、并行、全局搜索的方法，它能在搜索过程中自动获取和积累有关搜索空间的知识，并自适应的控制搜索过程以求得最优解。遗传算法操作使用适者生存的原则，在潜在的解决方案种群中逐次产生一个近似最优解的方案，在遗传算法的每一代中，根据个体在问题域中的适应度值和从自然遗传学中借鉴来的再造方法进行个体选择，产生一个新的近似解。这个过程导致种群中个体的进化，得到的新个体比原来个体更能适应环境，就像自然界中的改造一样。

针对定标问题设计的遗传算法伪代码如下：

（1）染色体的表示：初始染色体为51位的二进制串000...00，取点个数k时，将特征值点所对应的位置上的值置为1。首先随机生成一定大小的种群，这里我们设定初始大小为200。

（2）种群中个体适应度的确定：对每一个特征值串进行三次样条插值拟合求得拟合函数后计算平均成本作为个体适应度。平均成本越小个体适应度越大。

（3）遗传三大算子：

- 选择算子：使用轮盘选择算子。由于个体适应度与平均成本负相关，假设个体适应度分别为 $s_1, s_2, \dots, s_k$ ，经过变换 $s'_i = 2 \max\limits_{i=1,2,\dots,k} \{s_i\} - s_i, i = 1, 2, \dots, k$ 后 s'_i 与个体适应度正相关，然后在 $[0, 1]$ 区间上选取 $k+1$ 个分界点 $x_i, i = 0, 1, \dots, k$ ，且 $x_0 = 0$ ， $x_k = 1$ ， $x_i = \frac{\sum_{j=1}^{i} s'_j}{\sum_{j=1}^{k} s'_j}$ ，最后生成 $[0, 1]$ 上的随机数 r ，若存在 $i \in \{0, 1, 2, \dots, k-1\}$ 使得 $x_i \leq r \leq x_{i+1}$ ，则选择第 $i+1$ 个个体遗传到下一代。同时考虑到算子的随机性和变异算子、交叉算子的存在，强制性的将前一代中最优个体复制3个遗传到下一代。

- 变异算子：以一定的概率 $p_1 = 0.05$ 来进行变异。在保证个体染色体有效长度不变的情况下，我们先生成随机数 $r_1 \in \{a_1, a_2, \ldots, a_k\}$ 将 r_1 所对应位置上的值置为0，再生成随机数 $r_2 \notin \{a_1, a_2, \ldots, a_k\}$ 将r_2所对应位置上的值置为1。

- 交叉算子：以一定的概率 $p_2 = 0.25$ 来进行交叉。随机选取两个个体，然后随机选取特征值串的一个位置，将此位置之后的片段进行交换，完成交叉过程。

（4）设置遗传代数 $generation = 200$，选择算子、变异算子、交叉算子依次作用于每一代，最终输出结果几乎可以认为是在该取点个数下的最优解。

MATLAB实验结果如下：

取点个数	特征值组合	校准方案总体成本	实验耗时/s
4	{4,17,39,48}	127.6	3387
5	{4,13,26,39,50}	104.8	3374
6	{3,12,23,31,43,50}	93.7	3403
7	{1,9,20,26,31,43,49}	95.5	3346
8	{3,8,20,27,32,39,47,51}	102.9	3303
9	{2,6,15,19,26,31,37,44,50}	111.8	3612
10	{3,8,16,20,26,31,35,41,45,51}	122.7	3430

表 7: 遗传算法寻找最佳取点组合

对比表6可以看出，遗传算法比退火算法更加精确，但退火算法比遗传算法更快。

3.3.4 基于随机数函数算法

通过遗传算法和模拟退火算法我们可以发现取点相对均匀时得到的拟合效果相对越好，校准方案总体成本越小。因此我们可以采取另一种随机数函数算法。假设取点个数为 n ，则在 $\{1, 2, 3, \ldots, 51\}$ 这个整数区间上均匀的选取 n 个点 $\{a_1, a_2, \ldots, a_n\}$ ，确定波动范围为 $\Delta a = \frac{\sum_{i=1}^{n-1}(a_{i+1}-a_i)}{2(n-1)}$ ，（实际上波动范围是用平均的思想得到的，可以简单的处理为 $\Delta a = \frac{51}{2n}$ ）然后点 $x_i \in [a_i - \Delta a, a_i + \Delta a] \cap \{1, 2, 3, \ldots, 51\}, i = 1, 2, \ldots, n$ ，在此约束条件下用随机算子产生大量取点方式后找到校准方案总体成本最小的取点方式。

