统计推断在数模转换系统中的应用

第 63 组　彭诗奇 5130309120　　赖柏霖 5130309118

摘要：本报告主要展示了统计推断在数模、模数转换系统中的应用，以为某种电子产品模块的传感特性校准设计方案为研究实例，研究和分析 469 组电压信号 X 和被测物理量 Y 的关系，运用一定的数理统计方法，经过特征点选取、算法研究、拟合比较等一系列过程，借助 MATLAB 建立电压 X 和被测物理量 Y 的关系曲线模型，最终得到了一种只要六个点定标的取点方案，并有对其测试和评价标明该定标方案是有效的，实现以少量数据反映整体系统特性的效果，从而有效降低工程设计上的成本，提高效率。

关键词：样本，特征点，曲线拟合，样条差值拟合，遗传算法

APPLICATION OF STATISTICAL INFERENCE IN A DIGITAL TO ANALOG CONVERSION SYSTEM

Abstract: This report mainly shows the application of the statistical inference in ADC and DAC system, take some electronic module design scheme of sensing characteristics for instance, research and analyze 469 groups of X voltage signals and their relationships with another measured physical quantity Y, and use certain methods of mathematical statistics, feature selected points, algorithm, fitting comparison and a series of other procedures, using MATLAB to establish the curve model of the relationship of the voltage X and the measured quantities Y, finally getting a six-point set as the solution, and having tested and evaluated the calibration solution is effective to achieve our aim which is to use a small amount of data to reflect the characteristics of the overall system, thus effectively reducing the cost of designing the project and improving efficiency.

Keywords: sample, feature points, curve fitting, spline, Genetic Algorithm,

1 引言

在生产和研究中，对物理现象的研究可以通过确定其输入输出的关系来确定其性质，这就有必要提出一种解决方案来确定这二者的关系。但是实际研究生产中发现，我们很难找到一个能够精准描述二者关系的函数，这就要求我们运用实际测得的数据，通过曲线拟合来找到一个最符合输入与输出关系的函数。我们主要通过三次样条差值拟合，并借助于遗传算法的理论，不停地寻找不同的尽可能优的定标方案，使得总体的定标成本处于较低水平。本次统计推断的报告是通过研究电子产品输入的电压信号 X 与被监测的物理量 Y 的关系，分析大量的数据得出尽可能优的解，来说明统计推断在获得输入输出关系时的作用。

2 具体问题

2.1 研究对象

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。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。监测模块的组成框图如图 1。

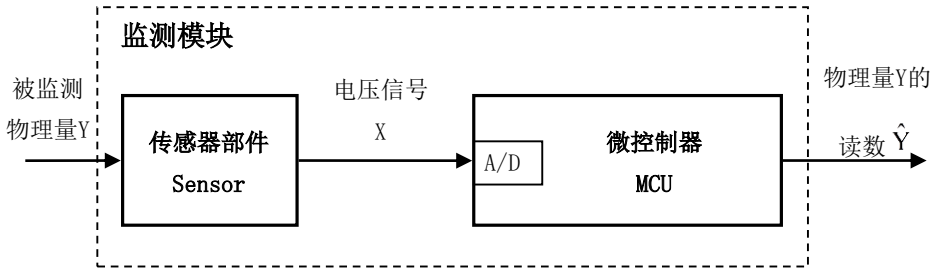


图 1 监测模块组成框图

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）。

我们需要针对某一特定传感部件个体，通过有限次测定，估计其 Y 值与 X 值间一一对应的特性关系的过程。数学上可认为是确定适用于该个体的估测函数 $\hat{y}=f(x)$ 的过程，其中 x 是 X 的取值，$\hat{y}$ 是对应 Y 的估测值。

2.2 寻找定标点组合

对于该电子产品，有相应的 Y-X 函数关系，由于该装置的函数关系为非线性，故现需要针对已有的 469 组实验测量数据，对 Y-X 函数关系进行分析，并寻找合适的拟合方法，给出求定标点组合的方法。

