统计推断在数模转换系统中的应用

——传感模块的传感特性校准方案

徐绍夫 5131309506 肖扬 5130309509

摘要：本文详述了关于传感模块的传感特性定标校准方案，根据出厂的样本实测的数据，运用均匀取点、模拟退火算法，计算校准方案的总体成本，取出最优的校准方案。为该模块的批量生产提供参考。

关键词：遗传算法，传感特性校准，拟合函数

1、 引言

新技术革命的到来，世界开始进入信息时代，再利用信息的过程中，首先要解决的就是获取准确可靠的信息，而传感器是获取自然和生产领域中信息的主要途径和手段。在现代工业生产尤其是自动化生产中，要用各种传感器来监视和控制生产中的各个参数，使设备工作在正常状态获最佳状态，并使产品达到最好质量。因此可以说，没有众多的传感器，现代化生产也就失去了基础。而如何让一个传感器变得可用呢。我们必须把引起传感器性质变化的物理量，通过一定的方法转化为另一种可观察可记录的物理量，通常来说输出都是电信号，因为电信号便于处理与运算。在转化的过程中，我们需要了解该性质变化与输出信号之间的对应关系，才能通过输出来观察传感器检测量的变化，本次研究的传感器的输出与输入呈现明显的非线性性，故需要进行运用不同的算法对其输出与输入之间的关系进行校准（及定标）。

2、 传感特性校准的理论基础

2.1、监测模块模型

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）。

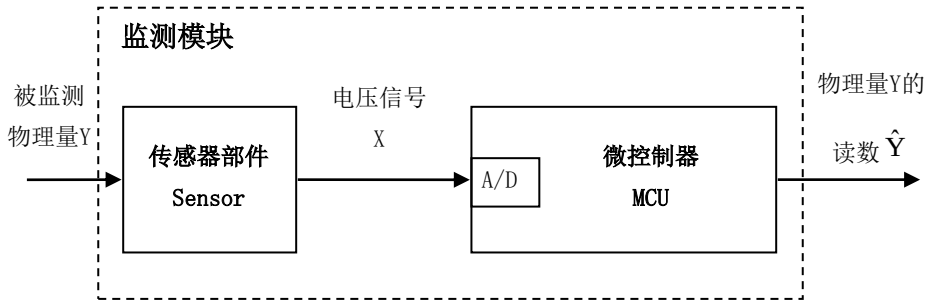


图 1 监测模块组成框图

将上述输出电压信号 X 与获得信号 $\hat{Y}$ 可记录在一个表格中进行处理。

2.2、传感器部件特性

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。

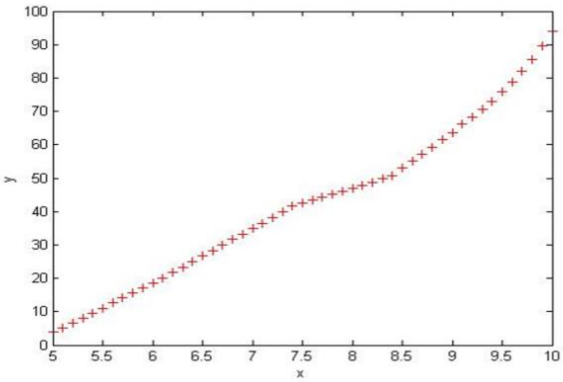


图 2　传感特性图示

2.3、三次样条插值

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。如图 3 所示，三次样条插值法每次选取相邻的 4 个点确定一条三次曲线，再取出中间两个点之间的三次曲线作为样条，然后借助样条来计算出插值点的估计值。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）。

