统计推断在数模转换系统中的应用

第 49 小组　　马带钢 5140309151；王晓星 5140309247

摘要：本课程目标为围绕一个数模、模数转换电路的非线性问题开展研究，训练学生在电子工程领域应用统计学方法的实际能力。课程中使用拟合、插值、启发式算法等知识对大量数据进行处理，且在一定的成本计算公式下得出时间、成本上最优化的拟合方案。课程中用到的数据处理、编程软件为 matlab。

关键词：三次样条插值拟合，遗传算法，优化成本，matlab

1. 引言

生产中常出现输入输出特性未知且呈非线性的元件，为了大致确定其输入-输出关系式，需要对离散的数据进行拟合。由于数据点较多，需要选出部分点进行拟合，但不可能采取穷举的方式确定取点方法，因此需要用到启发式算法，如遗传算法来优化取点。最后，不同的测定、取点、拟合方法在实际生产中所需的成本不同，所以需要按一定的方法计算总成本，从而得到生产过程中可行的最佳方案。

2. 课程简介

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]。

2.1 模型概述

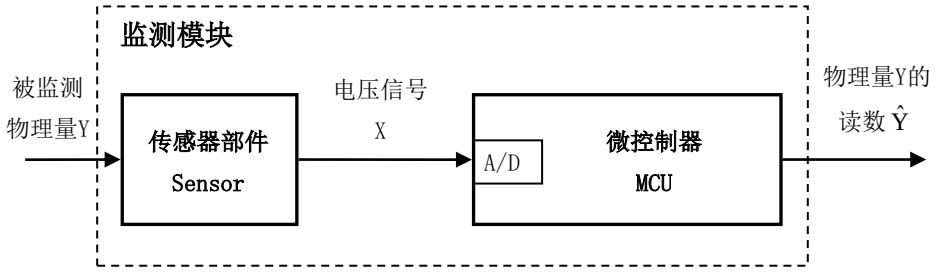

图 2-1 监测模块组成框图

监测模块的组成框图如图 1。其中，传感器部件（包含传感器元件及必要的放大电路、调理电路等）的特性是我们关注的重点。传感器部件监测的对象物理量以符号 Y 表示；传感部件的输出电压信号用符号 X 表示，该电压经模数转换器（ADC）成为数字编码，并能被微处理器程序所读取和处理，获得信号 $\hat{Y}$ 作为 Y 的读数（监测模块对 Y 的估测值）。

2.2 成本计算

为评估和比较不同的校准方案，特制定以下成本计算规则。

- 单点定标误差成本

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

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

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

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。

$$C=\frac{1}{M}\sum_{i=1}^{M}S_i \quad (3)$$

总成本较低的校准方案，认定为较优方案。

3. 取点与拟合方法

3.1 遗传算法简介

自从地球上有了生命以来，为了适应生存环境，便开始了漫长的进化过程。达尔文提出进化论学说，阐述了生物进化发展来源于三种运动：遗传、变异和选择。生物就是在遗传、变异的综合过程中不断地由低级向高级进化，而选择是通过遗传和变异起作用的，变异为选择提供新的基因，遗传巩固与保存选择的基因，选择则能控制遗传与变异的方向，使之朝着适应环境的方向发展。

仿生学根据进化理论学说，发展处适应现实世界复杂问题优化的模拟进化算法。通过对

自然界生物体进化过程的粗糙简化，来实现问题优化的效果。遗传算法主要包括选择、交叉和变异三个过程，分别以选择算子、交叉算子和变异算子进行实现。

1. 选择算子

从群体中选择优胜的个体，淘汰劣质个体的操作叫选择。选择算子有时又称为再生算子(reproduction operator)。选择的目的是把优化的个体(或解)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，目前常用的选择算子有以下几种：适应度比例方法、随机遍历抽样法、局部选择法。

2. 交叉算子

交叉算子根据交叉率将种群中的两个个体随机地交换某些基因，能够产生新的基因组合，期望将有益基因组合在一起。

3. 变异算子

遗传算法引入变异的目的有两个:一是使遗传算法具有局部的随机搜索能力。当遗传算法通过交叉算子已接近最优解邻域时，利用变异算子的这种局部随机搜索能力可以加速向最优解收敛。显然，此种情况下的变异概率应取较小值，否则接近最优解的积木块会因变异而遭到破坏。二是使遗传算法可维持群体多样性，以防止出现未成熟收敛现象。此时收敛概率应取较大值。

3.2 初期算法介绍及分析

初期算法采用的是十进制表示法，即，种群为一个二维数组，每一行是一组基因，每个基因点的范围是【1,51】，每组个体的个数在程序开始时设定。由于十进制表示法在后期交叉过程中，可能出现重复的基因点，因此在生成母本时，基因组数（m）与每组基因点个数（n）需要满足关系：m*n<=51，且每个母本没有共同的基因点。（这是十进制算法的缺陷之一，该缺陷在后期改进为二进制算法时得到了修补）。