3 通过穷举法的拟合实现

3.1 取点数目的分析

就该系统而言，测定 51 组数据后，系统特性已经被充分定标，若对每个样本都进行 51 组数据的完整测定，既费时间又耗成本，没有直接的工程实用意义。我们需要得到一种根据较少的点数就能得到较好的 Y-X 函数关系的一种实现方法。

经过后期多次拟合尝试的结果确定，我们在这里认为取 7 个点为能达到预期效果的保守取点数目（C（7,51）约为 1 亿，大概能接受，而 C（8,51）约为 6 亿，过大不太合适）。

3.2 区间的划分

为了大致确定这 7 个点应该存在的范围，故需要对 Y-X 特性做大致的划分，以便使 7 个点的选取不至于过于集中，使得 7 个点在相应的范围内变动，这样也可以大大减少接下来通过穷举法来确定特征点的计算量。

3.2.1 基本思路

根据每个电子元件的 Y-X 特性关系曲线形状具有相似性，故可以大致分为 3 个区间进行拟合。拟合时可选用不同的方法进行，最后通过计算比较拟合误差来选定拟合方法，进而给出求拟合表达式的方法。

3.2.2 区间划分的确定

区间划分应该具有普适性，即能够对 469 组实验数据同时具有划分的意义。由于 X 的坐标一样，故以 X 为横坐标，绘制 Y-X 图，并通过对 Y 求二阶差分来判断三个区间的电压 X 划分点（每组 51 个数据点的第一个点和最后一个点已经排除在外，实际计算为 49 个点）。

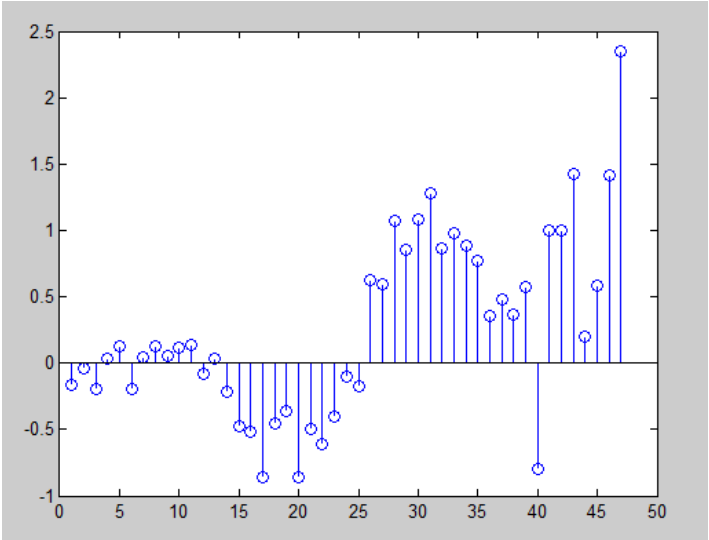


图 2 Y-X 的二阶差分离散图

由图 2 可以看出，图像大致分为三段，即 1-14 点为一段，15-25 点为一段，26-47 点为一段。

3.3 数学模型的建立

根据曲线的大致形状的划分方法，建立三次多项式拟合和三次样条插值法的数学模型。

3.3.1 三次多项式拟合

此方法为拟合的常用简单方法，即设一在区间[a, b]上的 n 阶多项式函数

$$P(x) = \sum_{k=0}^{n} a_k x^k \tag{1}$$

其中 a=x0<x1<…<xn=b 是数据点，则它为区间[a, b]上对数据点拟合的多项式。由定义可看出，它有 n+1 个待定系数。确定这些系数 ak 使

$$D = \sum_{k=1}^{51} [y_k - P(x_k)]^2 \tag{2}$$

最小。这时就得到误差平方拟合多项式。可以把它认为是推断函数。同样，我们利用 MATLAB 中已有的函数

$$a = \text{polyfit}(x_0, y_0, m) \tag{3}$$

进行多项式拟合。其中输入参数 x_0，y_0 为要拟合的数据，m 为拟合多项式的次数，输出参数 a 为拟合多项式系数

$$y = a_m x^m + L + a_1 x + a_0 \tag{4}$$

系数 a=[am,L,a1,a0]。多项式在 x 处的值 y 可用 MATLAB 中的 y=polyval(a,x)函数进行计算。

多项式拟合的次数在一定范围内越高，残差、方差等参数越小，拟合曲线与实际测量点的相关性越高，拟合程度越好。但次数的增高导致算法运行时间的延长，效率的降低，而且当次数超过一定范围时，甚至适得其反。例如用 5，6 次函数拟合，运行时间是很难让人接

