统计推断在数模转换系统中的应用

组号 17　　叶俊甫 5130309505，何沛骏 5130309501

摘要：统计推断是数模转换系统中的重要应用方法。本课题探究了快速、准确地为监测模块传感器特性定标的方法，以确定其输入电压信号与输出读数的关系。我们使用 MATLAB 进行程序设计，采用遗传算法选择合适的定标点，再使用几种不同的插值或拟合方法来完成对样品的定标。

关键词：统计推断，定标，三次样条插值法，多项式拟合，分段三次 Hermite 插值，遗传算法，MATLAB

Application of Statistic Inference in DA Converting System

ABSTRACT: Statistic inference is an essential application in the DA converting system. In this task, we do the research of approaching a calibration scheme for the sensor of a monitor sample quickly and accurately, so as to determine the X-Y function relationship. We want to achieve the excellent scheme by determining the points for calibration by using Genetic Algorithm and different interpolation or matching method in MATLAB programming.

Key words: Statistic inference, calibration, cubic spline interpolation, polynomial fitting, piecewise cubic Hermite interpolation, Genetic Algorithm, MATLAB

1 引言

1.1 课题内容的提出

假定有某种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。本课题要求为该模块的批量生产设计一种成本合理的传感特性校准（定标工序）方案[1]。

监测模块的组成框图如图 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）。

如果为了确定产品的 X-Y 关系， 对每个产品的所有 51 个数据点测量，成本太高。因此为了省时省工， 本课题研究了如何快速高效地， 通过尽量少的若干个测量点，以某种插值或拟合的方法，在一定的误差范围内， 推测出样品的其余实用设定点的 X-Y 关系。

本课题使用 MATLAB 作为编程语言来搜索、确定合适的定标方案。

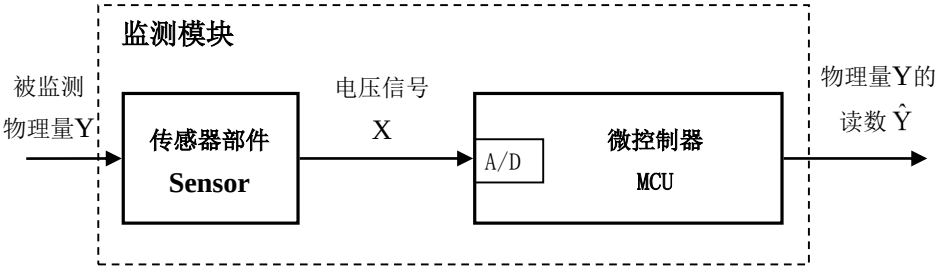

图 1 监测模块组成框图

1.2 课题研究的主要步骤

(1) 初步分析样本数据。
(2) 确定一种插值或拟合的方法。
(3) 使用搜索算法， 确定 7 个数据点，即找到定标准确度足够高的一个定标方案。
(4) 分析结果，优化或更换插值或拟合的方法，优化或更换搜索算法。
(5) 重复(2)(3)(4)，直到得到满意的定标方案。

2 样本数据分析

在所给的实验原始数据样本中，共有 469 件样品，每个样品提供了 51 个数据， 测量点的 X 的范围均在 5.0 至 10.0，每 0.1 提供一个数据点，而 Y 的范围则不确定。 如图 2 所示，通过对不同样品的 Y-X 函数曲线观察可以总结出 X-Y 函数关系有以下几个特点：

(1) 单调性，Y 随 X 单调递增而单调递增；
(2) 不同个体的特性曲线形态相似但两两相异；
(3) 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；
(4) 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；
(5) 不同个体的中段起点位置、终点位置有随机性差异。

