一个测量装置在大规模制造中的标定问题

组号 7　姓名 王山木　学号 519021910418

摘要：本文以某种温度传感器模块为研究对象，旨在寻找到一种标定工序的方案，使得在确保精度的前提下尽可能地提高标定的效率，并且降低标定成本。由于取点方案过多，采用穷举搜索的方法会消耗大量的时间。遗传算法是模仿自然界遗传机制的启发式搜索方法，文中通过随机选取测量点，采用三次样条插值法拟合分析数据，并采用遗传算法找到最优选点方案，最终设计出了测定成本较低，误差范围可以被接受的标定算法。

关键词：Matlab，标定工序，三次样条插值，遗传算法

The Calibration Problem of a Measuring Device in Large-Scale Manufacturing

ABSTRACT: This article takes a certain temperature sensor module as the research object, and aims to find a calibration procedure solution, so that the calibration efficiency can be improved as much as possible under the premise of ensuring the accuracy, and the calibration cost can be reduced. Because there are too many options for taking points, the exhaustive search method will consume a lot of time. The genetic algorithm is a heuristic search method that imitates the genetic mechanism of nature. In this paper, the measurement points are randomly selected, the cubic spline interpolation method is used to fit and analyze the data, and the genetic algorithm is used to find the most optimal point plan.

Key words: Matlab, Calibration procedure, Cubic spline interpolation, Genetic algorithm

1 引言

很多大规模制造的电子产品中包含有带测量功能的模块（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。任何测量装置在制造时一般都要经过标定。

本文假定一种测量装置。为便于理解，设定被测物理量为环境温度，可测范围是-20℃至70℃。所用核心传感器的输入-输出特性有明显非线性，且个体差异性比较明显。课题要求为该装置设计一种标定工序的方案，适合大规模高效率批量制造[1]。

测量装置的原理框图见图1.1。其中，传感器部件（包含传感器元件及必要的放大/调理电路）的输入信号，即被测温度，以符号T表示；它的输出信号是电压形式，用符号V表示。该电压信号经模数转换器（ADC）转为数字编码，被微处理器读取和程序处理，获得温度读数$\hat{T}$。所以，微处理器通过检测传感器电压信号，间接计算被测温度。这一操作须基于$V-\hat{T}$函数关系[1]。

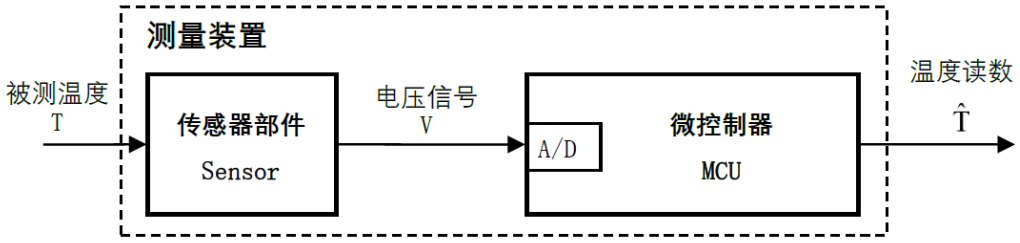


图1.1 装置原理框图[1]

2 标定成本计算方案[1]

据本课程规定，标定方案的成本包括标定误差成本和单点标定成本。对数据集中的任一样本 i，其在 $T_{i,j}$，$j \in \{1,2,\cdots,90\}$ 所表示的 90 个温度比较点上的单点和样本个体标定误差成本的计算规则如式（2-1）和式（2-2）所示：

$$s_{i,j}=\begin{cases}0, & \left|\hat{T}_{i,j}-T_{i,j}\right| \le 0.5\\ 1, & 0.5 < \left|\hat{T}_{i,j}-T_{i,j}\right| \le 1.0\\ 6, & 1.0 < \left|\hat{T}_{i,j}-T_{i,j}\right| \le 1.5\\ 20, & 1.5 < \left|\hat{T}_{i,j}-T_{i,j}\right| \le 2.0\\ 10000, & \left|\hat{T}_{i,j}-T_{i,j}\right| > 2.0\end{cases} \quad i \in \{1,2,\cdots,M\},\ \ j \in \{1,2,\cdots,90\} \tag{2-1}$$

$$S_i=\sum_{j=1}^{90} s_{i,j} \qquad i \in \{1,2,\cdots,M\},\ \ j \in \{1,2,\cdots,90\} \tag{2-2}$$