受的，所得结果误差反而越来越大，可见不一定次数越高，拟合曲线就越接近实际曲线。反复测试后，3 次曲线拟合的效果最好。

3.3.2 三次样条差值拟合

三次样条插值拟合，即用6段三次曲线来近似表示。选出7个特征点之后，对于中间的点而言，取中间点加上两侧的与中间点相邻的两个点作为研究对象。通过四个点确定出函数关系方程作为中间点之间曲线方程，以此类推。例如，对于S_3、S_6之间而言，由$S_3S_4S_5S_6$确定一条3次曲线，而后仅在S_4S_5之间以该曲线表示，其它点同理。而对于边缘上的点而言，如对点S_1和S_2，由$S_1S_2S_3S_5$确定一条3次曲线，而后仅在S_1S_2之间以该曲线表示。其优点在于它对于相邻两点之间都采用一条三次曲线来表示,这样就能使整体曲线非常平滑,且准确度高。但是该方案的缺点在于每两个点之间都用一条三次曲线拟合,其过程非常复杂，对于程序的编写和MATLAB的运算量都是一大考验。

三次样条插值函数在 MATLAB 中可以使用 spline 函数实现。

3.4 穷举法特征点的确定

3.4.1 穷举法特征点的划分

根据之前三段区间的划分，按比例分配大致为前两个区间各两个点，第三个区间取三个点。按照这种取点方法，一共有$C_{13}^2C_{12}^2C_{22}^3 = 7927920$种。这样可以找到拟合效果最好的 7 个点，但是计算量偏大。而由进一步观察分析可知，第一段区间的二阶差分基本在 0 附近，有较好的线性关系，故可将线性区三等分，取中间两个点，即第 5 点和第 9 点。这样就变成了$C_{12}^2C_{22}^3 = 101640$种取点方法。

3.4.2 特征点的数学评估方法

（1）对任意给定的对象，先测得一组观测值$\{(X_i, Y_i)\}$ 其中 $i = i_1, i_2, \dots, i_p$, p<51

（2）利用统计数学方法，估测 $\{(X_i, \hat{Y}_i)\}$ 其中 j = 1,2,3,…， 51

方法一：拟合法

当 i = j 时，允许$\hat{Y}_i \neq Y_j$

方法二：插值法

当 i = j 时，必须有$\hat{Y}_i = Y_j$

对于评价估测的准确程度，有两种方法。

（1） 增加测量，获得更多的观测值。

（2） 对于估测值和观测值，给出定量评价。

我们在本课题中规定评价函数如下：

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

$$C=\frac{1}{M}\sum_{i=1}^{M}S_i \qquad (6)$$

3.5 使用 MATLAB 编程计算

利用 MATLAB 编程计算，尝试左右 7 点下标组合，获得最低的定标成本，可得如下结果：

三次多项式拟合：MIN = 19.1087 + 20 * 7 = 159.1087， MATLAB 程序见附件。

上述计算结果是基于第一次的定标成本表达式算出的。后来发现遗传算法更适合解决该问题，而且暴力穷举法既费时间也得不到好的结果，故没有再次计算三次样条差值的定标成本。

4. 遗传算法

4.1 遗传算法简介

遗传算法（Genetic Algorithm）是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法。

遗传算法的基本运算过程如下：

a)初始化：设置进化代数计数器 t=0，设置最大进化代数 T，随机生成 M 个个体作为初始群体 P(0)。

b)个体评价：计算群体 P(t)中各个个体的适应度。

c)选择运算:将选择算子作用于群体。选择的目的是把优化的个体直接遗传到下一代或通过配对交叉产生新的个体再遗传到下一代。选择操作是建立在群体中个体的适应度评估基础上的。