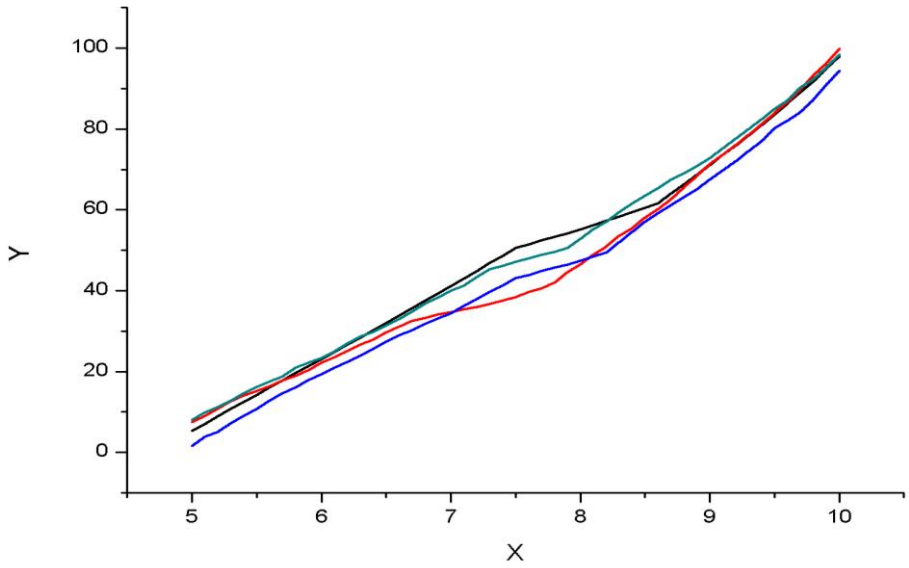

图 2 四个不同样本 X-Y 特性图

3 定标成本评价方法

3.1 定标准确度评价过程

对于某一确定的定标方案的定标准确度评价分以下几个阶段：

(1) 按照定标方案进行测量，获得样品的 7 个实测值。
(2) 借助(1)中所获实测值，按照定标方案进行定标，获得样品所有数据点的估测值。
(3) 增加测量，获得样品所有数据点的实测值。
(4) 对比估测值与实测值，给出定量的评价。

3.2 定标成本计算规则

- 单点定标误差成本

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

单点定标误差的成本按式（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。

- 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。

- 某一样本个体的定标成本

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

对样本 i 总的定标成本按式（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。

- 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。

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

总体成本较低的校准方案，认定为较优方案。

4 插值或拟合方法

由样本数据的定性分析可知，测量是存在误差的，某些样本的 X-Y 曲线存在不光滑或偏差较大的观察点。本段实验选择三次样条插值法，分段多项式拟合，以及分段三次 Hermite 插值来做对比，探究后确定深入研究三次样条插值法。

4.1 三次样条插值法[2]

4.1.1 方法概述

三次样条插值法一种非线性插值法，它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。如图 4 所示，三次样条插值法每次选取相邻的 4 个点确定一条三次曲线，再取出中间两个点之间的三次曲线作为样条，然后借助样条来计算出插值点的估计值。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（边界点的导数为 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）。

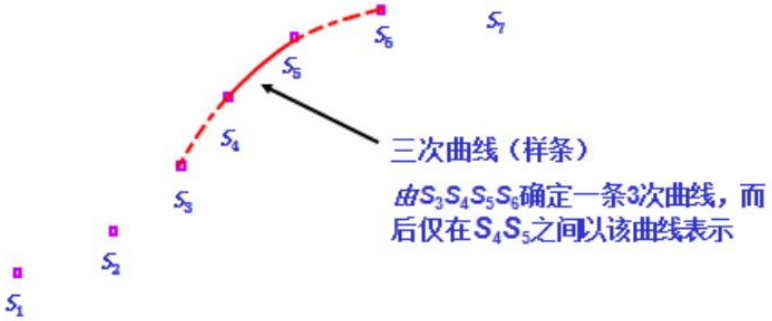

图 3 三次样条插值法简单示意图

4.1.2 优缺点分析

优点：插值所得曲线为光滑曲线，且曲线的导数曲线依然光滑。插值所得曲线通过每一个数据点，使得数据点的估计值等于观察值，无需对其另外处理。而且真实 X-Y 曲线的形态，对插值所得曲线与真实 X-Y 曲线的吻合度影响较小。

