统计推断在数模转换系统中的的应用

组号 63　戴逸飞 5140309033

摘要：为了以合理的成本对传感器进行定标而寻找合理的模型及算法。

关键词：定标，拟合，遗传算法，模拟退火算法

1.引言

随着科技日新月异的发展，各种对于不同量进行测定的传感器的要求也是越来越高。在制作传感器的过程中，对传感器进行定标的步骤也是必不可少的。在测量的过程之中总是少不了两个物理量之间的关系。对于这两个物理量之间的关系由函数建立一一映射关系。由一个物理量输入而得到另一个物理量的值。大多数两个物理量之间的函数关系都不是线性的，而是更为复杂的函数关系。但是随着测量要求的提高，我们发现很难找到一个函数，使其能够精确地描述两个物理量之间的关系。更多的情况下我们只能找到一个近似的解。不仅如此，对于某些材料，两种物理量之间的函数关系并不是确定的，他们只能由具有同一特征的某一类函数关系表示。为了对此类定标的话就不得不在测量范围之内密集地取点来拟合出一个函数进行定标。但是这在大规模生产之中明显是不现实的。这将会消耗大量的成本。

这就是本门课程所要解决的问题。在实际生产中，为了降低生产成本，我们会建立合理的数学模型，寻找应取样哪几个点并以何种函数关系进行拟合能够得到可以接受的测量精确度以及制造成本。

2.问题的提出与模型的建立

2.1 研究对象

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。

2.2 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）。

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，$i = 1, 2, 3, ..., 50, 51$。

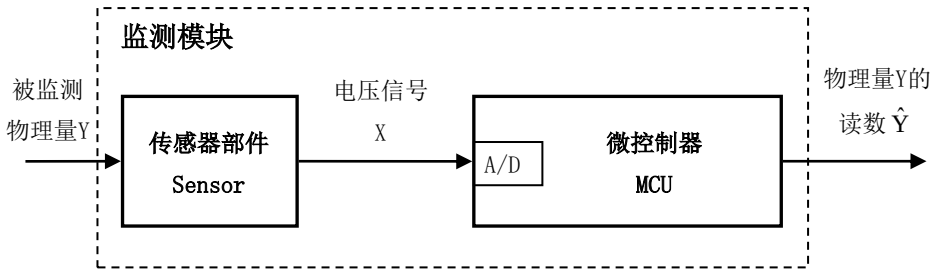

图 1 监测模块组成框图

2.3 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

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

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

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

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。

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

总成本较低的校准方案，认定为较优方案。

3.算法尝试

3.1 穷举法

考虑穷举法的时间复杂度，我们可以发现这是一个 NP-hard 问题。如果取 5 个点要尝试 C（5,51），约 2 百万次。如果取 6 个点，要尝试 C(6,61)，约 2 千万次。如果取 7 个点，要尝试 C(7,51)，约 1 亿次。如果取的点更多的话，要尝试的次数也大大增加。而这些计算次数所造成的巨大的时间成本明显是不可接受的。所以此方法不再尝试，仅仅在此列出。

3.2 模拟退火算法

模拟退火算法来源于固体退火原理，将固体加温至充分高，再让其徐徐冷却，加温时，固体内部粒子随温升变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。

以下是我使用退火算法（初稿程序）的几次试验（表 1）

初始温度	结束温度	运行时间	成本	选取点的集合
100	5	534	143	{5,5.3,6,6.5,7.3,8.3,9.8}
100	1	1677	119	{5.2,6.5,7.8,9.1,9.9}