d)交叉运算：将交叉算子作用于群体。遗传算法中起核心作用的就是交叉算子。

e)变异运算：将变异算子作用于群体。即是对群体中的个体串的某些基因座上的基因值作变动。

群体 P(t)经过选择、交叉、变异运算之后得到下一代群体 P(t+1)。

f)终止条件判断:若 t=T,则以进化过程中所得到的具有最大适应度个体作为最优解输出，终止计算。

4.2 借助 MATLAB 在本问题中的运用

（1）编码

采用二进制编码，即用一个二进制串来表示一个个体。在本问题中，每个样本都有 51 个点，故采用一个 51 位的 01 二进制串来表示一个个体，保存在一个矩阵中。在初始生成种群时随机产生 0 或 1,以保证初始种群个体的随机性。该二进制串的意义为，若某一位为 1，则选取该点进行定标，若为 0，则不选取该点定标。

（2）解码

解码即根据不同的编码计算每个个体相对应的适应度函数。选出某个个体的二进制编码为 1 的所有位，对这些点组成的集合进行曲线拟合并定标，算出来的定标值即为该个体的适应度函数。

详细代码参见附录中的 fitness.m。

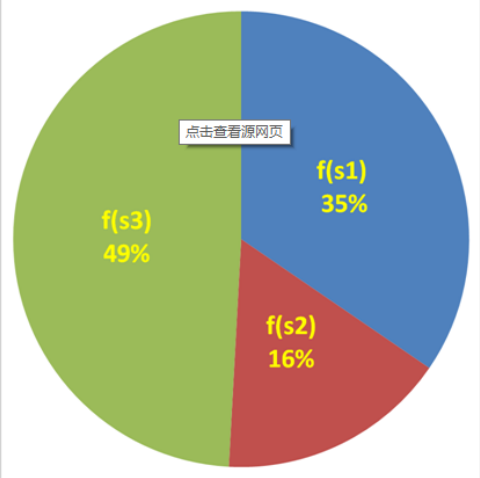
（3）初始化

在开始遗传算法的迭代过程之前，需要对种群进行初始化。设种群大小为 pop_size，每个个体的染色体长度为 chromo_size，种群的大小决定了种群的多样性，而染色体的长度则是由前述的编码过程决定的。一般是随机生成种群，但是如果知道种群的实际分布，也可以按照此分布来生成初始种群。本问题采用随机生成种群的办法，假设生成的初始种群为（v1,v2,…,vpop_size）。

详细代码参见附录中的 initialize.m。

（4）选择操作

选择操作即从前代种群中选择个体到下一代种群的过程。一般根据个体适应度的分布来选择个体。以初始种群（v1,v2,……,vpop_size）为例，假设每个个体的适应度为(fitness(v1),fitness(v2),…,fitness(vpop_size))，一般适应度就可以按照解码的过程进行计算。以轮盘赌的方式选择个体，如下图：



图三 适应度轮盘

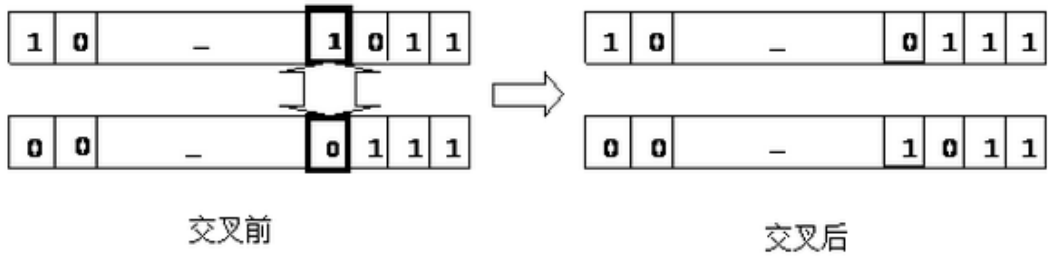
随机转动一下轮盘，当轮盘停止转动时，若指针指向某个个体，则该个体被选中。很明显，具有较高适应度的个体比具有较低适应度的个体更有机会被选中。但是这种选择具有随机性，在选择的过程中可能会丢失掉比较好的个体，所以可以使用精英机制，将前代最优个体直接选到下一代中。