其中$T_{i,j}$为样本 i 在温度比较点 j 的实际温度，$\hat{T}_{i,j}$为标定后样本 i 在温度比较点 j 的温度估计值，$s_{i,j}$为样本 i 在温度比较点 j 的单点标定误差成本，S_i为样本 i 的个体标定误差成本，M为数据集的样本总数。

对所有样本，其单点标定成本 Q 均相同，在本问题中设定 $Q=50$，由此得到样本 i 的个体标定成本如式（2-3）所示：

$$cost_i = S_i + Q \cdot n_i \qquad i \in \{1,2,\cdots,M\} \tag{2-3}$$

对某一标定方案 P，其在该数据集上的方案总体成本 C 如式（2-4）所示：

$$C=\frac{1}{M}\sum_{i=1}^{M} cost_i \tag{2-4}$$

3 遗传算法（Genetic algorithm）

3.1 遗传算法概述

遗传算法（Genetic Algorithm，GA）最早是由美国的 John Holland 于 20 世纪 70 年代提出，该算法是根据大自然中生物体进化规律而设计提出的。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。该算法通过数学的方式，利用计算机仿真运算，将问题的求解过程转换成类似生物进化中的染色体基因的交叉、变异等过程。在求解较为复杂的组合优化问题时，相对一些常规的优化算法，通常能够较快地获得较好的优化结果[2]。

遗传算法的流程如图 3.1 所示。

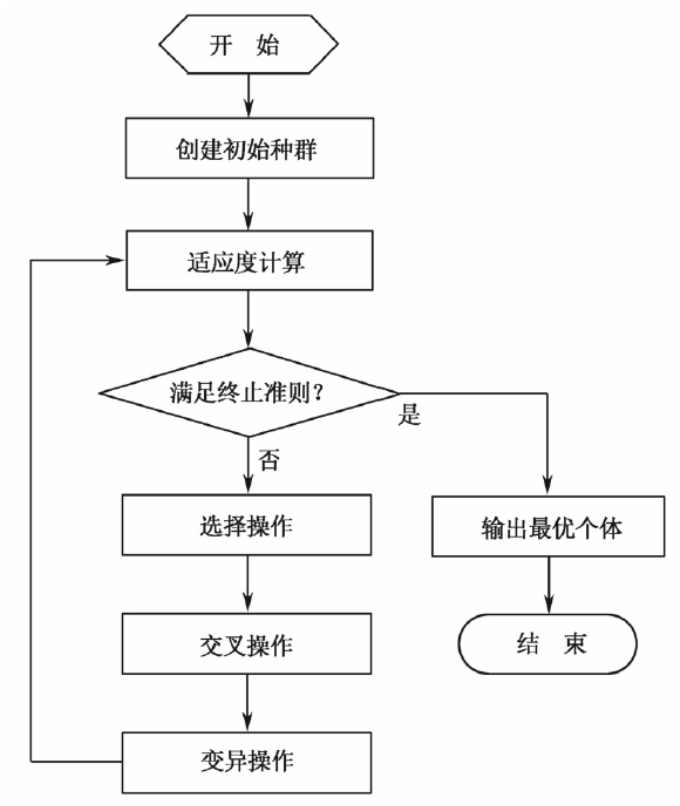


图 3.1 遗传算法的算法流程[3]

遗传学与遗传算法术语的对应关系如下见表 3-1：

表3-1 遗传学与遗传算法术语对应关系[3]

遗传学术语	遗传算法术语
群体	可行解集
个体	可行解
染色体	可行解的编码
基因	可行解编码的分量
基因形式	遗传编码
适应度	评价函数值
自然选择	选择操作
基因交叉	交叉操作
基因变异	变异操作

3.2 遗传算法参数设置

3.2.1 个体染色体编码方式

本文将每个个体的染色体编码为一个规模为90的0-1序列$X_i = (x_1, x_2, \cdots, x_{90})$，其中：

$$x_j = \begin{cases} 1, & \text{温度比较点 } j \text{ 处已设置标定点} \\ 0, & \text{温度比较点 } j \text{ 处未设置标定点} \end{cases}$$

在本案例中，由于后续三次样条插值的需求，每一个序列取值为 1 的点的个数至少要为两个，同时，根据实验的数据显示，取值为 1 的点过多会导致成本增长过快，理想的点的个数应该在 10 个点以内。

