统计推断在数模转换系统中的应用

组号：39 姓名 李晔璇 学号 5130309189，姓名 贾佳璐 学号 5130309190

摘要：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。

关键词：统计推断，线性拟合，样条插值，ADC，启发式搜索算法，遗传算法

1 引言

ADC，Analog-to-Digital Converter 的缩写，指模/数转换器或者模拟/数字转换器。是指将连续变量的模拟信号转换为离散的数字信号的器件。真实世界的模拟信号，例如温度、压力、声音或者图像等，需要转换成更容易储存、处理和发射的数字形式。模/数转换器可以实现这个功能，在各种不同的产品中都可以找到它的身影。

1.1 模型

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）。

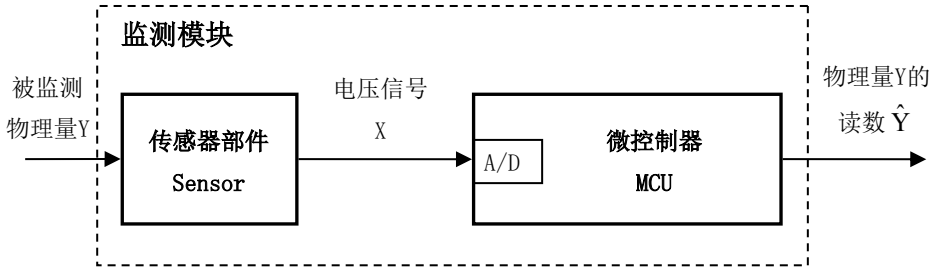


图 1-1 监测模块组成框图

我们把 X 为某个特定值时，针对相应 Y 的数值进行的测量过程，称作一次测定。测定需要付出一定成本。Sensor 特性呈非线性，不同 Sensor 个体间，特征的一致性不够强。由于数据量很大，我们要尽可能减少测定的次数，降低定标的成本，但也要保证准确性。这样可以由测定点的数据来推知其他点的数据。

1.2 传感部件特性

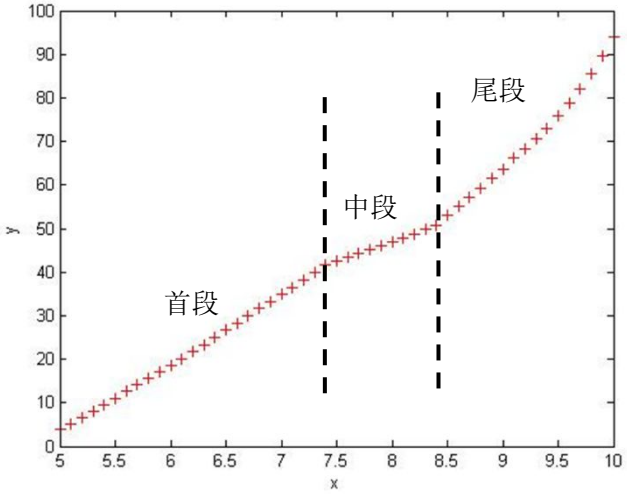


图 1-2 传感特性图示

一个传感部件个体的输入输出特性大致如图 2 所示，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。

为进一步说明情况，图 3 对比展示了四个不同样品个体的特性曲线图示。

1.3 标准样本数据库

前期已经通过试验性小批量生产，制造了一批传感部件样品，并通过实验测定了每个样品的特性数值。这可以作为本课题的统计学研究样本。数据被绘制成表格，称为本课题的“标准样本数据库”。

