参考他人报告或代码的申明

统计推断课程，2015 年秋季学期第 34 组，成员徐至盈学号 5140309242, 李杰锋学号 5140309376, 在报告编写过程中，以下方面参考了往届报告，现列表说明：

主要参考项目	说明
拟合方法描述方面	《统计推断在数模转换系统中的应用》，刘长风，黄奕斐，2014 年秋季学期，组号 21 参考了该组报告的拟合方法描述文字。
算法描述方面，包含流程图	《统计推断在数模转换系统中的应用》，游杰，陈佳明，2013 年秋季学期，组号 02 参考了该组报告的算法描述文字，引用了其流程图。

除了以上注明的参考内容，和报告中列出的引用文献注解，本报告其他部分都不包含任何其他个人或集体已经发表或撰写过的作品成果。

统计推断在数模转化系统中的应用

组号：34　　徐至盈 5140309242，李杰锋 5140309376

摘要：本文以传感器批量生产为背景，寻找校准的优化方案。我们运用统计方法结合 MATLAB 对样本中的数据进行分析，通过三次样条插值对数据进行拟合，采用粒子群算法找到了最优选点方案，得到了最优解，并对传统的粒子群算法进行了改进。

关键词：统计推断，曲线拟合，三次样条插值，艾尔米特插值，组合优化，粒子群优化算法

Application of Statistical Inference in DA Inverting System

Team: NO.34 Zhiying Xu 5140309242, Jiefeng Li 5140309376

ABSTRACT: In consideration of the mass production of sensors, we aim to find the optimized plan for the calibration. We analyze data in the Sample using statistical inference and MATLAB and eventually obtained the optimized plan with three fitting and PSO algorithm. We also make some improvements on the traditional PSO algorithm.

Key words: statistical inference, curve fitting, three fitting, Hermite interpolation, combinatorial optimization, PSO algorithm

1 引言

随着科技的发展，传感器得到广泛的应用，要在确保质量的前提下大批量生产传感器，就需要对样品进行单点测定。我们研究的问题寻找校准工序的优化方案。通过预先抽样研究多个样品的曲线特性分析，发现样品的曲线有如下特点：

（1）有确定的对应关系

（2）非线性或局部非线性且单调递增

（3）特性曲线形态相似但个体差异较大

所以需要通过几个特殊的测量点来有效拟合出产品的特性曲线并由此来对出厂产品进行标定。本实验中，我们通过 400 组有效实验数据，讨论如何在限定误差的范围内完成标定，需要讨论测量点的个数、测量点的选取、受控曲线的表达式确定方法。由于数据较为庞大，我们不再可能使用暴力穷举等方法来得出结论，考虑到拟合效果和运行时间我们最终三次样条插值两种方式来拟合出最佳匹配曲线。同时，为了能够有效而准确地筛选出测量点，我们选择使用遗传算法结合粒子群算法并对质进行了适当的改进，以得到最优秀结果[1]。

2 拟合方法选取研究

根据所给的 400 组有效的数据及其曲线形状，我们发现大多数数据都有着近似的曲线形状。考虑到要求对每一组数据都是用相同的一个最优解，故我们选择使用一个能够满足所有数据组的方法来得到曲线。建立如下评分规则[1]：

（1）公式单点定标误差成本按照式(2-1)评估

（2）某一样本的定标成本按照式(2-2)评估

（3）按照式(2-3)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。

依据这样的评分规则研究每一组数据，我们将得到的曲线点与其原数据值比较得到分数，取所有组得分的平均值作为本组解的得分。

$$S_{i,j}=\begin{cases}0 & if\ \left|\hat{y}_{i,j}-y_{i,j}\right|\le 0.4\\ 0.1 & if\ 0.4\le\left|\hat{y}_{i,j}-y_{i,j}\right|\le 0.6\\ 0.7 & if\ 0.6\le\left|\hat{y}_{i,j}-y_{i,j}\right|\le 0.8\\ 0.9 & if\ 0.8\le\left|\hat{y}_{i,j}-y_{i,j}\right|\le 1\\ 1.5 & if\ 1\le\left|\hat{y}_{i,j}-y_{i,j}\right|\le 2\\ 6 & if\ 2\le\left|\hat{y}_{i,j}-y_{i,j}\right|\le 3\\ 12 & if\ 3\le\left|\hat{y}_{i,j}-y_{i,j}\right|\le 5\\ 25 & if\ \left|\hat{y}_{i,j}-y_{i,j}\right|\ge 5\end{cases} \tag{2-1}$$