缺点：运算量较大，速度较慢。

4.1.3 MATLAB 实现

MATLAB 中实现插值的语句为：

Y1=interp1(X, Y, voltageX (i, :),'spline') ;

其中 X 和 Y 分别为数据点的 X 值和 Y 值，列表 voltageX(i,:) 为所有数据点的电压信号 X，Y1 为所有数据点 Y 的估测值列表。MATLAB 中将非扭结边界条件作为默认的边界条件。

4.2 多项式拟合 [3]

4.2.1 方法概述

多项式拟合是一种最为基本的拟合方式。对给定数据（x_i，y_i）在给定的函数类 Φ 中，求 p(x)∈Φ，使误差r_i=p(x_i) − y_i为最小，这样求得的函数 p(x)就是拟合出来多项式函数。

4.2.2 优缺点分析

优点:运算速度快，易于理解，可以很方便的进行操作和分析。拟合的效果可能产生极其符合实验点的拟合函数。

缺点：拟合的精度不能有平均的保证。 真实 X-Y 曲线的形状对拟合曲线的拟合精度影响较大。

4.2.3 MATLAB 实现

MATLAB 中的实现语句为：

p=polyfit(X, Y, 3);

Y1=polyval (p, voltageX (i, :));

其中 X 和 Y 分别为数据点的 X 值和 Y 值，voltageX(i,:)为所有数据点的电压信号 X，Y1 为所有数据点 Y 的估测值列表。

4.3 分段三次 Hermite 插值

4.3.1 方法概述

为使插值函数能更好地和原来的函数重合， 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这类插值称作切触插值，或埃尔米特(Hermite)插值。我们所采用的三次 Hermite 插值所做的事是通过平面上的两个点和它们的切线斜率来确定一个三次函数：

$$H(x) = h_0(x)f(x_0) + h_1(x)f(x_1) + \overline{h_0}(x)f'(x_0) + \overline{h_1}(x)f'(x_1)$$

$$h_0(x) = \left(1 + 2\frac{x-x_0}{x_1-x_0}\right)\left(\frac{x-x_1}{x_0-x_1}\right)^2 \qquad h_1(x) = \left(1 + 2\frac{x-x_1}{x_0-x_1}\right)\left(\frac{x-x_0}{x_1-x_0}\right)^2$$

$$\overline{h_0}(x) = (x - x_0)\left(\frac{x-x_1}{x_0-x_1}\right)^2 \qquad \overline{h_1}(x) = (x - x_1)\left(\frac{x-x_0}{x_1-x_0}\right)^2$$

4.3.2 优缺点分析

优点： 插值所得的函数曲线十分光滑， 并且其导数曲线也十分光滑，其他优点与三次样条插值比较相似。

缺点：运算量较大，速度慢。

4.3.3 MATLAB 实现

MATLAB 中实现插值的语句为：

Y1=interp1(X, Y,voltageX (i, :),'pchip') ;

其中 X 和 Y 分别为数据点的 X 值和 Y 值，voltageX(i,:)为所有数据点的电压信号 X，Y1 为所有数据点 Y 的估测值列表。

4.4 插值或拟合方法的选定 [4]

经过实验性运行，我们进行了拟合方法的筛选。

运行速度的比较方法是，使用三种不同方法分别运行遗传算法，得到 10 代的定标方案，每种方法运行 3 次，比较运行的时间消耗。三次样条插值法三次分别耗时 80s，84s，82s，平均 82s。多项式拟合法三次分别耗时 140s，133s，127s，平均 133s。三次 hermite 插值法分别耗时 421s，440s，398s 平均 420.7s。三次 hermite 耗时过长，首先淘汰。另外，三次样条插值运行多代后可以稳定得到成本 95 以下的优秀方案，而多项式拟合的表现不佳，成本不能稳定低于 95，所以淘汰这个方法。

