# 统计推断在数模转换系统中的应用

**组号：38　姓名：吴紫阳 学号：5140309418，姓名：裴帅 学号：5140309427**

**摘要：**本文是为了用遗传算法找到使定标成本最低的定标工序，从而根据提供的关于传感器部件的 400 组样本数据，对不同的拟合方式进行比较以找到最佳的拟合方式后，再对种群进化过程中交叉互换和基因突变的最优方式进行探究,最后运用找到的最佳方式得出我们想要的最优解以解决问题。

**关键词：**遗传算法，拟合方式，交叉互换，基因突变，比较，最优解

**引言：**假定有某型投入批量试生产的电子产品，其内部有一个模块，功能是监测某项与外部环境有关的物理量（可能是温度、压力、光强等）。该监测模块中传感器部件的输入输出特性呈明显的非线性。本课题要求为该模块的批量生产设计一种成本合理的传感特性校准（定标工序）方案。

## 1. 基于遗传算法的传感特性校准方案求解步骤

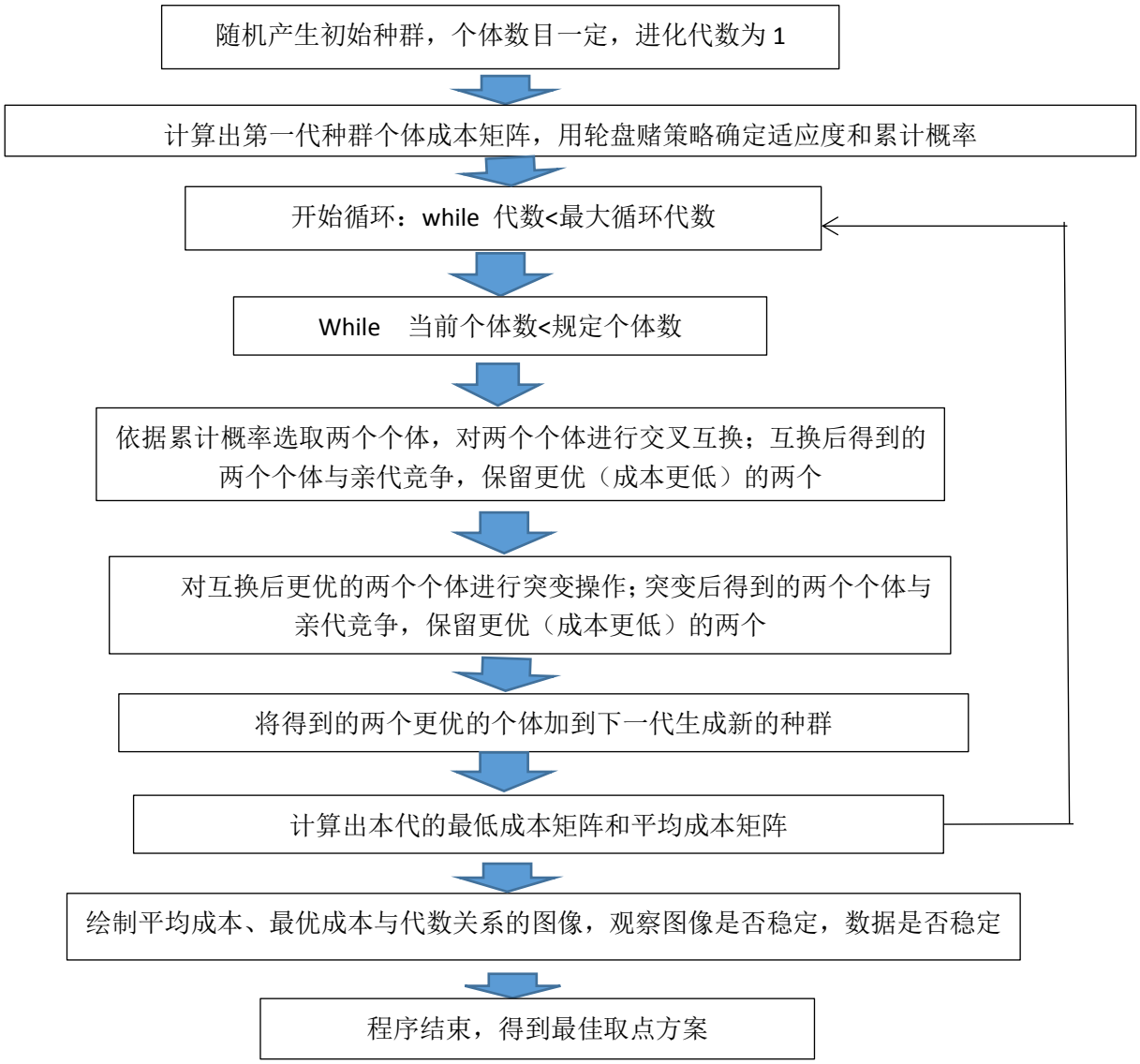
1、选择一种拟合（或插值）方法，也就是选择表达式
2、选择一种取点方案
3、根据 2 从样本库获取单点测定数值，按 1 的方案进行拟合，对所有样本逐一完成定标
4、根据 3 的定标结果，结合样本库原始数据，测算成本并记录
（若成本基本趋于稳定，转向 5，否则转向 2）
5、将成本最低的方案作为最终结果

