一个测量装置在大规模制造中的标定问题

组号：57　　小组成员：桑锐 518021911085

摘要：本报告主要分析研究了在大规模制造中，温度传感器的标定问题。基于上海交通大学电院电子系实验测得的共计 500 组输入信号温度 T 与输出信号电压 V 的数据，使用插值拟合、特征点选取以及遗传算法等分析方法，借助 Matlab 数学软件编程，进而得到在标定成本最小的情况下传感器的输入与被测物理量温度的关系，实现了较低成本的标定。
关键词：曲线拟合，样条插值，遗传算法，Matlab

Calibration of a measuring device in large-scale manufacturing

ABSTRACT: This report mainly analyzes the calibration of temperature sensors in large-scale manufacturing. Based on the data of a total of 500 sets of input signal temperature T and output signal voltage V measured by the Department of Electronics, Shanghai Jiao Tong University, using interpolation analysis, feature point selection, and genetic algorithm, etc., and programming with Matlab mathematical software, and then get The relationship between the input of the sensor and the temperature of the measured physical quantity under the condition of the smallest calibration cost, to achieve a lower cost calibration.
Key words: Polynomial fitting, Spline interpolation, Genetic algorithm, Matlab

1 引言

许多大规模制造的电子产品中包含带测量功能的模块（测量装置），用于监测某种物理量，比如环境温度、压力或光照强度等。任何一种测量装置在制造时一般都要经过标定。本课题假设一种环境温度测量装置，可测范围是-20℃至 70℃，输入输出呈现出非线性关系，且传感器存在个体差异。课题要求为该测量装置设计一种在大规模制造中的成本较低标定方案，以实现高效的标定工序[1]。

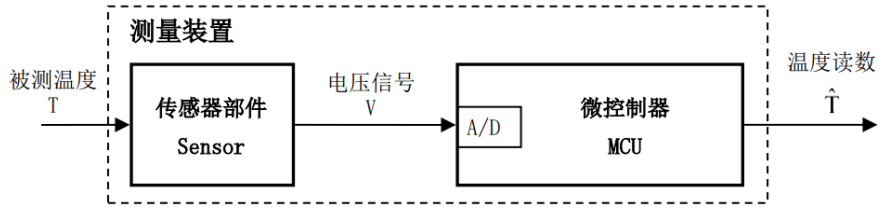

图 1　装置原理框图

2 拟合插值方法的选择

在数值分析中，可以使用线性插值，多项式插值，三次样条插值，立方插值等方法实现。在本次课题中，我们采取三次样条插值法进行插值的实现。

三次样条插值法是曲线插值最常用的插值方法之一。相比较于线性插值、二次多项式插值，三次样条插值的结果更加符合实际情况，得到的函数曲线也更加“平滑”。

在 Matlab 中，三次样条插值函数表示为：spline().

3 成本计算规则及成本函数的定义

在本次课题中，为评估不同标定方案的优劣，我们定义了标定成本函数。总体成本越低的标定方案，可以认定为较优方案。成本计算规则如下：

- 单点测定成本：

实施一次单点测定的成本记为符号 Q。本课题指定 Q = 50.

- 标定误差成本：

$$S_{i,j}=\begin{cases}0 & if\ \left|\widehat{T_{i,J}}-T_{i,j}\right|\le 0.5\\ 1 & if\ 0.5<\left|\widehat{T_{i,J}}-T_{i,j}\right|\le 1.0\\ 5 & if\ 1.0<\left|\widehat{T_{i,J}}-T_{i,j}\right|\le 1.5\\ 10 & if\ 1.5<\left|\widehat{T_{i,J}}-T_{i,j}\right|\le 2.0\\ 10000 & if\ \left|\widehat{T_{i,J}}-T_{i,j}\right|>2.0\end{cases} \tag{1}$$

