统计推断在数模转换中的应用

组号 70 王頔 5140309028 刘可 5140309025

摘要：本课题运用统计推断的方法为监测模块的批量生产设计一种成本合理的传感特性校准方案。本文根据已知的少量数据实现对其余更多的未知数据的推断。文中分析了不同的拟合方法及其比较，选取了一种适合于本监测模块较优的拟合方法，三次样条插值法，并采用遗传算法，模拟一定种群大小的生物遗传进化方式。通过不断比较优化，使总成本较低。
关键词：统计推断、遗传算法（GA）、三次样条插值法，拟合

1 课题内容与目标

1.1 课题内容的提出

本课题研究的内容为传感器部件监测对象 Y 与传感部件的输出电压信号 X 之间的函数关系，由于二者之间并不存在确定的函数关系，因而可以通过研究大量的数据寻找适当的拟合函数。由于样本存在个体差异因而对单个样本的研究不能代表整体。在对整体研究时，如果用所有测试点进行函数拟合，运算量大，成本高。因而，该课题的研究内容为找出所有样本共同的数据特征点，使得用这些点拟合的函数与样本数据的误差尽可能小，以至于工程上可以接受。

1.2 课题目标

1.2.1 概述

课题的目标是从 51 个数据中找到几个特征点，使得根据这几个特征点所对应数据确定的拟合函数在与实际数据进行比较评价后定标成本尽可能小。

1.2.2 成本计算

- 单点定标误差成本

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

单点定标误差的成本按式（1）计算，其中 $y_{i,j}$ 表示第 i 个样本之第 j 点 Y 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。

- 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

$$S_i = \sum_{j=1}^{51} s_{i,j} + q \cdot n_i \tag{2}$$

对样本 i 总的定标成本按式（2）计算，式中 n_i 表示对该样本个体定标过程中的单点测定次数。

- 校准方案总成本

按式（3）计算评估校准方案的总成本，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。

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

总成本较低的校准方案，认定为较优方案。

2 寻找满足条件的拟合函数

2.1 对于样本数据的观察

我们用 MATLAB 画出多个样本数据（以下列两个样本为例）对应的 X-Y 图像如下：

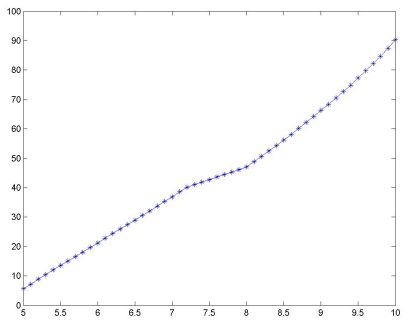


图 1：样本 16

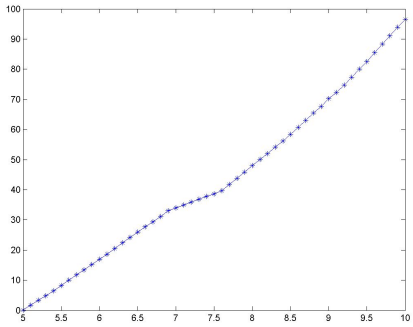


图 2：样本 117

观察样本发现有以下主要特征：

- Y 取值随 X 取值的增大而单调递增；
- X 取值在[5.0,10.0]区间内，Y 取值在[0,100]区间内；
- 不同个体的特性曲线形态相似但两两相异；
- 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。

通过观察分析，宜采用三次多项式或者三次样条插值的方法进行拟合。

3 基于特征点的差值拟合

3.1 概述

由于每组数据由 51 个 x-y 数值对构成，若使用 51 对数值对 x-y 关系进行研究，既增加了实际中测量的工作量，也使得通过全部 51 对被标定的数据得到的拟合曲线失去了工程上的实际意义。因而需要在这 51 个数值对中选取若干个特征点进行曲线拟合，一方面使得校准方案总成本最小，第二方面也使得拟合出来的曲线比较精确。因此需要解决取多少个点和怎么取的问题。

3.2 启发式算法的选择

特征点的确定直接决定了我们最终的拟合方案是否最优。我们首先想到的最简单的方式是用暴力搜索，我们需要排查 C_{51}^{k} (k=1,2,3,...,51)中情况，时间复杂度相当大，效率过低。因此我们需要采用其他的启发式搜索算法，有模拟退火算法和遗传算法。模拟退火算法计算过程简单，通用，健壮性强，适用于并行处理，可用于求解复杂的非线性优化问题，但是有收敛速度慢，执行时间长，算法性能与初始值有关及参数敏感等缺点，可能会找到局部最优解，而不一定能找到全局最优解，所以我们选用更具完备性和健全性的遗传算法。