## 2.遗传算法的具体实现

遗传算法基于自然选择的生物进化，是一种模仿生物进化过程的随机方法。它将取点集合看作是自然界中的种群，每个取点看作是种群之中的个体，按照自然界中的适者生存和优胜劣汰的原理，逐代演化产生出越来越好的个体，即近似解。但只能产生出近似解，这是因为个体是按某一种方式选择的，不能真正遍历到每一个个体，但是通过遗传算法所找到的近似解可以保证与实际解之间的差距足够小。

由此得到基本方案，我们选定要演化的代数 Generationmax 和种群的大小为 popsize，然后用一个长度为 BitLength（51）的二进制向量表示种群中的个体，这个实质就是表示取点方案，其中每一个 Bit 用 0 和 1 表示，0 表示舍弃该点，1 表示选取该点。之后随机产生初始种群，在循环增加代数条件下，将上一代种群进行交叉互换和变异，根据适应度 Fitness

选择下一代个体，适应度高的被选择，低的被淘汰；接着计算出本代最低成本矩阵 BestCost 和平均成本矩阵 AveCost；最后则是绘制出 BestCost 和 AveCost 与进化代数之间的图像，从而得到近似解。下面是我们算法对应的流程图：




关于在交叉互换和突变之前两个最优个体的选取，我们是选择了上一代得到的两个最优个体，这样就保证了具有良好基因的上一代个体一定能够传承到下一代，从而使每一代个体严格地向更适应环境的方向进化，并且这种进化的速度也是较快的。