表 1 模拟退火算法（初稿）的运行结果

接下来我来陈述一下我写退火算法的思路（图 2）。其中 l 为迭代次数。

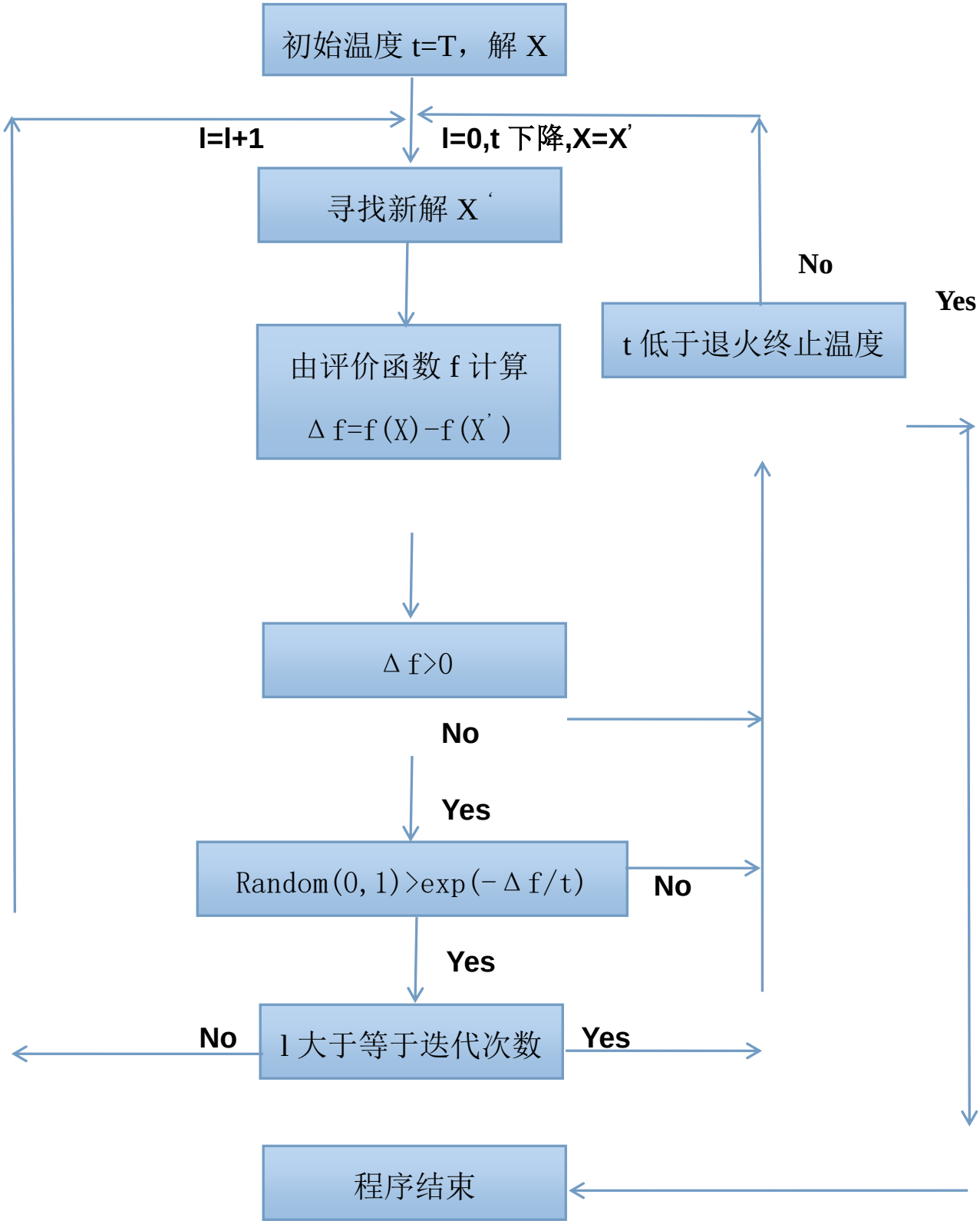

图 2 模拟退火算法流程图

对于寻找新解 X'这一步，首先我的编码方式是使用二进制编码，建立了一个 1*51 的矩阵。在提交初稿时，我是通过获取一个 1 到 51 间的随机数 x，然后改矩阵 x 处的值。不过

这种改变方式得出的结果不是很理想。每次只对一个点得到的新解可能并不能使我找到最优解而容易陷于局部最优解。原因如下：根据数学知识可知，取点数目相同的情况下当取的点分布的越宽广成本越小。而当我的解收敛到一个取点数量合适的情况时，如果需要降低成本则需要将取的点的位置变得尽可能合适。而如果每次只改变一个点所得到的解在选取点数量合适的情况下是不可能被接受的，所以被接受为新解只可能是由概率的变化来得到新解，这样会导致程序运行速度严重变慢。所以我后来采取了每次线获取一个 1 到 4 间的随机数来确定改变点的数量，然后在获得一个不重复的随机数矩阵来改变 X 获得 X ‘。这样会使新解被接受的概率大大提升，提高程序的运行速度。