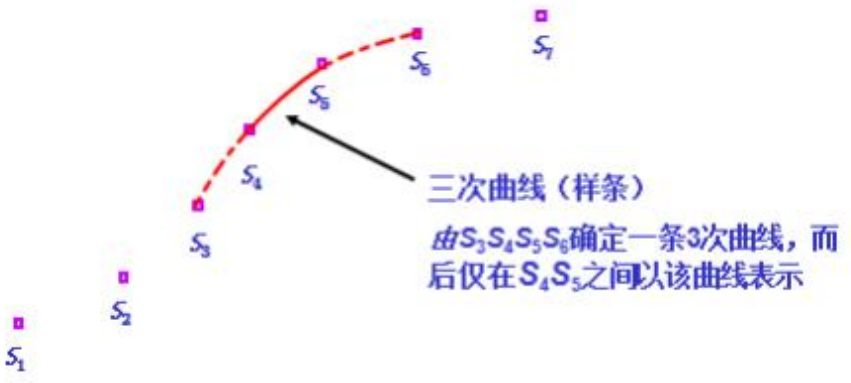


图 3　三次样条插值法示意图

2.4、遗传算法

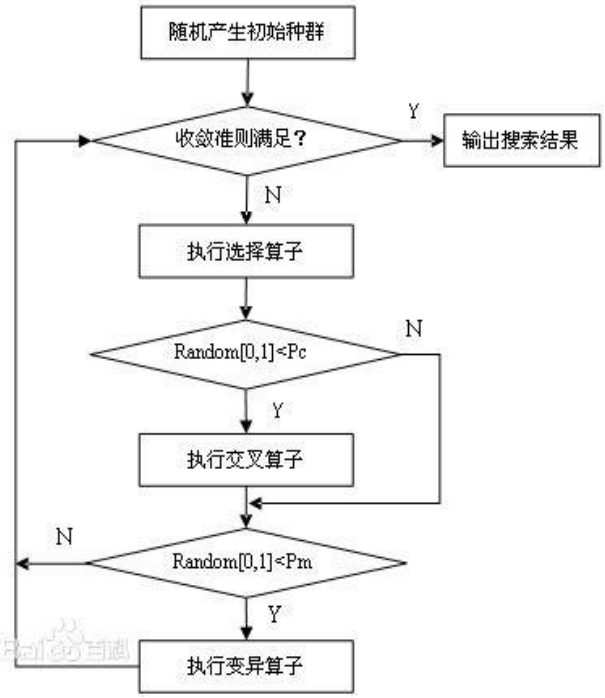
遗传算法（Genetic Algorithm）是模拟达尔文的遗传选择和自然淘汰的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法，简称 GA。

遗传算法是从代表问题可能潜在的解集的一个种群（population）开始的，而一个种群则由经过基因（gene）编码的一定数目的个体(individual)组成。每个个体实际上是染色体(chromosome)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（generation）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（fitness）大小选择（selection）个体，并借助于自然遗传学的遗传算子（genetic operators）进行组合交叉（crossover）和变异（mutation），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（decoding），可以作为问题近似最优解。

遗传算法的基本运算过程如下：

a) 初始化：设置进化代数计数器 g=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)。
b) 个体评价：计算群体 P(g)中各个个体的适应度。
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。
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。
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。
f) 群体 P(g)经过选择、交叉、变异运算之后得到下一代群体 P(t+1)。



