# 统计推断在数模转换系统中的应用

组号：17　　小组成员：虞盛炜 5130729065 张希伦 5140309248

**摘要：**在数学物理及许多工程科学中，实验往往有着举足轻重的作用，而在实验中取得的大量数据往往具有随机性强的特点。为了更好的找到不同影响因素之间的内在规律，建立数学模型并且应用数理统计规律来解决实际问题已经成为行之有效的方法和途径。本课题的问题背景是为某产品内部的一个测量模块寻求定标工序的优选方案。课题要求运用数学软件MATLAB，采用遗传算法选择合适的定标点，再使用插值或拟合方法来完成对样品的定标，进而得到待测电子产品的输入与被检测物理量之间的关系。

**关键词：**MATLAB，统计推断，遗传算法，三次样条插值法，多项式拟合法，定标

## 1 引言[1]

### 1.1 课题概述

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 模型

为了对本课题展开有效讨论，需建立一个数学模型，对问题的某些方面进行必要的描述和限定。

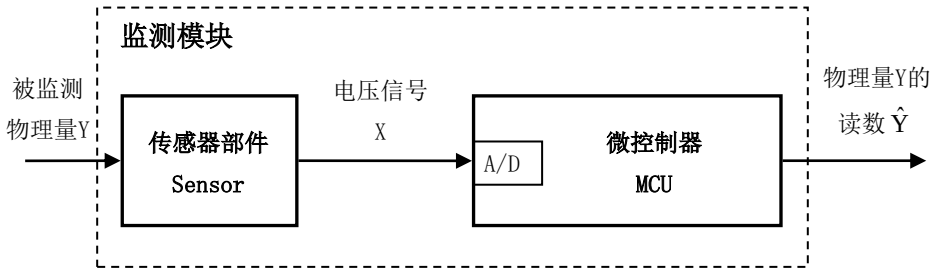


图 1.2.1 监测模块组成框图

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）。

所谓传感特性校准，就是针对某一特定传感部件个体，通过有限次测定，估计其 Y 值与 X 值间一一对应的特性关系的过程。数学上可认为是确定适用于该个体的估测函数 $\hat{y}=f(x)$ 的过程，其中 x 是 X 的取值，$\hat{y}$ 是对应 Y 的估测值。

考虑实际工程中该监测模块的应用需求，同时为便于在本课题中开展讨论，我们将问题限于 X 为离散取值的情况，规定

$$X \in \{x_1, x_2, x_3, ..., x_{50}, x_{51}\} = \{5.0, 5.1, 5.2, ..., 9.9, 10.0\}$$

相应的 Y 估测值记为 $\hat{y}_i = f(x_i)$， Y 实测值记为 $y_i$， $i = 1,2,3,...,50,51$。

传感部件特性

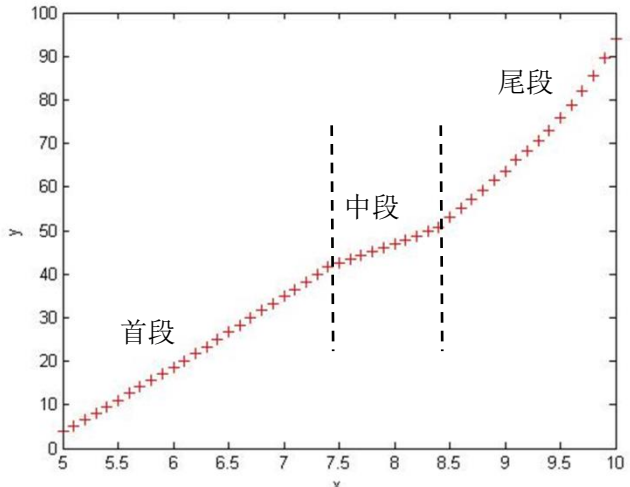


图 1.2.2 传感特性图示

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。

## 1.3 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。

# 2 拟合方法

科学或工程上可以通过实验等方法获得关于某个问题的若干离散数据。依靠数学方法，使用连续函数（也就是曲线）或者更加密集的离散方程尽量逼近（即最小二乘意义上的差别最小化）这些已知离散数据点集，此过程称为拟合。