3.3 遗传算法的介绍

遗传算法是计算数学中用于解决最佳化的搜索算法，是进化算法的一种。进化算法最初是借鉴了进化生物学中的一些现象而发展起来的，这些现象包括遗传、突变、自然选择以及杂交等。对于一个最优化问题，一定数量的候选解（称为个体）的抽象表示（称为染色体）的种群向更好的解进化。传统上，解用二进制表示（即 0 和 1 的串),但也可以用其他表示方法。进化从完全随机个体的种群开始，之后一代一代发生。在每一代中，整个种群的适应度被评价，从当前种群中随机地选择多个个体（基于它们的适应度），通过自然选择和突变产生新的生命种群，该种群在算法的下一代迭代中成为当前种群。

遗传算法的基本运算过程：

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)b)个体评价：计算群体 P(t)中各个个体的适应度。

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

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。

3.4 遗传算法的工作步骤

①参数列表：

a) 种群规模 popsize
b) 取点个数 dotnum
c) 交配概率 pcrossover
d) 变异概率 pmutation
e) 最大代数 genemax（也即终止条件）

②主程序：我们在最初就给定取点个数 dotnum 和最大代数 genemax。以循环 genemax 次作为函数的出口，最后一次的成本作为最终的取值。每一次循环中会顺序执行：选择、交叉（每选择两个个体，让二者有 pcrossover 的概率相互交叉）、变异并输出最优个体和成本。

√ 产生初始化种群：初始化种群为一个 popsize*51 的数组，种群中中有 popsize 个个体。每个个体为 51 元向量。在每一个个体中**随机**选出 dotnum 个点，将这些点的值**标记为 1**，其余为 0.标记为 1 的点就是我们的取点方案。

√ 选择操作：每个点被选择的概率已知的情况下，进行 popsize 次循环，每次从整个种群中，采用**赌轮选择**方法，随机选出一个个体。由于每个点被选择的概率不同，被选概率大（也即适应度高）的个体更容易被传到下一代，达到每一代都优化种群的效果。

√ 交配与变异：这两个函数使得种群被不断更新。合理地选取**变异概率**和**交配概率**（本实验中我们选择了 0.05 和 0.9），可以使得种群不至于出现收敛过早，出现部分最优解，成本偏高的情况；也不至于出现无法收敛，在终值附近震荡，无法达到种群成熟，得不到成本的情况。

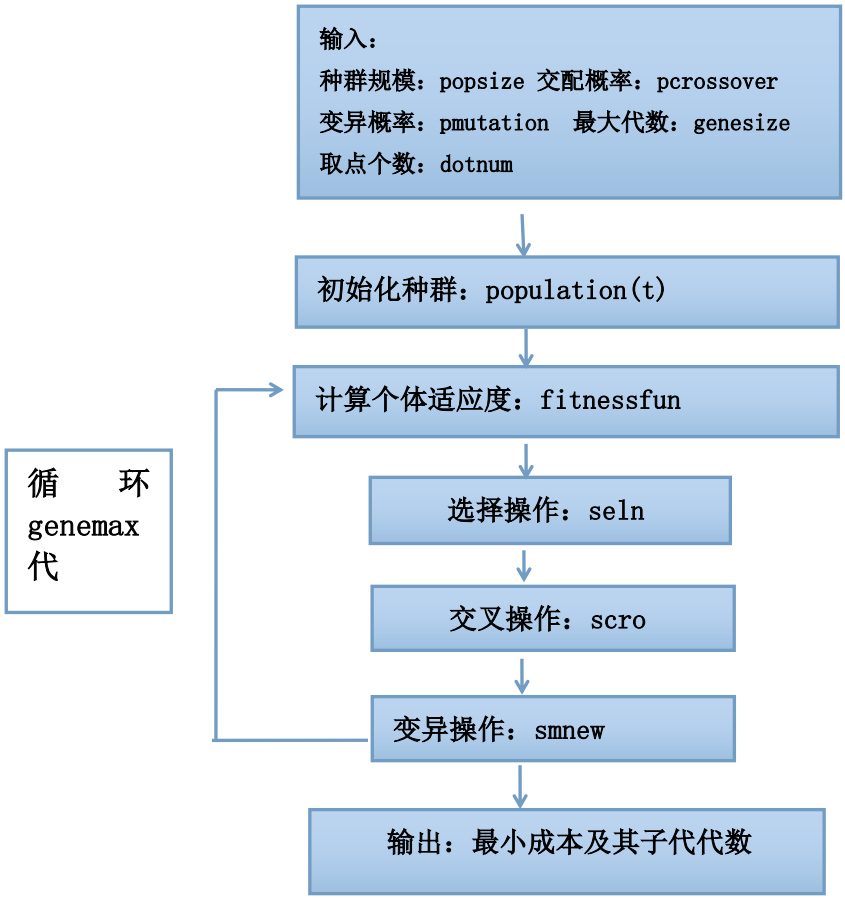
交配:选出两个个体,找出二者所有为 **1 的位置**,分别放在两个 dotnum 元的向量 num1，num2 中，然后让 num1 与 num2 交叉。达到让二者 1 的位置互相交换的效果。