其中 $y_{i,j}$ 表示第 i 个样本第 j 个点的实测值， $\hat{y}_{i,j}$ 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。

$$S_i=\sum_{j=1}^{51}S_{i,j}+12n_i \tag{2-2}$$

$$C=\frac{1}{M}\sum_{i=1}^{M}S_i \tag{2-3}$$

2.1 多项式拟合

2.1.1 概述

广泛观察了大量实验数据之后，我们发现曲线可大致分为首，中，尾三段，都不是完全线性的，有一定的弯曲度并且中段的斜率小于首段和尾段的斜率，且中段的起点位置和长度都带有随机性。因此我们认为可以通过多项式的性征来拟合出数据曲线。

2.2 三次样条插值拟合

2.2.1 概述

样条插值是使用一种名为样条的特殊分段多项式进行插值的形式。由于样条插值可以使用低阶多项式样条实现较小的插值误差，这样就避免了使用高阶多项式所出现的龙格现象。对于 $(n+1)$ 个给定点的数据集 $\{x_i\}$，我们可以用 n 段三次多项式在数据点之间构建一个三次样条。用公式(2-4)表示对函数 f 进行插值的样条函数，需要：

（1） 插值特性， $S(x_i)=f(x_i)$

（2） 样条相互连接， $S_{i-1}(x_i)=S_i(x_i),i=1,...n-1$

（3） 两次连续可导， $S'_{i-1}(x_i)=S'_i(x_i)$ 以及 $S''_{i-1}(x_i)=S''_i(x_i),i=1,...n-1$

$$S(x)=\begin{cases}S_0,\ x\in[x_0,x_1]\\ S_1,\ x\in[x_1,x_2]\\ \quad ...\\ S_{n-1}(x),\ x\in[x_{n-1},x_n]\end{cases} \tag{2-4}$$

由于对于每个三次多项式函数需要有四个参数如式(2-5)的形式，所以对于组成 S 的 n 个三次多项式而言，需要 $4n$ 个条件确定这些多项式。对于中间的点，我们可以两侧的邻点确定三次曲线。对于边界点，我们取有邻点的一侧确定二次曲线。

$$y = ax^3 + bx^2 + cx + d \tag{2-5}$$

2.2.2 MATLAB 自带三次样条插值

MATLAB软件自带三次样条拟合函数spline()，其输入值为两组已知的数据以及所求点，返回值为所求点在已知数据的三次样条拟合下的数值。

2.3 分段三次Hermite插值拟合

2.3.1 概述

Hermite插值是在给定的节点处，不但要求插值多项式的函数值与被插函数的函数值相同。同时还要求在节点处，插值多项式的一阶直至指定阶的导数值，也与被插函数的相应阶导数值相等。Hermite插值在不同的节点，提出的差值条件个数可以不同，若在某节点 x_i，要求插值函数多项式的函数值，一阶导数值，直至 $m_i - 1$阶导数值均与被插函数的函数值相同及相应的导数值相等。我们称 x_i 为 m_i 重插值点节。因此，Hermite插值应给出两组数，一组为插值节点 $\{x_i\}_{i=0}^n$，另一组为相应的重数标号 $\{m_i\}_{i=0}^n$。

$$\sum_{i=0}^{n} m_i = N + 1 \tag{2-6}$$

若满足式子(2-6)就说明了给出的插值条件有 $N+1$ 个，为了保证多项式的存在唯一性，这时的Hermite插值多项式应在 P_n 上求得，于是给出下面定义：

f 在 $[a,b]$ 上充分光滑函数，对给定的插值节点 $\{x_i\}_{i=0}^n$ 及相应的重数标号 $\{m_i\}_{i=0}^n$，式(2-6)成立时，若 $H(x) \in P_n$ 有满足

$$H^l(x_i) = f(x_i), l = 0,1,...,m_{i-1}; i = 0,1,...,n. \tag{2-7}$$

$H(x)$ 则称为 $f(x)$ 关于节点 $\{x_i\}_{i=0}^n$ 及重数标号 $\{m_i\}_{i=0}^n$ 的Hermite插值多项式。

2.3.2 MATLAB 自带分段三次Hermite插值

MATLAB中的插值函数为interp1，其调用格式为： yi = interp1(x,y,xi,'method')，其中x，y为插值点，yi为在被插值点xi处的插值结果；x,y为向量，'method'表示采用的插值方法。当'method'为'pchip'时，实现分段三次Hermite插值。