## 2.1 三次多项式拟合

用三次多项式近似拟合曲线，进行三次多项式拟合，令$f(x)=ax^3+bx^2+cx+d$，在本实验中，我们利用 MATLAB 中已有函数（1）进行拟合：[2]

$$a = polyfit(x0,y0,m) \quad (1)$$

其中 x0，y0 为拟合点坐标，m 为拟合多项式次数。

多项式在 x 处的 y 值可用 MATLAB 中已有的函数（2）进行计算：

$$y = polyval(a,x) \quad (2)$$

由于 m=3 时拟合效果较好，平均误差较小，故本课题中我们采用三次拟合。

## 2.2 三次样条插值法[3]

### 2.2.1 定义

三次样条插值是通过一系列形值点的一条光滑曲线，数学上通过求解三弯矩方程组得出曲线函数组的过程。

### 2.2.2 相关函数

三次样条函数:

定义:函数 S(x)∈C2[a,b] ，且在每个小区间[ $x_j$,$x_{j+1}$ ]上是三次多项式，其中 a =$x_0$ <$x_1$<...< $x_n$= b 是给定节点，则称 S(x)是节点 $x_0$,$x_1$,...$x_n$ 上的三次样条函数。若在节点 x j 上给定函数值 $Y_j$= f ($X_j$).( j =0, 1, , n) ，并成立 S($x_j$) =$y_j$ .( j= 0, 1, , n) ，则称 S(x)为三次样条插值函数。

实际计算时还需要引入边界条件才能完成计算。边界通常有自然边界（边界点的二阶导为 0），夹持边界（边界点导数给定），非扭结边界（使两端点的三阶导与这两端点的邻近点的三阶导相等）。一般的计算方法书上都没有说明非扭结边界的定义，但数值计算软件如 MATLAB 都把非扭结边界条件作为默认的边界条件。

# 3 遗传算法

## 3.1 基本概念

遗传算法（Genetic Algorithm）是一类借鉴生物界的进化规律（适者生存，优胜劣汰遗传机制）演化而来的随机化搜索方法。它是由美国的 J.Holland 教授 1975 年首先提出，其主要特点是直接对结构对象进行操作，不存在求导和函数连续性的限定；具有内在的隐并行性和更好的全局寻优能力；采用概率化的寻优方法，能自动获取和指导优化的搜索空间，自适应地调整搜索方向，不需要确定的规则。遗传算法的这些性质，已被人们广泛地应用于组合优化、机器学习、信号处理、自适应控制和人工生命等领域。它是现代有关智能计算中的关键技术。

对于一个求函数最大值的优化问题(求函数最小值也类同)，一般可以描述为下列数学规划模型：式中 x 为决策变量，式 2-1 为目标函数式，式 2-2、2-3 为约束条件，U 是基本空间，R 是 U 的子集。满足约束条件的解 X 称为可行解，集合 R 表示所有满足约束条件的解所组成的集合，称为可行解集合。

$$\max f(X) \qquad 2-1$$

$$x \in R \qquad 2-2$$

$$R \subset U \qquad 2-3$$

遗传算法也是计算机科学人工智能领域中用于解决最优化的一种搜索启发式算法，是进化算法的一种。这种启发式通常用来生成有用的解决方案来优化和搜索问题。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。遗传算法在适应度函数选择不当的情况下有可能收敛于局部最优，而不能达到全局最优。

### 3.2 基本步骤及具体实现[4]

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)。我们设定种群大小为 100。我们尝试了其他大小的种群，例如 50,200 等，但由于结果不是很理想（要么太小，很容易造成优势个体占领整个种群；要么太大，时间空间消耗过大），并与老师交流，最终确定了 100 为种群大小。

b)个体评价：计算群体 P(t)中各个个体的适应度。我们对每种取点法进行适应度计算。拟合后，先通过给定的计算成本方式计算成本，由于成本与适应度成近似于反比关系，可以直接取成本的倒数作为适应度。然而通过实际运算，我们发现取成本倒数的 2.5 次方比较合理。

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。我们通过适应度计算各个个体（取点法）的成活概率（总概率为 1），根据概率的大小随机生成下一代。

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。我们选择交叉概率为 50%，而且交叉点随机。由于选择操作的随机性，不失一般性，我们直接让第一个个体与第二个个体交叉，第三个与第四个交叉，以此类推。

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)。我们考虑到变异的不利性，仅取 1%的变异概率（对于每个基因来说）。

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。我们直接通过给定的遗传代数，尝试了 30,50,100,其中 50 代时已经较为稳定。