3.2.1 流程图

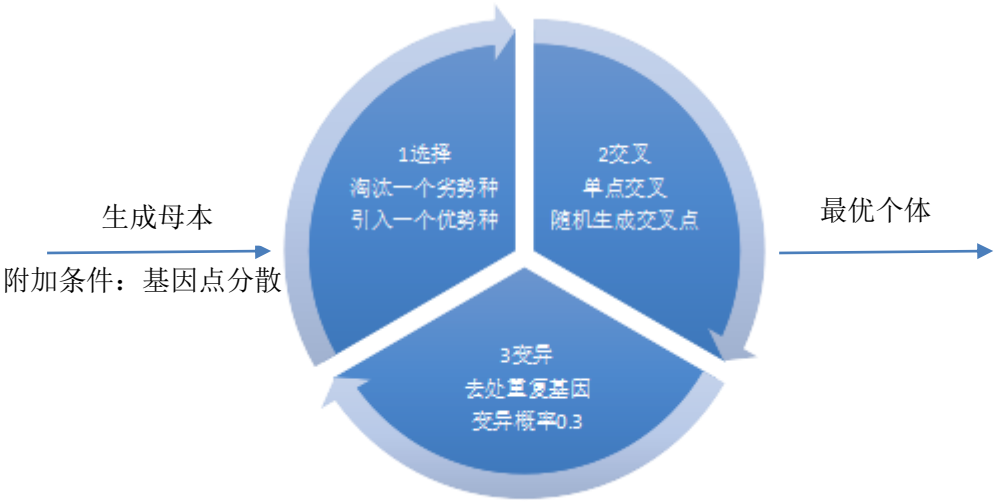

图 3-1 初期遗传算法流程图

3.2.2 生成母本

以每组基因 6 个基因点为例，（即 n=6，m=8）首先我们分析，当基因点分散时，最可能产生较好的拟合效果，因此，在设置母本时，人为将每个基因组中的基因点分散开，并满足上述母本条件（具体代码见附录：初期代码-createParents_fun.m）

3.2.3 基因选择

以指定的拟合方式，并根据每个基因组进行数据拟合，求出每个基因组的总体成本（具体拟合方式会在后面介绍），由于基因组个数的限制，所以我们决定每一代繁殖之前，将成本最大的基因组由成本最小的基因组代替，这样在交叉过程前，该种群中至少有两个完全相同的基因组，以这种方式进行基因组的选择和替换（具体代码见附录：初期代码-main.m 中的“%选择”部分）

3.2.4 基因交叉

在选择过程完成后，由于存在两个完全相同的基因组，那么令这两个基因组进行交叉就是没有意义的操作，因此我们需要进行预处理，将两个基因组尽可能的分散。对于处理后的基因组，我们采取单点交叉的方式，对每一对基因的交叉先随机产生交叉点，然后将两个基因组的后半基因点交换。（具体代码见附录：初期代码-main.m 中“%交叉”部分）

3.2.5 基因变异

随之遗传代数的增加，会有一部分基因点被保存，加上由于变异的过程可能使基因点的位置改变，因此在交叉后的种群中，每个基因组中可能包含相同的基因点。我们首先需要进行去重操作，对于每个基因组的基因点进行遍历（基因点是顺序排列），当发现重复时，随机产生过一个【2.50】的基因点进行代替。

由于十进制表示法基因组数少的缺陷，我们需要加大变异率来防止过早的收敛，所以我们预定的变异率为 1/3，变异时，先随机产生变异的个数，再产生变异点的位置，最后随机产生变异后的基因（具体代码见附录：初期代码-main.m 中“%变异”部分）

3.2.6 预置数据

经过我们大量的测试，我们发现当每组基因有 6~7 个基因点时，可以产生较小的拟合成本，则

遗传代数=50；　　每组基因点个数 n=6(7)；　基因组数 m=8(7)；　　变异率=0.3；

人为使每组基因点均匀分布与【1,51】

3.2.7 算法分析

好处：数据存储少，运行速度快。

缺陷：基因组个数少，收敛速度过快；基因更新比较依赖于变异操作；带有人为的附加条件（母本基因点分散）

十进制表示法在基因的交叉和变异操作中比较有优势，因为数据个数少，可以确保交叉操作的有效性，及变异操作的简便性。所以我们在后期进行了算法的改进，对于数据的存储采用二进制的表示法。

3.3 后期算法介绍及分析

后期改进算法采用二进制的存储方法，可以解除对基因组个数的限制，避免了收敛过速的问题，在二进制算法中由于基因组的更新不再过分依赖于变异操作，交叉操作的重要性比重提高，因此我们可以降低变异率，更好的模拟大自然环境。