以下是我使用退火算法（终稿程序）的几次试验（表 2）

初始温度	结束温度	运行时间	成本	选取点的集合
100	1	421	120	{5,1.3,2,6.3,8.2,9.7}
10	0.1	782	121	{5.2,6.3,7.1,8.3,9.9}
100	5	159	145	{5.2,6.1,7.4,7.8,8.5,9.4,9.6}
10	0.01	1302	115	{5.3,6.5,7.6,8.6,9.8}

表 2 模拟退火算法（终稿）的运行结果

通过表 2 与表 1 的比较可以发现明显终稿程序运行速度快于初稿程序。

对于评价函数 f，我是通过计算 2.3 中所提到的成本计算函数的值 x，然后为了是评价函数符合函数值越大解越能被接受，所以我取函数值为 10000-x。对于拟合方式的话，我们通过观察样本的图像（如图 3）情况可以发现样本图像普遍都分为 3 段，这符合 3 次函数的性质，所以我使用了 3 次拟合。

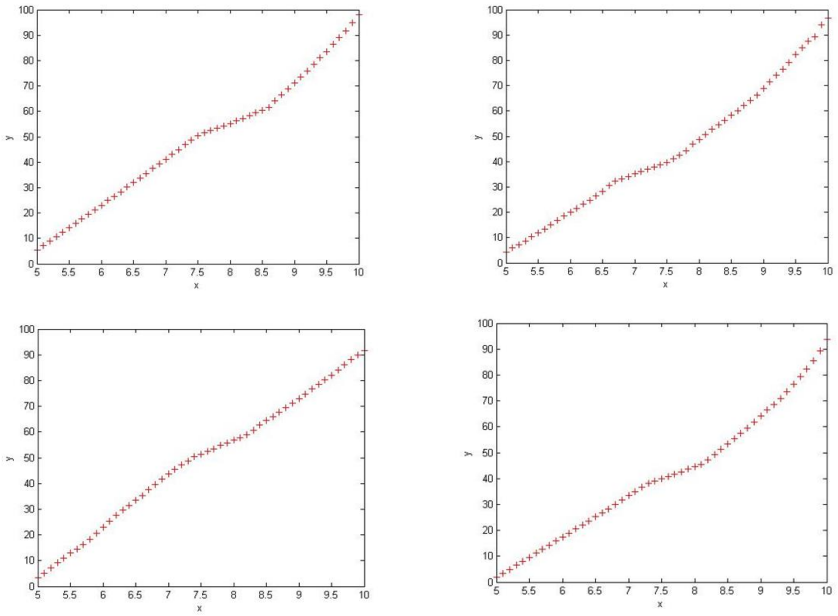

图 3 四个不同样本个体特性图示对比

还有一点就是初始温度与结束温度的选择。如果 t 的值太大而引起 $\exp(-\Delta f/t)$ 的值很小，那么会引起几乎所有的新解都是可被接受的情况，这样解就不会收敛而无法得到正确的

解。所以结束温度一定要选择在结束温度时 exp(-Δf/t)接近 1。

3.3 遗传算法

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。由于仿照基因编码的工作很复杂，我们往往进行简化，如二进制编码，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。