## 4 算法结果及其分析

| 拟合方式 | 取点数量 | 最优取点方案 | 成本 | 运行代数 |
|---|---|---|---|---|
| 三次多项式拟合 | 6 | 4, 16, 28, 35， 39, 46 | 129.763 | 50 |

| | | | | |
|---|---|---|---|---|
| 三次样条插值法 | 5 | 3, 13, 26, 39，49 | 105.8022 | 50 |
| 三次样条插值法 | 7 | 3, 12, 20，28, 36, 45, 49 | 97.6540 | 100 |

表 4.1　优化前得出的部分最优解

我们只列了部分数据，省去了部分无用比较，比如三次多项式拟合 100 次运算代数成本与 50 次几乎相等。这里要指出取点数量并不是我们制定的，只是通过初始化时随机产生，然后通过自然选择剩下来的最优解中取点的个数。通过上面数据可以得出，三次样条插值法拟合成本低于三次多项式拟合，最优解为（3, 12, 20，28, 36, 45, 49）。这是优化算法前的结果，时间成本很大，运行 100 代用了接近 1 个小时的时间，而且 50 代与 100 代还有较大差距，可见算法还有很大改进空间。这也促使我们改进算法。

# 5　优化算法

## 5.1 关于循环的简化与输出的优化

a.　优化思路与过程

通过与老师交流，我们得知 MATLAB 在运算大量 for 循环时有效率不足的问题。同时，MATLAB 在矩阵运算时有不错的效率优势，所以考虑修改部分代码来提高效率。改动最明显的地方在于计算成本时，利用了矩阵运算和布尔运算，大大简化了表达，更是提高了运行效率。

同时我们增加了交互性，每一代都输出最小成本供用户观察变化趋势。

还有就是不再取最后一代的最小成本取法为最优解，而是储存每一代的最优解取法，在 50 代的循环完成后，从这 51 个最优解（包括第 0 代，即初始化的一代）中找出最优解，进行输出（这是为了防止在“自然选择”操作中，“不小心”淘汰了最优解，导致得到的最后一代中的成本最小的取法不是我们循环中出现的最优情况）。最后还做出了每一代最低成本所连成的曲线（散点）。

b.　结果及其分析

| 拟合方式 | 取点数量 | 最优取点方案 | 成本 | 运行代数 |
|---|---|---|---|---|
| 三次多项式拟合 | 5 | 4,15,27,42,51 | 122.8035 | 50 |
| 三次样条插值法 | 6 | 1,9,22,31,42,50 | 97.2375 | 50 |
| 三次样条插值法 | 7 | 3,8,19,26,32,43,49 | 96.14325 | 200 |