i. 注：不直接让 51 元向量的个体交叉，是为了保持每个个体有 dotnum 个 1。比如若选取 dotnum=7，直接让个体交叉，可能使交换后的两个个体分别有八个 1 与六个 1，之后的程序就无法执行了。

变异：找出所有 1 的位置，放入一个**向量 num** 中，再在 num 中随机取点。达到在一个个体中随机选一个 1，再随机选一个 0，让 1 与 0 交换位置的效果。

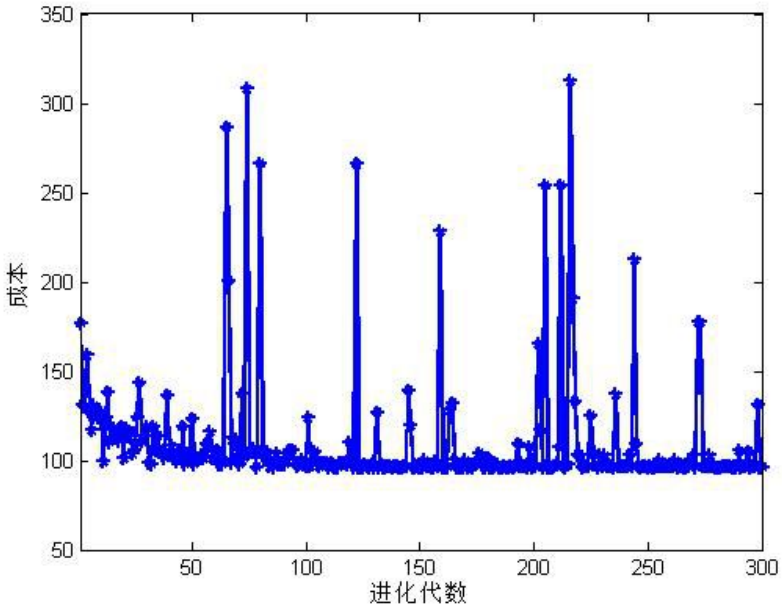
ii. 注：不直接在个体中选择两个交换位，是为了防止选出交换的是两个 0，而个体没有真正改变的情况。