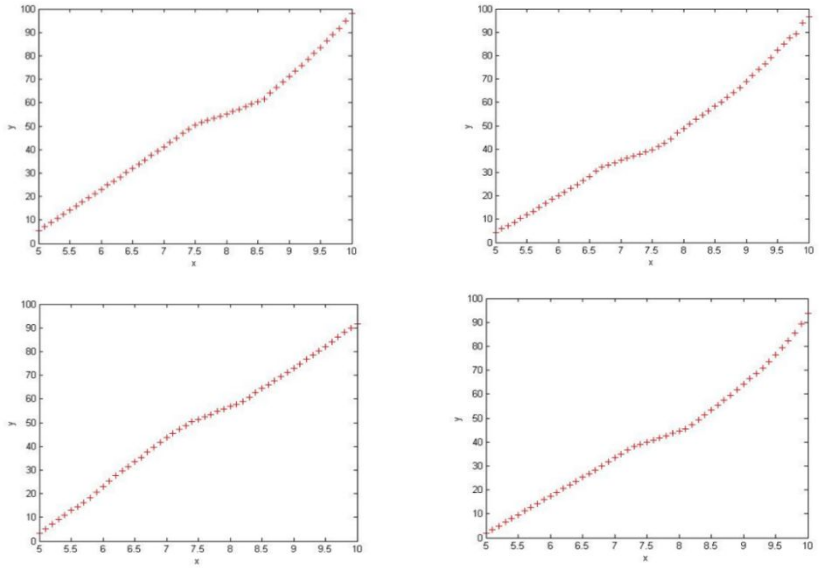


图 1-3 四个不同样本个体特性图示对比

2 选取最优拟合方式

2.1 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。

（1）单点定标误差成本

$$s_{i,j}=\begin{cases}0 & \text{if } \left|\hat{y}_{i,j}-y_{i,j}\right|\le 1\\ 1 & \text{if } 1<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 2\\ 2 & \text{if } 2<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 3\\ 4 & \text{if } 3<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 5\\ 10 & \text{if } \left|\hat{y}_{i,j}-y_{i,j}\right|>5\end{cases} \quad (1)$$

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。

（2）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=20。

（3）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。

（4）校准方案总体成本

按式（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。

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

总体成本较低的校准方案，认定为较优方案[1]。

2.2 多项式拟合分析

2.2.1 一段多次多项式拟合

多项式拟合是通过多项式来拟合曲线，使得残差平方和最小的方式，伪代码为：

（1）产生一组数据中 n 个随机特征点；

（2）通过 matlab 多项式拟合函数 p=polyfit（x,y,k）得到 k 次拟合曲线的系数 s_i；

（3）通过函数 y2=polyval（p,x1）获取全部数据的拟合值，再求评价分数（去掉每组的第一个和最后一个数据）；

（4）令 m=3、4、5，重复上述过程[2]。

2.2.2 三段多次多项式拟合

伪代码：

（1）获取一组数据（需判断此组数据未经使用）的 9 个特征点，且按照 3 个不同的分

段分别放在三个数组中；

（2）通过 matlab 多项式拟合函数 p=polyfit（x,y,k）得到三段 k 次拟合曲线的系数 s_i；

（3）通过函数 yy1=polyval（p,x1）获取全部数据的拟合值，再求评价分数；

（4）令 m=3、4、5，重复上述过程[3]。

2.3 插值拟合分析

由于多项式拟合拟合次数达到五次前成本随着次数的增大而减小，但是大于五次时拟合出的曲线发生严重扭曲，所以只能选择五次拟合但明显成本还是不够低，所以选择了插值拟合分析。