3.2.2 种群参数设置及种群的初始化

种群的初始化需要我们设定种群的规模 NP 和最大进化代数 N，并随机生成一个初代种群 F_0。在本案例中，我们将种群规模设置为 200，最大进化代数设置为 200 代。本案例生成初代种群的方法就是建立一个[0, 1]区间上 $NP \times 90$ 的随机数矩阵，将其中大于 0.9 的位置赋值为 1，小于 0.9 的位置赋值为 0，这么做的意义使初始种群中每个个体的点的个数较少，防止初始种群的个体的成本过高。

3.2.3 个体基因突变

本案例中，我们按照一定的突变概率随机选择一个点位，将该点位的取值取反，例如由 0 变为 1 或由 1 变为 0，设做为该处基因发生了基因突变。本案例中取变异概率为 0.001。

3.2.4 个体基因的交叉互换

遍历种群的基因库，即遍历每一个个体的基因，若满足了一定的交叉概率，则将该点位的取值与其下一个个体对应点位的值进行交换，设做两个个体的基因在此处发生了基因的交叉互换。本案例中取交叉概率为 0.30。

3.2.4 个体的适应度计算

个体的适应度是指一个个体（标定方案）符合我们的评价标准的程度。适应度越大的个体也就越符合我们的优化目标，更容易“存活”下去并“留下个体”。在本案例中，我们希望得到一个标定成本尽可能小的选择方案，因此我们倾向于保留标定成本更小的个体而筛选掉标定成本更高的个体，亦即标定成本更高的个体会具有更低的适应度。因此在本案例中，我们将成本的倒数的平方设为个体的适应度即个体 P 的适应度为：

$$f(X_i) = f(x_1, x_2, \cdots, x_{90}) = 1/C^2 \tag{3-1}$$

3.2.5 自然选择

自然选择就是让让适应度低的个体更容易被淘汰，但并非指适应度低的个体一定被淘汰，因此，我们以轮盘赌的方式从当前种群中选择适应度较高的个体，其具体过程为：

设第 k 代种群为$f_k = (X_1^{(k)}, X_2^{(k)}, \cdots, X_{NP}^{(k)})$，其中 $X_i^{(k)} = \left(x_{i1}^{(k)}, x_{i2}^{(k)}, \cdots, x_{i90}^{(k)}\right)$ 表示第 k 代种群中的第 i 个个体。根据已选定的适应度函数 f，可以得到个体 $X_i^{(k)}$ 的适应度为 $f\left(X_i^{(k)}\right)$

将每个个体的适应度进行归一化处理后得到：

$$\hat{f}\left(X_i^{(k)}\right) = \frac{f(X_i^{(k)})}{\sum_{j=1}^{M} f(X_j^{(k)})} \tag{3-2}$$

将前 i 个个体归一化后的适应度进行累加得到每个个体对应的一个 0~1 的数值$p_i^{(k)}$，即：

$$p_i^{(k)} = \sum_{j=1}^{i} \hat{f}\left(X_j^{(k)}\right) \tag{3-3}$$

接下来，我们生成一个 NP*1 的矩阵$(P_1, P_2, \cdots, P_{NP})$，数值为 0~1 随机取值并从小到大排好序，用作概率。在每轮选择中，均进行 NP 次个体选择，对当前种群$f_k = \left(X_1^{(k)}, X_2^{(k)}, \cdots, X_{NP}^{(k)}\right)$，若$P_i < p_j^{(k)}$，则个体$X_j^{(k)}$ 即为本次选择中被选中保留至下一代的个体, $i = i + 1$；否则，$j = j + 1$, 直至选出 NP 个个体构成新的种群。

伪代码如下：

Algorithm 1: Russian roulette

Input: the k-th generation population：$(X_1^{(k)}, X_2^{(k)}, \cdots, X_{NP}^{(k)})$
Output:the (k+1)-th generation population：$(X_1^{(k+1)}, X_2^{(k+1)}, \cdots, X_{NP}^{(k+1)})$

1. Calculate the fitness of each individual $f_i^{(k)}$;

2. Normalize the value: $\hat{f}_i^{(k)} = \frac{f_i^{(k)}}{\sum_{j=1}^{M} f_j^{(k)}}$;

3. Cumsum the value: $p_i^{(k)} = \sum_{j=1}^{i} \hat{f}\left(X_j^{(k)}\right)$

4. Generate NP numbers arranged in ascending order with value ranging from 0 to 1: $(P_1, P_2, \cdots, P_{NP})$

5. Set $i = 1, j = 1$

6. **While** (i < NP) **do**

7. **If** ($P_i < p_j^{(k)}$) **then**

8. Reserve individual $X_j^{(k)}$: $X_i^{(k+1)} = X_j^{(k)}$

9. $i = i + 1$

10. **Else**

11. $j = j + 1$

3.3 遗传算法的优化

3.3.1 优秀个体的特殊处理

在本案例中，为了防止上一代最优的个体因意外被淘汰或者往更坏的方向变异，我们人为地保留了上一代地最有个体，提高收敛的速度。

3.3.2 局部最优问题的解决

本案例的前期研究发现，当最大迭代次数为 200 代时，往往在一百多代时最低成本便已经收敛至一个局部最优解，此时的种群f_k中的个体的基因往往全部是一样的，即可行解趋同，种群过早地失去了活力、陷入了局部最优解。