接下来我来陈述一下我遗传算法的思路。

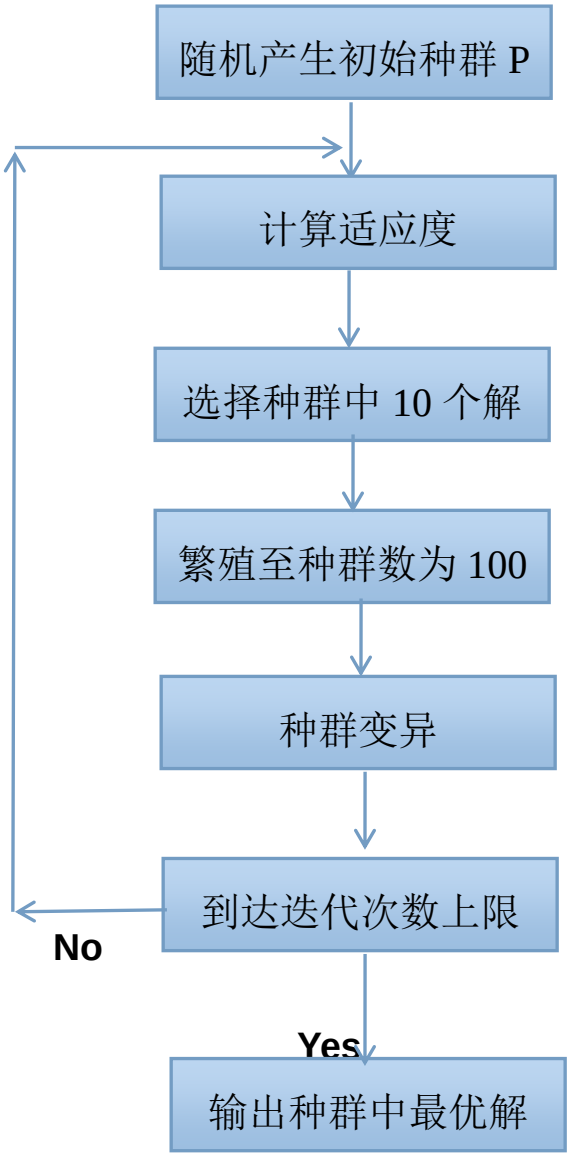

图 4 遗传算法流程图

对于适应度函数，我是通过计算 2.3 中所提到的成本计算函数的值 x，然后为了是适应度函数符合函数值越大解越能被接受，所以我取函数值为 1/x，而拟合方式的话，我在之前的退火算法中已经提到过，使用 3 次拟合。

对于选择过程，我使用俄罗斯转盘法。将每一个解的适应度函数值相加得到一个值 x 然后随机一个 0 到 x 的随机数（不一定是整数），每一个值被选中的概率就是他的适应度除以 x。

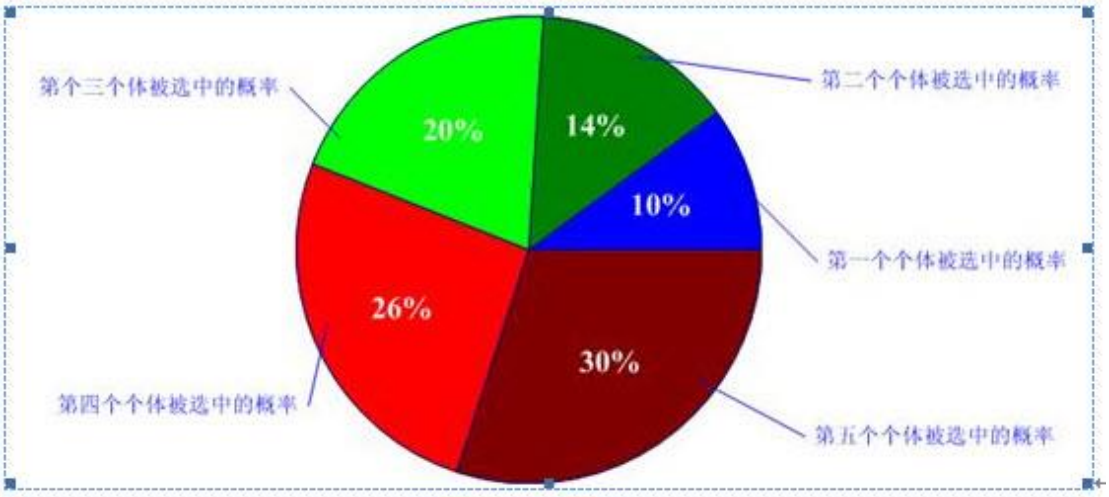

图 4 俄罗斯转盘法

对于繁殖过程，我使用的方法是染色体的单点交叉。如图 5。其中↑表示交叉点。

染色体 A：1001↑0011

染色体 B：1010↑1010

新染色体（后代）：1001↑1010

图 5 染色体的单点交叉

对于变异过程，由于大多数的变异对于结果是有害的，所以对于适应度越高的变异几率越小而且变异几率也不宜太大，所以我取变异几率为 0.01 除以适应度函数值。因为适应度大小一般为 0.01 左右，所以变异几率大约在 1%到 5%之间。

由于希望得到更低的成本，所以没有设收敛退出值。

由于代码（见附录）还存在一定问题，所以没有给出结果。

4.总结

本课程为了选取合适的点以合适的成本进行定标，通过程序计算得到了方案。由于是 NP hard 问题，所以使用了启发式算法来解决。我主要使用了模拟退火算法以及遗传算法。经过对于图像的分析，我认为可以使用 3 次曲线进行拟合，并写了两个程序的代码。其中模拟退火算法成功得到了合适的解。经过我的运算可得，由模拟退火算法得出的最佳解是取 5