观察到数据的中间部分线性关系比较明显，两端线性关系比较弱，所以可以用三次样条插值法来进行拟合。

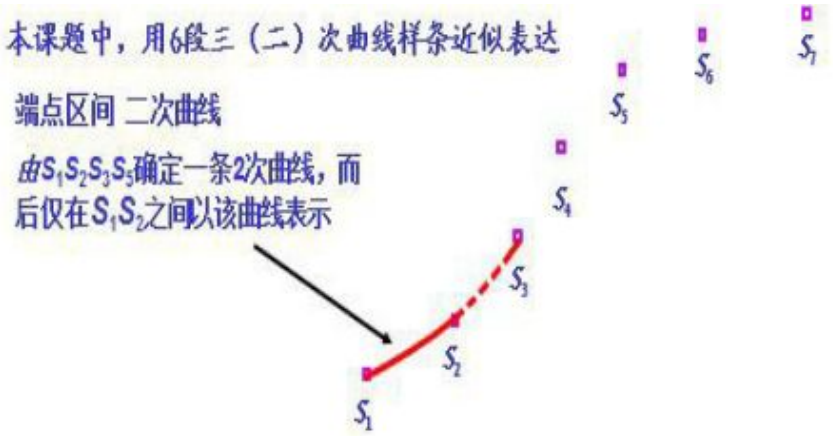


图 2-1 两端拟合方式

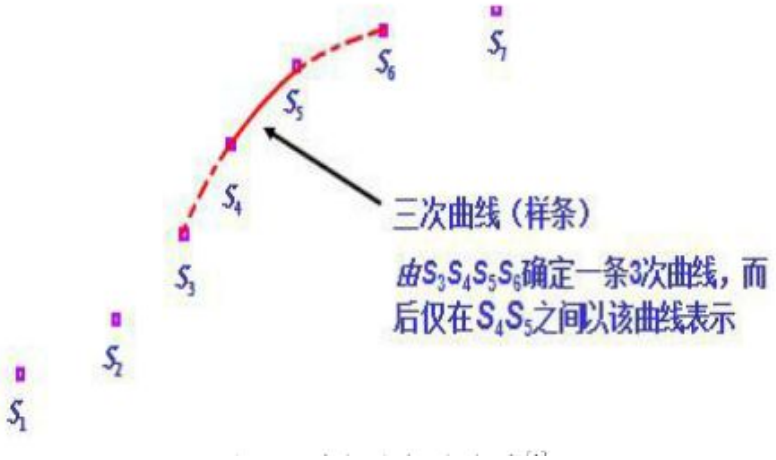


图 2-2 中间段拟合方式

用 matlab 实现插值拟合，这次实验中我们使用 interp1(x0,y0,xi,‘spline’)来实现拟合，伪代码为：

（1）产生一组数据中 n 个随机特征点；

（2）通过 matlab 函数 y_i=(x,y,x1 ,‘spline’)获取全部数据的拟合值，再求评价分数（去掉每组的第一个和最后一个数据）；

（3）处理所有数据，最后得出平均评价分数。

通过比较可以得出三次样条插值拟合方式更好，虽然耗时较长，计算更繁琐，但是其平均得分和样本都优于多项式拟合。

2.4 选取特征点

想得到更为精确的拟合结果，需要选择更优的拟合方式。在实验中，我们采用多项式拟合和插值拟合，我们直接使用 matlab 提供的拟合函数进行拟合，舍去了一些多余的不必要的点。再将数据大致分为若干段，在每一段中，采取随机数的方法各选取一个数字作为特征点下标，使用此组下标作为所有样本数据拟合所用特征点下标。

观察到第一个与最后一个数据点偏差较大，且为了方便计算，以舍去这两个数据，转化为在 49 个数据点中找起个特征值。如果采用将所有可能性计算的话，需要计算 C_{49}^{7}=85900584 次。数据量太大无法通过 PC 机实现。考虑到特征点分布不会太集中而是比较均匀，所以可以将剩下的 49 个数据平均分为 7 组，在每组中选取特征点，这样大大减少了计算量。

表 2-2 观测点选取范围

观测点序号	选取区间
1	[2,8]
2	[9,15]
3	[16,22]
4	[23,29]
5	[30,36]
6	[37,43]
7	[44,50]

3 用遗传算法寻找最优七点组合

3.1 原理概述

在选取最优 7 点组合的过程中，我们使用的是遗传算法（GA，Genetic Algorithm），它是一种“放生算法”，原理是把一个“生物个体”对应到一个可能的解答，需要把解答改写（编码）成一维数组形式（对生物基因染色体的仿生），计算适应度函数（每个个体“适者生存”的程度），然后模拟自然选择的过程，适者生存，不适者淘汰。其实现过程还包括了模拟生物的交配繁殖、基因突变、世代更替。这样一代一代地进化，最后就会收敛到最适应环境的一个“生物个体”上，它就是问题的最优解。遗传算法的特点包括对问题参数的编码组进行计算，而不是针对参数本身；搜索是从问题解的编码组开始搜素、而不是从单个解开始；使用目标函数值（适应度）这一信息进行搜索，而不需导数等其他信息；算法使用的选择、交叉、变异这三个算子都是随机操作，而不是确定规则。

