统计推断在数模转换系统中的应用

——数据定标中的遗传算法应用探究

第 41 组　程政瑜 5130309087，黄昌霖 5130309095

摘要：本文主要探究针对某一检测模块中输入输出特性呈明显的非线性关系的传感器部件进行传感特性校准的问题，在较短的时间和较小的运算规模内运用遗传算法和拟合搜索近似最优解，力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：遗传算法，近似最优解

1 引言

监测模块的组成框图如图 1-1。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）。而其中，一个传感部件个体的输入输出特性大致如图 1-2 所示。[1]

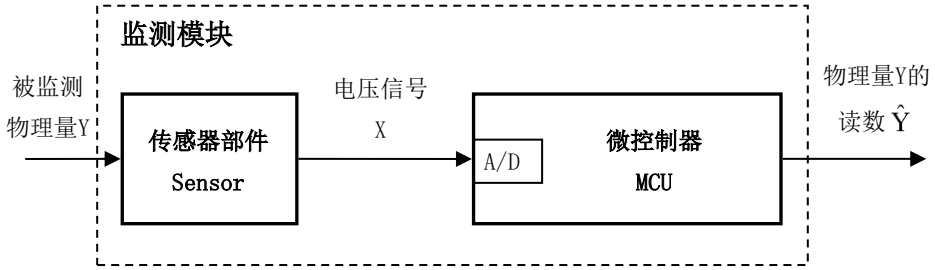


图 1-1 监测模块组成框图

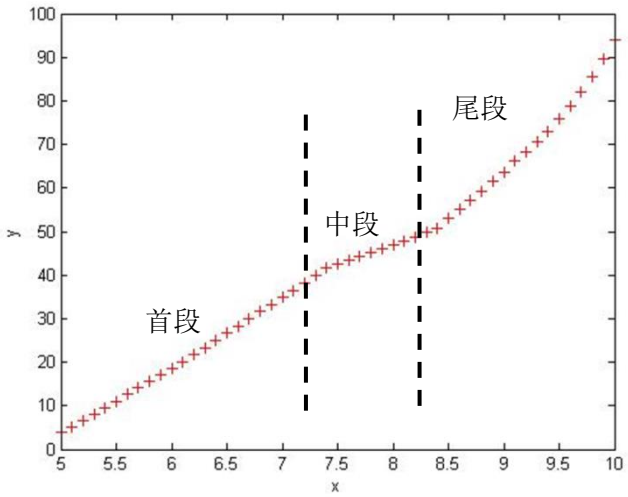


图 1-2 传感特性图示

2 拟合方案的对比探究

2.1 成本评估标准

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

$$s_{i,j}=\begin{cases}0 & \text{if } \left|\hat{y}_{i,j}-y_{i,j}\right|\le 0.5\\ 0.5 & \text{if } 0.5<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 1\\ 1.5 & \text{if } 1<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 2\\ 6 & \text{if } 2<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 3\\ 12 & \text{if } 3<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 5\\ 25 & \text{if } \left|\hat{y}_{i,j}-y_{i,j}\right|>5\end{cases} \tag{2-1}$$

式中 $y_{i,j}$ 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）。

（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

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

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

$$C=\frac{1}{M}\sum_{i=1}^{M}S_i \tag{2-3}$$

按式（2-3）计算校准方案的总体成本，总体成本较低的校准方案，认定为较优方案。

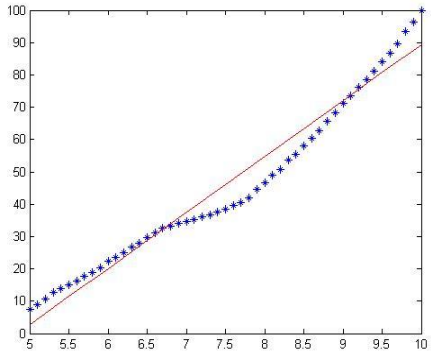
2.2 最小二乘法拟合

2.2.1 最小二乘法的介绍

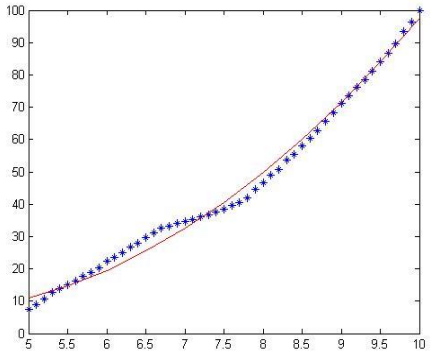
当将曲线进行多项式的拟合时，将数据点与拟合曲线上对应点的垂直距离定为误差，对各个数据点的距离求平方后累加起来得到曲线的误差平方和。最小二乘法即是使得误差平方和最小的方法，这样进行数据拟合较为便捷也易于理解。