个点比较合适。

5.参考文献

[1]百度百科 模拟退火算法、遗传算法词条

[2] “统计推断”课程设计的要求 V2.2 2015-9-22

附录（程序清单）

1.退火算法（初稿）

main.m

```
%统计推断 退火算法

tic;
clc;
global m;
global n;
t = 100;    %初始温度
l = 8;          %迭代次数
m = 51;
n = 400;

%初始化
global x;
global y;
data = csvread('20150915dataform.csv');
i = 2:2:800;
x = data(1,:);
y = data(i,:);
while true
    select = round(rand(1,m) * 3 / 4);
    length1 = 0;
    for i = 1:m
        if select(i)
            length1 = length1 + 1;
        end
    end
    if length1 > 3
        break;
    end
end
new = select;
length2 = length1;
%开始退火
while t > 1
    t = t * 0.98;
```

```
    %10 次迭代
    for i = 1:l
        %生成一个符合条件的新值
        while true
            change = randi(m);
            if new(change) == 0
                new(change) = 1;
                length2 = length2 + 1;
                break;
            elseif length1 > 4
                new(change) = 0;
                length2 = length2 - 1;
                break;
            end
        end
        %判断新值是否接受
        result1 = fitness(select,length1);
        result2 = fitness(new,length2);
        e = result1 - result2;
        if e < 0 || rand() < exp(-e/t)
            select(change) = new(change);
            length1 = length2;
            break;
        else
            new(change) = select(change);
            length2 = length1;
        end
    end
end
result = 10000 - fitness(select,length1);
toc;
```

Fitness.m

```
%适应度函数
function score = fitness(arr,long)
    global x;
    global y;
    global m;
    global n;
    choose = [];
    for k = 1:m
        if arr(k)
```

```
            choose = [choose k];
        end
    end
    score = 12 * long * n;
    for i = 1:n
        ynew = y(i,choose);
        fx = polyfit(x(choose),ynew,3);
        result = polyval(fx,x);
        compare = abs(result - y(i,:));
        for k = 1:m
            if compare(k) > 5
                score = score + 25;
            elseif compare(k) > 3
                score = score + 12;
            elseif compare(k) > 2
                score = score + 6;
            elseif compare(k) > 1
                score = score + 1.5;
            elseif compare(k) > 0.8
                score = score + 0.9;
            elseif compare(k) > 0.6
                score = score + 0.7;
            elseif compare(k) > 0.4
                score = score + 0.1;
            end
        end
    end
    score = score / n;
    score = 10000 - score;
end
```

2.退火算法（终稿）

main.m

```
%统计推断 退火算法

tic;
clc;
global m;
global n;
t = 10;    %初始温度
```

```
l = 8;          %迭代次数
m = 51;
n = 400;

%初始化
global x;
global y;
data = csvread('20150915dataform.csv');
i = 2:2:800;
x = data(1,:);
y = data(i,:);
while true
    select = round(rand(1,m) * 3 / 4);
    length1 = 0;
    for i = 1:m
        if select(i)
            length1 = length1 + 1;
        end
    end
    if length1 > 3
        break;
    end
end
%开始退火
me = select;
new = select;
while t > 0.01
    t = t * 0.98;
    %l 次迭代
    for i = 1:l
        %生成一个符合条件的新值
        while true
            time = randi(4);
            while true
                change = randi(m,1,time);
                flag = 0;
                for j = 2:time
                    for k = 1:(j - 1)
                        if change(j) == change(k)
                            flag = 1;
                        end
                    end
                end
                if flag == 0
```

```
                break
            end
        end
        new(change) = 1 - new(change);
        add = 0;
        if new(change) == 0
            add = add - 1;
        else
            add = add + 1;
        end
        length2 = length1 + add;
        if length2 > 3
            break
        end
    end
    %判断新值是否接受
    result1 = fitness(select);
    result2 = fitness(new);
    e = result1 - result2;
    if e < 0 || rand() < exp(-e/t)
        select(change) = new(change);
        length1 = length2;
        break;
    else
        new(change) = select(change);
    end
  end
end
result = 10000 - fitness(select);
toc;
```