表 5.1.1　优化循环与输出之后的最优解

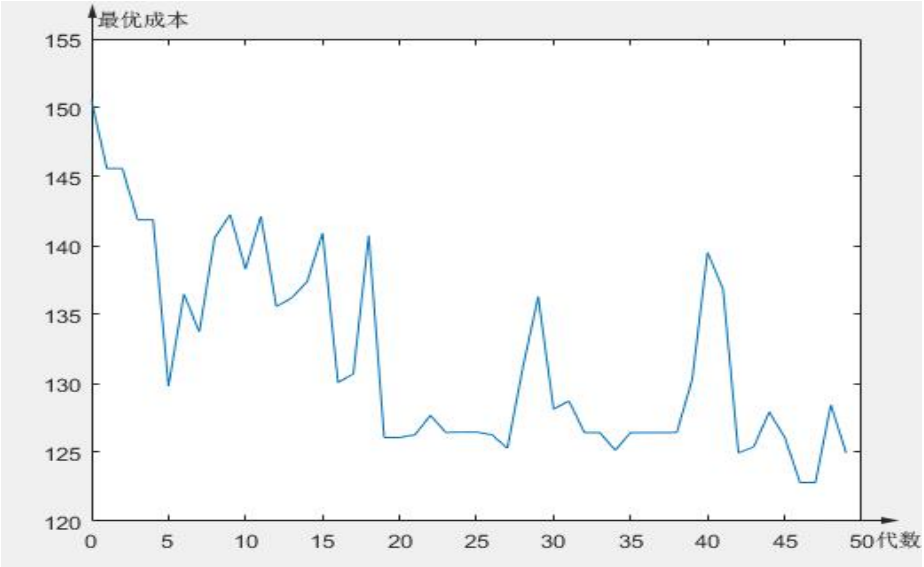


图 5.1.1　优化算法后三次多项式拟合最低成本变化曲线（50 代）

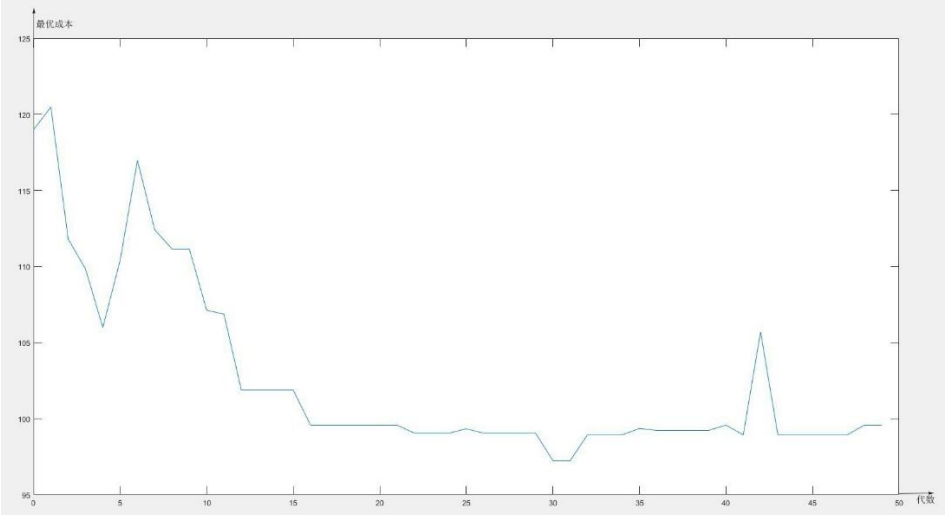


图 5.1.2　优化算法后三次样条插值最低成本变化曲线（50 代）

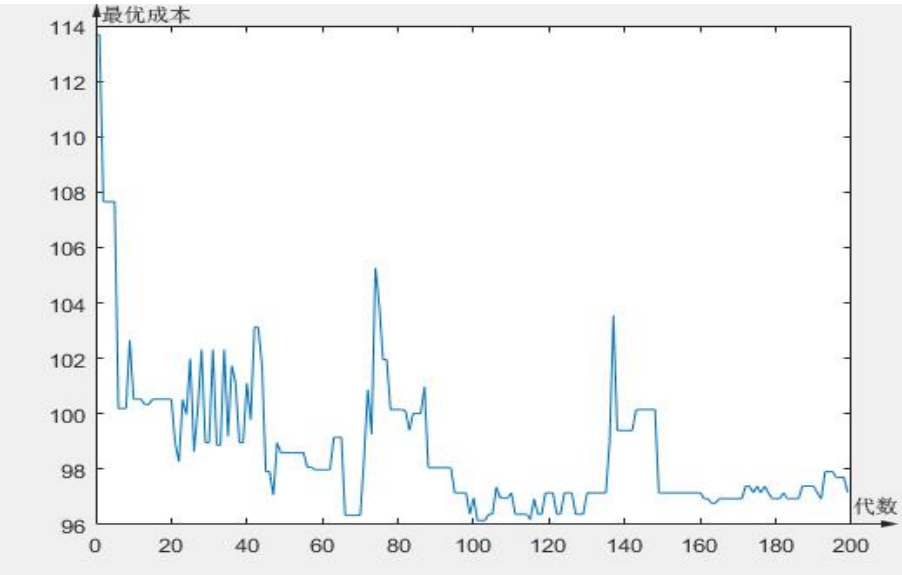


图 5.1.3　优化算法后三次样条插值最低成本变化曲线(200 代)

从结果我们可以看出，优化算法后的最优成本比之前有些许降低，三次样条插值法仍然比三次多项式拟合法更优秀。此时的最优解为（1,9,22,31,42,50），成本为 97.2375.而且代数在 20 多时已经趋于稳定。由于算法本质上没有改变，所以我们可以得知优化前的算法不仅消耗大量时间，而且会使原本取向稳定的结果发生变化导致成本增加，使最后代数不一定为最优解。最后我们还做了极限测试，代数设置为 200 代，得到的最优解为（3,8,19,26,32,43,49），成本为 96.14325。

