统计推断在数模转换系统中的应用

组号 28　赵忆漠 （5130309297）唐炜杰（5130309493）

摘要：统计推断是数模转换系统中的重要应用方法。本课题探究了快速、准确地为样品定标的方法，以确定其输入信号与输出信号的关系。我们使用 MATLAB 进行程序设计，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。
关键词：统计推断；定标；三次样条插值法；多项式拟合法；遗传算法；MATLAB

Application of Statistical Inference in DA System

Abstract: Statistic inference is the essential application in the Digital to analog conversion system.In this task,we search for the method of how to determine the Y-Y' relationship of a sample.We want to get to the best scheme by determing the fixed points by genetic algorithm and using defferent interpolation in MATLAB program design.
Key Words: Statistic inference,Calibration,Three times spline interpolation method,Polynomial fitting,Genetic algorithm.

1. 引言

1. 1 课题内容的提出

本课题研究围绕可调电源模块的输入输出受控关系展开。研究的实际物理对象是某型号产品内部的监测模块如图 1 所示，其输入为 **Y** 的信号，输出为电源输出电压 **Y'**，本课题研究的就是如何确定一个产品的 **Y-Y'**关系。

图 1　监测模块简化模型

1. 2 课题最终目标

本课题的最终目标是：找到一个定标方案，要求其经过给定的定标准确度评价函数在所有可行方案中评价最高。

1. 3 课题研究的主要步骤

(1) 初步分析样本数据。

(2) 确定一种插值或拟合的方法。

(3) 使用搜索算法，确定合适的测量点个数及位置，即找到定标准确度足够高的一个定标方案。

(4) 分析结果。优化或更换插值或拟合的方法。优化或更换搜索算法。

(5) 重复(2)(3)(4)，直到得到满意的定标方案。

2 样本数据分析

在所给的实验原始数据样本中，共有 469 件样品，每个样品提供了 51 个测量点，测量点的输出值 Y’ 的范围均大致在 5 至 10，大致每 0.1 提供一个测量点，而输入值的范围则不确定。如图 2 所示，通过对不同样品的函数曲线观察可以总结出函数关系有以下几个特点:(1) 单调性 (2) 在多数样品中函数表现为光滑曲线。(3) 曲线具有非线性或局部非线性。(4)不同样品间存在较大差异。

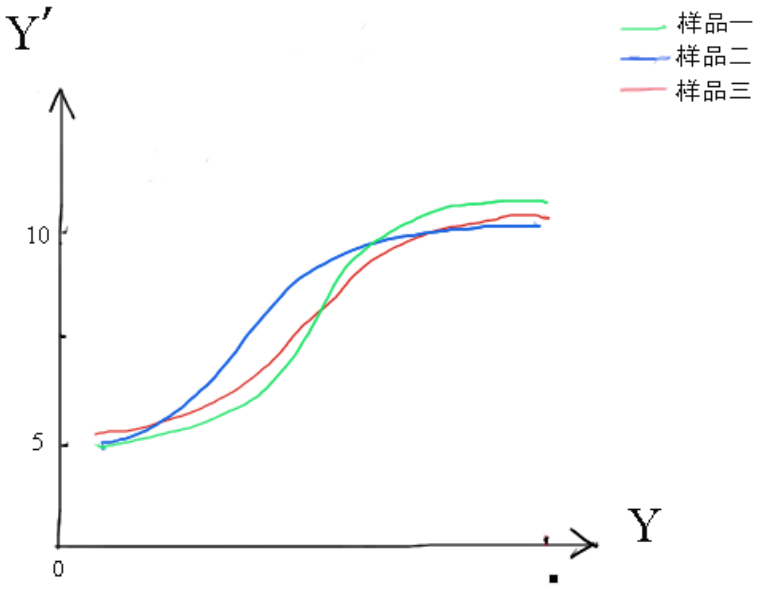

图 2　两个样品的函数曲线

3 定标准确度评价方法

定标准确度评价函数

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

$S_i=\sum_{j=1}^{51}s_{i,j}+q.n_i$　（其中，q=12 ）　　(3-2)

