统计推断在数模转化系统中的应用

组号：60　　姓名：黄汇 学号：5140309139　　姓名：刘劲涛 学号：5140309066

摘要：对某产品的传感特性进行校准定标，并寻找最优校准工序的方案。运用三次插值拟合方法得到样本数据拟合曲线，然后根据拟合曲线方程计算每个测定点的成本，以及由误差带来的损失成本。通过 Matlab 软件编码，运用遗传算法对不同的选点方案进行优化，降低校准定标的总成本，运用最终组合方案计算最终成本，并与其它组合方案进行比较，证明原组合方案的优化性。

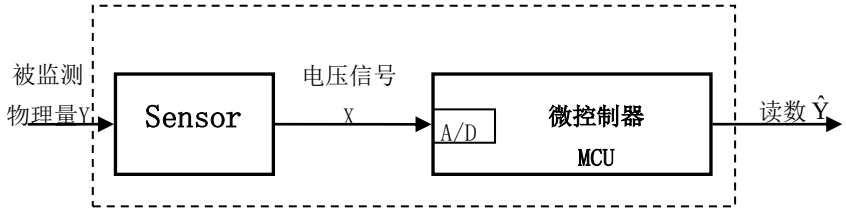
关键字：Matlab，校准定标，三次插值拟合，遗传算法

ABSTRACT: Calibration of the sensing characteristics of a product, and to find the optimal calibration procedure. The fitting curve of the sample data was obtained by using the three interpolation fitting method, and the cost of each measurement point was calculated according to the fitting curve equation. Through the MATLAB software code, using genetic algorithm to optimize different siting schemes, reduce total cost of calibration calibration, using the final combination scheme to calculate the final cost, and compared with other combination scheme that original combination scheme optimization.

Keywords: Matlab, calibration, Three interpolation fitting, Genetic Algorithm

1. 问题分析：

本课题在于对某工业器件的校准定标进行研究，并寻找最优校准工序的方案。
监测模块流程示意图：

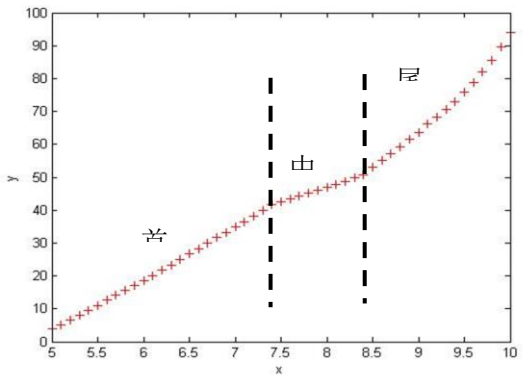


1.1 成本问题：1、测量点数量越多，测定成本越高
2、估测点数量越多，误差定标成本越高
因此我们要选出一个方案平衡二者的关系，进而得出最优方案。

1.2 降低定标成本：
原理：尽量选取较少的测试点数据，即减少测试次数，并选择合适的拟合函数，通过已有数据点计算出其他数据点数值。
方法 1：线性插值
方法 2：非线性插值
步骤：1、拟合——利用已测定点的数据，确定一条反映连续变化趋势的曲线

2、插值——利用这条曲线，确定未知点数据

2. 样本分析



由图例我们可以知道该产品物理量测试点的曲线特征有：

- Y 取值随 X 取值的增大而单调递增；
- 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；
- 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；
- 不同个体的中段起点位置、终点位置有随机性差异。