单点误差对应的成本值按式（1）计算，以符号$S_{i,j}$记。其中$T_{i,j}$表示第 i 个样本与第 j 点的实际温度值，$\hat{T}_{i,j}$表示标定后得到的估计值（读数）。

单个样本个体的标定误差成本用和式（2）计算。

$$S_i=\sum_{j=1}^{90}S_{i,j} \tag{2}$$

- 样本个体标定成本：

$$c_{i=}S_i+Q\cdot n_i \tag{3}$$

样本个体的标定成本是测定成本与误差成本之和，见式（3）。式中n_i表示对该样本个体标定过程中的测定点数目。

- 方案总体成本：

标定方案的总体成本值按式（4）计算。对选定的样本数据集里的每个样本个体逐一开展标定，取所有样本个体标定成本的统计平均。其中 M 是该数据集样本总数。

$$C=\frac{1}{M}\sum_{i=1}^{M}c_i \tag{4}$$

4 遗传算法的实现

4.1 遗传算法简介

遗传算法是本课题得出最优解的求解算法，也是本课题的核心问题。

遗传算法是一种基于自然选择和基因遗传学原理的随机并行搜索算法，是一种寻求全局最优解而不需要任何初始化信息的高效优化方法[2]。它将问题的解集看作一个种群，通过不断地选择、交叉、变异等遗传操作，使解的质量越来越好。该算法具有全局寻优能力、适应性强、解决非线性问题具有较强的鲁棒性、对问题没有特定限制、计算过程简单、对搜索空间没有特殊要求、易于与其他算法结合等特点,在函数优化、图像处理、系统辨识、自动控制、经济预测和工程优化等领域得到了广泛的应用[3]。

遗传算法的总特征可以概括为：

(1) 生成一组候选解，设为初始种群；

(2) 根据适应性条件，测算种群的适应度大小；

(3) 根据适应度大小，通过 “交叉配对”“变异”保留部分较优的解，放弃其他的候选解；

(4) 不停执行操作 2~3，直到满足迭代退出条件，退出迭代，得到最优解。

遗传算法的流程图见图 2。

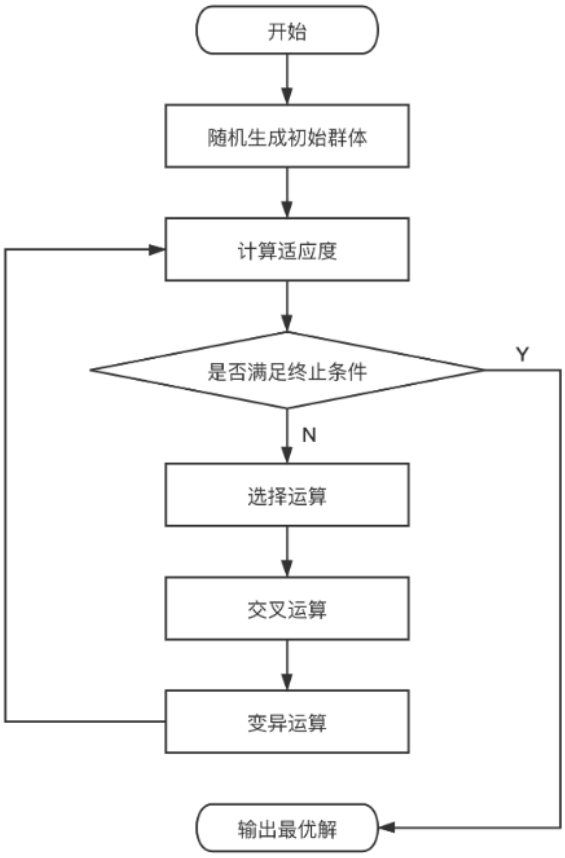

图 2 遗传算法的流程图

4.2 遗传算法在本课题中的实现

4.2.1 初始种群的生成

按照随机算法的思想，初始种群应随机生成初始种群。我们假设初始种群中有 100 组个体，每个个体的温度点为 90 个，故初始种群大小为 100*90。每一行代表一个个体，其中元素储存 1 或 0，分别代表改温度点是否被选中。考虑到标定的实际情况，我们假设每个元素值为 1 的概率为 0.4，为 0 的概率为 0.6。