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

公式 (3-1) 中下标 i 代表样品序号，下标 j 代表观测点序号。

公式 (3-2)代表第 i 件产品的成本，第一项为误差成本，第二项为测量成本。

公式 (3-3) 中 代表定标方案对整个样本的定标准确度，也即平均成本。

4 插值的方法

由样本数据的定性分析可知，测量是存在误差的，某些样品的函数曲线存在不光滑或偏差较大的观察点。本段实验选择三次样条插值和多项式拟合的方法。

4. 1 三次样条插值

4.1.1 方法概述

三次样条插值法一种非线性插值法，它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。

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（边界点的导数为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）。

图 4 三次样条插值简单示意图

4.1.2 优缺点分析

优点：插值所得曲线为光滑曲线；且曲线的导数曲线依然光滑。插值所得曲线通过每一个所取观测点，使得观测点的估计值等于观测值，无需对其另外处理。而且真实曲线的形态，对插值所得曲线与真实曲线的吻合度影响较小。

缺点：运算量较大，速度较慢。

4. 2 多项式拟合

4.2.1 方法概述

多项式拟合是一种最为基本的拟合方式。对给定数据（x_i，y_i），在给定的函数类Φ中，求 $p(x)\in\Phi$，使误差 $r_i=p(i)-y_i$ 为最小，这样求得的函数 p(x)就是拟合出来多项式函数。

4.2.2 优缺点分析

优点:运算速度快，易于理解，可以很方便的进行操作和分析。拟合的效果可能产生极其符合实验点的拟合函数

缺点：拟合的精度不能有平均的保证。真实 Y-Y’曲线的形状对拟合曲线的拟合精度影响较大。

5 启发式搜索

关于 7 个观测点的取法问题，可以通过搜索算法来完成。对于本课题，观测的取法共有约 1.54 亿种不同的组合，故此问题属于 NP 问题 (Non-deterministic Polynomial)， 因此无法使用穷举搜索的方法来求解此问题，但仍然可以采用启发式搜索来进行求解。 目前本小组采用遗传算法来进行求解。

5.1 遗传算法

遗传算法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）演化而来的随机化仿生搜索方法。以下简称遗传算法为 GA。

5.1.1 GA 基本框架

GA 将每一个可能的解看做一个生物个体，将若干生命个体组成种群。算法中需要对解进行编码，以模仿生物的基因染色体。GA 的流程图如图 4 所示。算法开始运行时，GA 将随机生成一个含有一定数量个体的初始种群。之后，GA 将计算种群中的个体的适应度，然后 GA 将模拟自然选择过程，下一步 GA 模拟生物交配繁殖，对两 个基因代码进行交叉，产生两个新个体。然后再模拟基因突变，最终产生新一代的种群。重 复本段所讲述过程直到满足终止条件即可。

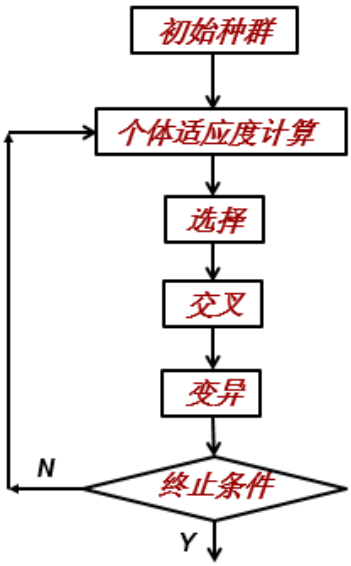

图 5 遗传算法流程图

5.1.2 本课题中 GA 的 MATLAB 实现

本文将罗列几个关键函数的实现过程，详细源代码请见附录。本次实验中将每一种 7 个点的取法作为一个生物个体。 其中一些重要参数及变量有：

popSize：代表种群大小；

pop：代表种群列表；

fitnessTable：代表适应度列表；

crossRate：代表交叉概率；

mutateRate：代表变异概率；

maxGeneration：代表最大子代数量；

以下为算法的具体实现：

(1) 编码方式 将 7 个样本点在全部 51 个观察点中所处的位置序号，编码为一个含有 7 个不重复整数的数列，数列中的整数按升序排列，将此数列作为个体的基因染色体。