3. 拟合方法的讨论

由对样本的初步分析可知，由于其为一个单调递增的函数，所以可以采用奇数次多项式拟合，又由图像分析可知三次插值拟合已经很好地拟合了曲线，所以我们本次程序采用之。

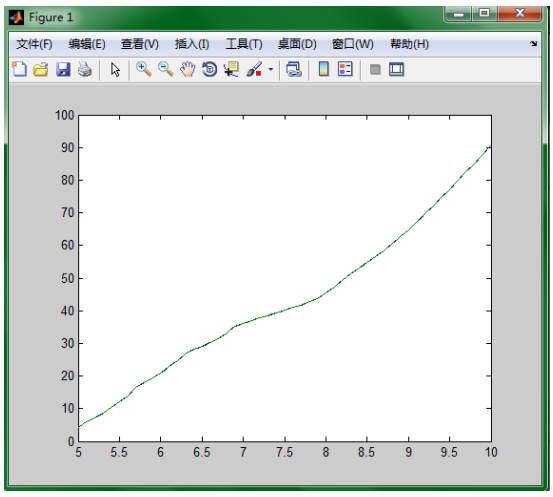


图 1：三次插值拟合

3.1 三次样条插值拟合法：

使用三次样条插值确定一个可能解，即对 n 个左右特征点组合 S={s1,s2,…,sN}进行拟合。由于曲线特征明显呈现三段分布，所以采用三次样条插值为佳。

方法介绍：

1、对中间的点，根据左右最邻近的 4 个点进行三次函数插值作为中间点之间的曲线方程；对于两边的点，则用靠近两端的三个点用二次函数对其插值。

2、选取中间的 49 个数据点根据拟合函数对每个样本点进行定标，计算出总体的成本。

3.2 三次样条插值使用情况：

在拟合选点过程中，一开始拟合原始数据点很大，即为初始随机基因点，而后第二、三次选择后，单个个体样本测试点数量基本稳定在 6、7 个点之间，说明在采用三次插值拟合时，选点在 6、7 个点之间为最佳。

3.3 spline()函数使用方法：

我们次课题最终采用三次样条插值拟合法。在算法中使用了 metlab 内置的 spline（）函数，进行三次样条插值。Spline()函数方法：

f=spline（xx,yy）　　（1）

out=ppval(f,X)　　（2）

其中，

式（1）为获得拟合函数 f(本课题中获得的 f 为以每个个体的拟合函数为元素组成的向量）。xx 为个体基因组成的矩阵，每行代表一个个体，yy 为个体矩阵对应的实际值组成的矩阵。

式（2）为获得有拟合函数计算得的理论值 out 矩阵，X 为矩阵，每行从 5.1 到 10,相邻两点间隔 0.1，每行相同，共 100 行，对应 100 个个体，100 个拟合函数。

4.特征值的选取

如果说拟合是让已选取的点发挥其最大效用,那么特征点的选取则是从一个单样本中选取最具有价值、最能体现整个函数特性的点集，因此它是优化中非常重要的一环。

4.1 特征点选取优化的必要性：

本课题中，样本容量为 51。

由上述分析可知，选点在 6、7 个点之间可以搜索到较佳的方案。如果考虑固定选 7 个点进行穷举法，共需进行 C_{51}^{7}=1,15775100 次，即亿次级别，而现在计算机普遍可以每秒运算亿次数级别（不考虑算法花费）。但是实际上计算机有时会几十次完成一次运算，所以这种计算花费大约达到了 1000M 的数量级。这样做耗费的资源大大违背了我们进行优化的初衷。对于这种备选方案数量巨大的问题，我们称其为 NP-hard 问题(即算法复杂度不能用问题的阶数 n 来表示)。

因此，如何选取特征点成了课题的核心问题之一。现今解决这类问题的主要方式还是以启发式搜索算法为主，较有名的有遗传算法，模拟退火算法，以及神经网络算法，粒子群算法等等。

本次程序主要采用遗传算法对每个选点基因进行组合交叉、自然选择、变异不断更新种群个体，直至稳定在一个较佳的水平上。

5.遗传算法

5.1 遗传算法简介：

遗传算法（Genetic Algorithm）是一类借鉴生物界的进化规律演化而来的随机化搜索

方法。其主要特点是直接对对象进行操作，不存在求导和函数连续性的限定；具有内在的隐性并行性和更好的全局寻优能力。

算法采用概率化的寻优方法，能自动获取和指导优化的搜索空间，自适应地调整搜索方案，无需确定的规则。

对于一个求函数最大值的优化问题(最小值类同)，可以描述为下列数学模型：

x 为决策变量，式 2-1 为目标函数式，式 2-2、2-3 为约束条件，U 是基本空间，R 是 U 的子集。

满足约束条件的解 X 称为可行解，集合 R 表示所有满足约束条件的解 X 所组成的集合，称为可行解集合。

$$\begin{cases} \max f(X) & 2-1 \\ x \in R & 2-2 \\ R \subset U & 2-3 \end{cases}$$