4.2.2 适应度函数的选择

因标定成本的要求，故适应度函数应满足与标定成本负相关的关系。起初，我们假设适应度函数$f(C)$与标定成本C之间的函数关系式为：

$$f(C) = \frac{1}{C} \tag{5}$$

但是，式（5）所表示的适应度函数在标定成本收敛至 600 左右时，易陷入局部最优解。因此，我们假设适应度函数为：

$$f(C) = \begin{cases} 100 & if\ C > 800 \\ 2^{800-C} & otherwise \end{cases} \tag{6}$$

这样，适应度函数在标定成本$C < 800$时也能拥有较大的斜率，使结果落在全局最优的概率更大。

4.2.3 交叉与变异的实现

“交叉配对”的代码实现意在模拟自然界中染色体的交叉配对过程。对于种群中第 i 个个体，我们在 90 个“基因”中选取编号 20~70 的某个“基因”，将一段染色体一分为二。后一段染色体保持不变，前一段染色体与种群中第（100 - i）的个体的对应片段交换，从而实现交叉配对。

“变异”的代码实现意在模拟自然界中基因的突变过程。由于基因突变的概率较小，我们假设综合突变概率为 0.2。若数值为“1”，则突变为“0”的概率为 0.9；若数值为“0”，则突变为“1”的概率为 0.1。通过设置“1”/“0”以不同的变异概率，防止了在迭代后期因“0”数量太多而出现的大规模变异情况，以至于使算法难以收敛至最优解。

4.2.4 自然演替的实现

根据适应度函数，我们得到种群中不同个体的适应度。“自然演替”通过选择的方式，保留适应度函数值大、更优的解，淘汰适应度函数值小的解。自然选择的原则是适应性强的个体为下一代贡献一个或多个后代的概率大。选择实现了达尔文的适者生存原则。

在具体代码的实现上，通过“轮盘选择”的方式，将选中的个体（“染色体”）储存到下一代的种群中，直到下一代的种群数量到达 100 个。

4.2.5 遗传迭代的退出条件

当遗传算法的解收敛到最优解时，解与解之间的标准差会趋近于 0。因此，当连续四次最优解的标准差小于 5 时，近似认为已经收敛到最优解，可以退出迭代循环。

4.2.6 遗传算法的输出

在程序到达结束时，输出最小的标定成本、标定选择的温度点、迭代次数，并绘制出迭代次数与标定成本的曲线图，显示程序运行时间。

4.3 遗传算法的不足

尽管遗传算法有着迭代速度快、求解效率高等优点，它也存在着不足。在本课题的研究中，遗传算法的不足主要表现在以下几点：

(1) 存在陷入局部最优解的情况。使用相同参数进行迭代时，训练集的最低成本可能会陷入局部最优解（平均测定成本 300 左右）而终止迭代。

(2) 变异概率过大时，会出现得到最优解又跳出最优解的情况，使得程序难以终止迭代，运行时间长，效率降低。

因此，在调试该算法时，将总变异概率设为 0.2 作为较为科学。这样的变异概率减小了陷入局部最优解的可能，同时避免从最优解跳出的情况。

4.4 遗传算法的参数对标定的影响

在遗传算法中，参数的选择会间接影响算法的解。通过调整规定的初始概率“possibility_chosen”、初始种群的规模大小、变异概率“possibility”，我们可以得到不同参数情况下的解，比较不同参数下该算法的求解精度与效率。

4.4.1 改变初始概率“possibility_chosen”

通过改变初始概率“possibility_chosen”，我们会获得不同的初始种群。在变异概率为 0.2，初始种群大小为 100*90 的条件下，不同的初始概率得到的程序运行结果见表 1.