## 5.2 对于选择、交叉、变异的优化与思考

由于选择、交叉、变异的操作会导致不稳定的结果，其中包含成本上升的可能，所以我们考虑是不是可以在每个操作最后加一个判断，如果成本升高，就不进行该项操作。以下是我们的一个结果。

| 拟合方式 | 取点数量 | 最优取点方案 | 成本 | 运行代数 |
|---|---|---|---|---|
| 三次样条插值法 | 6 | 1,12,24,29,40,48 | 104.15175 | 50 |

表 5.2.1　“优化”选择、交叉、变异操作之后三次样条插值法的最优解

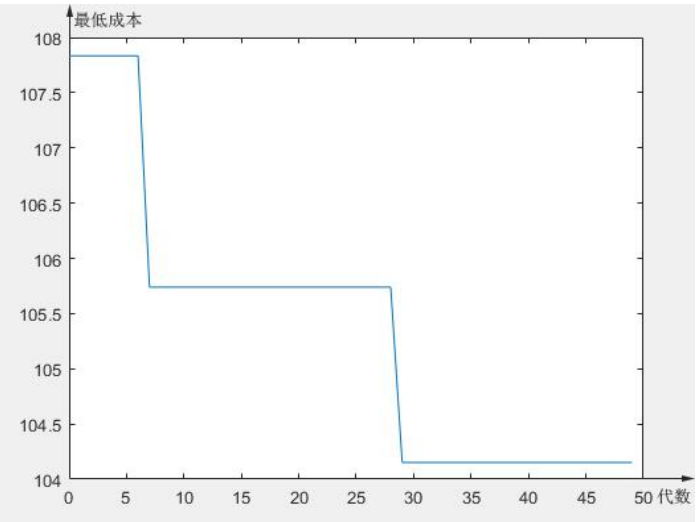


图 5.2.1 “优化”选择、交叉、变异操作之后三次样条插值法的最低成本变化曲线（50 代）

从结果中我们可以看到这样做的效果并不好，最低成本反而升高，而且从图中我们看到整个过程只出现了三个最优成本。分析后发现这样操作会严重影响后代的多样性。例如下一代的最优成本可能升高，但是最后得到的最优解可能就是需要这样一代经过交叉、变异之后才能得到。所以我们这样判断出成本升高就舍弃该操作会大大降低得到最优成本的可能，所以并不可取。

# 6 结论

三次样条插值法得到的最低成本比三次多项式拟合法的更低，时间消耗也更少。我们得到的最优解为（3,8,19,26,32,43,49），成本为 96.14325。

# 7 致谢

首先感谢袁焱老师，在上课过程中及之后的面谈解决了我们很多的疑惑。其次感谢课程助教老师，给我们的算法提供的修改意见优化了我们程序中的好多环节。在老师的帮助下，我们不但更加熟练地了 MATLAB，同时在科学探究的思想与方法上也有了很大提高。最后同时感谢为本文提出理论依据与的支持的作者，通过他们的文章，我们更加深刻地了解了遗传算法和三次样条插值法的具体实现过程，从而更加顺利地完成了本课程的任务。

最后，限于时间仓促和我们现阶段的知识水平，所写论文难免有不足之处，恳请老师批评和指正！

# 8 参考文献

[1]上海交大电子工程系. 统计推断讲座 1,2,3 ftp://202.120.39.248.

[2]MATLAB 数据拟合实用教程 ppt

[3]百度百科“三次样条插值”

http://baike.baidu.com/link?url=FsObXzyPtK1ScYHoDkRliM7aMP19nXZW2QdD0eEMx5G-k8mBkf1h1Wdl7Zrgq42hc60vHrAmFN5qv_DBa8a_1a

[4]百度百科“遗传算法”

http://baike.baidu.com/link?url=KZdKs4ltGjPyu5daV27GjrBFqdulOXOvkv81iWX9wiHwSx_FXcf2qdfZOUWgGpnjLKvpuryH0ypBvUy8NN3P7K

# 9 附录：MATLAB 程序

1. 优化前的代码（中期报告代码）

**main.m**

```
clear all;
close all;
global data;
data = csvread('20150915dataform.csv');
length=51;%每个样本的数据点数
populationsize=50;%种群大小（暂定）,偶数
maxcycle=50;%最大遗传次数（暂定）
%初始化样本，出现 1 的概率为 30%（暂定）
population=Initialfuc(populationsize,length);
cycle=1;
while (cycle<=maxcycle)
    [Fitvalue,ratelist]=fitness(population);
    population=selection(population,ratelist);%选择操作
    population=cross(population);%交叉（概率为 10%）
    population=variation(population);%变异（概率 0.1%）
 cycle=cycle+1;
end
[Fitvalue,ratelist]=fitness(population);
disp(1./Fitvalue);
disp(population);
```