5.2 遗传算法实现流程图：

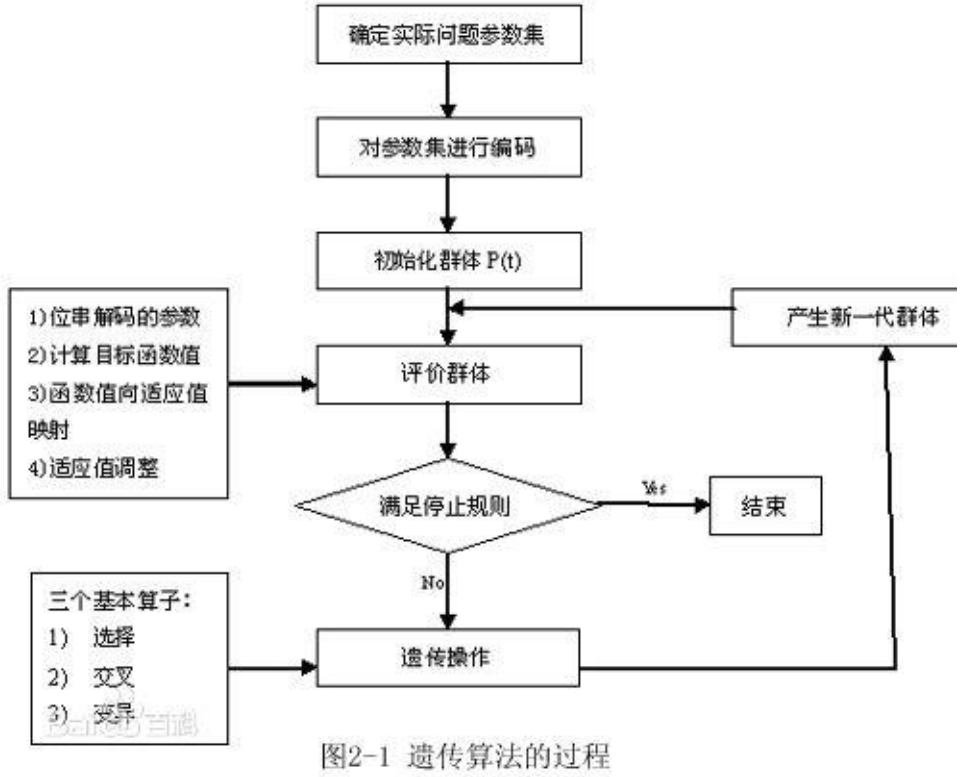


图2-1 遗传算法的过程

5.2 遗传算法在本课题中的应用简介：

在本次课题中，目标函数式 f(X)即为通过运用单个选点方案，对 400 组数据样本进行总的平均成本的计算（其中包括误差成本），f(X)即对应相应的计算公式与过程。 而每个选点方案即为遗传算法中的一个个体，本次课题中我们采用 100 组选点方案来组成一个个体种群，最后通过对这个种群进行：

1、筛选，我们采用的筛选规则为：保留存活的个体，死掉的个体用最优的个体代替，以保持种群数目的稳定以及种群的优化性。

2、杂交，我们采用的规则为：随机抽取两个个体，并随机选取个体中长度相同的一段基因序列，接着进行交换。

3、变异，我们设变异概率为一个小于 0.1 的数，第一个个体不参与突变，其余个体的第 1、51 个基因也不参与突变。这是因为 1 和 51 测试点有随机性误差，而且由取 1 和 51

基因的运行结果显示，输出最优个体时大多数是不包括 1 和 51 数据点的，因此可以去掉。最后剩下的个体基因参与随机性突变。

4、最后把进化后的种群保存在原数组中，继续上述循环，执行 N 代遗传。输出最终最优结果方案。

（**注**：遗传算法在适应度函数选择不当的情况下有可能收敛于局部最优，而不能达到全局最优。）

6.解决问题过程及具体程序设计思想

6.1 主程序

a)初始化：

设置最大进化代数 generation_size=200，交叉概率 cross_rate=0.9，突变概率 mutation_rate=0.09,以及一个 400×51 的 0 矩阵 Y(),用于存储提供的 800 组数据中的 400 组 51 个测试点的 y 数值。