而关于交叉互换和突变的方式的选择，不同的方式对进化速度和程度的影响是不同的，我们通过对不同方式的比较和验证选出了较合适的，具体情况会在下面进行比较分析。

## 3.拟合方式选择

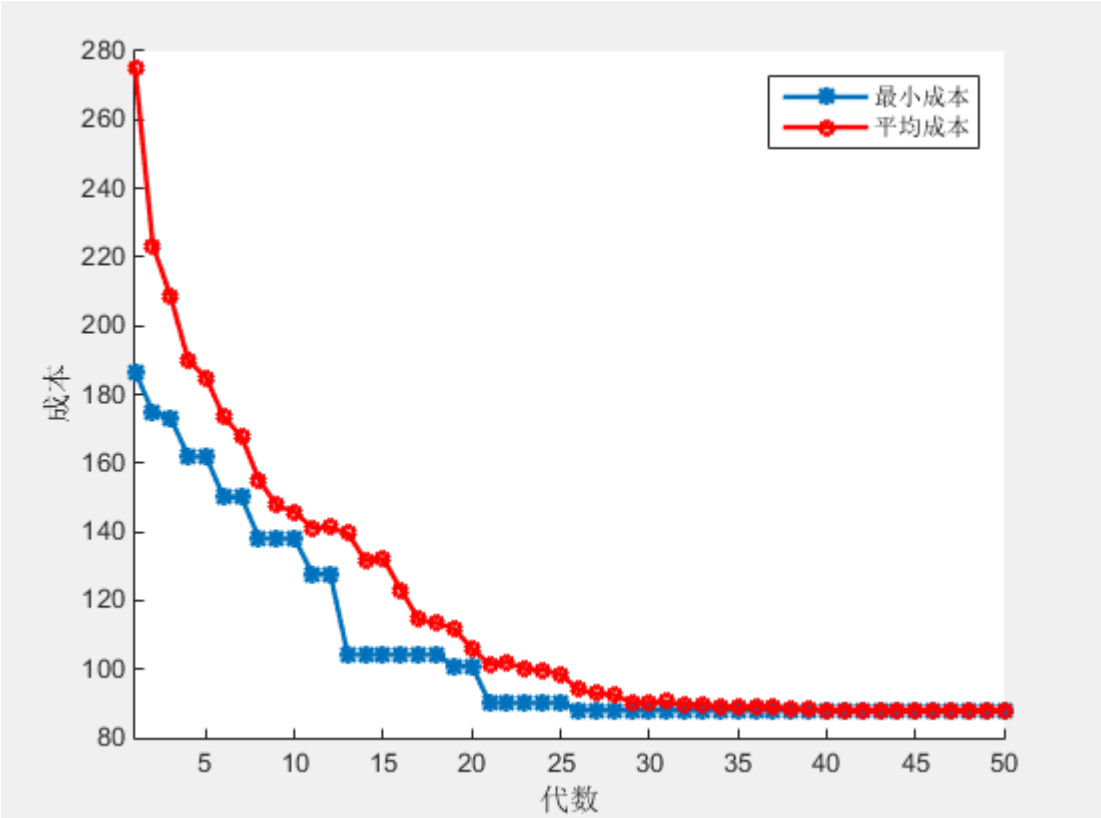
我们选取的进化代数为 50 代，种群的大小为 50，规定选取点的数目至少为 5 个，这是因为选取点数目过少是不合理的，易出现偶然性。接下来我们进行了对三次厄米多项式插值、三次多项式拟合和三次样条插值这三种拟合方式的比较和选择，得出我们认为较适合解决本题的拟合方式。

首先我们进行了“分段三次 Hermite 多项式插值”（pchip），最后得到的最低成本是 88.1557，对应的最佳选点方案是选取 2，18，27，36 和 48 这 5 个点。

然后我们进行了“三次多项式拟合”（polyfit），最后得到的最低成本是 116.2255，与之对应的最佳选点方案是选取这 4，17，27，38 和 49 这 5 个点。

最后我们进行了“三次样条插值”（spline），最后得到的最低成本是 97.8377，与之对应的最佳选点方案是选取 1，10，21，25，32，42 和 50 这 7 个点。

暂不考虑其他拟合方式，将以上三种方式进行比较，我们认为“分段三次 Hermite 插值多项式插值”（pchip）是目前最合理的拟合方式，由此得到的最低成本是 85.7335，最佳选点方案是选取 2，18，27，36 和 48 这 5 个点。对应的图像说明是：