3 基于粒子群优化算法寻找最优解

3.1 粒子群优化算法

3.1.1 概述

粒子群优化算法(Particle Swarm Opitimization)，缩写为 PSO，是一种基于群体的智能优化算法，它通过模拟鸟群的行为来解决最优化问题[2]。PSO 和遗传算法相似，从随机解出发，通过迭代寻找最优解，该算法具有以下特点：

（1）实现容易：没有遗传算法的“交叉”和“编译”过程，不断迭代以寻求最优解

（2）精度高：收敛点附近进行较多的搜索

（3）收敛快：粒子下一步状态只与粒子的最佳位置和全局的最佳位置有关，粒子能很

快收敛到最优解附近。

（4）容易陷入早熟收敛：收敛范围过早地聚集在最优解附近，容易陷入早熟收敛

3.1.2 算法流程图

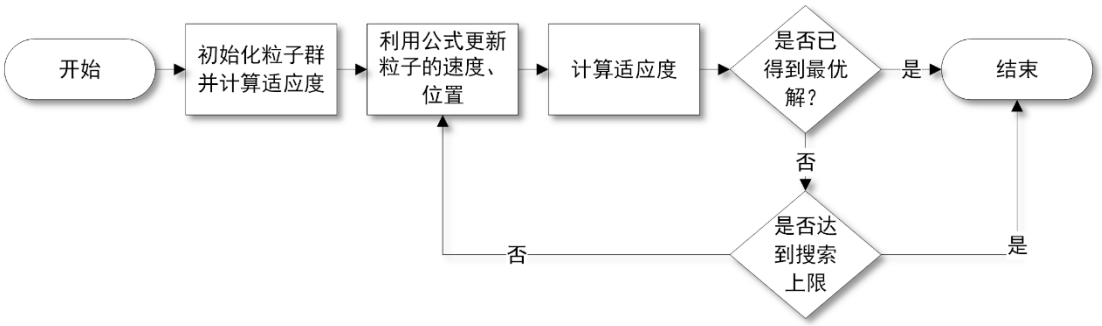

图 3-1 粒子群优化算法流程图

3.1.3 主要方程及参数设置

由 n 个粒子组成的群体对 Q 维（就是每个粒子的维数）空间进行搜。每个粒子表示为：$present_i = (x_{i_1}, x_{i_2}, x_{i_3}, ..., x_{i_Q})$，每个粒子对应的速度可以表示为 $v_i = (v_{i_1}, v_{i_2}, v_{i_3}, ..., v_{i_Q})$。初始化种群，初始化速度大小为 8，方向随机，即+8 或-8，每个粒子在搜索时要考虑两个因素：

（1）自己搜索到的历史最优值 $Pbest_i$， $Pbest_i = (p_{i_1}, p_{i_2}, p_{i_3}, ..., p_{i_Q})$

（2）全部粒子搜索到的最优值 $Gbest$， $Gbest_i = (g_1, g_2, g_3, ..., g_Q)$， $Gbest$ 仅有一个。

然后每次循环迭代，粒子都会更新，粒子的位置速度更新公式如公式(3-1)：

$$\begin{aligned} v_i' &= \omega v_i + c_1\eta_1(Pbest_i - x_i) + c_2\eta_2(Gbest - x_i) \\ present_i' &= present_i + v_i' \end{aligned} \qquad (3\text{-}1)$$

其中的参数说明及取值见表 3-1

表 3-1 粒子群优化算法参数说明及取值

参数	参数说明	参数取值
n	粒子总数	40
Q	粒子的维数	4,5,6,7,8,9
η_1 η_2	随机数	[0,1]
ω	惯性权重	0.729
c_1	跟踪自身最优值的权重系数	2
c_2	跟踪群体最优值的权重系数	1
$present_i$	第 i 个点当前的位置	$(x_{i_1}, x_{i_2}, x_{i_3}, ..., x_{i_Q})$
v_i	第 i 个点当前的速度	$(v_{i_1}, v_{i_2}, v_{i_3}, ..., v_{i_Q})$