在选择操作中，用到了排序算法，将当代所有个体按适应度函数从低到高排序，我采用的是选择排序，时间复杂度为 O(N^2)。

详细代码参见附录中的 rank.m 和 selection.m。

（5）交叉操作

交叉操作是对任意两个个体进行的（在这里我们实现的算法是直接对相邻的两个个体进行的）。随机选择两个个体，如下图所示：



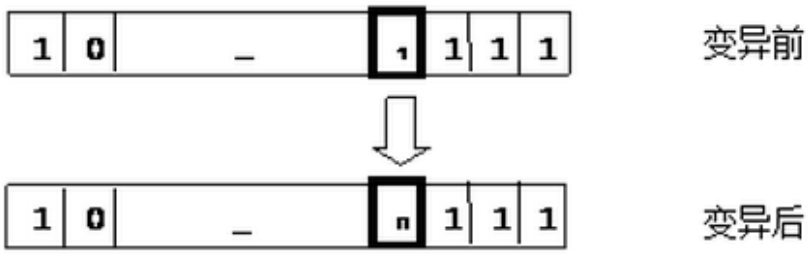
图四 交叉操作示意图

然后随机生成一个实数 0 <= r <= 1，如果 r < cross_rate， 0 < cross_rate < 1 为交叉概率，则对这两个个体进行交叉，否则则不进行。如果需要进行交叉，再随机选择交叉位置(rand * chromo_size)，如果等于 0 或者 1，将不进行交叉。否则将交叉位置以后的二进制串进行对换（包括交叉位置）。有时候还有多点交叉，我们在解决此问题的过程中，仅运用了单点交叉。

详细代码参见附录中的 crossover.m。

（6）变异操作

变异操作是对单个个体进行的。首先生成一个随机实数 0 <= r <= 1，如果 r < mutate_rate，则对此个体进行变异操作，0 < mutate_rate < 1 为变异概率，一般为一个比较小的实数。对每一个个体进行变异操作，如下图所示：



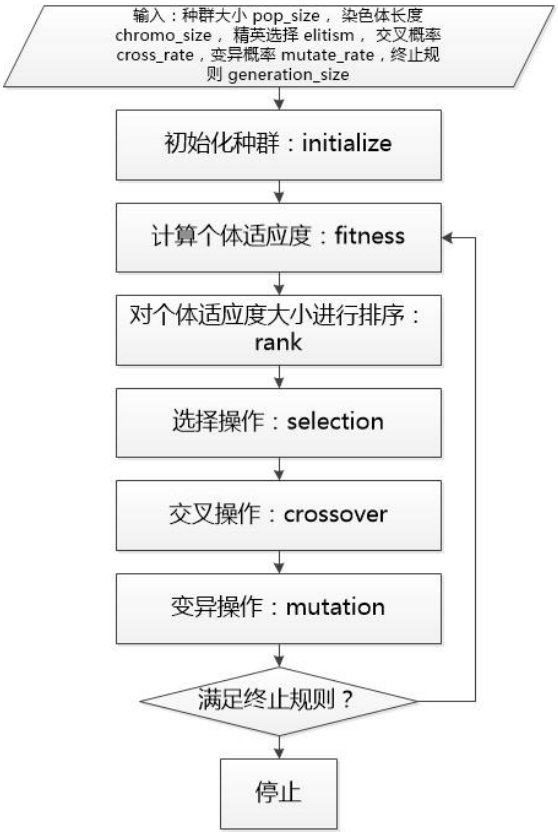
图五 变异操作示意图

如个体需要进行变异操作，首先需要确定变异位置(rand * chromo_size)，若为 0 则不进行变异，否则则对该位置的二进制数字进行变异：1 变成 0，0 变成 1。（当然也可以选择多点进行变异）

详细代码参见附录中的 mutation.m。

（7）遗传算法流程

遗传算法流程图如下图所示：



图六 遗传算法流程图

遗传算法的主函数详细代码参见附录中的 GeneticAlgorithm.m。

4.3 MATLAB 运行结果及其分析

由于遗传算法是用来求最大值的方法，而本问题是要求定标成本的最小值，所以我们首先得在计算适应度函数时将求最小值转化为求最大值，并且还得保证适应度函数为正数，否则在选择操作时无法处理。