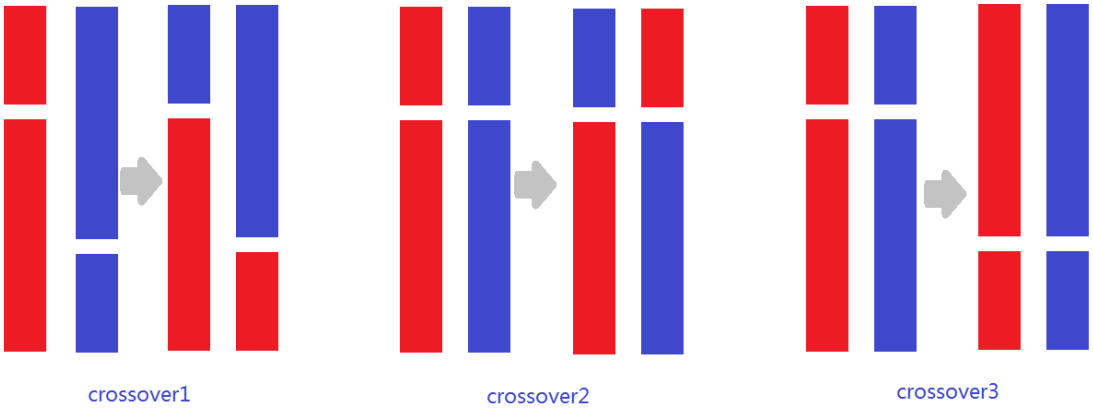

从所得的具体数据可以观测到，当进化代数越来越大，最低成本和平均成本虽然都还在减小，但这种减小是很微弱的，说明选择的个体已经基本上不再发生变化，个体进化程度已较高，因此我们认为种群已基本达到最优，可以将此解作为这种拟合方式下的最优解。

## 4.交叉互换方式选择

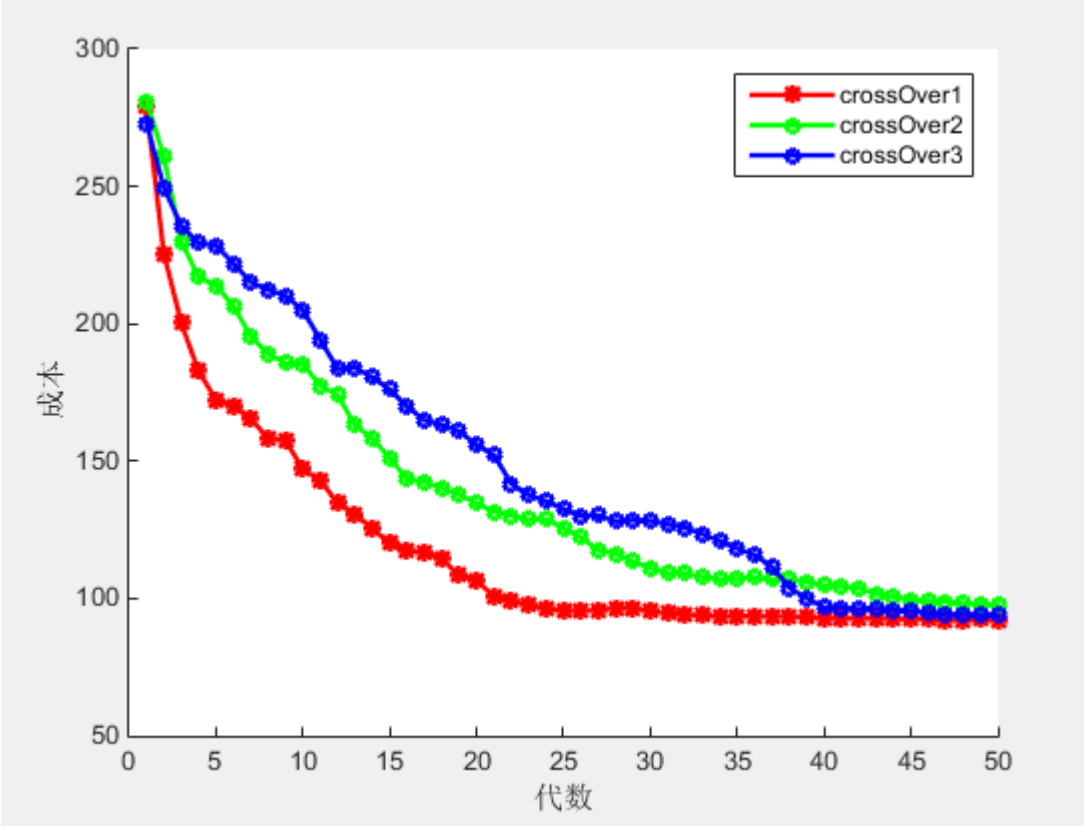
由于交叉互换对种群进化的速度和程度有着非常重要的影响，所以对种群个体交叉互换的方式的选取是非常有必要的，我们讨论的三种交叉互换方式如下图所示，选择这三种交叉互换方式是因为这是自然界中存在的较普遍的三种方式。我们在这三种交叉互换方式中选取较优的依据是作出每种方式下“进化代数与平均成本关系”曲线，较快达到稳定状态和成本更低的我们认为是更优的交叉互换方式。但事实上在运行过程中，我们选择的更优的交叉互换方式并不是每次都更快地使种群趋于稳定，也并不是每次得到的近似解都比其他方式更优，这是因为种群的进化速度和程度是由多个方面的因素决定的，因此，我们可以说我们选择的