3.2 原理实现

3.2.1 遗传算法的一般步骤

（1）初始化：随机生成 M 个个体作为初始群体 P。

（2）个体评价：计算群体 P 中各个个体 P_i 的适应度。

（3）选择运算：选择操作是建立在群体中个体的适应度评估基础上的。选择的目的是生存概率比较高的个体遗传到下一代，而这种遗传是完全遵从概率的。

（4）交叉运算：把两个通过选择算子遗传下来的父代个体的部分结构加以替换重组而生成新个体，这部分是模拟了生物学的基因重组。

（5）变异运算：对群体中的个体串的某些基因座上的基因值作变动。

（6）终止条件判断:若得到的满足终止条件个体，把其作为最终解输出，终止计算。

如不中止，则将变异之后的新群体带回到步骤（2），实现循环。

3.2.2 遗传算法的 Matlab 实现

3.2.2.1 初始群体的生成

初始化群体即为初始选择的七点组合，我们选择了初始化的七点组合一共100个，这些数均由matlab程序按照要求随机得出，存入P中。我们认为使用这些数组可以减少在程序中重新生成100组数据所耗费的时间，而且这些数组是具有普适性和代表性的。

3.2.2.2 适应性值评估检测

个体适应度，即用上述分段三次 Hermite 插值法对个体进行拟合，带入 938 组原始数据，所得评价函数的平均值。用 cost 表示适应度总成本。

3.2.2.3 选择

选择的目的是为了从当前群体个选出优良的个体，使它们有机会作为父代为下一代繁殖子孙。遗传算法通过选择过程体现这一思想，进行选择的原则是适应性强的个体为下一代贡献一个或多个后代的概率大。选择实现了达尔文的适者生存原则。

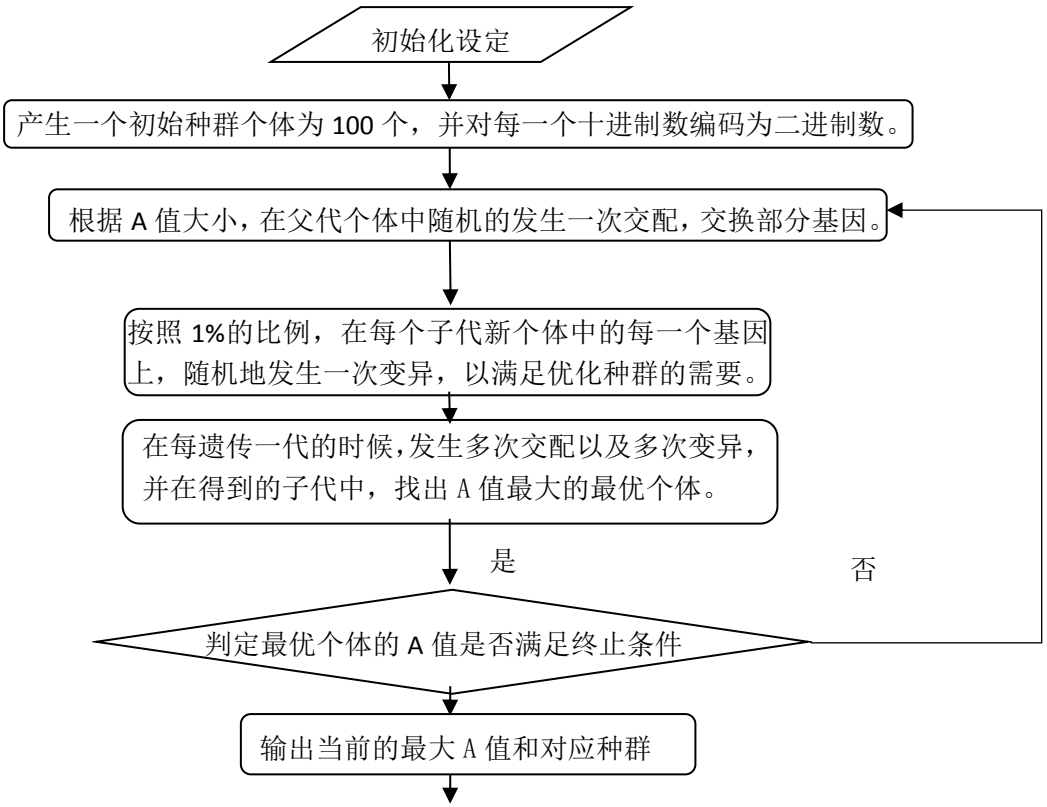
3.2.2.4 交叉

交叉操作是遗传算法中最主要的遗传操作。通过交叉操作可以得到新一代个体，新个体组合了其父辈个体的特性。交叉体现了信息交换的思想。在交叉过程中可能会出现基因重复现象，将其认为基因缺陷，直接淘汰，重新交换，直到每一个染色体内部没有重复的基因。