用一个 geneinit(100)函数用于随机生成每个个体的基因组成（由于随机性误差，第 1 和 51 个基因不选），其中用 round(rand(100,51)-0.2)；对每一行每一列生成 0、1 代码，1 表示选择数据点，0 则反之；减去 0.2 表示取 1 概率设置为稍大于取 0 概率。其后进入 200 代的遗传循环中。

b)遗传循环体：

find(gene(1,:)==1) 函数用于观察输出的每代最优个体的基因组成（选点编号），而后进行 fitness() 个体适应度的生成（每个选点方案的平均成本）。

随后依次进行 select() 选择、generate() 交叉、mutate() 变异。

c)最后用 evaluate() 函数体进行格式化输出最终方案。

6.2 个体适应度的确定：

这个过程由 fitness()函数来完成适应度初始化。首先生成一个 100×1 的矩阵 cost()用于存储 100 个不同选点方案的平均成本。其中 errorcost() 函数用于完成误差成本的计算，包裹在 fitness() 函数里使用。然后用 min(out)输出最小平均成本，即最优个体，供运行时观察。

6.3 选择运算及生存概率的确定:

在得到每个个体的平均成本后，我们小组讨论决定，用平均成本最大的个体与每个个体的平均成本作差（平均成本越小的个体，生存概率越大），然后对每个差结果求和得到基数，再用每个个体与最大个体的差值除以基数，得到一个相加得 1.0 的种群不同个体的生存概率。其中我们可以知道最大个体的生存概率为 0。

其后用 sortrows() （matlab 自带函数）对每个个体按生存概率从小到大排列以放方便选择过程的进行。即形成一个比较连续的生存选择区间[0,1]。

选择时，我们用 t=rand()函数生成一个 0-1 的小数，作为参数传入 search()函数中，如果个体的生存概率越接近 t(即最可能被选择的个体),则选择之。这个过程我们用二分法实现搜索过程。

其中有特殊情况即 t=0 或 1 的时候，0 表示个体死亡，我们规定用接近 0 概率的个体取代之，1 表示绝对存活，我们用最大生存概率的个体取代之，保证个体种群数目的稳定性及

平衡性。

6.4 交叉运算：

由于第一行为最佳个体的保留空间，所以不参与交叉，然后随机选择两个个体，用 i 和 j 实现；其后再随机取两个个体的部分基因序列进行交换。然后用 comepare()函数对子母代进行比较，保留最佳个体。（**注：**前几次为了较快观察结果，我们暂且不进行子母代比较保留，因为发现占用时间过长，且效果相差不大）

array 矩阵用于存储已交换过的个体，用于避免两个个体重复交叉。由于有 100 个个体，所以用 count 来计数种群交叉是否完成，若 count==25 则结束。

6.5 变异运算：

设置一个变异概率 mutation_tate(我们取了 0.01、0.09、0.1，最终选取 0.09 作为最终代码)，除去用于存储当前最优个体的第一行和第 1 和 51 个基因不参与突变。随后进行 99×49 次的循环，如果随机数 t=rand()小于变异概率则进行 0、1 取反。

群体 gene(t)经过选择、交叉、变异运算之后得到下一代群体 gene’(t)。

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

6.6 误差成本计算 errorcost()函数体：

为评估和比较不同的校准方案，选用课程提供的成本计算规则。

- 单点定标误差成本:

$$s_{i,j}=\begin{cases}0 & \text{if } |\hat{y}_{i,j}-y_{i,j}|\le 0.4\\ 0.1 & \text{if } 0.4<|\hat{y}_{i,j}-y_{i,j}|\le 0.6\\ 0.7 & \text{if } 0.6<|\hat{y}_{i,j}-y_{i,j}|\le 0.8\\ 0.9 & \text{if } 0.8<|\hat{y}_{i,j}-y_{i,j}|\le 1\\ 1.5 & \text{if } 1<|\hat{y}_{i,j}-y_{i,j}|\le 2\\ 6 & \text{if } 2<|\hat{y}_{i,j}-y_{i,j}|\le 3\\ 12 & \text{if } 3<|\hat{y}_{i,j}-y_{i,j}|\le 5\\ 25 & \text{if } |\hat{y}_{i,j}-y_{i,j}|>5\end{cases}$$