√ 计算个体适应度：因为成本越低，个体越好，适应度越高。所以用每个个体的成本的倒数作为其被选择概率。

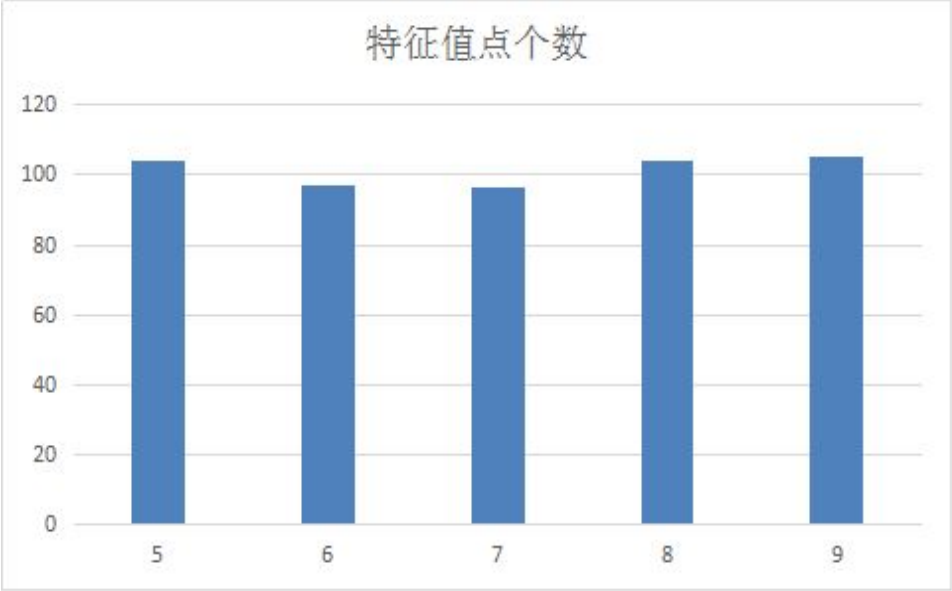


4. 分析与结论

4.1 成本曲线（以取点个数为 6, 变异概率为 0.05 为例）



4.2 结果数据



①平均生成下一代需要 6 秒左右时间。

②以三次多项式拟合得到的最优选点个数为 7 个，它们是 3 8 22 43 46 50，其最低成本为 96

以三次样条插值拟合得到的最优选点 **2 6 19 28 38 46 47**，其最低成本为 95

对比之下，我们选择以三次样条插值为拟合方法，点 2,6,19,28,38,46,47 作为选点的定标方案。

5.拓展

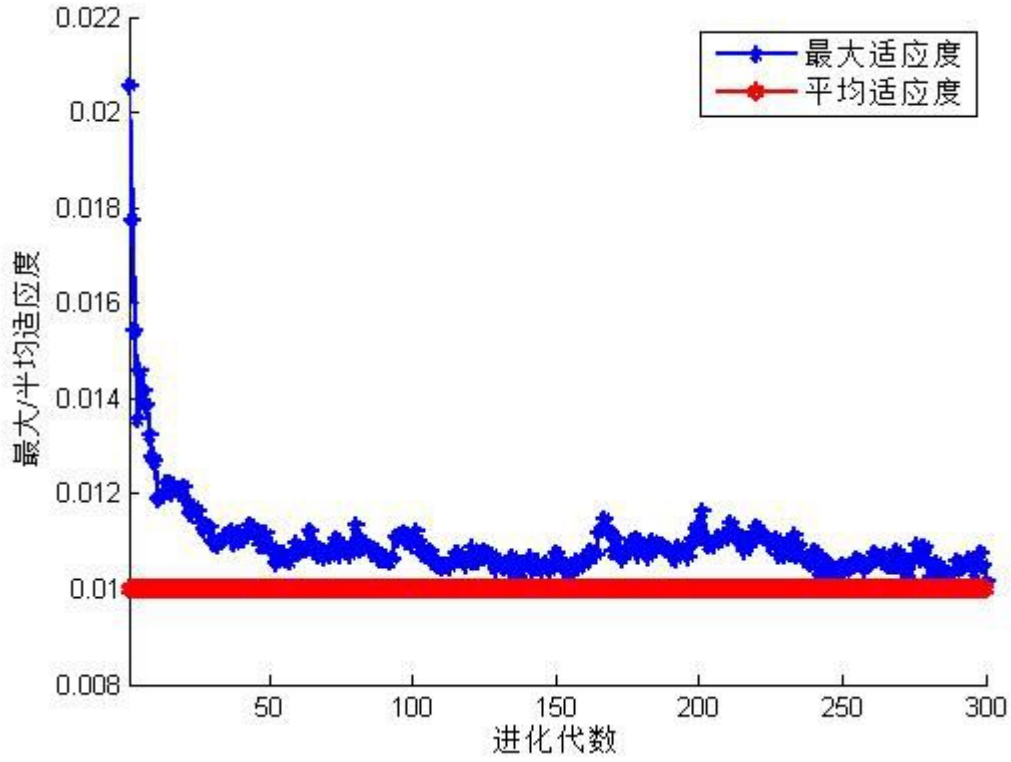
5.1 评估工作

因为我们的程序在循环一个给定次数 genemax 后，就会自动停下，而没有参考此时种群的状态。所以可能出现没有收敛或过早收敛的情况。为了规避这种弊端，我们采用了比较平均适应度和最大适应度的方式来评估收敛情况。