2.2.2 多项式阶数的影响

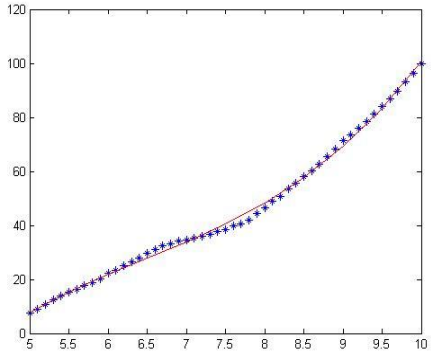
如图 2-1，三幅图任取一组数据（第七组），在样本 51 个数据中均匀间隔取 11 个点进行最小二乘法多项式拟合，拟合得到的图线与所有数据点进行对比、观察可知，本次数据最小二乘法处理宜采用 3 阶以上进行拟合，效果较好。



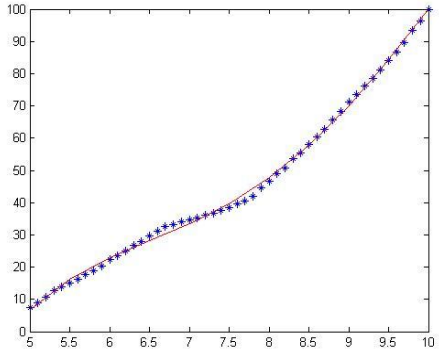
(a)一阶拟合



(b)二阶拟合



(c)三阶拟合



(d)四阶拟合

图 2-1 各阶最小二乘法拟合

2.2.3 成本探究

根据图 1-2 传感部件的特性，三段均宜取到三个点以上，为了初步以较少的点数计算比较探究最小二乘法拟合的成本，现均匀间隔取点 6~10 个，对于三阶、四阶、五阶的最小二乘法拟合进行对比，计算成本后列出表格如下。

表 2-1 最小二乘法拟合成本数据表

样本取样点数（均匀） / 拟合阶数	6	7	8	9	10
三阶	604.27	587．14	562.10	554.98	545.40
四阶	361.30	359.96	334.24	332.32	329.33
五阶	561.50	323.98	232.63	225.91	213.75
六阶	——	932.62	230.09	213.00	168.79

观察上表可知以下几点规律：

1． 在同等取点情况下，高阶拟合成本一般而言比低阶要低。

2． 一定范围内，取样点数越多拟合成本越低，且随着点数增多成本缩减幅度变小。

3． 高阶对于点数的依赖性更大，取点数与阶数相近时拟合效果甚至比低阶拟合差。

可初步总结，以均匀方法取点，随着拟合阶数与取样点数的提高，最小二乘法拟合可以得到更优化的方案，加上分析原理，不难推知阶数过高或点数过多时拟合成本会有一定上升，但过高的阶数会导致运行时耗费时间过长，而且点数较少时不能进行高阶取点，故此最小二乘法拟合存在很大的局限性，不适合作为本例中成本计算的拟合方法。

2.3 插值拟合

2.3.1 两种插值拟合的介绍

线性插值拟合：若实验数据为线性组合，用一个函数逼近所有实验数据得到若干个点，在用一条曲线来插值所有的点的操作叫做线性插值拟合。

三次样条插值拟合：用三次样条函数将实验数据拟合，再通过三弯矩算法或是三转角算法插值。

2.3.2 成本探究

根据图 1-2 传感部件的特性，为了初步以较少的点数计算比较探究插值拟合的成本，现均匀间隔取点 5~10 个，对于线性插值拟合和三次样条插值进行对比，计算成本后列出表格如下。

表 2-2 线性插值法拟合成本数据表

样本取样点数（均匀） / 插值拟合方法	5	6	7	8	9	10
线性插值	111.23	107.88	107.15	109.35	118.42	126.86
三次样条插值	113.71	96.41	99.49	103.42	112.54	122.64