交叉互换方式是更优的，它能够更大概率地让种群更快地趋于稳定。

上面说到种群进化速度和程度由多个方面因素决定，因此我们必须要控制变量，以保证结果的准确性。在这里我们除了种群的规模、进化的代数、基因突变的方式等的一致，最重要的是控制种群初始状态一致，因为种群的初始化是随机的，我们在产生初始种群后分别进行了三次交叉互换方式不同的进化。




Crossover1：第一个个体的头和第二个个体的尾交叉互换
Crossover2：第一个个体的头和第二个个体的头交叉互换
Crossover3：两个个体的头分别和自己的尾交叉互换




由上图像可知：

1、在 37 代左右之前，crossOver1<crossOver2<crossOver3，在 37 代左右之后，

crossOver1<crossOver3<crossOver2

2、crossOver2 和 crossOver3 所代表的平均成本基本上同时达到稳定，差异不大，crossOver1 所代表的平均成本达到稳定的时间明显更短

3、最后稳定是 crossOver1 为 89.2223，crossOver2 为 97.6612，crossOver3 为 91.9620，crossOver1 最小

由此我们得出 crossOver1 使种群进化速度最快，稳定程度也最高，是最好的交叉互换方式。但联系实际的话，在自然状态下 crossOver1 形式的交叉互换远没有 crossOver2 来得普遍，这似乎与自然规律违背，我们认为，这毕竟不是正常的自然演化，而是人为给定数据，数据存在的特殊性使得在此题情况下就是 crossOver1 是最优的交叉互换方式，我们最后选择的方式就是 crossOver1。