初始概率	最小标定成本	选定的温度点	迭代次数	程序运行时间（s）
0.1	269.37	[-17 -7 3 50 66]	5	83.64
0.2	266.92	[-18 -2 7 54 68]	5	85.47
0.3	264.59	[-18 -2 6 53 66]	10	165.90
0.4	263.37	[-18 -2 4 50 66]	14	234.33
0.5	270.58	[-18 -3 5 52 67]	16	267.08

表 1 改变初始概率参数得到的解

观察运行结果，我们可以得出结论：当初始概率较小时，程序迭代次数较少，运行速度较快；当初始概率较大时，程序迭代次数增加，运行速度减慢。

4.4.2 改变初始种群规模大小

通过改变初始种群的大小，我们会获得拥有不同规模的初始种群。在初始概率为 0.4，变异概率为 0.2 的条件下，不同的初始种群大小得到的结果见表 2.

初始种群大小	最小标定成本	选定的温度点	迭代次数	程序运行时间（s）
30*90	314.34	[-19 -4 3 46 57 65]	14	37.89
60*90	270.58	[-18 -2 7 59 61]	14	73.09
100*90	263.37	[-18 -2 4 50 66]	14	234.33
150*90	279.83	[-18 -7 2 45 66]	10	137.67
200*90	277.70	[-19 -3 1 51 65]	8	190.55

表 2 改变初始种群大小参数得到的解

观察运行结果，我们可以得出结论：初始种群规模越大，则代码的计算量越大，程序运行的时间越长。同时，规模较小的初始种群由于样本数量少，解的精度相应较低，难以获得最优解；规模较大的初始种群由于样本数量过大，程序运行效率低。

4.4.3 改变变异概率“possibility”

通过改变变异概率“possibility”，使得基因“变异”不确定性发生改变。在初始概率为 0.4, 初始种群大小为 100*90 的条件下，不同的变异概率得到的结果见表 3.

变异概率	最小标定成本	选定的温度点	迭代次数	程序运行时间（s）
0.05	271.59	[-16 -6 6 56 68]	17	163.04
0.10	310.43	[-16 -9 -3 15 48 66]	12	116.81
0.20	263.37	[-18 -2 4 50 66]	14	234.33
0.30	270.79	[-17 -2 9 57 69]	15	134.55
0.40	360.61	[-17 -15 -4 -2 15 46 65]	19	167.07

表 3 改变变异概率参数得到的解

在变异概率为 0.4 时，计算最优解的过程中出现了跳出最优解的情况，迭代曲线如图 3.

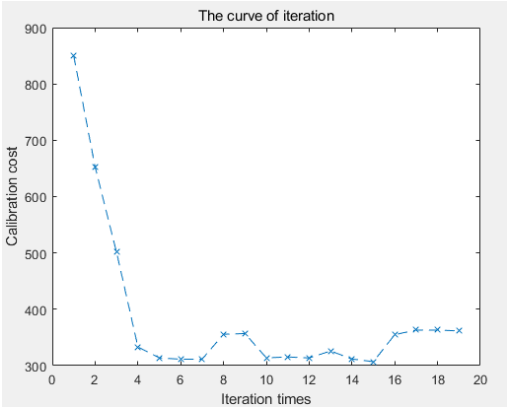

图 3 变异概率过大时跳出最优解的情况

观察运行结果，我们可以得出结论：当变异概率较小时，程序迭代次数多，运行时间长；当变异概率为 0.2 时，程序迭代次数较少，达到最优解的速度快；当变异概率继续增大时，则出现了跳出最优解的情况。因此，在本课题的遗传算法中，我们选择 0.2 作为变异概率。

5 结论

该课题要求为大规模制造中的测定装置设计一种成本合理的标定方案。借助遗传算法与三次样条插值等方法，预设定初始概率为 0.4、变异概率为 0.2 的情况下，通过 14 次基因迭代，我们得到的温度测定点为[-17 -4 6 52 67]，训练集最低的平均测定成本为 263.37。迭代过程见图 4.