5 启发式搜索

关于 7 个事前观察点的取法问题，可以通过搜索算法来完成。对于本课题，事前观察点的取法共有C_{51}^{7}（51 取 7 的组合数）种，也即有约 1.15 亿种不同的组合。其复杂度为 O(N!),故此问题属于 NP 问题 (Non-deterministic Polynomial)，因此无法使用穷举搜索的方法来求解此问题，但仍然可以采用启发式搜索来进行求解。

目前本小组采用遗传算法来进行求解。

5.1 遗传算法 [5]

遗传算法是一类借鉴生物界的进化规律（适者生存，优胜劣汰遗传机制）演化而来的随机化仿生搜索方法。以下简称遗传算法为 GA。

5.1.1 GA 基本框架

GA 将每一个可能的解看作一个生物个体，将若干生命个体组成种群。算法中需要对解进行编码，以模仿生物的基因染色体。GA 的流程图如图 5 所示。

算法开始运行时， GA 将随机生成一个含有一定数量个体的初始种群。之后，GA 将计算种群中的个体的适应度，然后 GA 将模拟自然选择过程，下一步 GA 模拟生物交配繁殖，对两个基因代码进行交叉，产生两个新个体。然后再模拟基因突变，最终产生新一代的种群。重复本段所讲述过程直到满足终止条件即可。

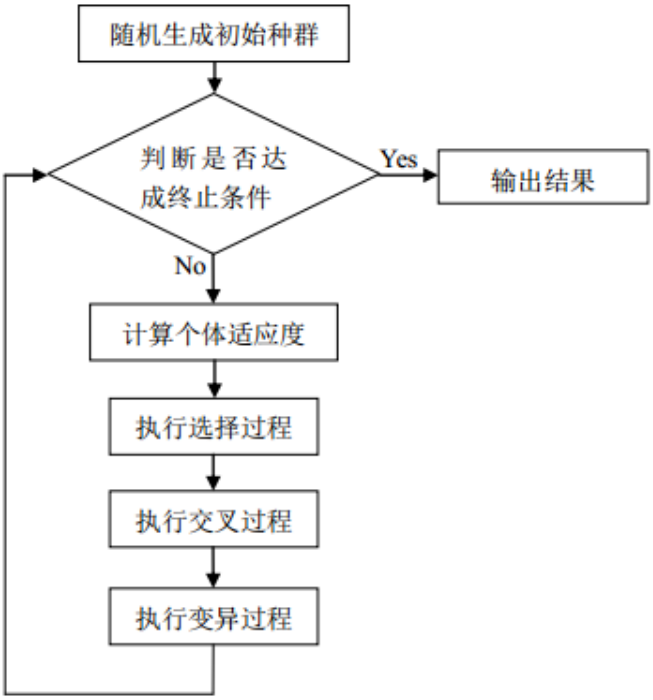

图 4 遗传算法流程图

5.1.2 GA 的优劣势

优势：

(1) 有全局择优性。遗传算法从串集开始搜索，覆盖面大，利于全局择优。

(2) 不采用确定性规则，而是采用概率的变迁规则来指导其的搜索方向。

(3) 具有自组织、自适应和自学习性。适应性较强的个体具有强生存能力。

(4) GA 具有隐性的并行性，且算法简单，易于实现。

劣势：

(1) 编码、交叉、变异的方式对 GA 的效率和准确性影响较大。

(2) 可能需要尝试多种的交叉率、变异率以提高 GA 效率。

(3) 通常 GA 的效率比其他传统的优化方法低。

(4) GA 的可行度、精确度、计算的复杂性等方面，难以定量分析。

(5) 对于某些实际问题，要判断是否已经获得全局内的最优解有一定难度。

5.1.3 本课题中 GA 的 MATLAB 实现

本文仅罗列几个关键点和关键函数，详细源代码请见附录。本课题中将每一种 7 个数据点的取法作为一个生物个体。

实现过程中一些重要参数及变量有：

popSize：代表种群大小；

pop：代表种群列表；

fitnessTable：代表适应度列表；

crossRate：代表交叉概率；

mutateRate：代表变异概率；

maxGeneration：代表最大子代数量；

aimScore：代表目标分数；

以下为算法的具体实现：

(1) 编码方式

将 7 个选定的用于拟合的数据点在全部 51 个数据点中所处的位置序号，编码为一个含有 7 个不重复整数的数列，数列中的整数按升序排列，将此数列作为个体的基因染色体。