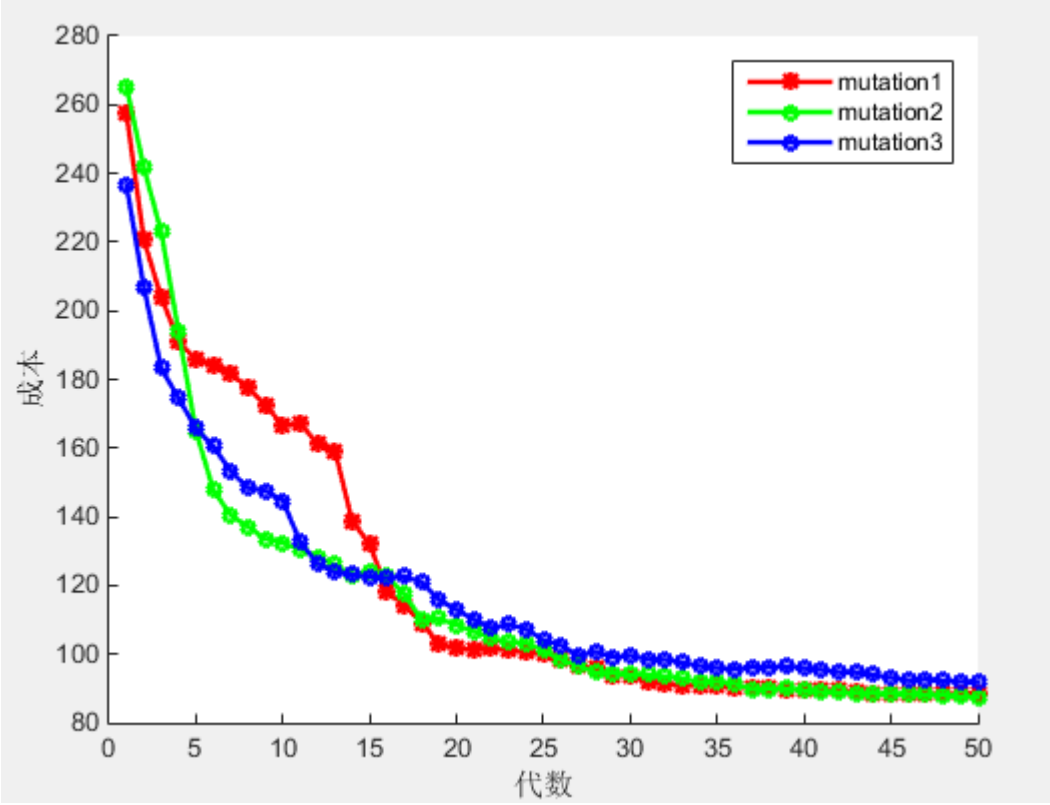
# 5.基因突变方式选择

与交叉互换类似，基因突变对种群进化的速度和程度同样有着非常重要的影响。我们也讨论了基因突变的三种情况：

（1）Mutation1：0.7 的概率随机选择一个位置变异;0.3 的概率随机选择两个位置变异

（2）Mutation2:以 0.5 的概率随机选择一个位置变异;0.5 的概率随机“丢弃”一个位置，即舍弃一个位置的基因，后段所有基因整体前移一位，第 51 位用 0 补齐；

（3）Mutation3:随机选择一个位置变异。




由上图像可得：

1、在前 5 代刚开始进化时三条曲线差别不大

2、三条曲线达到稳定的代数差别不大

3、最终达到稳定时有 mutation1=87.8273，mutation2=86.4430，mutation3=90.1272，mutation2<mutation1<mutation3

由此我们得出 mutation2 是最好的突变方式，这与我们推测的差不多，虽然自然界中舍弃一位基因编码整体前移会使基因发生严重改变，但是在我们的数据中这种方式却可以很大概率地将成本低的个体传承到下一代，故我们选择的突变方式是 mutation2。

# 6.结论

1、我们选择的最优插值方法是分段三次 Hermite 插值多项式插值（pchip）

2、我们选择的最优的交叉互换方式是 Crossover1（第一个个体的头和第二个个体的尾交叉互换）

3、我们选择的最优的基因突变方式是 mutation2

4、最后选择以上的方式运行一次得到的最低成本是 88.1557，对应的最佳选点方案是选取 2，18，27，36 和 48 这 5 个点

# 7.评价：

1、我们的优点是较好地运用遗传算法解决了问题，并基于遗传算法找到了更快的进化方式

2、缺点是我们只进行了遗传算法内部的比较，找出了最优的拟合和变异方式，却没有使用不同的算法进行比较；并且在对拟合和变异方式进行比较时，我们只能控制一定的无关变量，这会造成一定的误差

# 8.参考文献

[1]史峰，王辉，胡斐，郁磊. Matlab 智能算法 30 个案例分析. 北京航空航天大学出版社，2011.
[2]陈杰. Matlab 宝典. 电子工业出版社，2007.