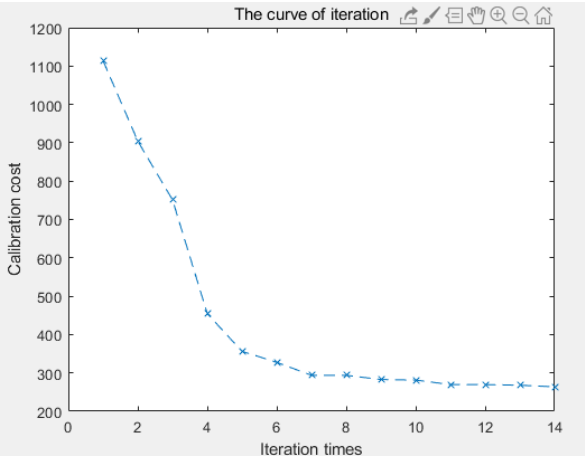

图 4 最优解的迭代曲线

6 参考文献

[1] 上海交通大学电子工程系. 工程问题建模与仿真之案例 1_V1.1 20180910[EB/OL].ftp://202.120.39.248.

[2] 任子武, 伞冶. 自适应遗传算法的改进及其在系统辨识中应用研究[J]. 系统仿真学报, 2006, 18(1): 41-43.(Ren Z W, San Y. Improved adaptive genetic algorithm andits application research in parameter identification[J]. J ofSystem Simulation, 2006, 18(1): 41-43.)

[3] 沐阿华, 周绍磊, 于晓丽. 一种快速自适应遗传算法及其仿真研究[J]. 系统仿真学报, 2004, 10(1): 122-125.(Mu A H, Zhou S L, Yu X L. Research on fast self-adaptive genetic algorithm and its simulation[J]. J of System Simulation, 2004, 10(1): 122-125.)

7 附录

7.1 程序清单

7.1.1 go.m

```
% 此文件是案例 1 的代码文件

tic
clear;

iteration_flag = 1;
cnt = 0; % 记录迭代次数
y_cost = []; % 记录每次迭代之后的测定成本

% 创建初始种群,大小为 100*90
group = get_initial_group();

% 读入训练数据
training_data = dlmread("dataform_train.csv"); % 大小为 1000*90，有 500 组 T/V 关系，首行是温度，次行是电压

cost = get_cost(group,training_data); % 对数据进行三次样条插值，分别对 100 个样本计算标定成本，返回一个 1*100 的向量，记录每个个体的标定成本

% 明确，我们要根据电压作为自变量拟合出温度与电压的关系，电压是自变量！
while iteration_flag
    cnt = cnt + 1; % 记录迭代次数

    % 计算适应度
    adaption = get_adaption(cost);

    % 淘汰
    group = eliminate(group,adaption);

    % 交叉互换，注意会引起数据点少于两个的情况
    group = exchange(group);

    % 变异，注意会引起数据点少于两个的情况
    group = mutate(group);

    % 检查配对、变异是否引起数据点少于两个的情况
```

```
    group = check_ones(group);

    cost = get_cost(group,training_data); % 对数据进行三次样条插值，分别对 100 个样本计算标定成本，返回一个 1*100 的向量，记录每个个体的标定成本

    y_cost = [y_cost min(cost)]; % 记录每次迭代后的最小平均测定成本

    % 判断是否退出循环
    iteration_flag = is_to_jump_out(y_cost);
end

% 输出最小的平均标定成本和标定点
point = print_result(group,cost); % 返回一个 1*90 的向量，记录着为 1 的数据点
fprintf("Iterated %d times.\n",size(y_cost,2));
x_iterate = 1:1:cnt;

% 画迭代曲线
plot(x_iterate,y_cost,'x--');
xlabel('Iteration times');
ylabel('Calibration cost');
title('The curve of iteration');

% 测试测试集数据
test(point);

toc
```