2.5、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

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

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

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

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。

$$C=\frac{1}{M}\sum_{i=1}^{M}S_i \quad (2\text{-}3)$$

总体成本较低的校准方案，认定为较优方案。

3、 制定传感模块的传感特性校准方案的过程

3.1、选取合适的拟合函数

从数据库里选出 5 个样本，依据实验数据描点作图，观察曲线走势，曲线斜率有明显的先升后降再升的趋势，因此决定进行三次曲线拟合（这里仅列出两幅样本的图像）

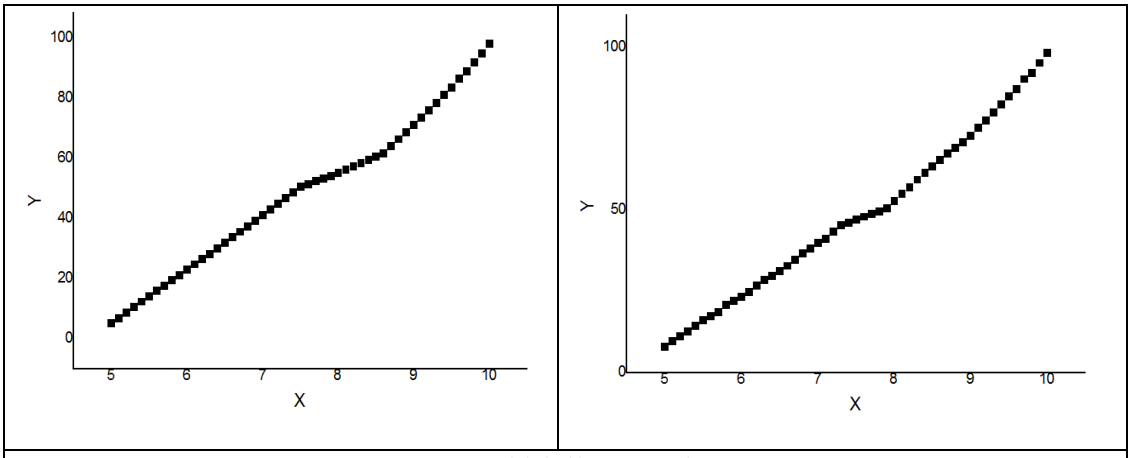


图 4　样本数据的图像

3.2、对样本中取点方法的讨论

本次取点，除了第一个点与最后一个点外，在 49 个点中选取 14 个点进行拟合，因拟合组合数目庞大，可视为每次取点都不相同。对于取点的讨论我们讨论了三中取点方式，前两种，均匀随机取点和分段随机取点作为易于理解的取点方式进行尝试；后一种，遗传算法方式作为本文重点进行讨论。

3.2.1 均匀随机取点法

即是将 49 个测试点分为 14 段，每段随机取出一个点作为需要拟合的样本点。

表一 均匀随机取点

测试点	2-4	5-8	9-11	12-15	16-18	19-22	23-25
取点	A	B	C	D	E	F	G
测试点	26-29	30-33	34-36	37-40	41-43	44-47	48-50
取点	H	I	J	K	L	M	N

穷举法计算总数为为 12^7=32831808，因为当试验次数足够多后，就不能在认为每次取点都不同，故经探讨认为，可只进行 10^8 次试验，大大减少了实验次数。

对运用均匀随机取点法得到的结果进行三次曲线拟合，随后计算此个样本的样本个体定标成本 S_i，随后将后续的样本按照同样方法取点，并且分别计算各自的 S_i，在 csv 文件中，共提供了 469 个样本，则对这 469 个样本进行同种取点方法拟合后，计算校准方案总体成本 C，最终得到当取点为 1，3,8,12,15,20,22,24,27，30,33,37,42,46,50,51 时，拟合函数为：

$$Y=0.471X^3+9.323X^2+78.888X-206.957$$

此时校准方案总体成本 C=329.73，其中单点测定成本 $q*n_i=16*20=320$，误差成本=9.73。

3.2.2 分段均匀随机取点

观察曲线走势，曲线的拐点大致在第 21 点和第 31 点，以这两点为分界线，随机取点,在每段内分别均匀取点

表二 分段随机均匀取点

测试点	2-21	22 -31	32-50
取点	A,B,C,D,E	F,G,H,I	J,K,L,M,N

穷举共 4^10*36=37748736，同样做 10^8 次试验。

对数据提供的 469 组数据拟合后定标并计算校准方案总体成本 C 发现，当取点为 1，

3,8,12,15,20,22,24,27，30,33,37,42,46,50,51 时，得到最低成本，此时

$Y=0.471X^3-9.323X^2+78.888X-206.957$

得到较优方案，C=329.38（20*16=320 为测定成本，9.38 为误差成本）。

3.2.3 启发式搜索——遗传算法取点

因为遗传算法的优点在于短时间内找出较优解，而非在上亿种组合中找出最优的解，所以在探讨以及几次试验之后，我们认为可以将平均成本小于 90 作为循环终止的条件，将此时得到的解作为最终结果，或将繁殖了两千代以后得到的解作为最终解。而事实证明，一般情况下，在繁殖了 30 代以后，得到的较优解基本保持不变，遗传算法大大的减少了找出较优解的时间。