由上表简单可知，在均匀取点的情况下，三次样条插值在整体的拟合程度上更为精确，成本往往更为节约。

在实际计算的过程中，也发现了线性插值取两个点（24，25）和三个点（24，25，26）时成本甚至能达到 24.00 和 36.00，说明其在本例中较为适用，但不符合设计传感特性校准方案的要求，不具有广泛的应用性，故此不建议拟合方式采用线性插值拟合。

再次观察上表，可以知道三次样条插值基本满足一般情况下均匀取点的拟合精准要求，可以广泛应用，故此采用三次样条插值法作为本例探讨中的拟合方法。

3 遗传算法取点的探究

3.1 遗传算法的介绍

遗传算法是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法。

遗传算法是从代表问题可能潜在的解集的一个种群开始的，而一个种群则由经过基因编码的一定数目的个体组成。每个个体实际上是染色体带有特征的实体。染色体作为遗传物质的主要载体，即多个基因的集合，其内部表现（即基因型）是某种基因组合，它决定了个体的形状的外部表现，如黑头发的特征是由染色体中控制这一特征的某种基因组合决定的。因此，在一开始需要实现从表现型到基因型的映射即编码工作。

初代种群产生之后，按照适者生存和优胜劣汰的原理，逐代演化产生出越来越好的近似解，在每一代，根据问题域中个体的适应度大小选择个体，并借助于自然遗传学的遗传算子进行组合交叉和变异，产生出代表新的解集的种群。这个过程将导致种群像自然进化一样的后生代种群比前代更加适应于环境，末代种群中的最优个体经过解码，可以作为问题近似最优解。

3.2 遗传算法的运用

3.2.1 基因编码

由于仿照基因编码的工作很复杂，我们往往进行简化，本例中采用二进制编码，即由51 位的二进制编码，代表取点的位置，若该点被选取为样本点则定为 1，反之该点编码则为 0，如[001001100010000100000100000001011110000100001000001]，根据表 2-2 得知，优秀个体基因编码取点数应较少，故此在初始化种群时设定随机概率为 0.3，即 51 位编码中取点最大概率为 15 个点左右，这样便于减少繁衍代数，减少程序运行时间。

3.2.2 基础繁衍模式

根据 2-1 成本评估标准和三次样条插值法对个体取样点进行适应度的计算，拟合后取其总成本的倒数作为该个体的适应度，即保证适应度越大的个体成本越少。此外，采用轮盘赌方式确定父本，随机确定母本，随机确定交叉点位置。

为了简化遗传模式，采用随机概率的方式，如 80%的概率父本和母本交叉，3%的概率变异，否则保留父本传入下一代。杂交采用单位点交叉，即确定交叉点后后续基因交换。变异采用单位点变异，即随机某一位置发生 0 和 1 的变换。

此后经过较多的代数，整体上适应度较优秀的会保留下来，较低劣的则不能保留，在较为优秀的末代中，取最优个体进行解码即为所得到的近似最优解。

3.3 遗传算法的思考与改进

3.3.1 杂交模式的思考

父本采用轮盘赌确定的方式能让较优秀的个体进行优化和保留，母本采用随机确定则能提供多样性的进化可能，但在一定程度上无法保证优秀个体的保留，在经过较多的代数和较大的种群个体数能很好地缓解这个问题。在尝试过程中，30 代以下往往不能出现较好的结果，但是实际上由于遗传算法的特点，在一次花费 3 个小时尝试进行 1000 代中出来的结果依然是成本为 98 左右的取点方式,相对于 100 代以内就能取得的结果并没有较大改善。当然，不排除偶然情况没有出现优良解，这一定程度上说明了本算法在促进种群多样性进化方面有很大的空间，但对于较优解往往偏于全面查找，而不能更精确地优化，这个问题在 3.3.3 中会加以解决。

交叉和变异均是单个位点的操作，交叉存在很大的随意性，变异对整体的情况也影响不大，需要更优化的控制。在本例中曾尝试多部分的交叉和多位置的变异，多部分的交叉即确定两个以上的交叉点，将其中某些片段进行交叉，但实际运用中导致运行时间增加和代码量繁杂，结果也并无明显优化，甚至在较少的代数中效果更差，本例中考虑到其必要性和时间运算的花费，故此认为其没有太大价值，不予采用；由于变异往往是对于种群进化较小的影响，但能很大程度上丰富其多样性，采用多位置的变异能很好地促进种群产生较优解的可能性，但同时也意味着种群进化波动，需要更多的时间花费，容易造成种群结构不够稳定，故此不予采用。针对交叉和变异的问题，在 3.3.2 中会进一步从遗传算子的角度进行改进。