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

```
t=abs(diffrernce);
t0=sum(sum(t<=0.4));
t1=sum(sum(t<=0.6))-t0;
t2=sum(sum(t<=0.8))-t0-t1;
t3=sum(sum(t<=1))-t0-t1-t2;
t4=sum(sum(t<=2))-t0-t1-t2-t3;
```

```
t5=sum(sum(t<=3))-t0-t1-t2-t3-t4;
t6=sum(sum(t<=5))-t0-t1-t2-t3-t4-t5;
t7=sum(sum(t>5));
out=0.1*t1+0.7*t2+0.9*t3+1.5*t4+6*t5+12*t6+25*t7;
```

out 即为 sum($s_{i,j}$)误差成本总和

- 计算出校准方案总成本

（**注**：本程序自定义的计算平均成本函数与老师提供的样例函数完全符合）

7.程序运行结果展示

（5）次循环：变异概率为：0.09　　(**注**：包括有 1 和 51 数据点

point: [3 10 20 28 35 44 51]　　cost: 95.960250
point: [2 9 20 26 34 44 50]　　cost: 95.283000
point: [2 10 21 30 40 49]　　cost: 95.486000
point: [3 9 19 26 33 43 50]　　cost: 95.264500
point: [2 9 20 27 35 44 51]　　cost: 95.819500
point: [3 11 22 31 43 50]　　cost: 94.924500

（5）次循环：变异概率为：0.01　　(**注**：无 comepare()函数

point: [2 9 20 26 34 43 50]　　cost: 95.237500
point: [2 9 19 26 32 43 50]　　cost: 95.286000
point: [2 9 19 26 33 43 50]　　cost: 95.114750
point: [3 12 22 31 43 50]　　cost: 94.898250
point: [2 9 20 26 33 43 49]　　cost: 95.621250

（5）次循环：变异概率=0.09

point: [3 10 20 27 35 43 50]　　cost: 95.449750
point: [3 11 21 31 42 50]　　cost: 95.444250
point: [3 9 20 26 33 43 49]　　cost: 95.760250
point: [2 9 19 26 33 43 50]　　cost: 95.114750
point: [2 9 20 27 34 44 50]　　cost: 95.214750

（5）次循环:变异概率=0.1

point: [2 9 20 26 34 43 50]　　cost: 95.237500
point: [2 9 19 26 32 43 50]　　cost: 95.286000
point: [2 9 19 26 33 43 50]　　cost: 95.114750
point: [3 12 22 31 43 50]　　cost: 94.898250
point: [2 9 20 26 33 43 49]　　cost: 95.621250
point: [3 10 20 27 34 44 50]　　cost: 95.243750

（50）次循环：变异概率=0.09　　（**注**：最终程序代码运行结果

point: [2 9 19 27 34 44 50]　　cost: 95.289750
point: [2 10 21 30 40 49]　　cost: 95.486000
point: [3 10 20 27 34 44 50]　　cost: 95.243750
point: [2 9 20 26 34 44 50]　　cost: 95.283000
point: [2 10 20 26 33 43 49]　　cost: 95.714750
point: [3 12 22 31 43 50]　　cost: 94.898250
point: [2 9 19 26 33 43 50]　　cost: 95.114750