同时，我们在采取十进制算法中发现基因交叉和变异过程以十进制的表示法会比二进制方便，所以我们在改进算法的基因交叉和变异操作中，现将二进制转化为十进制表示法，操作后再重新转化为二进制进行存储。

3.3.1 流程图

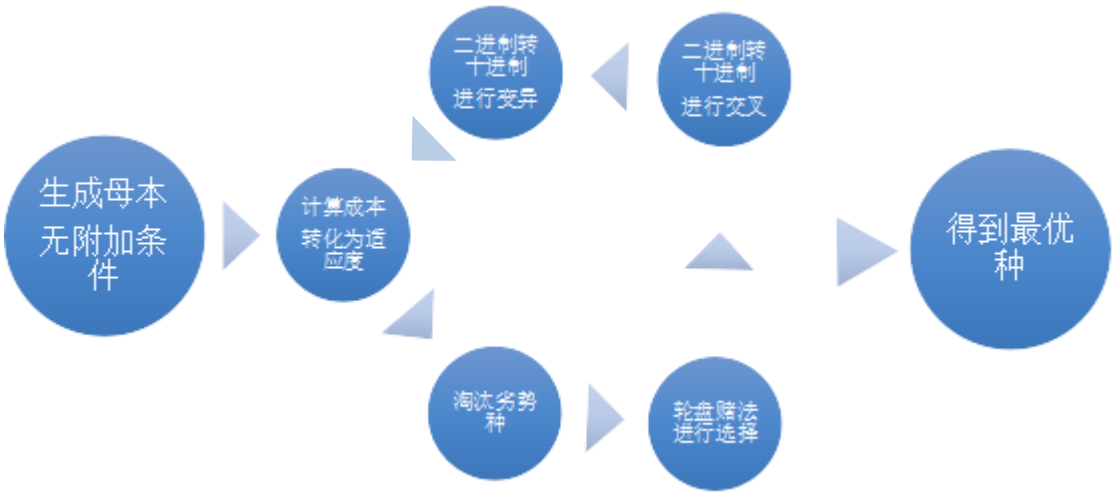

图 3-2 后期遗传算法流程图

3.3.2 生成母本

每个基因组由 51 个 0/1 数字表示，首先设置基因组个数 N，对于每个基因组，随机生成 51 个 0/1 数字（由于我们的前期操作，已知当基因点个数为 6~7 个时拟合成本最小，所以我们规定每一组随机生成的基因中有 6~7 个 1，其余为 0）最后，我们从保证数据测量的全面性考虑，在第 1~3 位和 49~51 位分别至少存在 1 个 1。（具体代码见附录：后期代码-main.m 中“%生成母本”部分）

3.3.3 基因选择

首先根据所选拟合方法进行数据拟合并计算拟合成本，对每个基因组的拟合成本进行保存，记录本代遗传的最小成本。并将成本转化成适应度（adaptProb=1/(a*cost+b)）

淘汰劣势种，当遗传代数大于 15 代后，之后的每次遗传将淘汰一个成本最大的个体，在前 10 次遗传中得到的最小成本中随机选取一个，进行替代。

采用轮盘赌的方法，首先由于其实基因组是随机选取的，因此前几代的拟合成本会有较大浮动，而我们在前几次迭代中应该保持较小的选择压力，以防止收敛过速，在迭代后期，我们则需要保持较高的选择压力，加速最优解的收敛，为了满足以上条件，我们借鉴 Boltzmann 选择法，借鉴退火算法，对适应度函数进行如下变换[2]

$$\delta(\mathrm{i}) = e^{\frac{adaptProb(i)}{H}} \tag{3-1}$$

其中 H 为随迭代次数(t)增加而减小的量，我们规定 H=80/t，对适应度进行变化后，定义一个轮盘 table 数组

$$s = \sum_{k=1}^{N} \delta(k) \tag{3-2}$$

$$\mathrm{table}(\mathrm{i}) = \sum_{k=1}^{i} \frac{\delta(k)}{s} \tag{3-3}$$

产生 N 次（0，1）的随机数，根据随机数所处区间，选择新的基因组。

对 Boltzmann 变换方法有效性的检验：我们分别绘制了在适应度函数在变换前和变换后的两组试验中，每次迭代后已知的最小点成本与迭代次数的图像（共迭代 100 次）