# 附录

## Matlab 代码

## 1.测试文件

## test_ur_answer.m

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

my_answer=[ 4,17,25,35,48 ];%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);
```

## mycurvefitting.m

```
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');
%p=polyfit(x_premea,y0_premea,3);
%y1=polyval(p,x);

end
```

# 2.遗传算法文件

## main.m

```
clear all;
close all;
BitLength=51;
popsize=30;
Generationmax=50;
global data;
data=csvread('data.csv');
%随机产生初始种群
population=zeros(popsize,51);
for i=1:popsize
    population(i,:)=RandomCreate();
end
%计算适应度函数，返回适应度 Fitness 和积累概率 accumRat
[Fitness,accumRat]=fitCal(population);
```

```
CrossOver=zeros(popsize,BitLength);     %储存交叉互换后新的种群
MutationOver=zeros(popsize,BitLength);  %储存突变后新的种群
BestOnes=zeros(Generationmax,BitLength);    %最优个体矩阵
BestCost=zeros(Generationmax,1);     %最优成本
AveCost=zeros(Generationmax,1);      %平均成本

Generation=1;
while Generation-1<Generationmax
    disp(Generation);
    [max1,position1]=max(Fitness);
    Fitness(position1)=0;
    [max2,position2]=max(Fitness);
    select(1)=position1;
    select(2)=position2;
    flag=false;
    Fitness(position1)=max1;
    for j=1:2:popsize
        if flag
            select=selection(population,accumRat);  %依概率选择个体
        end
       %完成交叉互换操作 cro 表示成功与否
        [selectPoints,cro1,cro2]=Cross(population,select);
       if cro1==0       %0 表示交换不成功，1 表示交换成功
          cost1=Fitness(select(1));
       else cost1=1./((costSum(selectPoints(1,:)).^2));
       end
       if cro2==0
          cost2=Fitness(select(2));
       else cost2=1./((costSum(selectPoints(1,:)).^2));
       end
       if Fitness(select(1))>=cost1
          CrossOver(j,:)=population(select(1),:);
          cost1=Fitness(select(1));
       else CrossOver(j,:)=selectPoints(1,:);
       end
       if Fitness(select(2))>=cost2
          CrossOver(j+1,:)=population(select(2),:);
          cost2=Fitness(select(2));
       else CrossOver(j+1,:)=selectPoints(2,:);
       end
       %完成变异操作，mut 表示成功与否
       %选择父子代中更优的
       [selectA,mut1]=mutation(CrossOver(j,:));
       [selectB,mut2]=mutation(CrossOver(j+1,:));
```

```
        if mut1==0
            cost3=cost1;
        else cost3=1./((costSum(selectA(1,:)).^2));
        end
        if mut2==0
            cost4=cost2;
        else cost4=1./((costSum(selectA(1,:)).^2));
        end
        if cost1>=cost3
            MutationOver(j,:)=CrossOver(j,:);
        else cost1=cost3;MutationOver(j,:)=selectA;
        end
        if cost2>=cost4
            MutationOver(j+1,:)=CrossOver(j+1,:);
        else cost2=cost4;MutationOver(j+1,:)=selectB;
        end
        Fitness(j)=cost1;
        Fitness(j+1)=cost2;
        flag=true;
    end
    population=MutationOver;        %更新种群
    %计算新种群的积累概率
    SumAccumCost=sum(Fitness);
    choiceRat=Fitness/SumAccumCost;
    accumRat(1)=choiceRat(1);
    for i=2:popsize
        accumRat(i)=accumRat(i-1)+choiceRat(i);
    end
    %记录当前代最优的适应度和平均适应度
    [maxVal,position]=max(Fitness);
    BestCost(Generation)=1./((maxVal).^0.5);
    disp(BestCost(Generation));
    AveCost(Generation)=mean(1./(Fitness).^0.5);

    x=population(position,:);
    BestOnes(Generation,:)=x;
    Generation=Generation+1;
end

[BestVal,minPos]=min(BestCost);      %取成本最小为最佳成本
BestMethod=BestOnes(minPos,:);
%绘制平均适应度和最大适应度曲线
figure(1)
hand1=plot(1:Generationmax,BestCost);
```

```
set(hand1,'linestyle','-','linewidth',1.8,'marker','*','markersize',6
)
hold on;
hand2=plot(1:Generationmax,AveCost);
set(hand2,'color','r','linestyle','-','linewidth',1.8,'marker','h','m
arkersize',6)
xlabel('代数');ylabel('成本');xlim([1 Generationmax]);
legend('最小成本','平均成本');
box off;hold off;
```

## fitCal.m