3.2.2.5 变异

由于初始种群是随机生成的，故可省略随机挑选 10%子体的步骤，直接选取子体进行变异。采用类似于上一步的等概率方法，让 7 个数各有 50%的概率变异，变异方法即使其+1，为了防止残疾的下一代，如果变异后的数超出初始设定的区间，则回到区间开始处第一个数。完成变异后的种群可视作下一个循环的初始种群[4]。

3.3 原理图



结束

4 结论

表 4-1 遗传算法和样条插值所得结果

组号	成本	最优点						
1	93.8561	2	9	20	26	33	43	50
2	93.8561	2	9	20	26	33	43	50
3	93.8998	2	9	20	26	34	43	50
4	93.9968	2	9	20	27	33	43	50
5	93.9968	2	9	20	27	33	43	50

由表 4-1 可以看出，本小组采取的分七段取点三次样条插值和遗传算法充分利用了数据的性质，有效的解决了数据量大，耗时长的问题。同时只要遗传代数足够多，选取的亲本够多，则成本还可以更低。但是我们还应看到，在特征点之外的数据无法对结果产生影响，从而导致数据的遗漏。

5 拓展一 选取更少的特征点

在七个特征点可以满足本课题要求的前提下，我们可以猜想更少的点应该也可以满足要求，所以通过改变程序取点的个数我们做出了 6 个、5 个、4 个特征点的结果。6 个特征点时选取区间重新分为[1,6]、[7,12]、[13,25]、[26,38]、[39,44]、[45,51]，5 个特征点的选取区间为[1,8]、[9,19]、[20,33]、[34,43]、[44,51]，4 个特征点的选取区间为[1,10]、[11,23]、[24,38]、[39,51]。将十次最低成本求平均值后得到下表所示结果。

表 5-1 选取不同特征点的结果

特征点个数	平均最低成本	最优点
6	93.0480	3,12,22,31,42,50
5	103.8962	3,14,26,39,49
4	127.7619	4,15,34,48

随着特征点数目的减少，寻求满足要求的特征点难度明显增大，并且成本逐渐变高，所以低于 7 个特征点的搜索基本不可以是实现，所以我们记录了表 5-1 中的不同特征点的最低成本以及对应的特征点。我们看出描述该问题的特征点可以选择 6 个点，且 6 个点的稍成本低于 7 个点的，所以特征点可以选取 7 个或 6 个，但是过少的特征点会造成拟合结果不能较高程度的代表曲线特征，同时还会增大电脑运算负担。