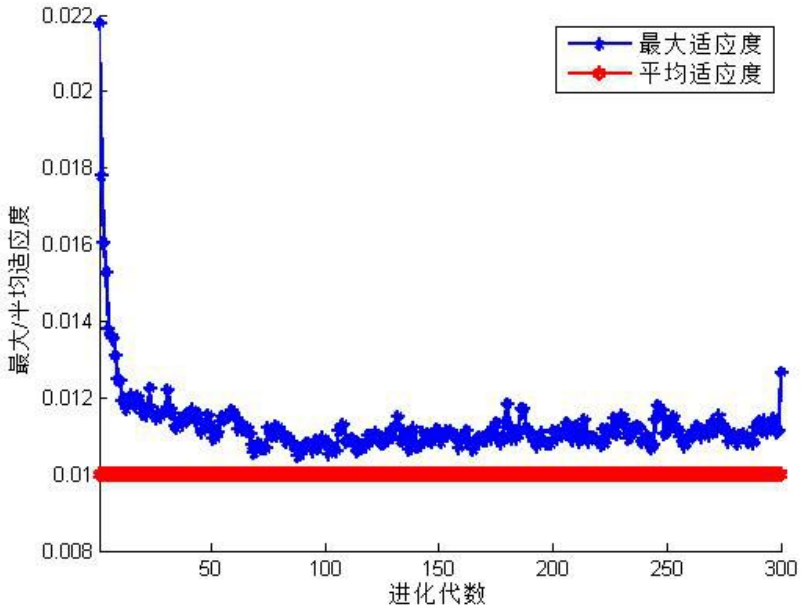
一般的，如果进化过程中种群的平均适应度和最大适应度在曲线上有相互趋同的形态，表示算法收敛进行得很顺利，没有出现震荡；在这种前提下，最大适应度个体连续若干代都没有发生进化表明种群已经成熟。[2]

我们的程序如果选择变异概率 0.05，两条曲线一直趋近，250 代之后就最终几乎重合，与此相对应，程序的结果也是每一代子代都几乎相同，可以认为到 300 代时种群已经成熟，答案是全局的最优解，结果可靠。

而相反地，比如选概率为 0.09,发现没重合，始终有间隙，而没有靠近的趋势，所以震荡，无法成熟，不能取解。



上图是在变异概率在 0.05 时的结果，最大适应度没有发生震荡情况，表示收敛的情况比较良好，种群已成熟。而变异概率在 0.09 时结果如下图



最大适应度在某一数值周围震荡，不趋近于平均适应度，说明收敛出现问题。
综上，我们选择了变异概率为 0.05。

6. 致谢

感谢袁焱老师的耐心指导与细致讲解，使我们发现报告的问题，为我们的报告提出宝贵意见，还有在同学们相互交流合作的氛围下，使我们逐渐掌握发现问题解决问题的能力！

参考文献

[1]上海交大电子工程系，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]卓金武 MATLAB 在数学建模中的应用（第 2 版）北京航空航天大学出版社 2011
[3]百度百科.遗传算法[M/OL].http://baike.baidu.com/遗传算法
[4]陈昭平; 张颖（计算机）MATLAB 程序设计与应用 高等教育出版社 2002
[5]苏金明; 阮沈勇 MATLAB 实用教程 电子工业出版社 第 2 版 2008