(2) 初始化种群函数 init.m。函数随机生成 popSize 个生物个体的染色体，储存于种群列表 pop 内，作为初始种群。

(3)适应度函数 fitness.m。将平均成本作为一个解的适应度。函数对 pop 内的每个染色体运用计算成本的函数进行计算成本，并将其得分储存在 fitnessTable 内。

(4) 选择过程函数 selection.m。函数根据适应度选择将要繁殖后代的生物个体，储存进 pop 作为下一代种群的亲代。第 i 个生物个体被选中的概率为：

$$p_i = \frac{第i个个体的适应度}{总适应度}$$

函数先产生一个 fitnesssum 的列表，含有 popsize 个元素，其中：

$$\text{fitnesssSum(i)} = \sum_{k=1}^{i} fitnessTable(k)$$

之后，函数每次生成一个浮点随机数 0≤r≤总适应度。并用二分法寻找最小的 x，满足 fitnessSum(i)≥r，再将 pop 中第 x 个元素拷贝进入新的种群 newPop 中。重复此操作 popsize 次，再令 pop=newPop。

(5) 交叉过程 cross.m 函数。由于选择过程中产生的亲代的顺序是随机的，故本函数直接对相邻的两个亲代生物的基因染色体在随机位置进行单点交叉，两个亲代生物间发生交叉的概率为 crossRate。交叉可能会出现一个染色体中出现重复的整数的情况，故必须对每个交叉产生的染色体使用 checkPop.m 函数进行校验并排序。若一个染色体中出现重复的数字，则将多余的重复数字赋值为一个 1 至 51 间的随机整数，并递归调用 checkPop.m 再次检验该染 色体，直到其变为合法染色体为止。

(6) 变异过程 mutation.m 函数。 一个染色体发生变异的概率为 mutateRate。变异的方法是，随机选定染色体内的一个整数，随机尝试对其+1 或-1。若产生的新染色体仍然合法，变异便完成；否则，则变异不发生。

(7) 主函数及终止条件终止条件被包含在主函数 main.m 内。主函数调用 readData.m 函数读取样本数据。然后调用 init.m 生成初始种群并调用 fitness.m 计算其适应度。之后循环调用 selection.m、cross.m、mutation.m、fitness.m，反复生成新一代种群并计算适应度，直到满足终止条件。 最后输出整个过程中所得的最优解，即平均成本最低的 7 个取样点的序号。

(8) 目前使用的参数设置

popSize=100; crossRate=0.9; mutateRate=0.5; maxGeneration=300; minCost=140;

6 改变参数和方法的研究

6.1 7 个取样点的优秀方案

通常，算法开始运行后，经过 30 分钟，可完成 200 代的循环。本小组采用 5.1.2 中所述参数设置，同时并行运行 3 次 GA，所得结果如表 1 所示。 其中最优秀的定标方案是：选取第 3，9，18，26，34，43，49 个观察点作为事前观察点，再利用三次样条插值法进行定标。

表 1 三次样条插值优秀定标方案

使用 三次样条插值	最优方案	平均成本
1	3，8，18，25，32，43，50	94.9904
2	2，9，20，26，34，43，50	93.8998
3	2，9，20，26，34，43，50	93.8998

6.2 减少、增多取样点的研究

依次减少一个取样点，观察成本的变化。本组更改了程序，增添一个改变取点个数的大循环，实现探究取点个数对成本变化的影响（具体实现详见附录代码）。表 2 为三次运行后的最优方案。

表 2　5、6、8 个取样点下的最优方案

使用三次样条插值	5 个取样点	6 个取样点	8 个取样点
1	103.6205	92.8774	102.0267
2	103.6205	92.8774	102.0267
3	103.6205	92.8774	102.0267

由表格可以看出，平均成本随着取点数目的增多先减小后增大，适当的减少取样点可以降低成本，本试验中，在取 6 个点的时候，cost 达到最小值，为 92.8774。其中优秀的取点方案为{3,12,22,31,43,50} 。