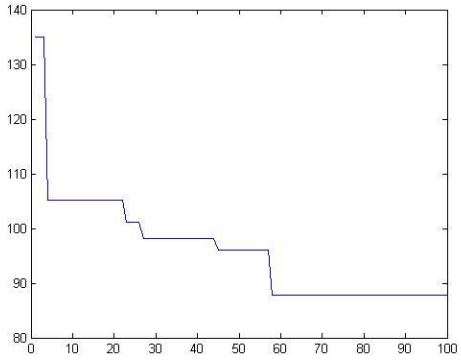

图 3-3-1.Boltzmann 变换后

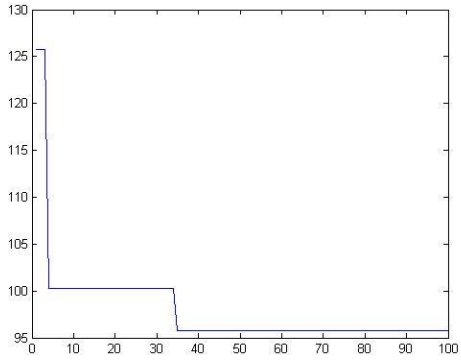

图 3-3-2.Boltzmann 变换前

由两个图形的比较，我们发现通过 Boltzmann 变换确实可以改变选择压力，使最终结果趋向最优解。[2]

最后，进行优势种的保留，在 N 个基因组中随机抽取一个替换成本次迭代中拟合成本最低的基因组。（具体代码见附录：后期代码-main.m 中“&选择”部分）

3.3.4 基因交叉

我们定义交叉概率 breedProb=0.9，在交叉操作中，由前面所述，先将二进制转化成十进制，根据两个交叉基因组的长度来确定交叉点取值范围，我们采取单点或多点交叉的方式，随机生成交叉点个数，再产生交叉点位置进行基因交叉操作。

需要注意的是，由于二进制交叉不需要考虑重复点的问题，所以在交叉后，可能使某一个基因组长度小于 3，致使无法拟合，所以我们规定，当基因组长度小于 5 时，随机加入新基因。在上述操作全部完成后，再将十进制转化为二进制重新储存到原基因组中。（具体代码见附录：后期代码-main.m 中“%交叉”部分）

3.3.5 基因变异

同样，我们采取十进制的变异方法，具体操作与初期基因变异类似，主要区别在于变异概率的大小以及十进制和二进制的转化（具体代码见附录：后期代码-main.m 中“%变异”部分）