7.1.2 get_initial_group.m

```
function group = get_initial_group()
% 获取初始种群

group = zeros(100,90);
possibility_chosen = 0.40; % 选择为 1 的可能性可以任意，没有很大差别

% 对每一个元素进行遍历，储存 1/0
for k = 1:size(group,1)
    cnt = 0;
    for m = 1:size(group,2)
        if rand() <= possibility_chosen
            group(k,m) = 1;
            cnt = cnt + 1;
        end
    end
```

```
    % 避免全部是 0 或者只有一个 1 的情况
    if cnt == 0 || cnt == 1
        for m = 1:size(group,2)
            if group(k,m) == 0
                group(k,m) = 1;
                cnt = cnt + 1;
            end
            if cnt == 2
                break;
            end
        end
    end
end
end
```

7.1.3 get_cost.m

```
function average_cost = get_cost(group,training_data)
% 对数据进行拟合、三次样条插值，分别对 500 个数据计算标定成本，返回一个 1*100 的向量，记录每个个体的标定成本

Q = 50; % 设置单次测定成本
average_cost = zeros(1,size(group,1));

% 对 200 种标定方案的每一个样本都进行计算标定成本
for k = 1:size(group,1)
    % 对 500 组中，每一组 t v 数据进行选点，拟合，插值，并计算成本，最后求得改标定方法的平均成本
    cost_sum = 0; % 此种标定情况下，成本的总和
    count = 0; % 累加测定次数
    y = []; % 保存选中的测定点的温度
    for m = 1:size(group,2)
        if group(k,m) == 1
            count = count + 1;
            y = [y m-21];
        end
    end

    for n = 1:(size(training_data,1)/2)
        % 将样本选择的点的电压值（横坐标）保存在向量 x 中
        x = [];
        cost = 0;
        for m = 1:size(group,2)
```

```
            if group(k,m) == 1
                tmp = training_data(2*n,m);
                x = [x tmp];
            end
        end

        % 对选中的数据点进行三次样条插值拟合
        yy = spline(x,y,training_data(2*n,:)); % 储存拟合后的温度值
        del_temperature = abs(yy - training_data(2*n-1,:)); % del_temperature 保存的是拟合后的温度差值

        % 计算单次、对一组数据的标定成本
        cost = count * Q;
        for m = 1:size(del_temperature,2)
            if del_temperature(1,m) <= 0.5
                continue;
            elseif del_temperature(1,m) > 0.5 && del_temperature(1,m) <=1
                cost = cost + 1;
            elseif del_temperature(1,m) > 1 && del_temperature(1,m) <=1.5
                cost = cost + 5;
            elseif del_temperature(1,m) > 1.5 && del_temperature(1,m) <=2.0
                cost = cost + 10;
            else
                cost = cost + 10000;
            end
        end

        % 对每组成本进行累加
        cost_sum = cost_sum + cost;
    end

    % 计算对 500 组数据标定的平均成本，并存储到 average_cost 对应的位置
    average_cost(1,k) = cost_sum/(size(training_data,1)*0.5);
end
end
```

7.1.4 get_adaption.m

```
function adaption = get_adaption(cost)
% 根据标定成本计算适应度
% 返回一个 1*100 的向量，储存每个标定方案的适应度大小

adaption = zeros(1,size(cost,2));
```

```
% solution:如果 cost 大于 800，那么适应度大小就设置成 100；如果小于 800，适应度大小就是
2^(800-cost)
for k = 1:size(cost,2)
    if cost(1,k) > 800
        adaption(1,k) = 100;
    else
        adaption(1,k) = 2^(800-cost(1,k));
    end
end

adaption_sum = sum(adaption);
adaption = adaption ./ adaption_sum;

end
```