我采用了两种方法解决了这个问题：

第一种是取相反数后再加上一个正常数，即

fitness(i)=-fitness(i) + C;

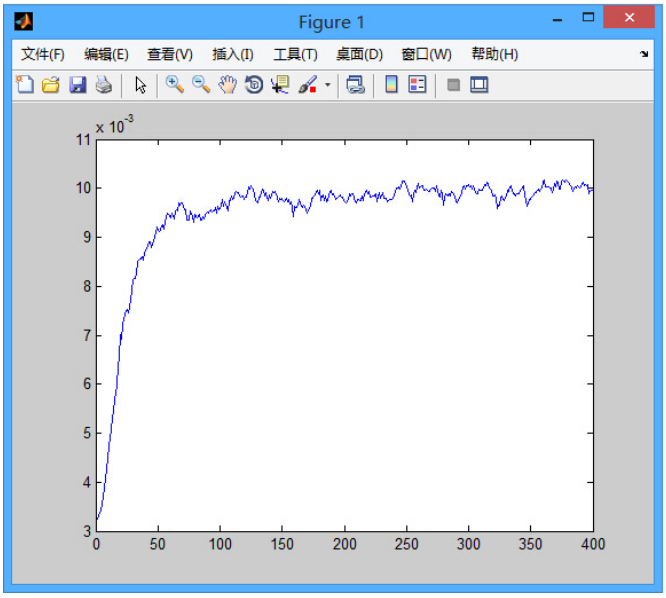
第二种是取倒数，即

fitness(i) = 1 / fitness(i).

在拟合方式为 spline 的情况下，运行结果如下表所示：

表一 spline 拟合运行结果表

运行次数	处理方式	最佳适应度	最佳取点方案	出现代数
1	相反数	105	1 9 15 22 29 38 48 50	444
2	相反数	95.4	3 10 18 27 34 44 49	345
3	相反数	95.3	1 8 20 26 34 45 50	226
4	倒数	93.4	2 10 21 30 41 50	260



图七 第四次运行历代平均值曲线图

其中第一次运行采用的是 fitness(i) = 100000 – fitness(i)，第二次和第三次运行采用的是 fitness(i) = 1000 - fitness(i)。

从图七中可以看出，遗传算法在运行了 100 代之后平均值趋于稳定，说明遗传算法能够很好地找到近似最优解的答案。

因为遗传算法的选择、交叉、变异等过程是完全随机的，一代代地淘汰不够优秀的个体，但是同时，产生新的优秀个体的过程也是随机的，我们无法控制最后的结果是 6 个点、7 个点还是 8 个点，只能通过增加种群数量来增加个体多样性，来提高产生更优秀个体的可能。前 3 次运行都是 500 代遗传，第 4 次运行是 400 代遗传，前两次运行的种群规模为 100，后两次运行的种群规模为 200，并由此可见种群数量对于结果的重要性。

由于第一次常数 100000 过大，所以在后期不同个体间的适应度函数差别不大，不太能显现出优秀个体的优越性，故最终答案为 8 个点，不够优秀。后改为 1000，明显可见最佳适应度有所降低，取点个数也由 8 个减少为了 7 个，获得了更优秀的定标方案。

通过后两次运行的比较可知，取倒数比取相反数的方法更适合来解决这个问题，并且能获得更优秀的解，最后取点方案成功减少至 6 个点，最佳适应度也降低至 93.4。

4.4 思考与改进

三次样条差值函数 spline 可以保证在各点处连续，但是不光滑，在此基础上可以有所改进。改进后的拟合方式为：分段三次 Hermite 插值。

三次 Hermite 插值即通过平面上的两个点及它们的切线斜率来确定一个三次函数，一般需要借助于 Lagrange 插值来实现。在此基础上，只要让相邻两点的导数相同就可以得到平滑的曲线。而且这样的分段插值是数值稳定的，不会产生高次多项式带来的龙格现象。