3.3.2 遗传算子的改进

本例中遗传算子简化为随机概率确定，但实际运用中确定合适有效的遗传概率则成为一个问题，最初本例采用杂交概率为 0.9，变异概率为 0.04，此后将其杂交概率改为 0.95，变异概率 0.03，以便种群更稳定以及进化更快。

这两个是参数实际上是影响遗传算法行为和性能的关键所在，直接影响到算法的收敛性，交叉概率却大，新个体产生的速度就越快。然而，交叉概率过大时遗传模式被破坏的

可能性也越大，使得具有高适应度的个体结构很快就会被破坏；但是如果交叉概率过小，会使搜索过程缓慢，以致停滞不前。为了取得更有价值的改进方案，在后续的过程中参考一些资料[2]，开始采用自适应的概率变化函数，其中交叉概率 Pc 的计算公式见式（3-1），变异概率 Pm 的计算公式见式（3-2）。

$$\mathrm{Pc}=\begin{cases}\frac{Pc_1\times(f_{max}-f)}{f_{max}-f_{avg}} & ,\ f\geq f_{avg}\\ Pc_2 & ,\ f<f_{avg}\end{cases} \tag{3-1}$$

$$\mathrm{Pm}=\begin{cases}\frac{Pm_1\times(f_{max}-f')}{f_{max}-f_{avg}} & ,\ f'\geq f_{avg}\\ Pm_2 & ,\ f'<f_{avg}\end{cases} \tag{3-2}$$

两式中 *fmax* 表示群体中最大适应度值，*favg* 表示群体平均适应度值，*f* 表示要交叉的两个个体中较大的适应度值，*f'*表示要变异个体的适应度值，Pc_1，Pc_2、Pm_1 和 Pm_2 表示基础设定常数。

此算法保证了当种群各个体适应度趋于一致或者趋于局部最优是，使得交叉概率和变异概率二者增加，而当群体适应度比较分散时，使得交叉概率和变异概率减少。在实际运用中本例采取了 Pc_1 = 0.5，*Pc2* = 0.9，*Pm1* = 0.02，*Pm2* = 0.05 为基础参数，并且在本例运用时取得了较大的突破，往往 30 代以上就较大有可能性可以取得 6 个点成本 95~100 的数据，较为完善的优化了遗传的模式，取得了很好地效果，自适应遗传算法代码见附录一。

3.3.3 近似最优解的进一步优化

对于末代种群的整体而言，适应度已得到了较大的提升，但这种启发式搜索算法往往无法定向的得到理想的结果，例如在本例实践中末代种群最优秀个体可能出来 6 个点或者 7 个点，无法控制其最终得出来的取点个数，当然，最简单的做法是扩大种群规模，例如扩大到 500 这样的个体数，但是这样很大程度上使得搜索过程和计算过程更加费时，而且扩大之后其花费的代价并不能突出性的得到回报。

根据遗传算法的局限性和优势，不改变固有参数的情况下，并借鉴其他启发式搜索中摇摆趋近的思想，更为省时有效的做法是针对遗传算法末代最优个体进行局部优化，从而取得更理想的近似最优解。据此编写出一个最优解简单优化的代码，即对最优个体的中间所有点数进行循环遍历，逐次单个的进行摇摆寻找更优解，由于在遗传算法结果基础上进行操作，往往轻易就能将不同的近似最优解成本下降到 95 以下。

3.4 结果与分析

在不同繁衍代数情况下多次运行代码，并适当进行优化所得结果如下表 3-1 所示。

表 3-1 遗传优化代码运行结果成本数据表

繁衍代数		10	30	70	100
第一次运行	原始解	134.9403	104.7484	102.0704	106.60
	优化后	——	95.70789	95.85501	92.8774
第二次运行	原始解	137.9574	118.9222	95.8817	98.3028
	优化后	——	93.39872	94.21429	96.6141
第三次运行	原始解	138.2729	97.84	107.15	121.0085
	优化后	——	93.40085	117.5384	111.98401