3.3.6 预置数据

遗传代数=80~100；　基因组个数 N=50；　基因组中基因点个数=6~7；

交叉概率 breedProb=0.9；变异概率 variationProb=0.05；

3.3.7 算法分析

优点：改进的二进制算法综合了原本二进制方法和十进制方法的优点，既消除了基因组个数的限制，也简化了交叉和变异操作。

缺陷：运行速率慢，存储量大。

3.4 拟合方式与结果讨论

我们采取的拟合方法主要为多项式拟合，spline 拟合，和 pchip 拟合，下面我们将展示初期算法和后期算法在不同拟合方法得出的结果。

3.4.1 多项式拟合

我们尝试了最高次幂分别为 3~7 的多项式拟合，采用 Matlab 中 polyfit 函数进行拟合，

最终得到的最小成本为 112.1170，选取的基因点为 2　　6　　19　　30　　43　　50。具体结果请看下表（n 为最高次幂）

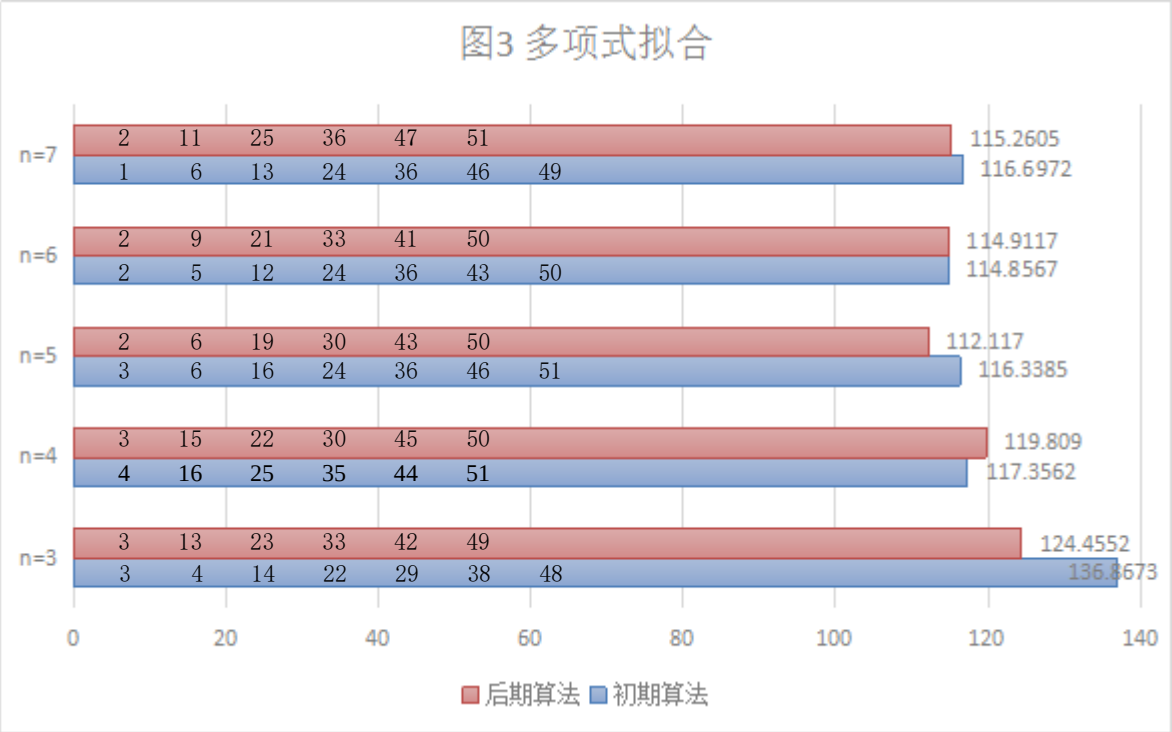

图 3-4-1 多项式拟合结果

我们分别用初期十进制算法和后期改进的二进制算法进行测量，我们发现多项式拟合方法的成本还是偏高，由于两个算法都是在最高次幂为 6 时取到最小值，当 n>6 时，成本又会增加，这种拟合方式差强人意，所以我们又采取了 spline 和 pchip 两种拟合方式进行比较。