point: [2 9 20 26 33 43 50]   cost: 95.133750
point: [2 8 20 26 34 43 50]   cost: 95.553500
point: [2 9 19 26 33 43 50]   cost: 95.114750
point: [2 9 19 27 34 44 50]   cost: 95.289750
point: [2 10 20 27 34 44 50]   cost: 95.248750
point: [3 11 22 31 41 50]   cost: 95.263750
point: [3 10 20 27 34 45 50]   cost: 95.891750
point: [3 12 22 31 41 50]   cost: 95.224250
point: [3 10 20 27 34 44 50]   cost: 95.243750
point: [2 9 19 27 34 44 50]   cost: 95.289750
point: [3 10 20 27 34 44 50]   cost: 95.243750
point: [2 9 20 26 33 44 50]   cost: 95.408250
point: [3 11 22 31 41 50]   cost: 95.263750
point: [2 9 19 26 33 44 50]   cost: 95.393000
point: [2 9 20 26 33 43 50]   cost: 95.133750
point: [2 10 21 29 40 49]   cost: 95.988750
point: [2 9 20 26 34 44 50]   cost: 95.283000
point: [2 9 20 27 34 43 50]   cost: 95.265250
point: [2 9 19 27 34 44 50]   cost: 95.289750
point: [3 10 20 28 34 44 50]   cost: 95.592250
point: [2 10 20 27 35 44 50]   cost: 95.346000
point: [3 11 22 31 43 50]   cost: 94.924500
point: [2 9 20 27 34 44 50]   cost: 95.214750
point: [3 10 20 27 34 44 50]   cost: 95.243750
point: [2 9 20 26 33 43 50]   cost: 95.133750
point: [2 9 20 27 34 43 50]   cost: 95.265250
point: [2 9 20 26 32 44 49]   cost: 96.080000
point: [3 10 20 27 34 44 50]   cost: 95.243750
point: [2 9 20 26 33 43 50]   cost: 95.133750
point: [2 9 19 25 34 44 49]   cost: 96.453500
point: [2 9 20 27 34 44 50]   cost: 95.214750
point: [3 10 20 26 33 43 50]   cost: 95.248750
point: [3 10 20 26 33 43 50]   cost: 95.248750
point: [2 9 19 26 33 43 50]   cost: 95.114750
point: [2 9 19 26 33 43 50]   cost: 95.114750
point: [3 12 22 32 43 50]   cost: 95.178250
point: [3 12 22 32 43 50]   cost: 95.178250
point: [2 9 19 26 33 43 50]   cost: 95.114750
point: [2 9 19 27 35 44 50]   cost: 95.442750
point: [2 9 20 27 34 44 50]   cost: 95.214750
point: [2 9 19 26 33 43 50]   cost: 95.114750
point: [3 10 20 26 33 43 50]   cost: 95.248750
point: [3 12 22 32 43 50]   cost: 95.178250

8. 结果分析

从上述结果我们可以知道，由小组讨论改进后的遗传算法可得最佳选点方案为：
[3 12 22 31 43 50] 对应的平均成本为：94.898250

可以发现每次运行后得出的最优个体方案变化不大，说明我们程序的稳定性，不过由于在某些方面，我们的程序仍没有做到很好的随机性，比如在处理死亡个体与绝对存活个体上，我们人为化地规定为就近原则取个体，这就导致第一和最后一个个体数量占优，种群个体丰富度不够，所以很难做到全局最优。因此我们很有可能只得到了局部最优解。

再者加上拟合函数的单一性，即只选择了 spline()函数，无法优化我们的遗传算法。

9. 小组心得体会

在前期工作中，我们不断地发现了我们算法上的不足，并且不断优化我们的算法，并在优化的过程中逐步深入理解遗传算法的实现过程。通过对 matlab 的运用，较好地完成了对遗传算法随机性的完善，以及对选点范围的讨论。并且在对该课题的研究中，我们能基本掌握了遗传算法的寻优思想，以及对这种自我更新完善的创新性算法思想有所启发。

由于前期工作进程缓慢，导致在与老师交流时不能很好的找出重点问题所在，只能在算法的实现过程上进行优化，没能实现算法或拟合方法的拓展，因此无法深入寻找到更好的求解方案。这是我们的不足之处。

总的来说，我们基本达到了要求，在本次课题中所采用的拟合方法下，通过运用遗传算法寻找到了个体样本定标方案的最优解。

附录：

程序代码：

（1）主程序

```
data = csvread('20150915dataform.csv');

pop_size = 100;    %种群的最大个体数
cross_rate=0.9;   %交叉概率
mutation_rate=0.09;   %突变概率
generation_size=200;   %遗传代数终止条件
Y=zeros(400,51);
Y(1:400,:)=data(2:2:800,:);      %读入 x 对应的 y 值，并存储在数组里
for j=1:50  %设置进行试验的次数
    h=[j,-1];
gene=geneinit(pop_size);  %初始化种群个体的基因组成
for G=1:generation_size     %遗传代数
    display(h);
    display(G);
    display(find(gene(1,:)==1));  %输出每一代个体中最小成本个体的基因序列(即选点方案)
    cost=fitness(gene,Y,pop_size);    %计算出每个个体的平均成本，用于适应度初始化
    gene=select(gene,cost,pop_size);  %对种群个体进行选择
    gene=generate(gene,pop_size,cross_rate,Y);   %进行交配产生下一代个体
    gene=mutate(gene,pop_size,mutation_rate);  %对下一代个体进行变异处理
end
xx=find(gene(1,:)==1);
evaluate(xx,Y) %格式化输出最终最优个体
end
```