上表中最优解为优化后的 92.8774，此时取点为[3 12 22 31 43 50]，详细数据见附录三。

观察表 3-1 可知，由于种群个数为 100，使得末代种群最优个体存在很大的波动性，但是通过采用 3.3.3 中的简单优化算法可以直接的得到更优化的解，一定程度弥补了遗传算法在寻找局部最优的局限性，可以看出 70 代或者 100 代并不能对于 30 代有过多的优越性，这一方面是由于种群规模较小，另一方面则是由于自适应算法的进化加快和优化代码发挥的作用。当然，如果在时间花费允许的情况下，种群规模有必要进行更大的扩展，那么将会大大减少出现成本为 100 以上的近似最优解，在此限于 matlab 运行时间过长，仅能通过这样的种群规模来大致反映遗传算法的特点及优化的必要性。

综上所述，在种群规模仅为 100 个个体，繁衍代数仅有 30 代的情况下，本例所采用的自适应遗传算法和最优解简单优化已经能够较快地粗略得出较为优秀的结果，从算法本身机制上进行优化确实能够起到很大的效果。

4 方案总结

本次探究主要针对了种群规模较小，繁衍代数较低情况下针对基础的遗传算法进行优化探究，拟合方案采用了普通的三次样条插值拟合以求能够广泛符合其他波动较大的特性曲线。在定标过程可以总结如下，针对较大的样本数据库，客观条件允许情况下，大规模种群的遗传算法启发式搜索往往能很好寻找优秀的结果，但是考虑到一些大规模计算的局限性和困难性，定标工序可以借鉴本例中对于数据的处理和探究思想，首先进行样本收集，其次设定校验标准，最后设定遗传算法基础参数，并在发挥遗传算法搜索优势的同时，针对其缺陷予以弥补和改进，其中在进化和稳定之间把握平衡是遗传算法的重要关键，于是就可以的想要寻找的近似最优解。

5 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] 龚纯,王正林. 精通 MATLAB 最优化计算[M]. 北京:电子工业出版社,2012.

附录 1　自适应遗传算法 Matlab 代码

```
Pc1=0.5;      %杂交概率1
Pc2=0.9;      %杂交概率2
Pm1=0.02;     %变异概率1
Pm2=0.05;     %变异概率1
NP=100;       %种群规模
NG=30;        %繁衍代数

x=zeros(100,51);
for i=1:NP                          %初始化种群
    for j=1:51
        sita1=rand();
        if sita1<0.3
         x(i,j)=1;
        end
    end

    fx(i)=fitness(x(i,:));
end

for k=1:NG
    disp(k);
    sumfx=sum(fx);
    Px=fx/sumfx;
    PPx=0;                          %轮盘赌确定父本
    PPx(1)=Px(1);
    for i=2:NP
        PPx(i)=PPx(i-1)+Px(i);
    end
    for i=1:NP
        sita2=rand();
        for n=1:NP
            if sita2<=PPx(n)
                SelFather = n;
                break;
            end
        end
        SelMother = floor(rand()*(NP-1))+1;  %随机母本
        posCut = round(rand()*49)+1;         %随机交叉点
        favg=sumfx/NP;                       %杂交概率变化函数
        fmax=max(fx);
        Fitness_f=fx(SelFather);
```

```
        Fitness_m=fx(SelMother);
        Fm=max(Fitness_f,Fitness_m);
        if Fm>=favg
            Pc=Pc1*(fmax-Fm)/(fmax-favg);
        else
            Pc=Pc2;
        end
        sita3=rand();
        if sita3<=Pc                        %杂交
            nx(i,1:posCut) = x(SelFather,1:posCut);
            nx(i,(posCut+1):51) = x(SelMother,(posCut+1):51);
            fmu=fitness(nx(i,:));
            if fmu>=favg                    %变异概率变化函数
                Pm=Pm1*(fmax-fmu)/(fmax-favg);
            else
                Pm=Pm2;
            end

            sita4=rand();
            if sita4<=Pm                    %变异
                posMut= round(rand()*49+1);
                nx(i,posMut)=~nx(i,posMut);
            end
        else
            nx(i,:)=x(SelFather,:);
        end
    end
    x=nx;
        for i=1:NP
            fx(i)=fitness(x(i,:));
        end
end
disp(1);
min_cost=-inf;
for i= 1:NP                                 %种群末代最优解及解码
    fitx=fitness(x(i,:));
    if fitx>min_cost
        min_cost=fitx;
        xsum=sum(x(i,:),2);
        my_answer=zeros(1,xsum);
        for j=1:51
            if x(i,52-j)==1
                my_answer(1,xsum)=52-j;
```

```
            xsum=xsum-1;
        end
    end
    disp(my_answer);
    disp(1/min_cost);
  end
end
%以下为适应度函数
function  yy = fitness( x_pre)
minput=dlmread('20141010dataform.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
index_temp=logical(x_pre);
x_op=x(:,index_temp);
y0_op=y0(:,index_temp);
x_tmp=[5.0:0.1:10.0];
for j=1:nsample
    y1(j,:)=interp1(x_op(j,:),y0_op(j,:),x_tmp,'spline');
end
Q=12;
errabs=abs(y0-y1);
le0_5=(errabs<=0.5);
le1_0=(errabs<=1);
le2_0=(errabs<=2);
le3_0=(errabs<=3);
le5_0=(errabs<=5);
g5_0=(errabs>5);
sij=0.5*(le1_0-le0_5)+1.5*(le2_0-le1_0)+6*(le3_0-le2_0)+12*(le5_0-le3
_0)+25*g5_0;
count=sum(x_pre,2);
si=sum(sij,2)+Q*ones(nsample,1)*count;
cost=sum(si)/nsample;
yy = 1/cost;
end
```