```
%计算适应度函数
function [Fitness,accumRat]=fitCal(population)
popsize1=size(population,1);
Fitness=zeros(1,popsize1);
accumRat=zeros(1,popsize1);
for i=1:popsize1
    x=population(i,:);
    Fitness(i)=costSum(x);
end
Fitness=1./((Fitness).^2);
fitsum=sum(Fitness);
Ppopulation=Fitness/fitsum;
accumRat(1)=Ppopulation(1);
for i=2:popsize1
    accumRat(i)=accumRat(i-1)+Ppopulation(i);
end
Fitness=Fitness';
accumRat=accumRat';
end
```

## RandomCreate.m

```
%个体随机生成
function individual=RandomCreate()
x=0;
while x<4   %每代至少 5 个个体被选取
    individual=round(rand(1,51));
    x=sum(individual);
end
End
```

## Cross.m

```
%判断交叉互换是否成功
function [selectPoints,cro1,cro2]=Cross(population,select)
cro1=1;
cro2=1;
BitLength1=size(population,2);
selectPoints=zeros(2,51);
while sum(selectPoints(1,:))<5||sum(selectPoints(2,:))<5
    CrossPos=round(rand*(BitLength1-2))+1;      %产生突变位子
    selectPoints(1,:)=[population(select(1),1:CrossPos)
population(select(2),1:51-CrossPos)];
    selectPoints(2,:)=[population(select(1),CrossPos+1:51)
population(select(2),51-CrossPos+1:51)];
end
end
```

## mutation.m

```
%判断变异是否成功
function [MutationOver,mut]=mutation(ss)
BitLength=size(ss,2);
MutationOver=zeros(1,51);
mut=1;
if rand<0.7
    while sum(MutationOver)<4
        mutPos=round(rand*(BitLength-1))+1;
        ss(mutPos)=1-ss(mutPos);
        MutationOver=ss;
    end
else
    while sum(MutationOver)<4
        mutPos=round(rand(1,2)*(BitLength-1))+1;
        ss(mutPos)=1-ss(mutPos);
        MutationOver=ss;
    end
end
```

## CostSum.m

```
%计算个体成本
function [average]=costSum(points)
global data;
currentLength=1;
for i=1:2:799
    x=data(i,:);
    y=data(i+1,:);
```

```
    c(currentLength)=costOne(x,y,points);
    currentLength=currentLength+1;
end
    average=mean(c);
end

%计算个体成本
function [average]=costSum(points)
global data;
currentLength=1;
for i=1:2:799
    x=data(i,:);
    y=data(i+1,:);
    c(currentLength)=costOne(x,y,points);
    currentLength=currentLength+1;
end
    average=mean(c);
end
```

## CostOne.m

```
%计算一个样本的成本
function [cost]=costOne(x,y,points)
pos=find(points==1);
x2=x(pos);
y2=y(pos);
%p=polyfit(x2,y2,3);
%q=polyval(p,x);
q=interp1(x2,y2,x,'pchip');
differ=abs(q-y);
sum=0;

for i=1:length(differ)
    if differ(i)>0.5 && differ(i)<=1
        sum=sum+0.5;
    end
    if differ(i)>1 && differ(i)<=2
        sum=sum+1.5;
    end
    if differ(i)>2 && differ(i)<=3
        sum=sum+6;
    end
    if differ(i)>3 && differ(i)<=5
        sum=sum+12;
```

```
    end
    if differ(i)>5
        sum=sum+25;
    end
end
cost=sum+12*length(pos);
end
```

## selection.m

```
%从种群中选择两个个体
function select=selection(population,accumRat)
select=zeros(2);
for i=1:2
    r=rand;
    randRat=accumRat-r;
    j=1;
    while randRat(j)<0
        j=j+1;
    end
    select(i)=j;
end
end
```