这样经过多次迭代之后，粒子群会整体趋向于最优解。

3.2 求解过程及注意点

首先，粒子每一维的速度都设置了一个最大值 8，当其中一维的速度大于 8 或小于-8 的时候，会将其设置为 8 或-8。这样可以防止在某些情况下，粒子跳动速度很快，收敛的最优解的速度会变慢。

其次，如果粒子的所在位置超过了边界，即 1 和 51，则将超出的部分设定为边界值，

以防出错。

最后，粒子的领域计算采用的环形拓扑结构，即按照下标，首尾相连地形成环，下标相邻的为邻居， 这个方法相对于全部版本的粒子群算法，是会使粒子的行为更加随机，有更大的可能性去搜索全部的解空间，尽可能地找到全局最优解而不是局部解。

3.3 基于求解结果对拟合方式的选择

3.3.1 不同拟合方式的求解结果

对于之前提到的不同拟合方式，我们分别用粒子群算法进行求解，得到如下结果：

表 3-2 不同拟合方式求解结果的比较

拟合方式	误差成本			
	4个点	5个点	6个点	7个点
三次多项式	129.26	113.90	123.78	134.68
四次多项式		116.58	107.00	117.47
五次多项式			108.23	108.47
六次多项式				108.50
三次样条插值	129.26	105.78	94.92	95.11
分段三次Hermite插值	98.97	84.75	86.61	92.08

3.2.2 对于不同拟合方式的评价与选择

在多项式拟合曲线中，能够较为精确反映出曲线的性状的应为四次、五次多项式。二阶阶次过小，不够精确，而六阶容易出现龙格现象，拟合效果没有明显提升。但是同时多项式拟合也反映出了其精度不高的问题，即便是效果最好的四次多项式，随机取点的样本中拟合出来并进行评分后所得到的最低总体成本也超过100分。

自带三次样条插值和分段三次Hermite插值的运行结果数据可以看出，使用MATLAB三次样条拟合函数拟合效果优于多项式拟合。而分段三次Hermite插值时的拟合精度拟合精度有了较大提升。而分段三次Hermite插值略优于三次样条插值。在分段三次Hermite插值的不同取点个数当中，取5个点所得的误差成本最小。

综上，我们将在之后的讨论中使用选取 5 个点的分段三次 Hermite 插值进行数据拟合。

4 对粒子群优化算法进行改进和优化

4.1 结合遗传算法引入选择算子

4.1.1 遗传算法简介

遗传算法的基本步骤如图 4-1。其中，选择运算把当前群体中适应度较高的个体按比例遗传到下一代群体中。一般要求适应度较高的个体将有更多的机会遗传到下一代群体中[3]。

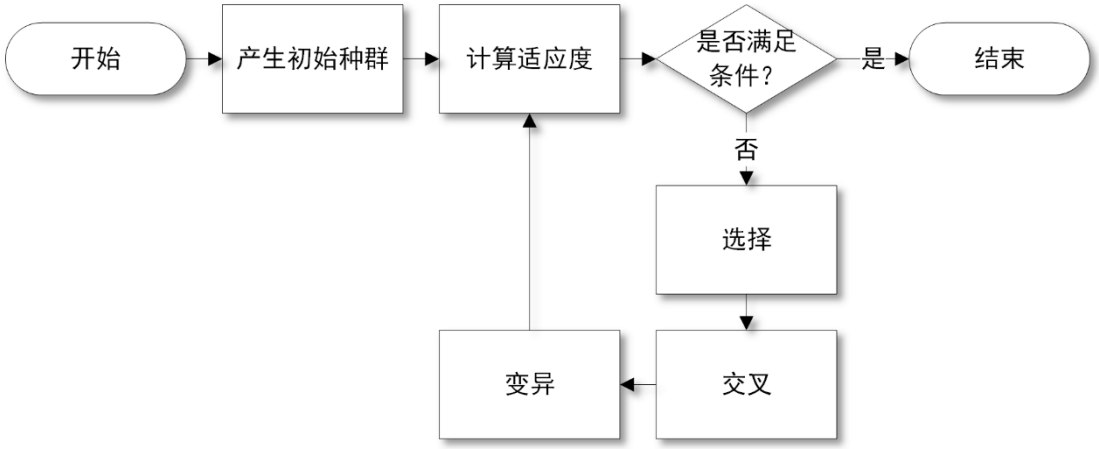

图 4-1 遗传算法流程图

4.1.2 引入遗传算子

在对领域最优解选取的时候，引入选择算子，即根据评分的大小，按概率选择一个解作为领域内的最优解。这样可以增加物种多样性，进一步减少了陷入局部最优解的可能。但同时，会耗费多一点时间进行迭代。