(2) 初始化种群

由函数 init.m 完成。函数随机生成 popSize 个生物个体的染色体，储存于种群列表 pop 内，作为初始种群。

(3) 评价适应度

由函数 fitness.m 完成。将定标成本的倒数作为一个解的适应度。函数对 pop 内的每个染色体运用定标成本计算函数进行计算，并将其得分储存在 fitnessTable 内。

(4) 选择过程

由函数 selection.m 完成。函数根据适应度选择将要繁殖后代的生物个体，储存进 pop 作为下一代种群的亲代。第 i 个生物个体被选中的概率为

$$p_i = \frac{\text{第 i 号个体适应度}}{\text{总适应度}}$$

函数先产生一个 fitnessSum 列表，含有 popSize 个元素，其中

$$\text{fitnessSum(i)} = \sum_{\text{k=1}}^{\text{i}} \text{fitnessTable(k)}$$

之后函数每次生成一个浮点随机数$0 \le \text{r} \le$ 总适应度，并用二分法在寻找最小的 x，满足 $\text{fitnessSum(x)} \ge \text{r}$， 再将 pop 中的第 x 个元素拷贝进入新种群 newPop。重复此步骤 popSize 次。

最后令 pop=newPop 即可。

(5) 交叉过程

由 cross.m 函数完成。

由于选择过程中产生的亲代的顺序是随机的，故本函数直接对相邻的两个亲代生物的基因染色体在随机位置进行单点交叉，两个亲代生物间发生交叉的概率为 crossRate。值得一提的是交叉可能会发生非法后代，即一个染色体中出现重复的整数。故必须对每个交叉产生的染色体使用 checkPop.m 函数进行校验并排序。若一个染色体中出现重复的数字，则将多余的重复数字赋值为一个 1 至 51 间的随机整数，并递归调用 checkPop.m 再次检验该染色体，直到其变为合法染色体为止。

(6) 变异过程

由 mutation.m 函数完成。

一个染色体发生变异的概率为 mutateRate。变异的方法是，随机选定染色体内的一个整数，随机尝试对其+1 或-1。若产生的新染色体仍然合法，变异便完成；否则，则变异不发生。这种变异方式对染色体的改变较小，不会使原本优秀的染色体被完全破坏，但仍可能带来新的更优解。

(7) 主函数及终止条件

终止条件被包含在主函数 main.m 内。

主函数调用 readData.m 函数读取样本数据。然后调用 init.m 生成初始种群并调用 fitness.m 计算其适应度。之后循环调用 selection.m、cross.m、mutation.m、fitness.m，反复生成新一代种群并计算适应度，直到满足终止条件。

最后输出整个过程中所得的最优解，即定标成本最低的 7 个拟合点的序号。

终止条件有两种： 找到目标成本小于 aimScore 的生物个体；或当前子代数大于 maxGeneration。

若希望找到全局内的最优解，应该去除前一种终止条件，即将 aimScore 参数设置为 90，并将 maxGenration 设置得足够大。

(8) 目前使用的参数设置

popSize=100;

crossRate=0.9;

mutateRate=0.05;

maxGeneration=1000;

aimScore=90;