为了解决过早地陷入局部最优解的问题，本案例中将 100 代后的变异概率由原来的 0.001 提升至 0.005，并将 150 代后的变异概率进一步提升至 0.01，从而提高种群的中个体的差异性。经过初步验证，这种处理在多数情况下能使种群进一步往更好的状态进化

3.3.3 成本函数计算优化

由于 Matlab 软件对于循环体的执行效率很低，在案例的研究的初级阶段，采用了双重 for 循环计算标定成本，从而不可避免地导致时间开销非常巨大，100 代的迭代需要画上一个多小时。

改进后的成本函数更多地采用了矩阵计算的思想，并把每个点的成本计算的 4 个判断语句改为了用布尔语句表示的函数，大大降低了运行时间，改进后的算法进行 100 轮的迭代仅需三百多秒。

4 曲线拟合插值方法

4.1 三次样条插值概述[4]

三次样条插值 (Spline Interpolation，简称 Spline)，是一种分段三次多项式插值方法，该方法要求拟合插值得到的曲线通过所有给定的样本点，并且在这些样本点处曲线的一阶、二阶导数连续，满足两个边界条件。

假设数据集中有 $n+1$ 个样本点$(x_0, y_0),(x_1, y_1),\cdots,(x_n, y_n)$，满足：

$$a = x_0 < x_1 < \cdots < x_n = b$$

样条曲线定义为：

$$y = q_i(x), \qquad i = 1,2,\cdots,n$$

其中 $q_i(x)$ 可统一写作式（4-1）的形式，并满足式（4-2）和式（4-3）：

$$q_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{4-1}$$

$$\begin{cases} q_i(x_{i-1}) = y_{i-1} \\ q_i(x_i) = y_i \end{cases} \tag{4-2}$$

$$\begin{cases} q_i'(x_i) = q_{i+1}'(x_i) \\ q_i''(x_i) = q_{i+1}''(x_i) \end{cases} \tag{4-3}$$

式（4-1）中共包含 $4n$ 个未知数，式（4-2）提供了 $2n$ 个方程，式（4-3）提供了 $2n-2$ 个方程，还需要 2 个方程才能够求出所有的未知数，这就要求我们对边界条件做出规定。本文中应用的三次样条插值均采取非节点(Not-A-Knot)边界条件：

$$\begin{cases} q_1'''(x_1) = q_2'''(x_1) \\ q_{n-1}'''(x_{n-1}) = q_n'''(x_{n-1}) \end{cases} \tag{4-5}$$

至此，方程与未知数的个数已经相等，可以求出每一段上三次多项式的系数，代入$[a,b]$区间内待插值的点即可得到目标函数的估计值。

4.1 三次样条插值的 Matlab 实现[5]

对 n 个样本点$(x_0, y_0),(x_1, y_1),\cdots,(x_n, y_n)$，可通过调用 Matlab 提供的函数 csape 得到含有多项式系数信息的矩阵。再输入一组数据$x = (x_1, x_2, \ldots x_m)$，通过调用 ppval 函数即可得到对应$x$的插值数据$y = (y_1, y_2, \ldots y_m)$。

5 仿真试验结果

在案例前期的研究中，笔者试验了如表 5-1 所示的参数组合并得到了相应的结果。在表格中变异概率中若存在多个数值，依次为“0-100 代/101 代-150 代/150 代后”的取值。

表5-1 不同参数对应的结果

仿真序号	变异概率	交叉概率	适应度函数	种群规模	迭代次数	最低成本
1	0.001	0.25	$1/\text{cost}^2$	200	10	361.46
2	0.001	0.25	$1/\text{cost}^2$	200	100	339.59
3	0.001	0.25	$1/\text{cost}^2$	200	100	351.92
4	0.001	0.25	$1/\text{cost}^2$	200	285	342.83
5	0.005	0.30	$1/\text{cost}^2$	200	150	335.55
6	0.005	0.30	$1/\text{cost}$	200	50	379.50
7	0.005	0.50	$1/\text{cost}^2$	200	200	337.14