Fitness.m

```
%退火适应度函数
function score = fitness(arr)
  global x;
  global y;
  global m;
  global n;
  long = 0;
  for i = 1:m
    if arr(i) == 1
        long = long + 1;
```

```
        end
    end
    choose = zeros(1,long);
    i = 1;
    for k = 1:m
        if arr(k)
            choose(i) = k;
            i = i + 1;
        end
    end
    score = 12 * long * n;
    for i = 1:n
        ynew = y(i,choose);
        fx = polyfit(x(choose),ynew,3);
        result = polyval(fx,x);
        compare = abs(result - y(i,:));
        for k = 1:m
            if compare(k) > 5
                score = score + 25;
            elseif compare(k) > 3
                score = score + 12;
            elseif compare(k) > 2
                score = score + 6;
            elseif compare(k) > 1
                score = score + 1.5;
            elseif compare(k) > 0.8
                score = score + 0.9;
            elseif compare(k) > 0.6
                score = score + 0.7;
            elseif compare(k) > 0.4
                score = score + 0.1;
            end
        end
    end
    score = score / n;
    score = 10000 - score;
end
```

3.遗传算法

main.m

```
%统计推断 遗传算法

tic;
clc;

global m;
global n;
global P;
global x;
global y;
m = 51;                                    %51 个数据点
n = 400;                                   %400 次测量
data = csvread('20150915dataform.csv');
i = 2:2:800;
x = data(1,:);                              %x 坐标（一行）
y = data(i,:);                              %y 坐标（全部）
T = 0;                                     %最大迭代次数
P = 100;                                   %种群大小
select_num = 10;                           %繁殖后代个数

p = round(rand(P,m));                      %生成初始种群

for i = 0:T
    selection = select(p,select_num);
    new = crossover(selection);
    p = mutation(new);
end

fit = fitness(p);
best = 999999;
line = 0;
for i = 1:m
    if fit(i) < best
        best = fit(i);
        line = i;
    end
end

result = p(line,:);
score = fit(line);

toc;
```

crossover.m

```
function new = crossover(selection)
global m;
global P;
select_num = size(selection,1);
new = zeros(P,m);
new(1:select_num,:) = selection;
for i = (select_num + 1):P
    a = randi([1 select_num]);
    b = randi([1 select_num]);
    position = randi([2 m]);
    new(i,1:position - 1) = selection(a,1:position - 1);
    new(i,position:m) = selection(b,position:m);
end
end
```

fitness.m

```
%遗传适应度函数
function result = fitness(arr)
    global x;
    global y;
    global m;
    global n;
    length = size(arr,1);
    result = zeros(1,length);
    for j = 1:length
        long = 0;
        for i = 1:m
            if arr(j,i) == 1
                long = long + 1;
            end
        end
        choose = zeros(1,long);
        i = 1;
        for k = 1:m
            if arr(j,k)
                choose(i) = k;
                i = i + 1;
            end
        end
        score = 12 * long * n;
        for i = 1:n
            ynew = y(i,choose);
```

```
            fx = polyfit(x(choose),ynew,3);
            result = polyval(fx,x);
            compare = abs(result - y(i,:));
            for k = 1:m
                if compare(k) > 5
                    score = score + 25;
                elseif compare(k) > 3
                    score = score + 12;
                elseif compare(k) > 2
                    score = score + 6;
                elseif compare(k) > 1
                    score = score + 1.5;
                elseif compare(k) > 0.8
                    score = score + 0.9;
                elseif compare(k) > 0.6
                    score = score + 0.7;
                elseif compare(k) > 0.4
                    score = score + 0.1;
                end
            end
        end
        score = n / score;
        result(j) = score;
    end
 end
```

select.m

```
function selection = select(p,select_num)
global P;
fit = fitness(p);
new = zeros(1,select_num);
tree = zeros(1,P);
all = 0;
for i = 1:P
    all = all + fit(i);
    tree(i) = all;
end
choose = rand(1,select_num) * all;
for i = 1:select_num
    choose = rand(1,select_num) * all;
    for j = 1:P
        if tree(j) > choose
            new(i) = j;
        end
```

```
    end
end
selection = p(new,:);
end
```

mutation.m

```
function result = mutation(new)
global P;
global m;
fit = fitness(new);
fit = 0.01 ./ fit;
for i = 1:P
    choose = rand() * 100;
    if choose < fit(i)
        time = floor(rand() * 4 + 1);
        change = round(rand(1,time) * (m - 1) + 1);
        new(i,change) = 1 - new(i,change);
    end
end
result = new;
end
```