附录 2　GA 最优解简单优化 Matlab 代码

```
n=6;
my_answer=[1 11 21 31 41 51];%此处输入GA算法所得最优解

minput=dlmread('20141010dataform.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
min=500;
while true
    my_answer_t=my_answer;

    for i=2:n-1
        for k=(my_answer(i-1)+1):(my_answer(i+1)-1)
        tmp=my_answer(i);
        my_answer(i)=k;

        my_answer_n=size(my_answer,2);
        my_answer_gene=zeros(1,npoint);
        my_answer_gene(my_answer)=1;


        index_temp=logical(my_answer_gene);
        x_optimal=x(:,index_temp);
        y0_optimal=y0(:,index_temp);
         for j=1:nsample
            y1(j,:)=mycurvefitting(x_optimal(j,:),y0_optimal(j,:));
         end

        Q=12;
        errabs=abs(y0-y1);
        le0_5=(errabs<=0.5);
        le1_0=(errabs<=1);
        le2_0=(errabs<=2);
        le3_0=(errabs<=3);
        le5_0=(errabs<=5);
```

```
        g5_0=(errabs>5);
sij=0.5*(le1_0-le0_5)+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;
        if cost<min
        min=cost;
        disp（min);
        disp(my_answer);
        else my_answer(i)=tmp;
            end
        end
    end
    if my_answer_t==my_answer break;
    end
end
disp（min);
disp(my_answer);
```

附录 3　代码运行数据结果

10 代：
第一次：1　9　13　20　22　25　28　38　40　50
　　成本 134.9403
第二次：4　5　10　12　15　22　28　32　41　48　51
　　成本 137.9574
第三次：4　8　18　24　25　26　30　35　42　49　50
　　成本 138.2729
30 代：
第一次：2　11　18　22　31　46　48 成本 104.7484
　　优化后：2　9　20　26　33　44　48 成本 95.70789
第二次：2　14　27　33　42　49　成本 118.9222
　　优化后：2　11　22　31　41　49　成本 93.39872
第三次：2　11　24　30　41　50 成本 97.84
　　优化后：2　10　21　30　41　50　成本 93.40085
70 代：
第一次：3　10　19　21　31　43　48 成本 102.0704
　　优化后：3　9　20　26　33　44　48 成本 95.85501
第二次：1　7　21　31　42　50 成本 95.8817
　　优化后：1　10　21　31　43　50　成本 94.21429
第三次：6　10　22　26　38　43　46　51 成本 117.5384
　　优化后：6　8　20　26　32　39　45　51　成本 107.75480
100 代：
第一次：3　10　22　37　43　50 成本 106.60
　　优化后：3　12　22　31　43　50 成本 92.8774
第二次：5　7　19　28　34　45　50 成本 98.3028
　　优化后：5　8　20　26　33　43　50 成本 96.61407
第三次：3　9　16　24　25　27　35　40　49 成本 121.0085
　　优化后：3　8　15　21　26　32　38　45　49　成本 111.98401