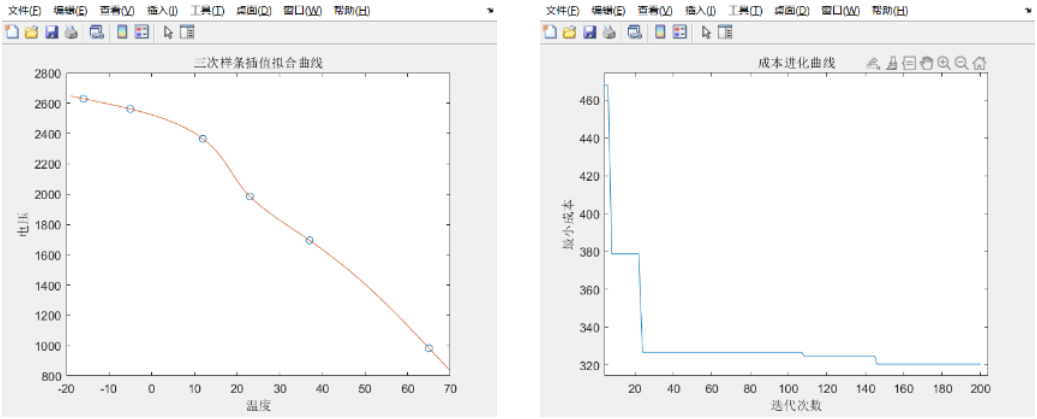
续表 5-1

仿真序号	变异概率	交叉概率	适应度函数	种群规模	迭代次数	最低成本
8	0.001/0.01/0.01	0.30	$1/cost^2$	200	100	335.09
9	0.001/0.01/0.01	0.30	$1/cost^2$	250	200	327.63
10	0.001/0.005/0.01	0.30	$1/cost^2$	300	200	322.53
11	0.001/0.005/0.01	0.30	$1/cost^2$	250	200	320.67
12	0.001/0.005/0.01	0.30	$1/cost^2$	250	200	329.55
13	0.001/0.005/0.01	0.30	$1/cost^2$	250	200	327.75

其中，第 11 次仿真给出了模型在训练集上的最优的一次解，其给出的标定方案如下：

{-16，-5，12，23，37，65}

运行结果如图 5.1 所示。



（a）拟合曲线　　（b）成本进化曲线

图 5.1 训练集上第 11 次试验的运行结果

根据前期在训练集上的运行结果，本案例中将模型的各项参数设置如表格 5-2 所示：

表 5-2 模型的各项参数设置

参数类型	数值设置
变异概率	0.001/0.005/0.01
交叉概率	0.30
种群规模	250
迭代次数	200
适应度函数	$1/cost^2$

按照表格 5-2 所示的参数设置，模型在测试集上运行的几次结果见表 5-3：

表 5-3 模型在测试集上的仿真结果

仿真序号	标定方案	最小成本
1	{-15，-9，10，21，42，69}	329.50
2	{-17，-3，14，28，34，68}	322.23
3	{-17，-8，7，14，22，42，68}	364.44
4	{-16，0，13，25，35，66 }	338.32
5	{-19，-8，2，20，43，68}	322.30
6	{-17，-8，11，28，32，67}	320.41
7	{-16，-4，12，29，31，64}	327.43
8	{-15，-10，13，28，38，64}	336.52