种群：将一个初始解设为一个个体，将每个解所取的点作为基因。最初我们打算将种群容量设为 100，即生成 100 个初始的取点方案，后来在老师的建议下，将种群容量改为 200，这样可以提高找出最优解的概率。

适应度：在选取适应度时，因为将成本设为了适应度，结果导致成本低的解容易被淘汰。在耐心检查后，我们发现了错误，并决定将适应度的倒数设为适应度，于是成本低的解适应度反而高，比其他解更容易遗传到下一代。

选择复制：此过程中，我们以各个体适应度在总适应度中所占的比率为选择概率，对种群选择复制。最终成本高，适应度小的比较容易被淘汰，成本低，适应度大的容易遗传下去。

交叉：我们将交叉概率设为 0.9。同时随机选择交叉的位置。交叉后每个个体可能出现重复的基因，对此我们进行了检查和更新。

在初次试验中我们将基因条数设为 7 条，即是取点容量设为 7 个。在运行到 30 代以上时最小成本几乎不发生变化，此时我们将这 7 个点作为最终的最优取点。这 7 个点为：2，9，18，26，33，44，50；最优平均成本为 94.5157。

3.3.1 改变取点的数量

由于取点的多少影响到三次样条插值的精确性以及和源数据的拟合程度，即是取得点数越多，与原数据拟合程度越好，从而误差成本越小，但是取点越多会造成测定成本的相应提高，所以在误差成本变化不大的情况下要考虑到越少取点测定成本越小，这样才能有效降低总成本。故我们在完成上述的确定基因条数为 7 的实验之后，决定改变基因条数即测试点个数来重复进行实验。进一步确定在改变基因条数之后最小平均成本的变化以及最佳取点个数。下表是改变取点个数的实验结果。

取点个数	*最佳取点方案*	*测定成本*	*误差成本*	*最小平均成本*
4	4, 20, 41, 49	48	84.0224	132.0224
5	3, 14, 27, 38, 49	60	45.6972	105.6972
6	2, 11, 22, 31, 41, 49	72	21.3987	93.3987
7	2, 9, 18, 26, 33, 44, 50	84	10.5171	94.5171
8	2, 8, 18, 25, 31, 38, 45, 51	96	5.9797	101.9797
9	2, 7, 15, 20, 26, 32, 39, 45, 50	108	3.4382	111.4382

由表格中可以看出当取点个数发生变化时，测定成本逐渐攀升，而误差成本则逐渐下降，他们的总和最小平均成本出现一个向下再向上的性态。可见，当取点个数为 6

的时候最小平均成本最低可达到 93.3987。

3.3.2 改变拟合方式

另外，由于拟合方式不同，所产生的误差也就不同，因而改变拟合方式也事寻找最低成本的方法之一。观察样本数据分布的图像，与三次多项式的图像相似，于是我们尝试将拟合方式改为三次多项式拟合，并且得到了如下结果：

基因条数	最佳取点方案	测定成本	误差成本	最小平均成本
4	5, 18, 37, 48	48	82.033	130.033
5	3, 13, 25, 35, 48	60	51.9403	111.9403
6	3, 8, 20, 30, 39, 49	72	50.1983	122.1983
7	3, 7, 14, 24, 32, 38, 48	84	49.9478	133.9478
8	3, 4, 13, 22, 31, 37, 45, 50	96	50.5725	146.5725
9	3, 4, 11, 19, 26, 31, 36, 42, 49	108	48.1898	156.1898

由上表可以发现，除选择四个测试点时误差比较大，其他五组实验中误差成本比较接近，可以人为仅需五点就能较准确的拟合出三次多项式曲线。因此在三次多项式拟合的情况下，结合误差成本与测定成本，应选取五个测试点比较合适，这五个点是：3,13,25,35,48。

3.3.3 对算法的思考

遗传算法大大的减少了找出较优解的时间，但经过探讨，我们认为可以对其进行改进，对每次得到的最优个体施加保护，在交叉和变异的过程中保持不变。这样就可以提高运算的效率，节约更多找到最优解的时间。