4.2 递增粒子领域

4.3.1 粒子领域的递增

粒子的领域是随着迭代次数的增大而增大的，一开始领域大小只有 1，即左右各一个邻居，随后每迭代一次，领域范围就增大 1。最后，随着迭代次数的越来越多，领域的范围会扩大至整个种群。

4.3.1 运行结果及分析

表 4-1 递增粒子领域的测试结果

粒子领域	出现最优解时迭代次数	收敛时迭代次数
不变	35~45	70~80
随迭代次数递增	20~30	55~65

这样做的好处是一开始粒子的行为会更加随机，避免全部进入到局部最优解的附近。而随着迭代次数增加，粒子的行为趋于一致，于是会逐渐整体向最优解移动，增加了收敛的速度，提高效率。

4.3 递减惯性权重

4.3.1 惯性权重的更改

粒子的惯性权重是随着迭代次数增大而减少的，一开始最大的惯性权重最大值是 0.9，并设置最小值为 0.4。经过多次运行程序后发现，平均经过 30 次迭代即可找到测量成本小于 85 的解。于是设置惯性权重 ω 根据公式 4-1 进行线性递减，其中 k 为迭代的次数。

$$\omega_k = 0.9 + \frac{0.4-0.9}{30} \times k \tag{4-1}$$

4.3.2 运行结果及分析

表 4-2 不同惯性权重的测试结果

ω	出现最优解时迭代次数	收敛时迭代次数
0.729	40~50	130~150
从 0.9 到 0.4 线性递减	20~30	55~65

运行修改后的程序发现，刚开始时由于惯性权重大，粒子的随机性较好，能大范围地对整个解空间进行搜索。随着迭代次数逐渐增加，惯性权重逐渐减小，粒子的跳跃性减少，收敛的迭代次数比原来更少，行为也更加稳定。

由此可以发现，设置递减的惯性权重，不仅能在一开始减少陷入局部最优解的可能性，而且在迭代的后期能加快粒子收敛的速度，精确地定位到最优解。

4.4 边界粒子回弹

4.4.1 对于处于搜索空间边界处粒子位置更新方法的更改

在根据公式 3-1 更新粒子位置时，会出现粒子的位置值超过边界上下限的情况。在原算法中，我们采用的是将超过上下限的部分设置成边界值，让粒子粘连在边界上，以确保程序运行不出错。但这只是妥协的结果，因为这样的处理方法不符合粒子群的运动模式，从概率上说，运动边界处的点与其它点的概率不一样，甚至对粒子的速度造成了浪费，即对自身历史最优解与邻域最优解的利用不充分。

在改进后的算法中，我们模拟了真实粒子的运动方式。当粒子的更新后的位置超过边界值时，我们将其设置成回弹，即粒子朝反方向运动，速度也相应地设置成反方向。如此一来，粒子的速度得到了充分的运用。其中，粒子的位置更新方法如式 4-2：

$$\begin{cases} present_i = 2 \times 51 - present_i & (present_i > 51) \\ present_i = 2 \times 1 - present_i & (present_i < 1) \end{cases} \tag{4-2}$$

4.4.2 运行结果及分析

表 4-3 改进更新位置公式的测试对比结果

位置更新方法	出现最优解时迭代次数	收敛时迭代次数
粘连法	30~45	70~90
回弹法	20~30	55~65

由运行结果可知，回弹法减小了迭代的次数，提高了算法的效率。

4.5 更改跟踪最优值的权重系数

4.5.1 对于跟踪最优值的权重系数的尝试

原算法的 c_1 和 c_2 在迭代过程中均保持不变，c_1=2，c_2=1，此时的迭代次数为 100 左右，所有粒子才会收敛到同一个值。由于迭代次数较多，我们尝试通过更改 c_1 和 c_2，希望得到更少的迭代次数，加快收敛速度。