3.4.2 spline 拟合

针对 spline 拟合方式，我们以两种算法进行了多次试验，下图为最小的 4 组数据。

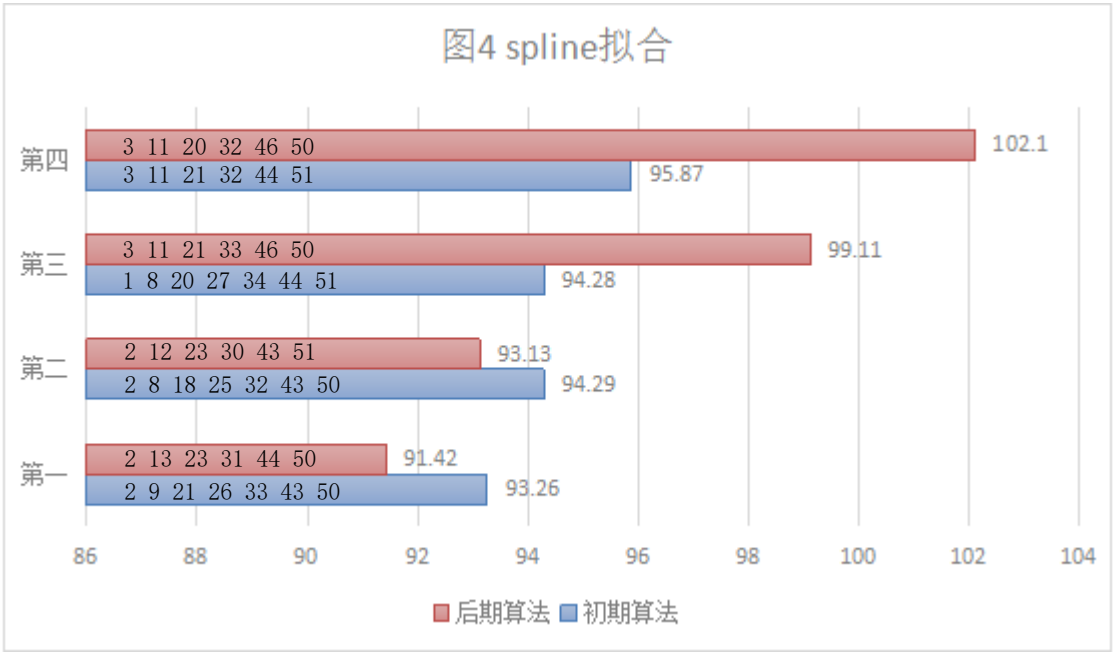

图 3-4-2 spline 拟合结果

3.4.3 pchip 拟合

针对 pchip 拟合方式，我们仍对两种算法进行了大量试验，其中成本最小的 4 组数据如下

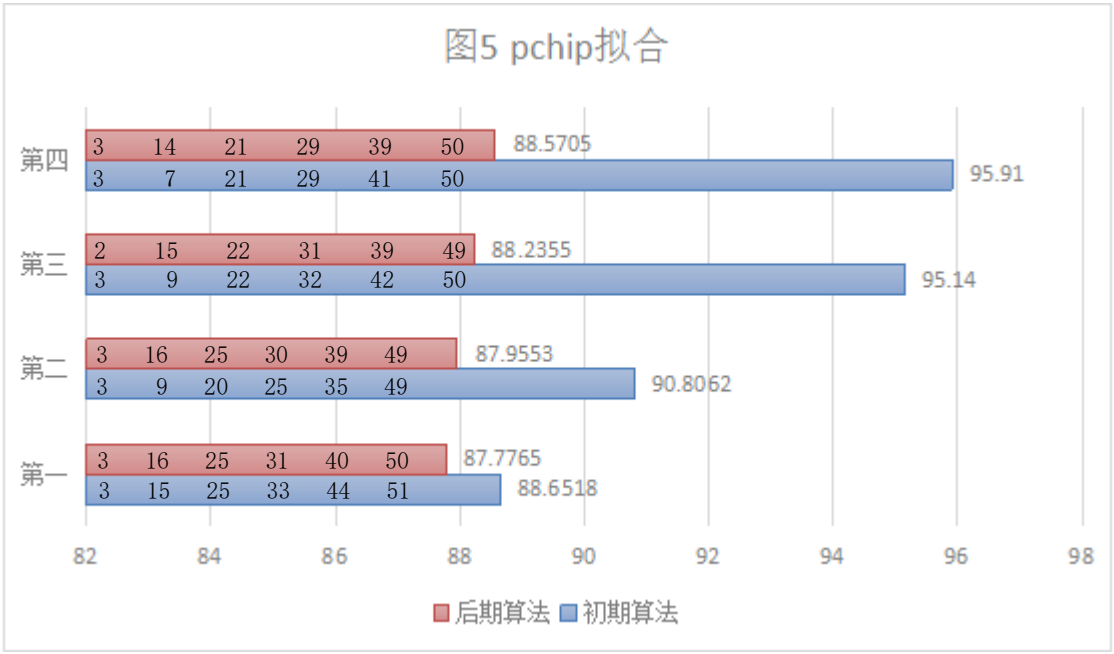

图 3-4-3 pchip 拟合结果

通过 pchip 拟合的试验，我们发现后期算法对最优解有较好的选择性，而初期算法容易过早收敛而得到次优解。所以整体来看后期算法较优于初期算法。

从拟合方法来看，最优解是在 pchip 拟合中得到的，而多项式拟合得到的成本最高，因此最终结果我们选取后期算法通过 pchip 拟合得到的最低成本。

4. 结果分析

最终我们采用了二进制存储、十进制交叉变异、轮盘赌法选择为核心的遗传算法，并使用 pchip 拟合，得到的最佳结果为：

基因点：3　16　25　31　40　50

最小成本：87.7765

5. 思考感悟

在《统计推断》这门课程的学习中，我们学到了编程方法在实际生产成本估计中的应用，收获很大。最初上理论课的时候，我们根本不知道这节课的内容和目的，也不了解成本计算、拟合等基本操作，基本是一头雾水。但在参考往届同学报告的基础上，我们逐步理解了课题内容，并尝试编写了初步的遗传算法代码。

在编写代码的过程中，我们也遇到了很多问题。我们最初版本的遗传算法代码比较简单，且有人为的预置条件，存在收敛速度过快、变异率过大等问题。在老师的指导下，我们改进了代码，由十进制存储改为二进制，减小变异率、增大交叉率，并采用了轮盘赌方法等方式进行筛选，最终使结果成本得到了极大改进。

在此感谢指导老师的帮助，也感谢往届学长们的报告给我们的借鉴作用，更要感谢组员的辛勤劳动。希望我们在今后的学习中勤于思考、再接再厉！

6. 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] 蒋腾旭，谢枫. 遗传算法中防止早熟收敛的几种措施[J].计算机与现代化，2006(12)：54-56