4、 结论

本次实验，我们以遗传算法为主，配合三次样条插值和三次多项式拟合，对如何降低测定成本进行了探讨。最终得到以下结论：

1、对取点数不同的情况，进行多次实验后，发现取六个点或七个点时成本较接近，得到的最优解为以下六点：2, 11, 22, 31, 41, 49。也就是说，在以后的测试过程中，只需选这些点为测试点，利用三次样条插值，即可较为准确的估计其他点的数据。

2、三次样条插值与三次多项式拟合相比，三次样条插值拟合结果更优。因此在测试时应选择三次样条插值拟合。而且与三次多项式拟合相比，三次样条插值能够提高拟合的精确度。在一些需要控制误差成本的工程中，三次样条插值拟合方法更加优越。

本次实验我们加强对于 matlab 的掌握，了解了启发式搜索方法原理及应用，同时结合实际应用进行了一些改进。在这些过程中收获良多。

5、 参考资料

[1] 上海交通大学电子工程系.统计推断在数模转换中的应用课程讲义，ftp://202.120.39.248/

[2] 传感器. 仪器仪表世界网. 2013-01-03. http://www.1718world.com/ziliao/201208/21/ziliao-75.html/

[3] 三次样条插值. 百度百科. http://baike.baidu.com/link?url=vlvn3iH4kYG0U3n-pWp8KYs7E7KN9M7rKWdMGKSsf4e56S_DKcsKIZF7ixrj-vMTeg3PGwPgSKU4gpWmQamkPq/

[4] Adaptive Particle Swarm Optimization. IEEE Transaction on Systems, man, and Cybernetics. 2009, 39(6): 1362-1381. 2009-04-07.

[5] MATLAB 6.5 辅导优化计算与设计. 飞思科技产品研发中心编著. 电子工业出版社 2003.1.
[6] 遗传算法及其应用. 陈国良等编著. 人民邮电出版社 1996.6.

6、 附录

MATLAB 程序代码

文件名：**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
%                  g 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 y;                %样本列表
global x;            %样本占空比列表
global len;                  %样本组数量

popSize=100;                 %种群大小
codeSize=9;                  %观察点数量
crossRate=0.9;               %交叉概率
mutateRate=0.01;              %变异概率
maxGeneration=2000;        %最大子代数
aimCost=90;                %目标
minCost=500;                  %

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<aimCost
return;
end
g=1;
while g<=maxGeneration
 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<aimCost
      break;
 end
 g
 g=g+1;
end
```

文件名：**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
```

文件名：**readData.m**

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

文件名：**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
```

文件名：**fitness.m**

```
    function [score] = fitness(method)
    global y;
    global x;
    global len;
    codeSize=length(method);
    sum_A=0;
    U=zeros(1,codeSize);
    D=zeros(1,codeSize);
    for i=1:len
      A=0;
      for j=1:codeSize
            U(j)=y(i,method(j));
            D(j)=x(i,method(j));
      end
```

```
  a=polyfit(D,U,3);
  U1=polyval(a,x(i,:),3);
  %U1=interp1(D,U,x(i,:),'spline');
  uxTable=abs(U1-y(i,:));
  for j=1:51
        ux=uxTable(j);
        if ux<=0.5
              A=A+0;
        elseif ux<=1
              A=A+0.5;
        elseif ux<=2
              A=A+1.5;
        elseif ux<=3
              A=A+6;
        elseif ux<=5
              A=A+12;
        elseif ux>5
              A=A+25;
        end
  end
  A=A+108;
  sum_A=sum_A+A;
 end
 score=sum_A/len;
 end
```

文件名：**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
```

文件名：**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
```

文件名：**selection.m**

```
function [] = selection(popSize,codeSize)
global pop
global fitnessTable;
fitnessSum(1)=1.0/fitnessTable(1);
for i=2:popSize
 fitnessSum(i)=fitnessSum(i-1)+1.0/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;
```