6 尝试减少事前观测点数目的研究

对于拟合点而言，更少的取点可以节约研究成本，让我们的研究更加高效、迅速的开展。

6.1 事前观察点减少为 6 个

将 7 个拟合点减少为 6 个，考察前 10 代的定标方案的定标成本。

表 1　6 个拟合点 10 代定标成本

定标方案代数	种群内最优的拟合点						定标成本
1	2	3	12	25	39	47	128.6716
2	4	13	25	37	45	46	126.3454
3	4	13	25	37	47	49	116.5490
4	4	13	25	37	47	49	116.5490
5	4	8	17	28	41	48	115.0981
6	4	8	17	28	41	49	113.7393
7	3	14	24	29	40	50	112.6375
8	3	14	24	29	40	50	112.6375
9	2	11	23	29	39	48	111.1866
10	2	11	23	29	39	48	111.1866

可以发现，6 个拟合点的定标方案可以达到 115 以下的定标成本，之后要达到 110 以下难度较大。劣于 7 个拟合点。本次运行总共进行了 96s。

6.2 事前观察点减少为 5 个

将 7 个拟合点减少为 5 个，考察前 10 代的定标方案的定标成本。

表 2　5 个拟合点 10 代定标成本

定标方案代数	种群内最优的拟合点					定标成本
1	1	14	26	39	48	133.2313
2	2	14	25	39	48	131.7036
3	2	16	24	39	48	130.0650
4	2	16	24	39	48	130.0650
5	4	14	26	39	50	129.2281
6	4	14	26	39	50	129.2281
7	4	14	26	39	49	128.2985
8	4	14	26	39	49	128.2985
9	4	14	26	39	49	128.2985
10	4	14	26	39	49	128.2985

可以发现，5 个拟合点的定标方案只能达到 130 以下的定标成本，比 6 个拟合点的方案更为劣势。本次运行总共进行了 104s。

7 结论

使用上述 MATLAB 实现的遗传算法可以成功的找到若干种成本在 95 以下的定标方案。

其中拟合或插值方法使用三次样条插值方法，该方法相对于多项式拟合以及三次 Hermite 插值法更为优秀，兼具运行效率和准确度。

通常，算法开始运行后，经过 5-10 分钟，即可找到第一种成本在 95 以下的定标方案。

本小组采用 5.1.3 中所述参数设置，同时并行运行了 3 次 GA，所得结果如表 3 所示。

表 3　优秀定标方案

使用三次样条插值	拟合点序号							定标准确度
定标方案 1	1	8	19	26	33	44	50	94.8582
定标方案 2	2	9	20	27	34	43	50	93.9456
定标方案 3	3	9	18	26	34	43	50	94.5480

其中最优秀的定标方案是：选取第 2、9、20、27、34、43、50 个数据点作为拟合点，再利用三次样条插值法进行定标，可以使定标成本低至 93.9456。

8 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] 三次样条插值_百度百科 http://baike.baidu.com/link?url=bZefaKxXPmng2K5GyD42brivz4vPJSlqP1GUgnWTRoUR1m_uIRyH24LwJhD84X_jK1X2vl6YMEgW6HohLhcKsq

[3] 最小二乘法的基本原理和多项式拟合 MATLAB 实现 http://wenku.baidu.com/link?url=3uc1mMLwUy5emTE8O0lo3sv2clWWFd6k_3BZ9NMkA__m93GrYCzqzWsDuON1cWjE3UU8gnG6dQQqSf1qumW6N_U559Po-ZFueRIJmy0604W

[4] MATLAB 中插值函数汇总和使用说明 http://wenku.baidu.com/link?url=RaIyc4FLzG1m2OOs9eh2pKCPgu1Zu2F0NJR4O6fU7NolatGB3Md2d6skym2_7RsGAk5j2wJTuGt-7llICGogOXoaNkSOIPeZKVy1rFpGDDS

[5] 遗传算法_百度百科 http://baike.baidu.com/link?url=O0b4P0D0fdVtkhWRtVDs_ZzsdRYeuBtNk5rONQ1Sh5S7EYYeYy3C-JYRFWQ6dN2Atbt6QIIdIaqAwplb4Ci9dq