（2）种群初始化

```
function out = geneinit(pop_size)     %随机产生初始种群
out=round(rand(pop_size,51)-0.2);
out(:,1)=0;
out(:,51)=0;
end
```

（3）计算每个个体总平均成本

```
function out =fitness (gene,Y,pop_size) %计算每个个体平均成本
out=zeros(pop_size,1);   % 用于存储每个个体的平均成本
X=5:0.1:10;
for i=1:pop_size
    c=sum(gene(i,:)==1); %测试点数量
```

```
    pos=find(gene(i,:)==1); %测试点位置
    xx=5+(pos-1)*0.1; %测试点 X 值
    yy=Y(:,pos); %测试点 Y 值
    f=spline(xx,yy);
    difference=ppval(f,X)-Y; %理论值与实际值的差
    out(i)=12*c+errorcost(difference)/400; %单个个体平均成本
end
min(out) %输出每一代个体中的最小成本个体的成本
mean(out);
end
```

（4）选择算法

```
function out = select(gene,cost,pop_size)%选择
out=zeros(pop_size,51);
cost0=max(cost)-cost;  %每个个体成本与最大成本个体之差
s0=sum(cost0);   %因为最大成本生存概率最小，所以相加后总概率得 1.0
cost0=cost0/s0;  %生成每个个体的生存概率，成本最大的个体生存概率为 0,100*1 的矩阵
s=[[1:pop_size+2]',zeros(pop_size+2,1),zeros(pop_size+2,1)]; %生成 102*3 的矩阵
s(:,1)=s(:,1)-1;
s(pop_size+2,3)=1;    %最后一列最后一行置 1 表示区间上限
s(2:pop_size+1,3)=cost0;   %确定生存概率并赋给每个个体使用,在第三列
s=sortrows(s,3);    %按第三列生存概率大小顺序排列,但其原位置不变
s(pop_size+2,2)=1;
for i=2:pop_size+1  %第一行保存最佳个体不参与自然选择
    t=rand();
    j=search(t,s,1,pop_size+2); %寻找生存概率最接近自然选择概率的最可能存活个体
    out(i,:)=gene(j,:);  %j=1~100 之间的一个数，表示由几率取第 k 个个体基因（组合方案）
end
sort0=[[1:pop_size]',cost];  % '表示转置行为列
sort0=sortrows(sort0,2);   %把成本（即）按增序排列，1 列对应相应的个体位置
out(1,:)=gene(sort0(1,1),:); %在一次组合中，通过选择后，留下最小成本的那个个体样本
end
```

（5）寻找可以存活个体函数

```
function [out] = search(in,s,l,r)%查找
%使用二分法
mid=floor((l+r)/2);
if in<=s(mid,3)   %如果自然概率小于等于生存概率则继续查找
    if in>=s(mid-1,3)  %如果大于等于下一个生存概率则取之
```

```
        if(in~=0)  %不等于 0
            out=s(mid-1,1); %取位置编号
        else
            out=s(mid,1);
        end
    else
        %下一个小的部分仍大于自然概率则往小的部分继续查找
        out=search(in,s,l,mid);
    end
else      %如果自然选择概率大于生存概率则选择
    if in<s(mid+1,3)
        out=s(mid,1); %取位置编号
    else
        %如果下一个大的部分仍小于自然概率则继续往大的部分查找更接近生存概率的个体
        out=search(in,s,mid,r);
    end
end
end
```

（6）交叉运算的实现函数

```
function out = generate(gene,pop_size,cross_rate,Y) %交叉
array=zeros(1,100);   count=0;

while count<25
    i=floor(rand()*99)+1; %1-100
    j=floor(rand()*99)+1;
    if(i==1||j==1||i==j) continue; end   %所有个体参与交叉,除第一行保留的最佳个体外
    if(i==array(1,i)||j==array(1,j)) continue; end  %如果两个个体已发生交叉，则继续
    count=count+1;  array(1,i)=i; array(1,j)=j; %放入数组 array，表示第 i,j 个个体发生过交叉
    out=gene;       %导入种群基因序列
    mida=floor(rand()*23)+2;  %基因前半部分发生交叉概率地产生，1 和 51 基因不参与交叉
    midb=floor(rand()*24)+26;
    q=floor(rand()*25);
    midb1=midb+q;   %基因后半部分发生交叉概率地产生
    mida1=mida+q;
    if(midb1>=51)
        midb1=midb;
        mida1=mida;
    end
```

```
    t=rand();
    if t<=cross_rate   %在交叉概率内则进行交叉
    out(i,mida:midb)=gene(j,mida1:midb1);
    out(j,mida1:midb1)=gene(i,mida:midb);
    end
    out(i,:)=comepare(i,out,gene,Y);  %(比较函数)子代与母本比较，取最佳
    out(pop_size-i+2,:)=comepare(pop_size-i+2,out,gene,Y);
end

end
```