6 拓展二 用模拟退火算法选取特征点

6.1 原理概述

模拟退火算法是一种通用概率演算法，其模拟的是自然界的降温过程，使得升高温度的粒子群能够在降温过程中更好的趋于能量最低状态。

6.2 算法实现

我们可以利用和遗传算法相同的编码规则和成本计算函数。

（1）设定初温为充分大，我们取定为 300℃，降温系数取为 0.95,终止温度为 1℃。随机产生一个初始解 S_1。

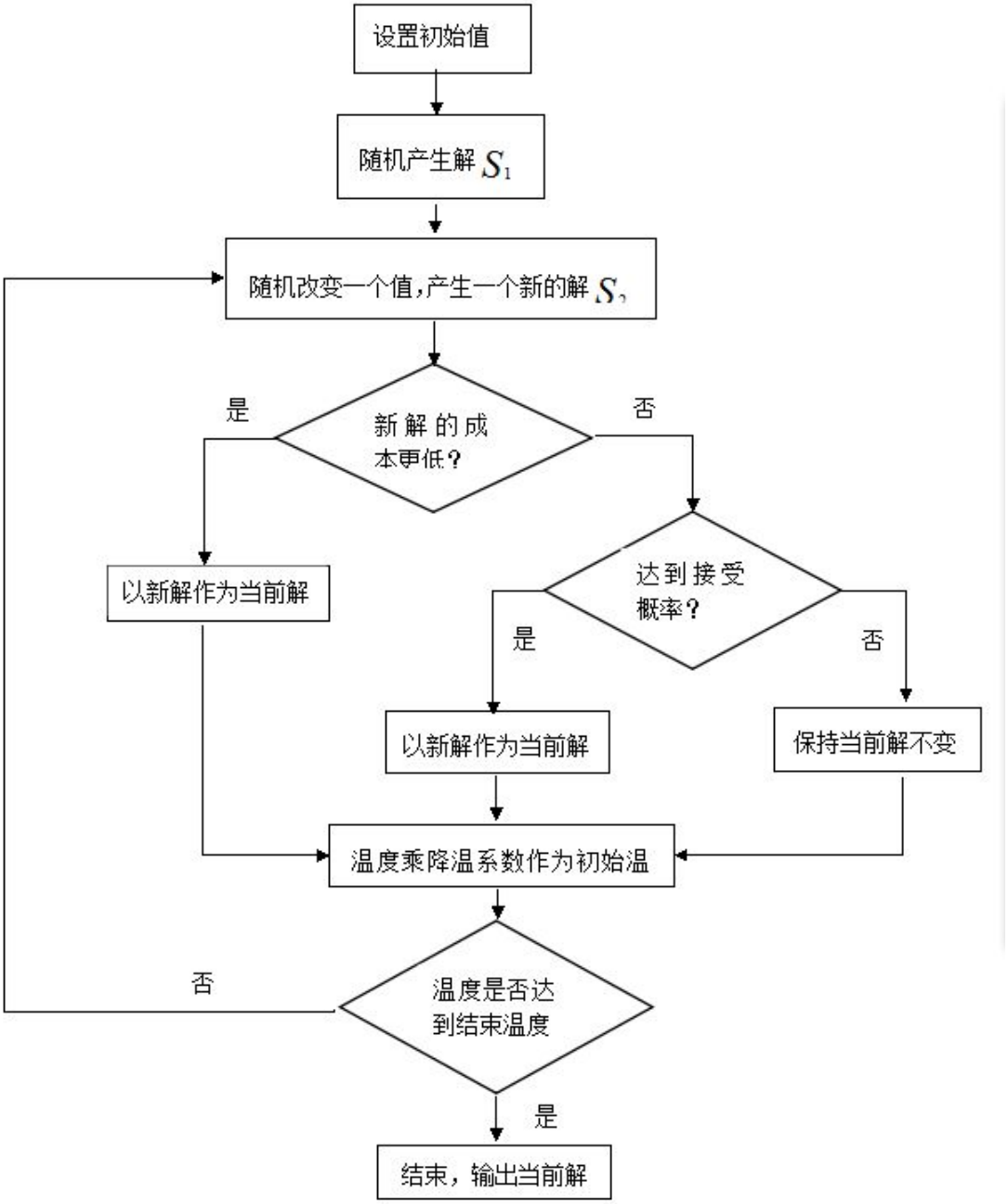
（2）对当前的解使用随机数改变其中一个值，得到一个新解 S_2。

（3）若产生的新解的成本低于原来的解，则新的S_2解作为当前解；否则计算接受概率$e^{-df/T}$，随机产生（0,1）上的均匀分布随机数 rand，若$e^{-df/T}$>rand,则接受S_2作为当前解，否则保持当前解为S_1不变。

（4）温度变为初始温度乘降温系数所得的温度。

（5）重复上面过程，直至温度满足第于 1℃[5]。

6.3 原理图



6.4 结论

选取测试的 20 次中成本最低的 5 次记录出下表所得数据。

表 6-1 模拟退火算法所得结果

组号	成本	最优点						
1	93.8561	2	9	20	26	33	43	50
2	93.8998	2	9	20	26	34	43	50
3	93.9456	2	9	20	27	34	43	50
4	93.9712	2	10	20	26	33	43	50
5	94.0011	2	10	20	27	33	43	50

可以看出模拟退火算法和遗传算法的成本差不多，但是模拟退火算法的所需时间只有十几秒，运行很快。

7 鸣谢

非常感谢袁炎老师的指导以及其他小组的帮助，从刚接触 matlab 和统计推断方法的陌生到现在可以较熟练的应用 matlab 计算和作图离不开老师、助教和同学们的帮助，袁老师对我们不懂的问题及时的回答以及鼓励给了我们很大的信心。同时，也感谢学长流传下来的报告，让我们可以尽快熟悉报告类型和模板，站在巨人的肩膀[6]。