MATLAB实验结果如下：

取点个数	特征值组合	校准方案总体成本	实验耗时/s
4	{4,17,38,48}	128.40	166
5	{3,12,25,38,48}	105.51	176
6	{3,10,21,31,41,50}	93.42	164
7	{3,9,19,26,32,43,51}	95.38	161
8	{1,8,16,24,31,37,44,50}	102.56	164
9	{2,7,13,20,27,33,39,45,50}	111.83	165
10	{2,7,13,19,24,30,35,40,45,50}	122.39	162

表 8: 随机数函数算法寻找最佳取点组合

对比表7和表8可以发现：

- 随机数函数算法更高效；

  相对于遗传算法，随机数函数算法省去了大量的遗传、变异、交叉的过程，并通过合理的划分区间大大地缩小了取点范围。

- 在取点个数很小时，遗传算法求得最佳取点组合方式的准确率更高；

  取点个数较小时，随机数函数算法划分区间较少，区间长度较大，趋近于暴力搜索，导致效率下降；而遗传算法不存在这个问题。

- 在取点个数很大时，随机数函数算法求得最佳取点组合方式的准确率更高。

  取点个数较大时，随机数函数算法的高效性充分体现了出来，但变异算子、交叉算子的存在使得遗传算法丢掉比较好的取点组合方式的可能性增大。

3.3.5 取点个数的确定

最终目标是找到一个合适的取点方案使得成本最小，取点方案包括两方面的内容：

- 取点个数
- 取点位置

取点位置的确定以上两个算法已经基本解决，取点个数 n 可以理论分析如下：
根据校准方案总体成本的定义（1）（2）（3），可得

$$\begin{aligned} C(n,f) =& \frac{\sum_{i=1}^{M} S_i}{M} \\ =& \frac{\sum_{i=1}^{M}(\sum_{j=1}^{51} s_{i,j} + q \cdot n)}{M} \\ =& q \cdot n + \frac{\sum_{i=1}^{M}\sum_{j=1}^{51} s_{i,j}}{M} \end{aligned}$$

$C(n,f)$ 中 n 代表取点个数 f 代表取点方式。

取点个数 n 越大， $q \cdot n$ 越大，而相应的拟合效果就会越好， $\frac{\sum_{i=1}^{M}\sum_{j=1}^{51} s_{i,j}}{M}$ 越小。这时就要考虑一个增长率的问题。当 n 很小时， $q \cdot n$ 的增长率比 $\frac{\sum_{i=1}^{M}\sum_{j=1}^{51} s_{i,j}}{M}$ 的减小率小， $C(n,f)$ 呈下降态势，当 n 比较大时， $q \cdot n$ 的增长率比 $\frac{\sum_{i=1}^{M}\sum_{j=1}^{51} s_{i,j}}{M}$ 的减小率大， $C(n,f)$ 呈上升态势。故在拟合方式一定，对于每个取点数量 n 都采用最佳取点位置的情况下， $C(n,f) - n$ 的图像应呈抛物线状。MATLAB实验结果如下：