7.1.5 eliminate.m

```
function new_group = eliminate(group,adaption)

new_group = zeros(size(group,1),size(group,2));
round = zeros(1,size(group,1));
flag = 1;

for k = 1:size(group,1)
    for m = 1:k
        round(1,k) = round(1,k) + adaption(1,m);
    end
end

while flag < size(group,1) + 1
    tmp = rand();
    for k = 1:size(round,2)
        if tmp <= round(1,k)
            new_group(flag,:) = group(k,:);
            flag = flag + 1;
            break;
        end
    end
end

end
```

7.1.6 exchange.m

```
function group = exchange(group)
% 交叉互换的代码实现

group_size = size(group,1); % 取出 group 的行数

for k = 1:1/2 * group_size
    % 选择交换的点，设置成在 20~70 之间的数值
    position = ceil(rand()*70) - ceil(rand()*20);

    % 基因交叉配对（片段交换）
    for j = 1:position
        medium = group(k,j); % 用作复制的中间环节，a 与 b 交换的临时介质
        group(k,j) = group(group_size - k + 1,j);
        group(group_size - k + 1,j) = medium;
    end
end

end
```

7.1.7 mutate.m

```
function group = mutate(group)
% 变异的实现

% 这样设置的变异概率，对于 1 变 0 是 0.08,0 变 1 是 0.02
possibility = 0.2;
possibility_0 = 0.1; % 0 变 1 的概率
possibility_1 = 0.9; % 1 变 0 的概率

for k = 1:size(group,1)
    for m = 1:size(group,2)
        if rand() < possibility
            if group(k,m)
                if rand() < possibility_1
                    group(k,m) = 0;
                end
            else
                if rand() < possibility_0
                    group(k,m) = 1;
                end
            end
```

```
        end
    end
end

end
```

7.1.8 check_ones.m

```
function group = check_ones(group)
% 检查是否不满足三次样条插值的情况

for k = 1:size(group,1)
    cnt = 0;
    for m = 1:size(group,2)
        if group(k,m) == 1
            cnt = cnt + 1;
        end
    end
    % 避免全部是 0 或者只有一个 1 的情况
    if cnt == 0 || cnt == 1
        for m = 1:size(group,2)
            if group(k,m) == 0
                group(k,m) = 1;
                cnt = cnt + 1;
            end
            if cnt == 2
                break;
            end
        end
    end
end

end
```

7.1.9 is_to_jump_out.m

```
function flag = is_to_jump_out(y_cost)
% 判断是否要退出迭代（是否收敛）
% 若 y_cost 最后四个数据的标准差小于 5，就认为收敛，可以退出

cost_size = size(y_cost,2);
if cost_size > 4
    a = [y_cost(1,cost_size-3) y_cost(1,cost_size-2) y_cost(1,cost_size-1) y_cost(1,cost_size)];
```

```
    if std(a) < 5
        flag = 0;
    else
        flag = 1;
    end
else
    flag = 1;
end

end
```

7.1.10 print_result.m

```
function min_result = print_result(group,cost)
% 输出执行结果，返回 1*100 向量记录要选择的点

min_cost = min(cost);
[m,n] = find(cost == min_cost);
min_result2 = group(n,:); % 如果有多个相同的值，那么 find 返回也是一个矩阵！每行内容会相同
min_result = min_result2(1,:);
temperature_point = [];

for k = 1:size(min_result,2)
    if min_result(1,k) == 1
        temperature_point = [temperature_point k-21];
    end
end

fprintf("The minimun cost is: %f4.\n",min_cost);
disp("The chosen temperature point is :");disp(temperature_point);

end
```

7.1.11 test.m

```
function test(point)
% 使用选定的标定方案，计算测试集中的平均标定成本

test_data = dlmread("dataform_testA.csv");
cost_test = get_cost(point,test_data);
fprintf("\nThe test data's average cost is %f4.",cost_test);

end
```