7. 附录

7.1 初期代码

初期代码-main.m

```
data=xlsread('C:\Users\Victor\Desktop\20150915dataform.xlt');

%前期预置数据
syms t m n Rmax Rmin rmin;
T=50;n=6;  %n>=7
m=floor(50/n);r=[]; %r成本
rmin=500;minParents=[];

%创建父辈
parents=creatParents_fun(m,n); %  m*(n-2)<50

%开始迭代
for times=(1:T)

%选择
    for i=(1:m)
        totalCosts=0;
```

```
    for j=(1:2:800)
        xdata=data(j,:);
        ydata=data(j+1,:);
        xtry=xdata(parents(i,:));
        ytry=ydata(parents(i,:));
        ytheory=spline(xtry,ytry,xdata);
%    pp=6;
%    P=polyfit(xtry,ytry,pp);
%   ytheory=polyval(P,xdata);
        totalCosts=totalCosts+accountCost(ydata,ytheory,n);
    end
    r(i)=totalCosts/400;
    end

    times
    rmin(times)=min(r)
    pos=find(r==rmin(times),1);
    chozenGenes=parents(pos,:)
    minParents=[minParents;chozenGenes];

    clear xtry ytry Cell ytheory;

    minr=find(r==min(r),1);
    maxr=find(r==max(r),1);
    if minr==maxr
        maxr=minr+1;
    end
    parents(maxr,:)=parents(minr,:);

%交叉
    if(minr > maxr)
        tmp=minr;
        minr=maxr;
        maxr=tmp;
    end
    if(minr==1 && maxr==m)
        tmp=parents(maxr,:);
        parents(maxr,:)=parents(maxr-1,:);
        parents(maxr-1,:)=tmp;
    end

    t=minr;
    while (t<=minr && t>0)
        tmp=parents(t,:);
```

```
        pret=mod_fun(t-1,m);
        point=randi([3,n-2],1);
        parents(t,(point:n))=parents(pret,(point:n));
        parents(pret,(point:n))=tmp(point:n);
        t=t-2;
    end
    t=mod_fun(t,m);
    while(t>=minr+2)
        tmp=parents(t,:);
        pret=mod_fun(t-1,m);
        point=randi([3,n-2],1);
        parents(t,(point:n))=parents(pret,(point:n));
        parents(pret,(point:n))=tmp(point:n);
        t=t-2;
    end


%变异
    for k=[1:m]
        parents(k,:)=sort(parents(k,:));   %去重
        for i=[1:n-1]
            if(parents(k,i)==parents(k,i+1))
                j=randi([2,50],1);
                while(ismember(j,parents(k,:)))
                    j=randi([2,50],1);
                end
                parents(k,i)=j;
            end
        end

        random=rand(1);
        if(random >0.3)
            continue;
        end
        varyNum=randi([0,floor(n/3)],1); %随机产生变异个数
        g=randperm(n);
        g=g(1:varyNum); %随机产生变异点位置
        syms h;
        h=1;
        while(h<varyNum)
            varypoint=randi([1,51],1); %随机产生新基因点
            if ~ismember(varypoint,parents(k,:)) %判断是否重复
                h=h+1;
                parents(k,g(h))=varypoint;
```

```
            end
        end
        parents(k,:)=sort(parents(k,:));
    end

    clear h g varypoint varyNum;
end
finalmin=min(rmin)
pos=find(rmin==finalmin);
minParents(pos,:)
x=[1:T];
plot(x,rmin);
```

初期代码-createParents.m

```
function p=creatParents_fun(m,n)
%´´½¨m×é¸ö±²µã¼¯£¬Ã¿×éÓÐn¸öµã

for i=(1:floor(m/3))
    p(i,:)=[1,[i+1:m:m*(n-2)+1],49];
end
for i=(floor(m/3)+1:floor(2*m/3))
    p(i,:)=[2,[i+1:m:m*(n-2)+1],50];
end
for i=(floor(2*m/3)+1:m)
    p(i,:)=[3,[i+1:m:m*(n-2)+1],51];
end
```

初期代码-计算成本

```
function cost=accountCost(y1,ytheory,n)
%¼ÆËã×Ü¹²µÄÎó²î³É±¾ ·µ»ØÖµdouble
everyCost=errorCost_fun(y1,ytheory);
c=0;
for i=1:51
    c=c+everyCost(i);
end
cost=c+12*n;

function e=errorCost_fun(y1,ytheory)
%Ã¿¸öµãµÄÎó²î³É±¾ ·µ»ØÖµÊý×é
e=abs(y1-ytheory);
for i=[1:51]
    if e(i)<=0.4
        e(i)=0;
    elseif e(i)<=0.6
```

```
        e(i)=0.1;
    elseif e(i)<=0.8
        e(i)=0.7;
    elseif e(i)<=1
        e(i)=0.9;
    elseif e(i)<=2
        e(i)=1.5;
    elseif e(i)<=3
        e(i)=6;
    elseif e(i)<=5
        e(i)=12;
    else e(i)=25;
    end

end
```

初期代码-小工具

```
function mm=mod_fun(x,m)

if(x==0) mm=m;
elseif(x==-1) mm=m-1;
elseif(x>=m+1) mm=x-m;
else mm=x;
end
```