分段三次 Hermite 插值在 MATLAB 中可以借助于 pchip 函数来实现。

运行结果如下：

表二 pchip 拟合运行结果表

运行次数	最佳适应度	最佳取点方案
1	86.24	3 12 22 29 38 49
2	85.71	3 13 23 30 38 49
3	85.09	4 15 23 30 38 49

三次运行都是 500 代遗传，种群数量为 200，采用 pchip 拟合方式得出的结果。由最佳适应度可明显看出，pchip 比 spline 能拟合出更优的样本特性。而且，上述 spline 找出的最优取点集合[2 10 21 30 41 50],将其用 pchip 拟合后，适应度由 93.4 降低为 90.75，也同样说明了 pchip 比 spline 拟合后误差更小。

5 总结

从以上的结果可以看出，遗传算法可以有效地解决这种组合优化问题，用 spline 可以较好的拟合出样本特性，而 pchip 可以得出更优的拟合曲线。最后运用遗传算法，在两种拟合方式下均得到了较好的结果。

最后，本课题的结论：在误差允许的范围内，给定六个点，可以用三次样条插值或三次 Hermite 插值来完成标定，取点方案见表三：

表三 问题的解

拟合方式	最佳取点方案	最佳拟合值
三次样条插值	2 10 21 30 41 50	93.4
三次 Hermite 插值	4 15 23 30 38 49	85.09

6 致谢

感谢袁焱老师在讲座及面谈中的悉心指导！

感谢同学们在我们困难时给予的帮助！

7 参考文献

[1]上海交通大学电子工程系.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.曲线拟合-维基百科，自由的百科全书[M/OL].[2013-10-26].http://zh.wikipedia.org/wiki/曲线拟合

[3]维基百科.样条插值-维基百科，自由的百科全书[M/OL].[2013-10-26].http://zh.wikipedia.org/wiki/样条插值

[4]百度百科——遗传算法：http://baike.baidu.com/view/45853.htm

[5]简单遗传算法 MATLAB 实现.
http://www.cnblogs.com/biaoyu/archive/2011/12/03/2274604.html

[6]Web 技术研究所. 分段三次 hermite 插值.
http://www.web-tinker.com/article/20267.html

8 附录

附上程序代码清单：（注意：运行时 chromo_size 必须设置为 51）

initialize.m

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

function initialize(pop_size, chromo_size)
global pop;
for i=1:pop_size
    for j=1:chromo_size
        pop(i, j) = round(rand);
    end
end

clear i;
clear j;
```

fitness.m

```
%计算种群个体适应度
%pop_size: 种群大小
%chromo_size: 染色体长度

function fitness(pop_size, chromo_size)
global fitness_value;
global pop;
global G;

for i=1:pop_size
    fitness_value(i) = 0;
end

data=csvread('20141010dataform.csv');
a=1:2:(length(data)-1); %奇数行编号
b=2:2:length(data);     %偶数行编号
X=data(a,:);            %横坐标数据向量
Y=data(b,:);            %纵坐标数据向量

for i=1:pop_size
    point = [];         %选取的点
```

```
    for j=1:chromo_size
        if pop(i, j) == 1
            fitness_value(i) = fitness_value(i) + 12 * length(a);
            point = [point, j];
        end
    end

    if length(point) < 3
        fitness_value(i) = 0;
        continue;
    end

    point_x = X(:,point);        %469 组的所有选取点的横坐标
    point_y = Y(:,point);        %469 组的所有选取点的纵坐标

    for q=1:length(a)            %计算 469 组数据
        point_x_q = point_x(q,:);        %第 q 组选取点的横坐标
        point_y_q = point_y(q,:);        %第 q 组选取点的纵坐标

        %p3 = polyfit(point_x_q, point_y_q, 3);  %3 次多项式拟合的系数
        point_x_q_all = X(q,:);          %第 q 组所有点的横坐标实际值
        point_y_q_all = Y(q,:);          %第 q 组所有点的纵坐标实际值
        %y_current = polyval (p3, point_x_q_all);     %第 q 组所有点的纵坐标拟合
值
        y_current = spline(point_x_q, point_y_q,point_x_q_all);