附录

```
%遗传算法主程序
function avgcostmin = genetic_all()
bitlength=51;   %染色体长度
popsize=100;    %初始种群大小
genemax=300;    %最大代数
pcrossover=0.9; %交配概率
pmutation=0.05; %变异概率

dotnum=7;       %选点个数

mdata=xlsread('20150915dataform.csv');

%产生初始化种群
population = initialize(popsize,bitlength,dotnum);
%计算适应度，返回适应度 fitvalue 和累计概率 cumsump
[fitvalue,cumsump,sp_x,sum] = fitnessfun(population,popsize,dotnum,mdata);

gene=1;
smnew=zeros(popsize,bitlength);
for gene=1:genemax-1
    gene
    for i=1:2:popsize
        %选择操作,选出两个个体，将其序号存储在二元数组 seln 中
        seln = selection(population,cumsump);
        %交配操作,将两个交叉过的个体存储在 2*bitlength 的数组 scro 中
        scro = crossover(population,seln,pcrossover,bitlength,dotnum);
        %变异操作
        smnew(i,:)   = mutation(scro(1,:),pmutation,bitlength,dotnum);
        smnew(i+1,:) = mutation(scro(2,:),pmutation,bitlength,dotnum);
```

```
    end
    population=smnew;
    [fitvalue,cumsump,sp_x,sum] = fitnessfun(population,popsize,dotnum,mdata);
    %记录当前代最好的适应度和平均适应度
    [fmax,nmax] = max(fitvalue);
    fmean = mean(fitvalue);
    ymax(gene) = fmax;
    ymean(gene) = fmean;
    %计算最小成本及其对应子代,並輸出
    [avgcostmin,bestkid] = finalcosting(sum,sp_x);
    bestkid
    avgcostmin
    avgcostminArr(gene) = avgcostmin;
end

gene = gene + 1
    for i=1:2:popsize
        %选择操作,选出两个个体，将其序号存储在二元数组 seln 中
        seln = selection(population,cumsump);
        smnew(i,:)   = mutation(scro(1,:),pmutation,bitlength,dotnum);
        smnew(i+1,:) = mutation(scro(2,:),pmutation,bitlength,dotnum);
    end
    population=smnew;
    [fitvalue,cumsump,sp_x,sum] = fitnessfun(population,popsize,dotnum,mdata);
    %记录当前代最好的适应度和平均适应度
    [fmax,nmax] = max(fitvalue);
    fmean = mean(fitvalue);
    ymax(gene) = fmax;
    ymean(gene) = fmean;
    %计算最小成本及其对应子代,並輸出
    [avgcostmin,bestkid] = finalcosting(sum,sp_x);
    bestkid
    avgcostmin
    avgcostminArr(gene) = avgcostmin;
```