8 参考文献

[1]袁炎. 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]戚非.闫勇.田应.基于 MATLAB 的多项式拟合.实验室科学 2006 年 10 月第 5 期
[3]张志涌.精通 MATLAB6.5.北京航空航天大学出版社,2003.03
[4]陈国良.遗传算法及其应用.人民邮电出版社,1996.06
[5]姚新.陈国良.木偶你退火算法及其应用.中国科技大学计算机科学技术系.计算机研究与发展,1990.07
[6]顾婕昱.毕雨来.统计推断应用报告,2012.12

9 附录

9.1 遗传算法代码

主函数 testmain

```
function [ P,Costmin ] = testmain()
tic
N=100;  T=30;  p=0.8;  q=0.01;  n=7;
data=xlsread('20141010dataform.xlsx');
t=1;
allCost=zeros(1,N);
population=Initialize(N,n);
for i=1:1:N
    %计算初始种群每一个的成本并储存
    allCost(i)=Costing(population(i,:),data,n);
 end
Costmin=allCost(N);
while t<=T
    [population,allCost]=Mutate(population,allCost,N,q,data,n);          %变异
    [allCost,Index]=sort(allCost,'descend');
    population=population(Index,:);
    [population,allCost]=Crossover(population,allCost,N,p,data,n);       %交叉
    [allCost,Index]=sort(allCost,'descend');
    population=population(Index,:);
    [population,allCost]=Select(population,allCost,data,n);              %选择
    if Costmin>allCost(N)
        Costmin=allCost(N);
        P=population(N,:);
    end

    disp('子代数');                                  %输出适应度最高的子代及其成本
    disp(t);
    disp(Costmin);
    disp(population(N,:));
    t=t+1;
end
toc
end
```

初始化函数 Initialize

```
function [ population ] = Initialize(N,n)
switch n
    case 7
        population=zeros(N,n);
        for i=1:1:N
        population(i,1)=1+ceil(rand*7);
        population(i,2)=8+ceil(rand*7);
        population(i,3)=15+ceil(rand*7);
        population(i,4)=22+ceil(rand*7);
```

```
            population(i,5)=29+ceil(rand*7);
            population(i,6)=36+ceil(rand*7);
            population(i,7)=43+ceil(rand*7);
            end
        case 6
            population=zeros(N,n);
            for i=1:1:N
            population(i,1)=ceil(rand*6);
            population(i,2)=6+ceil(rand*6);
            population(i,3)=12+ceil(rand*13);
            population(i,4)=25+ceil(rand*13);
            population(i,5)=38+ceil(rand*6);
            population(i,6)=44+ceil(rand*7);
            end
        case 5
            population=zeros(N,n);
            for i=1:1:N
            population(i,1)=ceil(rand*8);
            population(i,2)=8+ceil(rand*11);
            population(i,3)=19+ceil(rand*14);
            population(i,4)=33+ceil(rand*10);
            population(i,5)=43+ceil(rand*8);
            end
        case 4
            population=zeros(N,n);
            for i=1:1:N
            population(i,1)=ceil(rand*10);
            population(i,2)=10+ceil(rand*13);
            population(i,3)=23+ceil(rand*15);
            population(i,4)=38+ceil(rand*13);
            end
end

end
```

成本计算函数 Costing

```
function [ singleCost ] = Costing( P,data,n )
singleCost=0;
sort(P);
X=data (1,:);
for k=2:2:938
    Cost=0;
    Y=data (k,:);
```

```
    Px=zeros(n,1);
    for j=1:1:n
        Px(j,1)=X(P(j));
        Py(j,1)=Y(P(j));
    end
    Ytheory=interp1(Px,Py,X,'spline');
     for i=1:1:51
         s=abs(Y(i)-Ytheory(i));
         if s<=0.5
             c=0;
         elseif s<=1
             c=0.5;
         elseif s<=2
             c=1.5;
         elseif s<=3
             c=6;
         elseif s<=5
             c=12;
         else c=25;
         end
         Cost=Cost+c;
     end
     singleCost=singleCost+Cost;
end
 singleCost= singleCost/469+12*n;
end
```