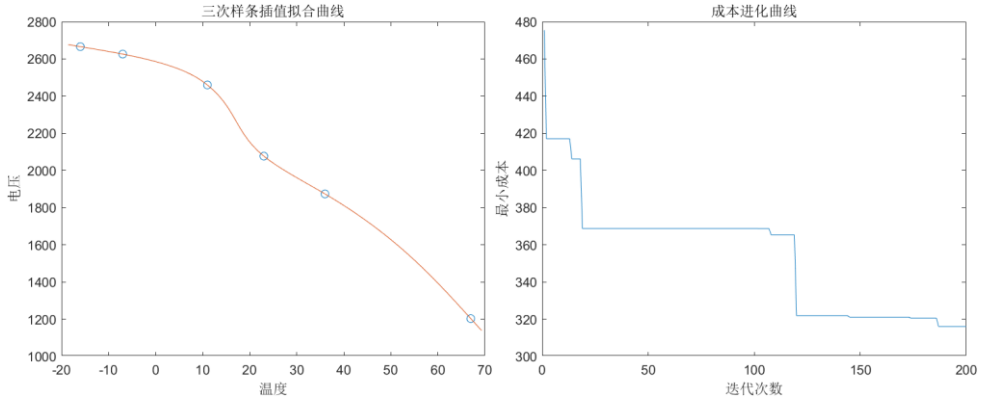
续表 5-3

仿真序号	标定方案	最小成本
9	{-19，-7，10，23，36，66}	326.12
10	{-16，-7，11，23，36，67}	316.08

其中，第 10 次仿真给出了模型在测试集上的最优的一次解，其给出的标定方案如下：

{-16，-7，11，23，36，67}

其对应的最小成本为 316.08，运行结果如图 5.2 所示。



（a）拟合曲线　　（b）成本进化曲线

图 5.2 测试集上第 10 次试验的运行结果

在训练集和测试集上进行了多次仿真实验后，不难发现以下结论：

1. 优化结果是逼近最优解的一个较好的可行解：本测定方案的优化结果并不唯一，且基本上能确定的是，最终的优化结果只是接近最优解，尚未能精准地得到最优解。
2. 优化结果在接概率情况下会偏离最优解较远：绝大多数的仿真实验给出了 6 个点的标定方案，但在测试集上进行的 10 次试验中，仍然有一次（第 3 次）给出了 7 个点的优化方案，最小成本来到了 340 以上，说明本优化方案是需要多进行几次试验来得到一个相对最优解的。
3. 优化结果与初始条件相关性大：最终的优化结果与初始种群的基因关联性很大，在初始种群整体较为优秀的情况下，在 40-50 代左右便能得到一个很接近最终优化结果的解；但在初始种群相对表现较差时，优化至 150 代之后还会出现较大幅度的进化。

6 拓展：不同拟合插值方法的应用

在接下来的扩展内容中，我们额外尝试了分段三次 Hermite 插值（pchip）和修正 Akima 分段三次 Hermite 插值（makima）两种插值方法与三次样条插值法（spline）进行对比。

6.1 pchip 插值的 Matlab 实现及试验结果[6]

对 n 个样本点$(x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n)$，可通过调用 Matlab 提供的函数 pchip 得到含有多项式系数信息的矩阵。再输入一组数据$x = (x_1, x_2, \ldots x_m)$，通过调用 ppval 函数即可得到对应$x$的插值数据$y = (y_1, y_2, \ldots y_m)$。

使用 pchip 插值进行 5 次试验后得到结果如表格 6-1 所示：

表 6-1 采用 pchip 模型在测试集上的仿真结果

仿真序号	标定方案	最小成本
1	{-19，-7，10，26，34，68}	321.39
2	{-17，-6，7，23，42，61}	310.78
3	{-17，-7，8，22，45，68}	312.17
4	{-17，5，25，43，68}	319.16
5	{-17，-4，6，26，37，69}	313.52

6.2 makima 插值的 Matlab 实现及试验结果[7]

对 n 个样本点$(x_0, y_0),(x_1, y_1),\cdots,(x_n, y_n)$，可通过调用 Matlab 提供的函数 makima 得到含有多项式系数信息的矩阵。再输入一组数据$x = (x_1, x_2, \ldots x_m)$，通过调用 ppval 函数即可得到对应$x$的插值数据$y = (y_1, y_2, \ldots y_m)$。

使用 makima 插值进行 5 次试验后得到结果如表格 6-2 所示：

表 6-2 采用 makima 插值的模型在测试集上的仿真结果

仿真序号	标定方案	最小成本
1	{-17，-11，8，28，37，63}	310.39
2	{-20，-10，9，26，42，69}	304.61
3	{-19，-8，9，23，43，63}	313.14
4	{-20，-6，10，26，43，68}	310.23
5	{-19，-8，10，23，39，61}	315.81

6.3 扩展部分小结

通过上述将不同插值方法应用在本案例研究的试验结果，我们可以发现，相较于 spline 插值，在本案例中，pchip 及 makima 插值方法具有更好的优化性能，可以使得标定成本进一步下降，最优的一次方案由 makima 插值方法得到，选择的标定点为：

{-20，-10，9，26，42，69}

其对应的成本仅为 304.61。

另一方面，pchip 和 makima 插值表现出了更高的稳定性。在用 pchip 和 makima 插值两种方法各进行的五次试验中，结果给出的方案的标定成本都很接近，相比于 spline 插值进行的试验具有更高的稳定性。同时，已有的结果显示出 makima 插值方法在本案例中具有最高的稳定性。