%绘制经过遗传运算后的适应度曲线。一般的，如果进化过程中种群的平均适应度和最大适应度在曲线上有相互趋同的形态，

%表示算法收敛进行得很顺利，没有出现震荡；在这种前提下，最大适应度个体连续若干代都没有发生进化表明种群已经成熟

```
figure(1)
hand1 = plot(1:genemax,ymax);
set(hand1,'linestyle','-','linewidth',1.8,'marker','*','markersize',6)
```

```
hold on;
hand2 = plot(1:genemax,ymean);
Set
(hand2,'color','r','linestyle','-','linewidth',1.8,'marker','h','markersize',
6)
xlabel('进化代数');ylabel('最大/平均适应度');xlim([1 genemax]);legend('最大适应度
','平均适应度');
box off;hold off;

%绘制遗传运算每一代的成本变化趋势
figure(2)
hand = plot(1:genemax,avgcostminArr);
set(hand,'linestyle','-','linewidth',1.8,'marker','*','markersize',6)
xlabel('进化代数');ylabel('成本');xlim([1 genemax]);


%以下为子函数
%产生初始化种群
function population = initialize(popsize,bitlength,dotnum)
   population = zeros(popsize,bitlength);
   for i=1:popsize
      for j=1:dotnum
         n=randi(51);
         while(population(i,n)==1)
            n=randi(51);
         end
         population(i,n)=1;
      end
   end

 %选择操作,选出两个个体，将其序号存储在二元数组 seln 中
 function seln = selection(population,cumsump)
    seln=zeros(1,2);
    for i=1:2
       r=rand;
       prand=cumsump-r;
       j=1;
       while prand(j)<0
          j=j+1;
       end
       seln(i)=j;
    end
```

```
%交叉操作,将两个交叉过的个体存储在 2*bitlength 的数组 scro 中
function scro = crossover(population,seln,pc,bitlength,dotnum)
    num=zeros(2,dotnum);
    for i=1:2
        k=1;
        for j=1:bitlength          %找到被选取的 7 个点的编号放进数组 num,
          if population(seln(i),j)==1
              num(i,k)=j;
              k=k+1;
          end
        end
    end
    num;

     chb = randi(dotnum-1)+1; %在[1,bitlength-1]的范围内随机产生一个交换位 changebit

        while num(1,chb)<=num(2,chb-1) || num(2,chb)<=num(1,chb-1)
            chb = randi(dotnum-2)+1;
        end
        num1(1,:) = [num(1,1:chb-1) num(2,chb:dotnum)];
        num1(2,:) = [num(2,1:chb-1) num(1,chb:dotnum)];
    else
        num1=num;
    end
    num1;
    scro=zeros(2,bitlength);
    for i=1:2
        for j=1:dotnum
            scro(i,num1(i,j))=1;
        end
    end

%变异操作 此处规定变异是随机选出一个 1 与一个 0 交换位置
%此处的 smnew 和 scro 都是一维数组
function smnew = mutation(scro,pm,bitlength,dotnum)
     num=zeros(1,dotnum);
        k=1;
        for j=1:bitlength          %找到被选取的 7 个点的编号放进数组 num,
          if scro(1,j)==1
              num(1,k)=j;
              k=k+1;
          end
        end
```

```
    smnew = scro;
    if rand < pm
        chb1 = num(randi(dotnum)); %在[1,bitlength]的范围内随机产生第一个交换位
changebit
        chb2 = randi(bitlength); %在[1,bitlength]的范围内随机产生第二个交换位
changebit

        while chb1==chb2 %如果两个 changebit 重合，则重新选第二个
            chb2 = randi(bitlength);
        end

        smnew(chb1) = scro(chb2);
        smnew(chb2) = scro(chb1);
    end



function [fitvalue,cumsump,sp_x,sum] =
fitnessfun(population,popsize,dotnum,mdata)
sp_x=zeros(popsize,dotnum);   %将放入被选择的 7 个数据编号
sp_x1=zeros(1,dotnum);        %将放入被选择的 7 个 x 值
sp_y=zeros(400,dotnum);   %将放入每一个个体的取点方案下的所有 y 值
M=800;
N=51;
x=zeros(1,N);
for i=1:N          %把各个数据点的 x 的值用一个向量表示
    x(1,i)=mdata(1,i);
end
y0=zeros(1,N);    %放置三次插条拟合出来的数值
y=zeros(M/2,N);   %将放置所有的数据值
for i=1:M/2
    y(i,:)=mdata(2*i,:);
end
sum=zeros(1,popsize);   %放入每个个体的总成本

for i=1:popsize      %对于每一个样本（每一種選點方案）
    k=1;
    for j=1:N          %20 行到 26 行找到被选取的 7 个点的 x 编号放进数组 spy_x,
       if population(i,j)==1
           sp_x(i,k)=j;  %被選擇的數據編號
           sp_x1(1,k)=x(1,j); %被選擇的 7 個 x 值
            for number=1:400
                sp_y(number,k)=y(number,j); %一列一列地放入所有每個個體，被選擇的 7 個 y
值
```

```
            end
           k=k+1;
        end
    end
    for number=1:400
        y0=interp1(sp_x1,sp_y(number,:),x,'spline');  %对这 7 个数据点作 3 次插条拟合
        singlecost=12*dotnum; %测量成本
        for data=1:51            %对 51 個测量点 400 组数据的误差总和
            singlecost=singlecost+costing(y0(1,data),y(number,data));
        end
        sum(1,i) = sum(1,i)+ singlecost;
    end
end


%接下來计算适应度 fitvalue（適應度就是被選中的概率，所有個體的 fitvalue 求和爲 1）
fitvalue=zeros(1,popsize);
Psum = 0;    %每個個體(1/成本)的總和
for p=1:popsize         %为了使成本越低，适应度越高，所以求倒数所占百分比
    Psum=Psum+1/sum(1,p);
end
for m=1:popsize
    fitvalue(1,m)=1/(sum(1,m)*Psum);
end
cumsump = zeros(1,popsize);
cumsump(1,1)=fitvalue(1,1);
for q=2:popsize     %累計概率
    cumsump(1,q)=cumsump(1,q-1)+fitvalue(1,q);
end

%计算单点误差成本
function singlecost = costing(y1,y)
       s=abs(y1-y);
       if s<=0.4
           cost=0;
       elseif s<=0.6
           cost=0.1;
       elseif s<=0.8
           cost=0.7;
       elseif s<=1
           cost=0.9;
       elseif s<=2
           cost=1.5;
       elseif s<=3
```

```
            cost=6;
        elseif s<=5
            cost=12;
        else cost=25;
        end
    singlecost = cost;


%计算一代中的最小成本及其对应子代
function [avgcostmin,bestkid] = finalcosting(sum,sp_x)
%寻找最小值并记录编号，输出对应的子代
tmp=sum(1,1);
num=1;
for i=2:100
    if tmp>sum(1,i)
        tmp=sum(1,i);
        num=i;
    end
bestkid=sp_x(i,:);
avgcostmin=sum(1,i)/400;
end
```