通过更改 c_1 和 c_2，得到了如下结果。

表 4-4 不同权重系数的测试结果

c_1	c_2	出现最优解时迭代次数	收敛时迭代次数
2	1	20~30	95~105
1	2	70~80	>200
3	1.5	未出现	收敛失败
递减	递增	20~30	55~65

4.5.2 对于测试结果的分析

c_1=1，c_2=2 时，理论上，自身认知系数比社会认知系数小，粒子的运动更具有群体性。但是实际运行过程中，收敛效果并不理想。经过仔细分析后，我们认为是因为自身认知系数太小，导致粒子群开发全局的能力太弱，整个群体从一开始就陷入一个小范围的漩涡中，很难脱离。并且领域社会认知系数太大，由于开发能力弱，邻域中的粒子成本都很高，导致接近最优解的粒子会被邻域中其它粒子影响，而离最优解越来越远。

把自身认知系数增大，社会认知系数减小，c_1=1，c_2=2 时，运行的结果是收敛失败，一直迭代到 800 代以上还未收敛，而且全体粒子的成本都居高不下。经过分析，发现是自身认知系数过大，粒子容易“我行我素”，整个粒子群的行动不趋于一致，群体性太弱，难以收敛。

4.5.3 追踪最优值的权重系数的改良

经过以上尝试可以看出要想粒子的运功更有群体性，加快收敛速度，和避免优秀粒子被邻域的不良粒子拖累牵引，这两者是矛盾的。因为要加强群体性，就要增加 c_2/c_1 的比值，但增加 c_2/c_1 就会让优秀粒子被牵引。因此我们想出了一个改良的办法，就是把当前邻域最优解 *Gbest*，改成当前邻域历史最优解。这样的话不良粒子的牵引能力就会下降，因为即使是不良粒子，其历史最优解的成本也会逐渐减小，对优秀粒子的“拖累”也会减弱。同时，把 c_1 和 c_2 从恒定不变，改成随迭代次数变化而变化。经过多次实验，我们发现，将参数设置成 c_1 从 3 开始，线性递减到 1，c_2 从 1 开始，线性递增到 1.5，收敛效果最好，如式 4-3。

$$\begin{aligned} c_1 &= 3+(1-3)\times times/40 \\ c_2 &= 1+(1.5-1)\times times/40 \end{aligned} \tag{4-3}$$

这样改动是与生物群体的群体觅食行为是吻合的。因为，正如人群的行动，在一开始毫无目的地寻找时，大家的行动都会倾向于随机，会依靠自己去寻找，此时自身认知系数大。等大家寻找过一段时间后，每个人都有一些经验，知道哪里的成本会比较低（有自己的历史最优解），此时，大家会一起商量，会更多地依靠群体的经验去寻找，此时邻域社会认知系数增大。

经过测试，发现经过此次改动，粒子群的收敛速度有很大的提升，从原来的 100 次左右迭代，到现在平均 60 次左右，收敛速度提高了 40%，而且刚开始自身认知系数大，使得粒

子开发全局的能力变强，因此加快了收敛速度的同时，也没有减少种群的多样性。

5 结论

以上得到的结果表明，使用粒子群优化算法可以较快较好地解决这种组合优化问题。但是粒子群算法容易陷入早熟收敛，所以我们用粒子群算法的局部版本与遗传算法的选择算子相混合的办法，其最大的优点在于提高了种群的多样性，减少陷入局部解的可能，最终得到了很好的结果。

结论：通过粒子群算法，我们取5个点，通过分段三次Hermite插值拟合，实现了最小总体成本(表5-1)，同时，我们对传统的粒子群优化算法进行了改进。

表5-1 问题的解

实验条件	最优解	成本
分段三次Hermite插值	[4,16,26,35,48]	84.75

6 参考文献

[1] 上海交大电子工程系. 统计推断在数模转换同种的应用课程讲义[EB/OL].ftp://202.120.39.248.

[2] 李丽,牛奔. 粒子群优化算法[M]. 冶金工业出版社,2009:25-33.

[3] 邢文训,谢金星. 现代优化计算方法[M]. 清华大学出版社,2005:113-148.