此外，值得一提的是采用 pchip 和 makima 插值方法时，在相同的参数设置下，时间开销更小。采用 spline 插值时，种群的一轮迭代大约需要 3 至 4 秒，而采用 pchip 和 makima 插值方法时，一轮迭代只需 1.5 秒左右，平均运行的时间减少了 6 至 7 分钟。

综合看来，在针对本案例的研究中，采用 makima 的插值方法能取得更优的结果。

7 结论

本文以某种温度传感器模块为研究对象，研究了大规模制造中测量装置的标定问题。文中通过随机选取测量点，采用三次样条插值法拟合分析数据，并采用遗传算法找到最优选点方案。案例中，我们得到采用三次样条插值法得到的最优的标定方案为：设置 6 个标定点，温度分别为： -16℃，-7℃，11℃，23℃，36℃，67℃。

此外，本文还研究了 pchip 和 makima 两种插值方法对标定方案的影响，结果表明：相较于三次样条插值，pchip 和 makima 的插值方法能得到更低的成本标定方案，同时也具有

更高的稳定性和更小的时间开销。

在未来的工作中，我希望进一步优化遗传算法中各个参数的设置，从而获得更加稳定、时间开销更小的优化方案。此外，我也将学习不同的启发式算法，如模拟退火算法等，用不同的思路来解决标定问题。

8 致谢

在本次标定问题的研究中本人得到了老师和同学们的帮助，我对此表示衷心的感谢：

蒋乐天老师在授课过程中讲解了问题的研究思路、程序的设计思路以及论文的写作技巧，为本人的研究提供了很大的指导作用。同时，蒋乐天老师也在我进行工作进度汇报后向我提出了进一步优化研究的建议，这为我后续的研究和最终报告的完成提供了极大的帮助。

此外，在初稿工作汇报中，2019 级周彬彬同学对自己的探究思路的阐述，对我在一些关键参数的设置上起了帮助作用；以及 2019 级李春一同学提到的用矩阵运算代替 for 循环的优化方向，显著提高了算法的时间性能。此外，在案例一的模型、算法设计以及论文写作过程中，我也与其他同学就一些问题进行了交流和探讨，限于篇幅这里不逐一列出，本人同样要向他们表达诚挚的谢意。

9 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课程讲义[EB/OL]. ftp://202.120.39.248.

[2] 郑树泉．工业智能技术与应用[M]．上海：上海科学技术出版社,2019：250-251

[3] 包子阳,余继周,杨杉.智能优化算法及其 MATLAB 实例[M].中国工信出版社:北京,2018:34-35.

[4] 许小勇,钟太勇.三次样条插值函数的构造与 Matlab 实现[J].兵工自动化,2006(11):76-78.

[5] CSDN. Matlab 三次样条曲线的绘制（spline 和 csape 函数详解）[OL].https://blog.csdn.net/weixin_42943114/article/details/84843632.

[6] MathWorks. Piecewise Cubic Hermite Interpolating Polynomial (PCHIP)–Matlab pchip –MathWorks China [OL].https://ww2.mathworks.cn/help/matlab/ref/pchip.html.

[7] MathWorks. Modified Akima piecewise cubic Hermite interpolation–Matlab pchip–MathWorks China [OL]. https://ww2.mathworks.cn/help/matlab/ref/makima.html?lang=en.