**Initialfuc.m**

```
function [ ini ] = Initialfuc( populationsize,length )
%初始化
ini=zeros(populationsize,length);
for i = 1:populationsize
    for j = 1:length
        r=rand;
        if r<0.3
            ini(i,j)=1;
        end
        clear r;
    end
end
end
```

**fitness.m**

```
function [ fitvalue,ratelist ] = fitness( population )
```

```
%计算适应度，返回当前种群各单位的适应度以及他们各自的累计概率（用于随机生存、淘汰）
column=size(population,1);
fitvalue=zeros(1,column);
ratelist=zeros(1,column);
for i=1:column
    x=population(i,:);
    fitvalue(i)=costall(x);
end
fitvalue=1./(fitvalue); %暂定倒数，可以是 1/x^2 等等
rate=fitvalue/sum(fitvalue);
ratelist(1)=rate(1);
for i=2:column
    ratelist(i)=ratelist(i-1)+rate(i);
end
end
```

**costone.m**

```
function [ cost ] = costone( x,y,points )
%计算单个样本的成本 cost，x 为 x 轴坐标，y 为 y 轴坐标，points 为单个选点方案
pos=find(points);
x2=x(pos);
y2=y(pos);

p=polyfit(x2,y2,3);%三次多项式拟合
q=polyval(p,x);%在 x 处的拟合值
%q=spline(x2,y2,x);%三次样条插值
difference=abs(q-y);

sum=0;
for i=1:length(difference)
    if difference(i)>0 && difference(i)<=0.4
        sum=sum+0;
    end
    if difference(i)>0.4 && difference(i)<=0.6
        sum=sum+0.1;
    end
    if difference(i)>0.6 && difference(i)<=0.8
        sum=sum+0.7;
    end
    if difference(i)>0.8 && difference(i)<=1
        sum=sum+0.9;
    end
```

```
    if difference(i)>1 && difference(i)<=2
        sum=sum+1.5;
    end
    if difference(i)>2 && difference(i)<=3
        sum=sum+6;
    end
    if difference(i)>3 && difference(i)<=5
        sum=sum+12;
    end
    if difference(i)>5
        sum=sum+25;
    end
end
cost=sum+12*length(pos);
end
```

**costall.m**

```
function [ cost ] = costall( points )
%定标方案总成本
global data;
costlist=zeros(400,1);
for i=1:2:799
    x=data(i,:);
    y=data(i+1,:);
    costlist((i+1)/2)=costone(x,y,points);
end
cost=mean(costlist);
end
```

**selection.m**

```
function [ pop ] = selection( population,ratelist )
%选择操作
select=zeros(size(population,1));
for i=1:size(population)
    r=rand;
    j=1;
    while (r>ratelist(j))
        j=j+1;
    end
    select(i)=j;
end
pop=zeros(size(population,1),size(population,2));
for i=1:size(population)
    pop(i,:)=population(select(i),:);
```

```
end
end
```

**cross.m**

```
function [ pop ] = cross(population)
%交叉（概率为 10%）
num=size(population,1)/2;
pop=population;
for i=1:num
    r=rand;
    if r>0.9
        s=ceil(rand*51);
        pop(2*i-1,:)=[population(2*i-1,1:s),population(2*i,s+1:size(population,2))];
        pop(2*i,:)=[population(2*i,1:s),population(2*i-1,s+1:size(population,2))];
    end
end
end
```

**variation.m**

```
function [ pop ] = variation(population)
%变异（概率 0.1%）
pop=population;
for i=1:size(population,1)
    for j=i:size(population,2)
        r=rand*1000;
        if r<1
            pop(i,j)=1-pop(i,j);
        end
    end
end
end
```