附录1：

```
%%%%%%% 答案检验程序 2015-11-04 %%%%%%%%

my_answer=[ 4,16,26,35,48 ];%把你的选点组合填写在此
my_answer_n=size(my_answer,2);

% 标准样本原始数据读入
minput=dlmread('20150915dataform.csv');
[M,N]=size(minput);
nsample=M/2; npoint=N;
x=zeros(nsample,npoint);
y0=zeros(nsample,npoint);
y1=zeros(nsample,npoint);
for i=1:nsample
    x(i,:)=minput(2*i-1,:);
    y0(i,:)=minput(2*i,:);
end
my_answer_gene=zeros(1,npoint);
my_answer_gene(my_answer)=1;


% 定标计算
index_temp=logical(my_answer_gene);
x_optimal=x(:,index_temp);
y0_optimal=y0(:,index_temp);
for j=1:nsample
    % 请把你的定标计算方法写入函数mycurvefitting
    y1(j,:)=mycurvefitting(x_optimal(j,:),y0_optimal(j,:));
end

% 成本计算
Q=12;
errabs=abs(y0-y1);

le0_4=(errabs<=0.4);
le0_6=(errabs<=0.6);
le0_8=(errabs<=0.8);
le1_0=(errabs<=1);
le2_0=(errabs<=2);
le3_0=(errabs<=3);
le5_0=(errabs<=5);
g5_0=(errabs>5);
```

```
sij=0.1*(le0_6-le0_4)+0.7*(le0_8-le0_6)+0.9*(le1_0-le0_8)+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(nsample,1)*my_answer_n;
cost=sum(si)/nsample;

% 显示结果
fprintf('\n经计算，你的答案对应的总体成本为%5.2f\n',cost);

function y1 = mycurvefitting( x_premea,y0_premea )

x=[5.0:0.1:10.0];

% 将你的定标计算方法写成指令代码，以下样式仅供参考
y1=interp1(x_premea,y0_premea,x,'pchip');


end
```