6.3 更换拟合或插值的方法的研究

本组将三次样条插值更换为多项式拟合，来研究更换插值或拟合的方法对降低成本的影响，表 3 为取 7 个观察点时多项式拟合下两次运行的最优方案。

表 3　多项式拟合下两次运行的最优方案

使用多项式拟合	最优方案	平均成本
1	2，9，20，26，33，44，50	94.0757
2	2，8，20，26，33，43，，50	94.1812

由表 1 和表 3 可看出，大多数时候多项式拟合的精度不如三次样条插值高。

7 结论

本小组采用 5.1.2 中所述参数设置，同时并行运行了 3 次 GA，所得最终结果如表 4 所示。 其中最优秀的定标方案是：选取第{ 3、12、22、31、43、50}观察点作为事前观察点，再利用三次样条插值法进行定标，可以使定标的平均成本达到 92.8774。

表 4　最终的优秀方案

使用多项式拟合	最优方案	平均成本
1	3，12，22，31，43，50	92.8774
2	3，12，22，31，43，50	92.8774

使用上述 MATLAB 实现的遗传算法可以成功的找到若干种定标平均成本在 95 以下的定标方案。其中拟合或插值方法使用三次样条插值方法，该方法相对于多项式拟合以及三次 hermite 插值法更为优秀，兼具运行效率和准确度。

为了能使结果更加直观化，我们以直方图的形式表示出了最终最优解的单个样本的计算成本 vs 成本数、所有点的误差绝对值统计、按评分标准的所有点的误差绝对值统计、所有样本的平均误差、所有样本的最大误差五个变量。结果如下：