附　录

代码清单

minimize_cost.m 主函数

```
%%代码功能：运用遗传算法(GA)解决标定问题
%%学号：519021910418
%%姓名：王山木

clear;
clc;

NP = 250;            %种群规模
p1 = 0.25;           %交叉概率（查阅资料：取 0.25~1 为宜）
p2 = 0.001;          %变异概率（查阅资料：取 0.001~0.1 为宜）

N = 200;            %最大迭代次数
train = load('dataform_train-2021.csv'); %读取训练集数据
test = load('dataform_testA-2021.csv'); %读取测试集数据

[m,L] = size(train); %分别获得训练集的行数和列数

f = ceil(rand(NP,L)-0.9); %随机获得 NP 个初始种群
%0 表示不选取点，1 表述选取；90%概率为 0，10%概率为 1

%对生成的初始种群做初步处理
for i = 1:NP
    while ((sum(f(i,:)) <= 2 )||(sum(f(i,:)) >=10))
        f(i,:) = ceil(rand(1,L)-0.9);
    end
end

%%遗传算法的 N 轮循环
for k = 1:N
    tic
    cost = zeros(1,NP); %提前生成成本矩阵
    fit = zeros(1,NP); %提前适应度矩阵
    %计算每一个个体的适应度，函数为 suitability
    disp(k);

    if (k == 100)
        p2 = 0.005;
    end
```

```
if (k == 150)
    p2 = 0.01;
end

for i = 1:NP
    cost(i) = calculate_cost(f(i,:),train,L,m);
end
fit = 1./(cost.^2);

maxfit = max(fit);                  %获得最大适应度
minfit = min(fit);                  %获得最小适应度

location = find(fit == maxfit);     %最优个体的位置
fbest = f(location(1,1),:);         %历代最优个体

%%轮盘赌决定去留
sum_fit = sum(fit);
fitvalue = fit./sum_fit;
fitvalue = cumsum(fitvalue);
%这里面的第 n 个元素是初始 fitvalue 中前 n 个元素的累加
change_p = sort(rand(NP,1));
%生成 NP*1 的矩阵，数值为 0~1 随机并从小到大排好序，用作概率

fiti = 1;
newi = 1;
while newi <= NP
    if(change_p(newi) < fitvalue(fiti))
    %说明落在该区域，留下 fiti 对应的个体
        new_f(newi,:) = f(fiti,:);
        newi = newi + 1; %newi + 1,直至筛选出 NP 个新的个体
    else  %概率不在 fiti 指示的个体内，该个体被筛掉
        fiti = fiti + 1;
    end
end

%%基因交叉操作
for i = 1:2:NP                  %隔一个是保证有一条基因能与之交叉
    p = rand;
    if p < p1                   %说明满足了交叉的概率
        q = randi([0,1],1,L);   %随机生成哪一个基因要被换
        for j = 1:L
            if q(j) == 1        %为 1 则交换该位置 i 和 i+1 对应的基因
                temp = new_f(i+1,j);
                new_f(i+1,j) = new_f(i,j);
```

```
                new_f(i,j) = temp;
            end
        end
    end

    %%基因变异操作
    for m = 1:NP
        for n = 1:L
            r = rand;
            if r < p2                          %说明满足变异概率
                new_f(m,n) = ~new_f(m,n);    %直接取反，设为变异
            end
        end
    end

    f = new_f;
    f(1,:) = fbest;
    %人为保留上一轮最佳个体，防止因意外导致该个体被淘汰/往差的方向变异
    mincost(k) = calculate_cost(fbest,train,L,m);
    disp(mincost(k));
    toc
end

disp('选择方式：')
x = [];
y = [];
for i = 1:L
    if (fbest(i)==1)
        x = [x,train(1,i)];
        y = [y,train(location(1,1)+1,i)];
    end
end
disp(x);
disp('最小成本：')
disp(calculate_cost(fbest,train,L,m))

pp = csape(y,x,'second');
% pp = pchip(y,x);  %pchip 插值
% pp = makima(y,x); %makima 插值

yi = train(location(1,1)+1,:);
xi = ppval(pp,yi); %插值后的估计函数值（温度值）
plot(x,y,'o',xi,yi);
```

```
xlabel('温度')
ylabel('电压')
title('三次样条插值拟合曲线')
figure
plot(mincost)
hold on
hold off
xlabel('迭代次数')
ylabel('最小成本')
title('成本进化曲线')
```

calculate_cost.m 计算标定成本的函数

```
function result = calculate_cost(f,train,L,m)
    ox = -20:1:69;
    x = [];
    y = [];
    for i = 1:L
        if (f(i)==1)
            x = [x,train(1,i)];    %用来标定的点的温度 , 1*sum(f(:))矩阵
            y = [y,train(2:2:m,i)];
            %用来标定的点的电压 , m/2*sum(f(:))矩阵
        end
    end

    yi = train(2:2:m,:);
    %每次插值时对应的自变量（电压值）,数据集中所有电压的值，500*90 矩阵
    for z = 1:m/2
        pp = csape(y(z,:),x,'second'); %spline 插值
        % pp = pchip(y(z,:),x);  %pchip 插值
        % pp = makima(y(z,:),x); %makima 插值
        xi(z,:) = ppval(pp,yi(z,:)); %插值后的估计函数值（温度值）
        delta(z,:) = abs(xi(z,:) - ox);
    end

    tmp = sumcost(delta);
    cost = sum(tmp,2) + 50 * sum(f(:));
    result = mean(cost);
```

sumcost.m 分段处理成本的函数

```
function m = sumcost(t)
m = 1.*(t > 0.5 & t <= 1) + 6.*(t > 1 & t <= 1.5) + 20.*(t > 1.5
& t <= 2) + 10000.*(t > 2 | t < 0);
end
```