附录2：

```
function A_result = particleXgenetic  %粒子群算法x遗传算法
    data = xlsread('D:\20150915dataform.csv');
    num = 40;  %粒子数量
    Dimen=5; %维度
    par_v = zeros(num,Dimen);  %速度
    par_w = 0.9;  %惯性权重
    par_c1 = 3;par_c2 = 1;  %自身认知&领域社会认知
    %r = 1;  %约束因子
    par_selfbest = zeros(num,Dimen);  %每个粒子的历史最优解
    par_selfbestValue = zeros(num,1);  %每个粒子历史最优解的评分
    par_neighbor=zeros(num,Dimen); %每个粒子的领域最优解
    par_neighborValue=zeros(num,1); %每个粒子领域内最优解的评分
    cnt=0; %领域范围
    times=0; %迭代次数
    flag=1; %找到后置为0
    Index=0;


    %产生粒子，初始化速度，惯性权重，认知系数
    Par = [];
    for i=1:1:num
        Par = [Par;sort(randperm(51,Dimen))];
        par_v(i,:) = [2*round(rand(1,1))-1 2*round(rand(1,1))-1
2*round(rand(1,1))-1 2*round(rand(1,1))-1 2*round(rand(1,1))-1];
```

```
%初始速度随机取为正负1
    %par_v(i,:)=par_v(i,:)*2;
    par_neighborValue(i)=intmax('uint64');
    par_selfbestValue(i)=intmax('uint64');
end

while(1)
    if(cnt<=20)
        cnt = cnt + 1;
    end
    times=times+1;
    times


    %算个体适应度,更新个体历史最优解
    A_result = zeros(num,1);
    for i = 1:1:num
        A_result(i) = total_grade(Par(i,:),data);
        if(A_result(i)<=par_selfbestValue(i))
            par_selfbestValue(i)=A_result(i);
            par_selfbest(i,:)=Par(i,:);
        end
    end

    %par_selfbestValue

    for i=1:1:num
        if(A_result(i)<84)
          Index=i;
          flag=0;
          break;
        end
    end

    if(flag==0)
      break;
    end

    %更新领域最优解
    for i = 1:1:num
        neighbor_set=zeros(num,2*cnt+1,Dimen);
        neighborIndex_set=zeros(num,2*cnt+1,1);
        weight=[];
        total_weight=0;
```

```
            num_in=0; %

            %寻找领域中评分小于200的加入到领域集合中进行选取
            for k = (i-cnt):1:(i+cnt)
                kk=mod(k-1,num)+1;
                if(par_selfbestValue(kk)<=200)
                    num_in=num_in+1;
                    neighbor_set(i,num_in,:)=par_selfbest(kk,:);
                    neighborIndex_set(i,num_in,1)=kk;
                    weight=[weight;1/(A_result(kk)*A_result(kk))];
                    %按评分倒数的平方进行概率选择

total_weight=total_weight+1/(A_result(kk)*A_result(kk));
                end
            end

            %用选择算子进行选取领域最优解

            if(numel(weight)==0)
               tmp_par=zeros(1,Dimen);
               tmp_value=zeros(1,1);
               tmp_value(1)=intmax('uint64');
               %如果周围评分都是负，直接找局部最优，不作选择
               for k = (i-cnt):1:(i+cnt)
                   kk=mod(k-1,num)+1;
                   if(A_result(kk)<=tmp_value(1))
                       tmp_value(1)=A_result(kk);
                       tmp_par(1,:)=Par(kk,:);
                   end
               end
                par_neighbor(i,:)=tmp_par(1,:);
                par_neighborValue(i)=tmp_value(1);
            else
                select=total_weight*rand(1);
                selectIndex=1;

                for j=1:1:num_in
                    if(select<=weight(j))
                        selectIndex=j;
                        break;
                    end
                end
                par_neighbor(i,:)=neighbor_set(i,selectIndex,:);
```

```
par_neighborValue(i)=A_result(neighborIndex_set(i,selectIndex,1));
          end
       end


       par_w=0.8+(0.2-0.8)*times/40;
       par_c1=3+(1-3)*times/40;
       par_c2=1+(1.5-1)*times/40;
       if(par_w<0.1)
          par_w=0.1;
       end
       %更新位置信息
       for i = 1:1:num
          %更新速度
          tt=Par(i,:);
          vt=par_v(i,:);
          while(1)
              %par_v(i,:)
              %par_selfbest(i,:);
              %Par(i,:)
              Par(i,:)=tt;
              par_v(i,:)=vt;
              %par_selfbest(i,:)
              %par_neighbor(i,:)

              par_v(i,:) =
par_w*vt+par_c1*rand(1)*(par_selfbest(i,:)-
Par(i,:))+par_c2*rand(1)*(par_neighbor(i,:)-Par(i,:));
              for k=1:1:Dimen
                 if(par_v(i,k)>8)
                    par_v(i,k) = 8;
                 elseif(par_v(i,k)<-8)
                    par_v(i,k)=-8;
                 end
              end
              %更新位置
              tt=Par(i,:)+par_v(i,:);
              Par(i,:)=round(Par(i,:)+par_v(i,:));
              %判断是否超出范围
              for j = 1:1:Dimen
                 if(Par(i,j)>51)
                    Par(i,j)=2*51-Par(i,j);
                    par_v(i,j)=-par_v(i,j);
                 elseif(Par(i,j)<1)
```

```
                    Par(i,j)=2-Par(i,j);
                    par_v(i,j)=-par_v(i,j);
                end
            end
            t=Par(i,:);
            t=unique(t);
            if(length(t)==Dimen)
               break;
            end
         end
      end


      A_result_sort=sort(-A_result);
      A_result_sort=-A_result_sort;
      par_selfbestValue
      -unique(-A_result_sort)
      if(length(unique(A_result_sort))==1)
         break;
      end;
      %par_v
   end
   'the best'
   %A_result(Index)
   Par
   'end'

function total_res = total_grade(select,data)

   %data = xlsread('D:\20150915dataform.csv');
   total_res = 0;

   for i = 1:2:800
      d1 = data(i,select);
      u1 = data(i+1,select);
      total_res = total_res +
grade(data(i,1:51),data(i+1,1:51),d1,u1);
   end
   total_res = total_res/400;

function s = grade(d,u,d1,u1)
   uu = interp1(d1,u1,d,'pchip');
   %uu = spline(d1,u1,d);
   s = 0;
```

```
for i=1:1:51
    item = abs(uu(i)-u(i));
    if item <= 0.4

    elseif item <= 0.6
        s = s + 0.1;
    elseif item <= 0.8
        s = s + 0.7;
    elseif item <= 1
        s = s + 0.9;
    elseif item <= 2
        s = s + 1.5;
    elseif item <= 3
        s = s + 6;
    elseif item <= 5
        s = s + 12;
    else
        s = s + 25;
    end
end
s=s+12*5;
```