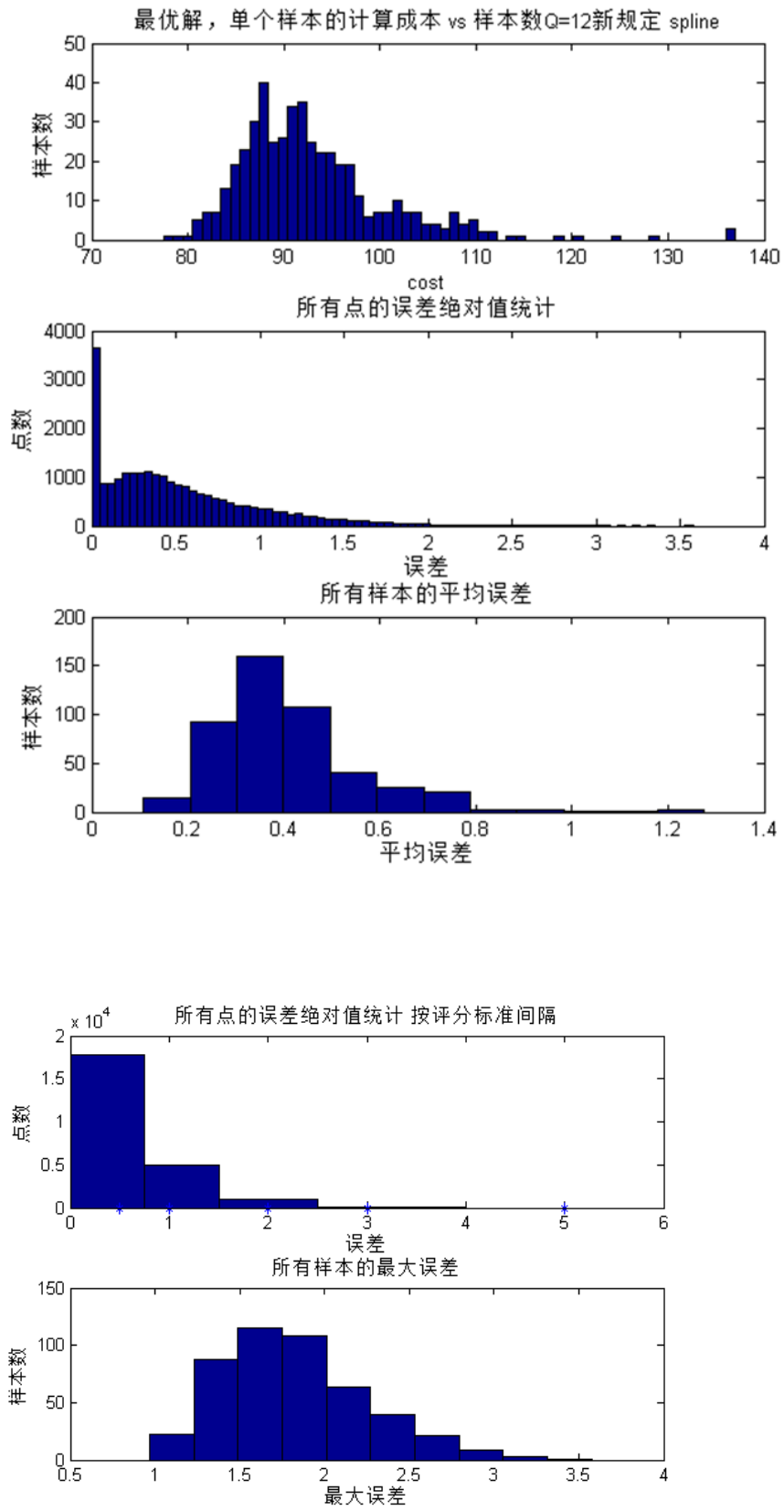

图 6　6 个点最优结果直方图

8 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] 三次样条插值_百度百科
http://baike.baidu.com/link?url=hawlHS9fng6mXe26tnK1BL9rJqS03x02LYb139acnZNa4ZdrI__P4jOBfO6PV8ek8sS9O_K0ltb0RtBlHw09ta
[4] Matlab 中插值函数汇总和使用说明
http://www.cnblogs.com/yujunyong/archive/2011/05/10/2042335.html
[5] 遗传算法_百度百科
http://baike.baidu.com/link?url=ja1jVPDGw1qtukHIPg0wGLO2TLvSF3F_ATJyenpz2hhffzh T-dTi2cRPDNWbPyDz
[6]《Matlab 遗传算法工具箱及应用》 雷英杰著
[7]《Matlab 语言及实践教程》 朱衡君著

附录：

附录中包含了所有本实验中的代码。

1 main.m

```
%使用三次样条插值的遗传算法
%运行过程中显示的 tmpMin 表示当前种群中的最高适应度
%                  tmpAvg 表示当前种群中的平均适应度
%                  minCost 表示目前整个运行中出现过的最高适应度
%                  minCostMethod 表示上述最高适应度对应的事前观察点
%                  gen 表示当前种群的子代数

clear;

global pop;                  %取观察点种群
global fitnessTable;       %适应度列表
global x;                %样本 X
global y;            %样本 Y
global len;                  %样本组数量

popSize=100;                 %种群大小
codeSize=6;                  %观察点数量
crossRate=0.9;               %交叉概率
mutateRate=0.5;              %变异概率
maxGeneration=300;         %最大子代数
leastCost=20*codeSize;               %最低成本
minCost=500+20*codeSize;                   %最低可行成本初始为最高成本值
GGAP=0.95;                    %代沟

rand('state',sum(100*clock));

readData();
init(popSize,codeSize,51);
fitnessTable=zeros(popSize,1);
for i=1:popSize
    fitnessTable(i)=fitness(pop(i,:));
end
tmpMin=min(fitnessTable)
tmpAvg=mean(fitnessTable)
if tmpMin<minCost
    minCost=tmpMin;
    minCostLocation=find(fitnessTable==tmpMin);
    minCostMethod=pop(minCostLocation,:);
end
```

```
minCost
minCostMethod
if minCost==leastCost
    return;
end
gen=1;
while gen<=maxGeneration
    for i=1:popSize
        fitnessTable(i)=500+20*codeSize-fitnessTable(i);
    end
    selection(popSize,codeSize);
    cross(popSize,codeSize,crossRate);
    mutation(popSize,codeSize,mutateRate);
    for i=1:popSize
        fitnessTable(i)=fitness(pop(i,:));
    end
    tmpMin=min(fitnessTable)
    tmpAvg=mean(fitnessTable)
    if tmpMin<minCost
        minCost=tmpMin;
        minCostLocation=find(fitnessTable==tmpMin);
        minCostMethod=pop(minCostLocation,:);
    end
    minCost
    minCostMethod
    if minCost==leastCost
        break;
    end
    gen
    gen=gen+1;
end
```