附录 代码的实现

① main.m

```
%使用三次样条插值的遗传算法
%运行过程中显示的tmpMin表示当前种群中的最低成本
%tmpAvg表示当前种群中的平均成本
%minScore表示目前整个运行中出现过的最低成本
%minScoreMethod表示上述最低成本对应的七个拟合点
%g表示当前种群的子代数

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=7;                %观察点数量
crossRate=0.9;             %交叉概率
mutateRate=0.05;           %变异概率
maxGeneration=1000;        %最大子代数
aimScore=90;               %目标成本
minScore=200;              %最低成本

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

readData();
init(popSize,codeSize,51);
fitnessTable=zeros(popSize,1);
for i=1:popSize
    fitnessTable(i)=fitness(pop(i,:));
end
tmpMin=min(fitnessTable)

if tmpMin<minScore
    minScore=tmpMin;

minScoreLocation=find(fitnessTable==tmpMin);

minScoreMethod=pop(minScoreLocation,:);
end
minScore
minScoreMethod
if minScore<aimScore
    return;
end
g=1;
while g<=maxGeneration
    selection(popSize,codeSize);
    cross(popSize,codeSize,crossRate);
    mutation(popSize,codeSize,mutateRate);
    for i=1:popSize
        fitnessTable(i)=fitness(pop(i,:));
    end
    tmpMin=min(fitnessTable)

    if tmpMin<minScore
        minScore=tmpMin;

minScoreLocation=find(fitnessTable==tmpMin);

minScoreMethod=pop(minScoreLocation,:);
    end
    minScore
    minScoreMethod
    if minScore<aimScore
        break;
    end
    g
    g=g+1;
end
```

② readData.m

```
function readData()
origin=xlsread('dataform.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
```

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
```

④ fitness.m

```
function [score] = fitness(method)
global voltageX;
global outputY;
global len;                          %样本组容量
codeSize=length(method);          %codeSize=7
sum_A=0;
X=zeros(1,codeSize);
Y=zeros(1,codeSize);
for i=1:len
    A=0;
    for j=1:codeSize
        X(j)=voltageX(i,method(j));
        Y(j)=outputY(i,method(j));
    end
    Y1=interp1(X,Y,voltageX(i,:),'spline');
    yTable=abs(Y1-outputY(i,:));
    for j=1:51
        dif=yTable(j);
        if dif<=0.5
            A=A+0;
        elseif dif<=1
            A=A+0.5;
        elseif dif<=2
            A=A+1.5;
        elseif dif<=3
            A=A+6;
        elseif dif<=5
            A=A+12;
        elseif dif>5
            A=A+25;
        end
    end
    sum_A=sum_A+A;
end
score=sum_A/len+84;
end
```

⑤ 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 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]);
    tmp=pop(i,p:codeSize);
    pop(i,p:codeSize)=pop(i+1,p:codeSize);
    pop(i+1,p:codeSize)=tmp;
    checkPop(i,codeSize);
    checkPop(i+1,codeSize);
end
end
```

⑦ checkPop.m

```
function checkPop(n,codeSize)
global pop;
pop(n,:)=sort(pop(n,:));
flag=0;
for j=2:codeSize
    if pop(n,j-1)==pop(n,j)
        pop(n,j)=randi(51);
        flag=1;
    end
end
if flag
    checkPop(n,codeSize);
end
end
```

⑧ mutation.m

```
function
mutation(popSize,codeSize,mutateRate)
global pop;
for i=1:popSize
    r=rand();
    if r<mutateRate
        r=randi(codeSize);
        if rand()>0.5
            x=1;
        else
            x=-1;
        end
        tmp=pop(i,r)+x;
        if (r==1)
            if
(tmp>=1)&&(pop(i,r+1)~=tmp)
                pop(i,r)=tmp;
            end
        elseif (r==codeSize)
            if (tmp<=51)&&(pop(i,r-
1)~=tmp)
                pop(i,r)=tmp;
            end
        else
            if (pop(i,r-
```

```
1)~=tmp)&&(pop(i,r+1)~=tmp)
                        pop(i,r)=tmp;
                    end
                end
                %checkPop(i,codeSize);
            end
        end
    end
```