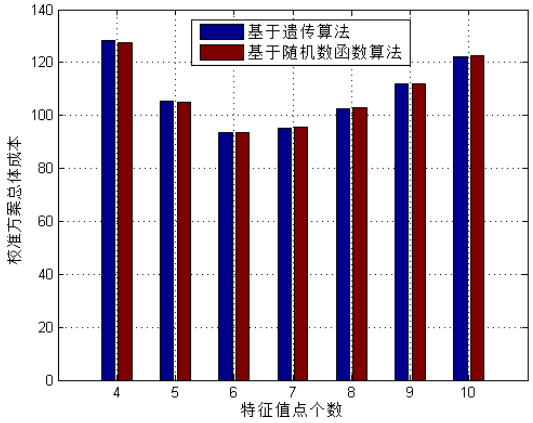

图 7: 取点个数的确定

因此为了确定最佳取点个数可以设计算法如下：
（1）利用随机数函数算法或遗传算法可得到针对每个取点个数的最佳取点方式以及在此方式下的校准方案总体成本；
（2）从最小的取点个数 $k = 4$ 开始（工程上一般认为取1、2、3个点的拟合结果不可靠），在（1）的基础上计算最小校准方案总体成本 $C(k)$ ；
（3）根据理论分析的结果， $C(k)$ 先减后增，当 k 满足 $C(k-1) > C(k) < C(k+1)$ 时可以认为 k 就是最佳的取点个数。

3.3.6 小结

模拟退火虽快但不够准确，遗传算法虽准但不够快速，而在两者基础上衍生出来的随机数函数算法则弥补了两者的缺陷，再结合确定最佳取点个数的算法便得到了一种折衷的校准方案。

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[M/OL].http://zh.wikipedia.org/zh/样条插值

[3] 维基百科. 模拟退火[M/OL].http://zh.wikipedia.org/zh/模拟退火

[4] 维基百科. 遗传算法[M/OL].http://zh.wikipedia.org/zh/遗传算法

附录

计算取点组合的成本

```
%% answer_check.m
function cost=answer_check(my_answer)
size_of_my_answer=length(my_answer);
mdata=csvread('20141010dataform.csv');
[M,N]=size(mdata);
x=zeros(M/2,N);
y0=zeros(M/2,N);
y1=zeros(M/2,N);
for i=1:M/2
    x(i,:)=mdata(2*i-1,:);
    y0(i,:)=mdata(2*i,:);
end
my_answer_gene=zeros(1,N);
my_answer_gene(my_answer)=1;
index_temp=logical(my_answer_gene);
x_optimal=x(:,index_temp);
y0_optimal=y0(:,index_temp);
for i=1:M/2
    y1(i,:)=curvefitting(x_optimal(i,:),y0_optimal(i,:));
end
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(M/2,1)*size_of_my_answer;
cost=sum(si)/(M/2);
```

```
%% curvefitting.m
function y=curvefitting(x_premea,y0_premea)
x=[5.0:0.1:10.0];
y=interp1(x_premea,y0_premea,x,'spline');
```

模拟退火算法

```
%% simulated annealing algorithm
tic;
rng(100);
T=100;
size_of_sample=4;
```

```
answer=randperm(51,size_of_sample);
new_answer=answer;
cost=answer_check(answer);
L=20;
while(T>0)
    for i=1:L
        index=randperm(size_of_sample,1);
        tmp=mod(new_answer(index),51)+1;
        while(ismember(new_answer,tmp))
            tmp=mod(tmp,51)+1;
        end
        new_answer(index)=tmp;
        new_cost=answer_check(new_answer);
        r=rand();
        if(new_cost<cost)
            answer=new_answer;
            cost=new_cost;
            for j=1:size_of_sample
                fprintf('%2d ',answer(j));
            end
            fprintf('\n cost is %3.2f \n',cost);
        end
    end
    T=T-2;
end
toc;
```