2 init.m

```
function init(popSize,codeSize,maxNum)
global pop;
pop1=zeros(popSize,codeSize);
for i=1:popSize
    for j=1:codeSize
        pop1(i,j) = randi(maxNum);
        k=1;
        while (k<=j-1)
            if (pop1(i,j) == pop1(i,k))
```

```
                pop1(i,j) = randi(maxNum);
                k=0;
            end
            k=k+1;
        end
    end
end
pop=sort(pop1,2);
end
```

3 selection.m

```
function [] = selection(popSize,codeSize)
global pop
global fitnessTable;
fitnessSum=fitnessTable;
for i=2:popSize
    fitnessSum(i)=fitnessSum(i-1)+fitnessTable(i);
end
popNew=zeros(popSize,codeSize);
for i=1:popSize
    r=rand()*fitnessSum(popSize);
    left=1;
    right=popSize;
    mid=round((left+right)/2);
    while 1
        if r>fitnessSum(mid)
            left=mid;
        else
            if r<fitnessSum(mid)
                right=mid;
            else
                popNew(i,:)=pop(mid,:);
                break;
            end
        end
        mid=round((left+right)/2);
        if (mid==left)||(mid==right)
            popNew(i,:)=pop(right,:);
            break;
        end
    end
end
pop=popNew;
```

4 mutation.m

```
function mutation(popSize,codeSize,mutateRate)
global pop;
for i=1:popSize
    r=rand();
    if r<mutateRate
        r=randi(codeSize);
        if rand()>0.5
            x=1;
        else
            x=-1;
        end
        tmp=pop(i,r)+x;
        if (r==1)
            if (tmp>=1)&&(pop(i,r+1)~=tmp)
                pop(i,r)=tmp;
            end
        elseif (r==codeSize)
            if (tmp<=51)&&(pop(i,r-1)~=tmp)
                pop(i,r)=tmp;
            end
        else
            if (pop(i,r-1)~=tmp)&&(pop(i,r+1)~=tmp)
                pop(i,r)=tmp;
            end
        end
        %checkPop(i,codeSize);
    end
end
end
```

5 cross.m

```
function [] = cross(popSize,codeSize,crossRate)
global pop;
for i=1:2:popSize
    r=rand;
    if r>crossRate
        continue;
    end
    p=randi([2,codeSize]);
    tmp=pop(i,p:codeSize);
```

```
        pop(i,p:codeSize)=pop(i+1,p:codeSize);
        pop(i+1,p:codeSize)=tmp;
        checkPop(i,codeSize);
        checkPop(i+1,codeSize);
end
end
```

6 checkpop.m

```
function checkPop(n,codeSize)
global pop;
pop(n,:)=sort(pop(n,:));
flag=0;
for j=2:codeSize
        if pop(n,j-1)==pop(n,j)
                pop(n,j)=randi(51);
                flag=1;
        end
end
if flag
        checkPop(n,codeSize);
end
end
```

7 readData.m

```
function readData()
origin=csvread('20141010dataform.csv');
global x;
global y;
global len;
len=length(origin)/2;
x=origin(1:2:(len)*2-1,:);
y=origin(2:2:(len)*2,:);
end
```

8 fitness.m

```
function [cost] = fitness(method)
global x;
global y;
global len;
codeSize=length(method);
sum_A=0;
X=zeros(1,codeSize);
Y=zeros(1,codeSize);
for i=1:len
```

```
        A=0;
        for j=1:codeSize
            X(j)=x(i,method(j));
            Y(j)=y(i,method(j));
        end
        Y1=interp1(X,Y,x(i,:),'spline');%三次样条插值运算
        %p=polyfit(X,Y,3);%三次多项式拟合
        %Y1=polyval(p,x(i,:));%三次多项式拟合
        YTable=abs(Y1-y(i,:));
        for j=1:51
            e=YTable(j);
            if e<=0.5
                A=A+0;
            elseif e<=1
                A=A+0.5;
            elseif e<=2
                A=A+1.5;
            elseif e<=3
                A=A+6;
            elseif e<=5
                A=A+12;
            else
                A=A+25;
            end
        end
        %A=max(A-2,0);
        sum_A=sum_A+A;
    end
    cost=sum_A/len+12*codeSize;
    end
```