（7）变异函数

```
function out = mutate(gene,pop_size,mutation_rate)%变异
out=gene;
for i=2:pop_size; %第一个个体不参与突变
    for j=2:50;  %其余个体除第一或最后一个基因外的基因有突变的可能,第 1,51 不参与突变
        t=rand();
        if t<=mutation_rate
            out(i,j)=~out(i,j);  %如果发生变异，则将相应位置上的单个基因（单个测试点）取反
        end
    end
end
```

（8）comepare()函数，用于比较子代母代个体

```
function out=comepare(hpos,aout,gene,Y)
pos=hpos;
X=5:0.1:10;

out1=aout;  %子代
out2=gene;   %母本
c1=sum(out1(pos,:)==1); %测试点数量
c2=sum(out2(pos,:)==1);
pos1=find(out1(pos,:)==1);%测试点位置
pos2=find(out2(pos,:)==1);

x1=5+(pos1-1)*0.1; %测试点 X 值
x2=5+(pos2-1)*0.1;
y1=Y(:,pos1); %测试点 Y 值
y2=Y(:,pos2);
f1=spline(x1,y1);
f2=spline(x2,y2);
differ1=ppval(f1,X)-Y;%理论值与实际值的差
```

```
differ2=ppval(f2,X)-Y;

mon1=12*c1+errorcost(differ1)/400;%子代单个个体平均成本
mon2=12*c2+errorcost(differ2)/400;%母本单个个体平均成本

if(mon1>mon2)
    out=gene(hpos,:);
else
    out=aout(hpos,:);
end
end
```

(9)误差成本计算函数

```
function [out] = errorcost(diffrernce)%单个个体平均误差成本
t=abs(diffrernce);
t0=sum(sum(t<=0.4));
t1=sum(sum(t<=0.6))-t0;
t2=sum(sum(t<=0.8))-t0-t1;
t3=sum(sum(t<=1))-t0-t1-t2;
t4=sum(sum(t<=2))-t0-t1-t2-t3;
t5=sum(sum(t<=3))-t0-t1-t2-t3-t4;
t6=sum(sum(t<=5))-t0-t1-t2-t3-t4-t5;
t7=sum(sum(t>5));
out=0.1*t1+0.7*t2+0.9*t3+1.5*t4+6*t5+12*t6+25*t7;
end
```

（10）最终最优成本格式化输出函数

```
function [out] =evaluate(in,Y)%计算成本
out=length(in)*12;
X=5:0.1:10;
xx=5+(in-1)*0.1;
yy=Y(:,in);
f=spline(xx,yy);
difference=ppval(f,X)-Y;
out=out+errorcost(difference)/400;

fid=fopen('result4.txt','a');
fprintf(fid,'\n point: [ ');
fprintf(fid,'%2d ',in);
fprintf(fid,']    cost: %7f\n\n',out);
fclose(fid);
end
```

参考资料：
[1]统计推断讲座 2_问题的提出和基本求解思路 ppt

[2]网络资源
https://pt.sjtu.edu.cn/login.php（葡萄网）
[视频][MATLAB 数值分析与应用（第 2 版）] （注：matlab 教学视频）

[3]网络资源
http://baike.baidu.com/link?url=lBmxNPKM1rWRLb_axONRnYPptOQzkMJX6maZUJD5Yp_mf2czTudr0CAsGU4xi0KpKKYBOS4ANtdXKopbBlLeN_ （注：遗传算法介绍）

[4]附录中仅参考了于哲昆学长（第 43 组）的论文中的拟合方法讨论及程序设计代码。