遗传算法

```
%% genetic algorithm
tic;
rng(100);
size_of_pop=200;
size_of_sample=7;
pop=zeros(size_of_pop,size_of_sample);
for i=1:size_of_pop
    pop(i,:)=randperm(51,size_of_sample);
end
cost_pop=zeros(1,size_of_pop);
for i=1:size_of_pop
    cost_pop(i)=answer_check(pop(i,:));
end
for generation=1:100
    selection;
    crossover;
    mutation;
    for i=1:size_of_pop
        cost_pop(i)=answer_check(pop(i,:));
    end
    minpop=min(cost_pop);
    fprintf('%2d   %5.1f   %5.1f   \n',generation,mean(cost_pop),min(cost_pop)
        );
    for i=1:size_of_pop
        if(cost_pop(i)==minpop)
            for j=1:size_of_sample
                fprintf('%2d    ',pop(i,j));
            end
            fprintf('\n');
            break;
```

```
        end
    end
end
toc;
```

```
%% selection.m
temp=-cost_pop+2*max(cost_pop);
ppop=temp/sum(temp);
qpop=zeros(1,size_of_pop);
for i=1:size_of_pop
    qpop(i)=sum(ppop(1:i));
end
size_of_tmp_pop=size_of_pop;
tmp_pop=zeros(size_of_tmp_pop,size_of_sample);
pmin=min(cost_pop);
for i=1:size_of_pop
    if(cost_pop(i)==pmin)
        break;
    end
end
tmp_pop(1,:)=pop(i,:);
tmp_pop(size_of_pop,:)=pop(i,:);
tmp_pop(size_of_pop/2,:)=pop(i,:);
for i=[2:size_of_tmp_pop/2-1,(size_of_tmp_pop/2+1):(size_of_tmp_pop-1)]
    r=rand();
    for j=1:size_of_pop
        if(qpop(j)>=r)
            break;
        end
    end
    tmp_pop(i,:)=pop(j,:);
end
pop=tmp_pop;
size_of_pop=size_of_tmp_pop;
```

```
%% crossover.m
p0=0.25;
flag=0;
times=10;
for i=1:times
    for k=1:size_of_pop
        r=rand();
        if(r<p0 && flag==0)
            a=k;
            flag=1;
        end
        if(r<p0 && flag==1)
            b=k;
            flag=2;
            break;
        end
    end
    if(flag==2 && a~=b )
        r=randi([2 size_of_sample],1);
        tmp=pop(a,r:size_of_sample);
        pop(a,r:size_of_sample)=pop(b,r:size_of_sample);
        pop(b,r:size_of_sample)=tmp;
    end
end
```

```
%% mutation.m
pm=0.05;
for i=1:size_of_pop
    r=rand();
    if(r<pm)
        pop(i,:)=imutation(pop(i,:));
    end
end
```

```
%% imutation.m
function new_answer=imutation(answer)
N=51;
answer_gene=zeros(1,N);
answer_gene(answer)=1;
index=randi(N,1);
if(ismember(index,answer))
    flag=1;
    answer_gene(index)=0;
else
    flag=0;
    answer_gene(index)=1;
end

index=randi(N,1);
if(flag)
    while(ismember(index,answer))
        index=randi(N,1);
    end
    answer_gene(index)=1;
else
    while(~ismember(index,answer))
        index=randi(N,1);
    end
    answer_gene(index)=0;
end
point=1:51;
new_answer=point(logical(answer_gene));
```

随机数函数算法

```
%% random function algorithm
tic;
rng(100);
size_of_sample=4;
points=1:51;
answer=zeros(1,size_of_sample);
new_answer=zeros(1,size_of_sample);
for i=1:size_of_sample
    answer(i)=randsample(points(floor(51*(i-1)/size_of_sample)+1:floor(51*i
        /size_of_sample)),1);
end
cost=answer_check(answer);
for times=1:1000
    for i=1:size_of_sample
    new_answer(i)=randsample(points(floor(51*(i-1)/size_of_sample)+1:floor
        (51*i/size_of_sample)),1);
    end
    new_cost=answer_check(new_answer);
    if(new_cost<cost)
```

```
18          answer=new_answer;
19          cost=new_cost;
20          for i=1:size_of_sample
21              fprintf('%2d ',answer(i));
22          end
23          fprintf('\n cost is %3.2f \n',cost);
24      end
25  end
26  for i=1:size_of_sample
27      fprintf('%2d ',answer(i));
28  end
29  fprintf('\n cost is %3.2f \n',cost);
30  toc;
```