变异函数 Mutate

```
function [ population,allCost ] =Mutate( population,allCost,N,q,data,n)
for i=1:1:N
    if rand<q
        while 1
        if allCost(i)<(n*12+10)
            len=randi([-2,2]);
        else
            len=randi([-5,5]);                          %变异步长(1-3)
        end
        pos=randi(n);                                   %变异位置（1-7）
        child=population(i,:);
        child(1,pos)=mod(population(i,pos)+len,49)+2;
        if length(unique(child))==n                     %检查变异后个体是否合法
           Cost=Costing(child,data,n);
           if allCost(i)>Cost;                          %变异之后要更新分数列
```

```
                population(i,:)=child;
                allCost(i)=Cost;
            end
            break;
        end
      end
    end
end
end
```

交叉函数 Crossover

```
function [population,allCost] = Crossover( population,allCost,N,p,data,n)
%适应度高的一半内部进行交叉
for i=N:-2:(N/2+2)
    sort(population(i,:));                          %把个体中的元素重新排列
    sort(population(i-1,:));
    if rand<p
        point=randi(n);                                    %单点交叉
        child1=[population(i,1:point),population(i-1,point+1:n)];
        child2=[population(i-1,1:point),population(i,point+1:n)];
        child1=Check(child1,n);                        %检查交叉结果是否合法
        child2=Check(child2,n);
        if (length(unique(child1))==n )&&(length(unique(child2))==n )
          Cost1=Costing(child1,data,n);                 %计算交叉结果的成本
          Cost2=Costing(child2,data,n);
          if allCost(i)>Cost1                %交叉后的个体成本更低，就取交叉后的个体
              population(i,:)=child1;
              allCost(i)=Cost1;
          end
          if allCost(i-1)<Cost2
              population(i-1,:)=child2;
              allCost(i-1)=Cost2;
          end
          sort(population(i,:));
          sort(population(i-1,:));
        end
    end
end
%适应度低的与适应度高的进行交叉，下标从N/2到1
for i=N/2:-1:1
    sort(population(i,:));                          %把个体中的元素重新排列
    if rand<p
        point=ceil(rand*4);
        child1=[population(i,1:point),population(i+N/2,point+1:n)];%与第i+N/2个元素交叉
```

```
            child1=Check(child1,n);                                    %检查是否合法
            Cost1=Costing(child1,data,n);                              %计算新得到结果的分数
            if allCost(i)>Cost1                          %交叉后的个体成本更低，就取交叉后的个体
                population(i,:)=child1;
                allCost(i)=Cost1;
            end
            sort(population(i,:));
        end
end
end
```

选择函数 Select

```
function [ population,allCost] = Select( population,allCost,data,n)%根据适应度高低进行选择
population([1,2,3,4],:)=Initialize(4,n);
allCost(1)= Costing(population(1,:),data,n);
allCost(2)= Costing(population(2,:),data,n);
allCost(3)= Costing(population(3,:),data,n);
allCost(4)= Costing(population(4,:),data,n);
end
```

检查函数 Check

```
function [x] = Check(x,n)                                  %检查子代是否有重复
k=1;
sort(x);
new=unique(x);
if length(new)~=n
    x=Initialize(1,n);
end
end
```

9.2 模拟退火算法代码

主函数 testmain

```
function [ X,Costmin ] = testmain( )
T=300;
n=7;
rate=0.97;
Tout=5;
data=xlsread('data.xlsx');
l=0;
X=Initialize(1,n);
Costmin=Costing(X,data,n);
tic
while (T>Tout)
    Xnew=X+round((rand(1,n)-0.5)*2);
```

```
    while
(Xnew(1)<1||Xnew(n)>51||abs(Xnew(1)-Xnew(2))<3||abs(Xnew(2)-Xnew(3))<3||abs(Xnew(2)-Xne
w(3))<3||abs(Xnew(3)-Xnew(4))<3||abs(Xnew(4)-Xnew(5))<3||abs(Xnew(5)-Xnew(6))<3||abs(Xne
w(6)-Xnew(7))<3)
        Xnew=X+round((rand(1,n)-0.5)*2);
    end
    costnew=Costing(Xnew,data,n);
    d=costnew-Costmin;
    if (d<0)
        X=Xnew;
        Costmin=costnew;
    else
        p=exp((-d)*1000/T);
        if (rand<p)
            X=Xnew;
            Costmin=costnew;
        end
    T=T*rate;
    end
    l=l+1;
    disp(Costmin);
end
toc
disp('最好结果=');
disp(X);
disp(Costmin);

end
```

其余初始化与成本计算函数与遗传算法相同