2. 优化循环后的代码

**main.m**

```
clear all;
close all;
global data;
global bestpoint;
data = csvread('20150915dataform.csv');
length=51;%每个样本的数据点数
populationsize=100;%种群大小（暂定）,偶数
```

```
maxcycle=200;%最大遗传次数（暂定）
%初始化样本，出现 1 的概率为 20%（暂定）
population=Initialfuc(populationsize,length);
cycle=1;

while (cycle<maxcycle)
      [Fitvalue,ratelist]=fitness(population);
      [minfit,num]=max(Fitvalue);
      minfit= (1/minfit)^0.4;
      pri=[cycle-1,minfit];
      fprintf('第%d 层的最低成本为%f\n',pri);
      history(cycle)=minfit;
      bestway(cycle,:)=population(num,:);

      population=selection(population,ratelist);%选择操作
      population=cross(population);%交叉（概率为 50%）
      population=variation(population);%变异（概率 1%）
 cycle=cycle+1;
end
[Fitvalue,ratelist]=fitness(population);
[minfit,num]=max(Fitvalue);
minfit=(1/minfit)^0.4;
pri=[maxcycle,minfit];
fprintf('第%d 层的最低成本为%f\n',pri);
history(maxcycle)=minfit;
bestway(maxcycle,:)=population(num,:);

[minminfit,nthcycle]=min(history);
pri2=[nthcycle-1,minminfit];
fprintf('所有结果中最优解为第%d 层，成本为%f\n,取点方案为',pri2);
bestpoint=find(bestway(nthcycle,:));
disp(bestpoint);
x=0:1:maxcycle-1;
y=history(x+1);
plot(x,y);
```

**Initialfuc.m**

```
function [ ini ] = Initialfuc( populationsize,length )
%初始化
ini=zeros(populationsize,length);
a=rand(populationsize,length);
finda=find(a<0.2);
ini(finda)=1;
end
```

**fitness.m**

```
function [ fitvalue,ratelist ] = fitness( population )
%计算适应度，返回当前种群各单位的适应度以及他们各自的累计概率（用于随机生存、淘汰）
row=size(population,1);
fitvalue=zeros(1,row);
ratelist=zeros(1,row);
for i=1:row
    fitvalue(i)=costall(population(i,:));
end
fitvalue=1./(fitvalue.^2.5); %暂定 1/x^2.5，可以是 1/x 等等
rate=fitvalue/sum(fitvalue);
ratelist(1)=rate(1);
for i=2:row
    ratelist(i)=ratelist(i-1)+rate(i);
end
end
```

**costall.m**

```
function [ cost ] = costall( points )
%定标方案总成本
global data;
data_2n=data(2:2:end,:);
data_1=data(1,:);
for i=1:400
    costlist(i)=costone(data_1,data_2n(i,:),points);
end
cost=mean(costlist);
end
```

**costone.m**

```
function [ cost ] = costone( x,y,points )
%计算单个样本的成本 cost，x 为 x 轴坐标，y 为 y 轴坐标，points 为单个选点方案
pos=find(points);
x2=x(pos);
y2=y(pos);
%p=polyfit(x2,y2,3);%三次多项式拟合
%q=polyval(p,x);%在 x 处的拟合值
q=interp1(x2,y2,x,'spline');%三次样条插值
difference=abs(q-y);
sum=numel(difference,difference>0.4 & difference<=0.6)*0.1+numel(difference,difference>0.6 &
difference<=0.8)*0.7+numel(difference,difference>0.8 &
```

```
difference<=1)*0.9+numel(difference,difference>1 &
difference<=2)*1.5+numel(difference,difference>2 &
difference<=3)*6+numel(difference,difference>3 &
difference<=5)*12+numel(difference,difference>5)*25;
cost=sum+12*length(pos);
end
```

**selection.m**

```
function [ pop ] = selection( population,ratelist )
%选择操作
select=zeros(size(population,1));
for i=1:size(population)
    r=rand;
    j=1;
    while (r>ratelist(j))
        j=j+1;
    end
    select(i)=j;
end
pop=zeros(size(population,1),size(population,2));
for i=1:size(population)
    pop(i,:)=population(select(i),:);
end

end
```

**cross.m**

```
function [ pop ] = cross(population)
%交叉（概率为 50%）
num=size(population,1)/2;
pop=population;
for i=1:num
    r=rand;
    if r>0.5
        s=ceil(rand*51);
        pop(2*i-1,:)=[population(2*i-1,1:s),population(2*i,s+1:size(population,2))];
        pop(2*i,:)=[population(2*i,1:s),population(2*i-1,s+1:size(population,2))];
    end
end
end
```

**variation.m**

```
function [ pop ] = variation(population)
```

```
%变异（概率 1%）
pop=population;
r=100*rand(size(population,1),size(population,2));
pos=find(r<1);
pop(pos)=1-pop(pos);
end
```