        Q = abs(point_y_q_all - y_current);

        for j=1:51
            if Q(j) > 5
                fitness_value(i) = fitness_value(i) + 25;
                else if Q(j) > 3
                        fitness_value(i) = fitness_value(i) + 12;
                        else if Q(j) > 2
                                fitness_value(i) = fitness_value(i) + 6;
                                else if Q(j) > 1
                                        fitness_value(i)  =  fitness_value(i)  +
1.5;
                                        else if Q(j) > 0.5
                                                fitness_value(i)              =
fitness_value(i) + 0.5;
                                                else if Q(j) >= 0
                                                        fitness_value(i)      =
fitness_value(i) + 0;
```

```
                                                    end
                                                end
                                            end
                                        end
                                    end
                                end
                            end
                        end
                    end

    fitness_value(i) = fitness_value(i) / 469;
    %fitness_value(i) = 1000 - fitness_value(i);
    fitness_value(i) = 1 / fitness_value(i);
    %if fitness_value(i) < 0
    %    fitness_value(i) = 0;
    %end
end

clear i;
clear j;
```

rank.m

```
%对个体按适应度大小进行排序，并且保存最佳个体
%pop_size: 种群大小
%chromo_size: 染色体长度

function rank(pop_size, chromo_size)
global fitness_value;
global fitness_table;
global fitness_avg;
global best_fitness;
global best_individual;
global best_generation;
global pop;
global G;

for i=1:pop_size
    fitness_table(i) = 0.;
end

%插入排序,从小到大
min = 1;
temp = 1;
temp1(chromo_size)=0;
```

```
for i=1:pop_size
    min= i;
    for j = i+1:pop_size
        if fitness_value(j) < fitness_value(min);
            min = j;
        end
    end
    if min~=i
        temp = fitness_value(i);
        fitness_value(i) = fitness_value(min);
        fitness_value(min) = temp;
        for k = 1:chromo_size
            temp1(k) = pop(i,k);
            pop(i,k) = pop(min,k);
            pop(min,k) = temp1(k);
        end
    end
end

%计算累积适应度
for i=1:pop_size
    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(pop_size)/pop_size;  %第 G 代的平均适应度

if  1 / fitness_value(pop_size) < best_fitness
    best_fitness = 1 / fitness_value(pop_size);
    best_generation = G;
    for j=1:chromo_size
        best_individual(j) = pop(pop_size,j);
    end
end

G
1/fitness_value(pop_size)
best_fitness

clear i;
clear j;
```

```
clear k;
clear min;
clear temp;
clear temp1;
```

selection.m

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

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

for i=1:pop_size
    r = rand * fitness_table(pop_size);
    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

if elitism
```

```
    p = pop_size-1;
else
    p = pop_size;
end
for i=1:p
   for j=1:chromo_size
        pop(i,j) = pop_new(i,j);
   end
end

clear i;
clear j;
clear pop_new;
clear first;
clear last;
clear idx;
clear mid;
```

crossover.m

```
%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;
```

mutation.m

```
%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*chromo_size);
        if mutate_pos == 0
            continue;
        end
        pop(i,mutate_pos) = 1 - pop(i, mutate_pos);
    end
end

clear i;
clear mutate_pos;
```

plotGA.m

```
%打印算法迭代过程
%generation_size: 迭代次数

function plotGA(generation_size)
global fitness_avg;
x = 1:1:generation_size;
y = fitness_avg;
plot(x,y)
```

GeneticAlgorithm.m

```
%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 fitness_value;%当前代适应度矩阵
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_generation;%最佳个体出现代

fitness_avg = zeros(generation_size,1);

fitness_value(pop_size) = 0;
best_fitness = 9999;
best_generation = 0;
initialize(pop_size, chromo_size);%初始化

for G=1:generation_size
    fitness(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, elitism);%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
end

plotGA(generation_size);%打印算法迭代过程
m = best_individual;%获得最佳个体
n = best_fitness;%获得最佳适应度
p = best_generation;%获得最佳个体出现代

%获得最佳个体变量值
q = [];
for j=1:chromo_size
    if best_individual(j) == 1
            q = [q, j];
    end
end
```

```
m
n
p
q

clear i;
clear j;
```