7.2 后期代码

后期代码-main.m

```
%预置数据
T=100;%遗传代数
N=50;%种群数量
a=1;b=0; % adaptProb=1/(a*allCost+b);
prioNum=1;
deadNum=1;
rmin=[inf];
breedProb=0.9;
variationProb=0.05;
minGenes=ones(1,51);
chozenGenes=[];
community=zeros(N,51);

%创建母本
for i=(1:N)
    num=randi([4,4],1);
    parents=randperm(45)+3;
```

```
    parents=parents(1:num);

    pos1=randi([1,3],1);
    pos2=randi([49,51],1);
    parents=[pos1,parents,pos2];

    community(i,parents)=1;
end
clear num parents pos1 pos2;

%开始迭代
for t=(1:T)

 %选择

    allCost=genesCost(community,N);  %allCost 1*N
    minCost(t)=min(allCost);
    minPos=find(allCost==minCost(t));

    t
    chozenGenes{t}=find(community(minPos,:)==1);
    chozenGenes{t}
    minCost(t)

    if(rmin(t)>minCost(t))
        rmin(t+1)=minCost(t);
        minGenes=chozenGenes{t};
    else rmin(t+1)=rmin(t);
    end

    if(t>15)
        c=sort(allCost,'descend');          %淘汰劣势种，变为优势种
        c=c(1:deadNum);
        for i=(1:deadNum)
            pos=find(allCost==c(i));
            community(pos,:)=zeros(1,51);
                p=randi([t-10,t],1);
                parents=minGenes;
                community(pos,parents)=1;
                allCost(pos)=rmin(p+1);
        end
    end

    H=80/t;
```

```
    adaptProb=exp(allCost./H);       %Boltzmann变换
    adaptProb=(1./(a.*allCost+b));

    sumOfProb=sum(adaptProb);
    table=[0];                                    %生成轮盘
    for i=(1:N)
        table=[table,table(i)+adaptProb(i)/sumOfProb];
    end

    tmp=[];

    for i=(1:N)          %轮盘赌法进行选择
        random=rand(1);
        pos=find(table<random);
        p=find(pos,1,'last');
        pos=pos(p);
        tmp=[tmp;community(pos,:)];
    end

    c=sort(allCost);          %精英选择
    c=c(1:prioNum);
    for i=(1:prioNum)
        pos=randi([1,t],1);
        p=randi([1,N],1);
        parents=chozenGenes{pos};
        tmp(p,:)=zeros(1,51);
        tmp(p,parents)=1;
    end

    community=tmp;
    clear pos tmp p parents adaptProb allCost;

%交叉
    n=1;
    while(n<N)
      prob=rand(1);
      if(prob>breedProb)
            n=n+1;
      else
        p1=find(community(n,:)==1);  l1=sum(community(n,:));
        p2=find(community(n+1,:)==1); l2=sum(community(n+1,:));
        if(l1 > l2)
            l=l2;
        else l=l1;
```

```
        end
        num=randi([1,l-4],1);
        pos=randperm(l-2)+1;
        pos=pos(1:num);
        pos=sort(pos);

        p=interbreed(p1,p2,pos,num,l1,l2);
        p1=p{1};
        p2=p{2};

        community(n,:)=zeros(1,51); community(n+1,:)=zeros(1,51);
        community(n,p1)=1;
        community(n+1,p2)=1;
        for i=n:n+1
            if(sum(community(i,:))<=5)
                pos=randi([4,48],1);
                while(ismember(pos,p1)==1)
                    pos=randi([4,48],1);
                end
                community(i,pos)=1;
            end
        end
        n=n+2;
      end
    end
    clear pos tmp p1 p2 p prob;

%变异
    for i=(1:N)
        tmp=rand(1);
        if(t>30)
            variationProb=0.1;
        end
        if(tmp>variationProb)
            continue;
        end
        Genes=find(community(i,:)==1);
        s=sum(community(i,:));
        Genes(1)=randi([1,3],1);
        Genes(s)=randi([49,51],1);
        community(i,:)=zeros(1,51);
        num=randi([2,s-2],1);  %随机产生变异个数
        g=randperm(s-2)+1;
        g=g(1:num); %随机产生变异点位置
```

```
        h=1;
        while(h<=num)
            varypoint=randi([4,48],1); %随机产生新基因点
            if ~ismember(varypoint,Genes) %判断是否重复
                Genes(g(h))=varypoint;
                h=h+1;
            end
        end
        community(i,Genes)=1;
    end
    clear tmp num pos Genes g s varypoint h;
end

rmin(T)
minGenes
t=[1:T];
plot(t,minCost);
```

后期代码-计算成本

```
function Costs=genesCost(community,N)

data=xlsread('C:\Users\Victor\Desktop\20150915dataform.xlt');
parents=[];
Costs=[];
totalCosts=0;

for i=(1:N)
    Genes=community(i,:);
    parents=find(Genes==1);
    for t=(1:2:800)
        xdata=data(t,:);
        ydata=data(t+1,:);
        xtry=xdata(parents);
        ytry=ydata(parents);
        ytheory=spline(xtry,ytry,xdata);
      %     pp=3;
      %  P=polyfit(xtry,ytry,pp);
      %  ytheory=polyval(P,xdata);
        totalCosts=totalCosts+accountCost(ydata,ytheory,sum(Genes));
    end
    Costs=[Costs,totalCosts/400];
end
end
```

mycurvefitting.m

```
function y1 = mycurvefitting( x_premea,y0_premea )

x=[5.0:0.1:10.0];

y1=interp1(x_premea,y0_premea,x,'pchip');


end
```

test_er_answer.m

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

my_answer=[3    16    25    31    40    50];%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
minput=xlsread('C:\Users\Victor\Desktop\20150915dataform.xlt');
[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

% 显示结果
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);
```