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

组号：72 姓名：童迅 学号 5140309221 姓名：边登科 学号 5140309222

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 Yˆ 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]

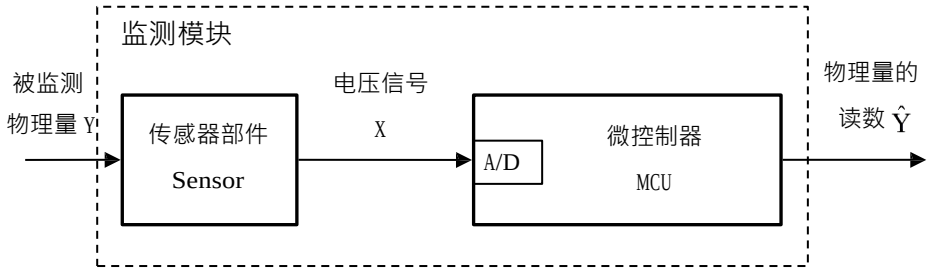


图 1-1 监测模块组成图

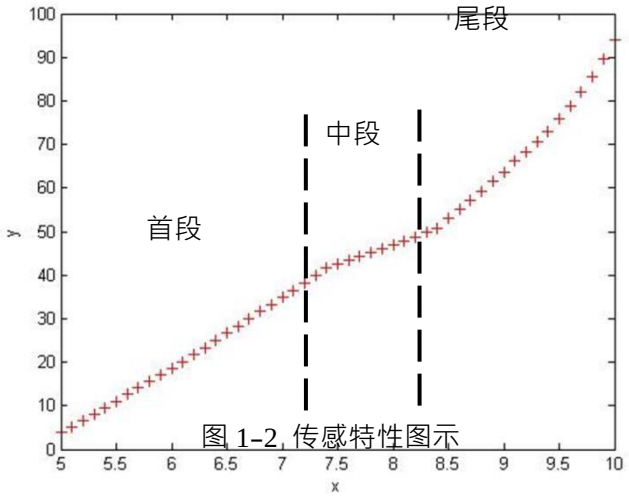


图 1-2 传感特性图示

2 拟合方案的对比探究

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

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

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

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

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。

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

总成本较低的校准方案，认定为较优方案。

2.1.2 遗传算法

遗传算法是基于达尔文的生物进化论和孟德尔的遗传学的一种通用的求解优化问题的适应性搜索方法，其基本原理是仿效生物界中物竞天择、适者生存的演化法则。遗传算法把问题参数编码为染色体，再利用迭代的方式进行选择、交叉以及编译等运算来交换种群中染色体的信息，最终生成符合优化目标的染色体。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。

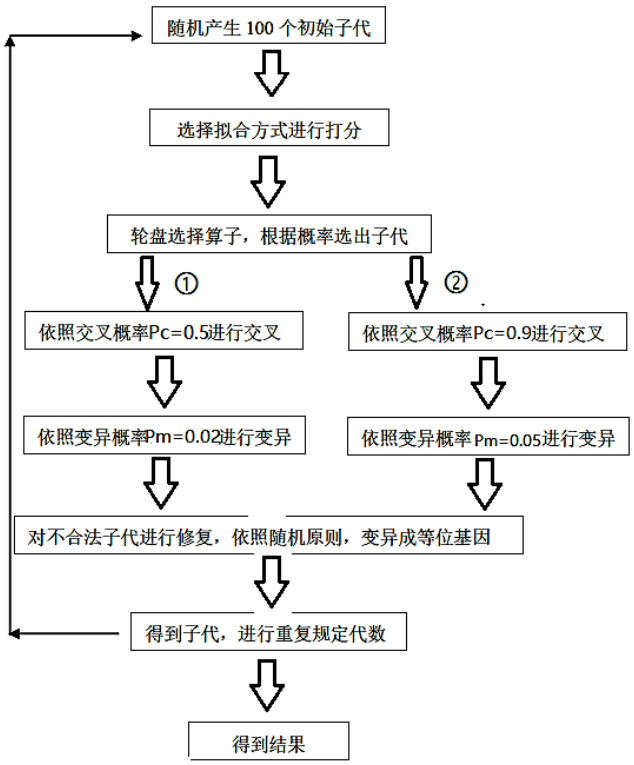
（1）选择（selection），选择父代中优秀的个体直接传递给下一代，即保留父代。

（2）交叉（crossover），以两个父代的基因为基础，进行交叉重组，生成下一代，好的基因会有更大的概率传给子辈。

（3）变异（mutation），每个父代个体都有一定的概率发生随机的基因突变。

评价基因好坏的依据就是适应度。在遗传算法的步骤中起着突出作用的其实是变异。正因为有了变异，算法才存在在全局范围内寻找到最优解的可能性。但是遗传算法需要通过反复试验，才有可能给出最优解。

遗传算法流程图如图 2-1 所示。



①：当群体适应度比较分散时，使得交叉概率和变异概率减少。

②：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

图2-1 遗传算法流程简图

交叉概率 Pc，变异概率 Pm 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）。

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

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

两式中f_{max}表示群体中最大适应度值，f_{avg}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。

2.1.3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。

2.2 拟合方法

图像中观察到的跳变是因为整体系统特性与其电路系统中晶体管元件特的工作特性可分为前后衔接的连续区间：截止区、线性区和饱和区，在跨越区间时物理特性是连续而非跳跃间断的。

2.2.2 三次样条差值法

考虑到中间部分线性关系比较明显，而在两端线性关系比较薄弱，所以在找到特征点之后可以分成三段来拟合 D-U 关系函数。

三次样条插值法的介绍如图 2-2、图 2-3 所示。

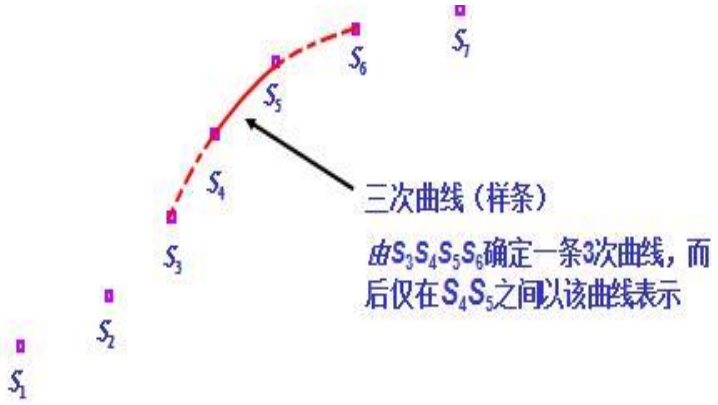


图 2.2 中间段拟合方式[1]

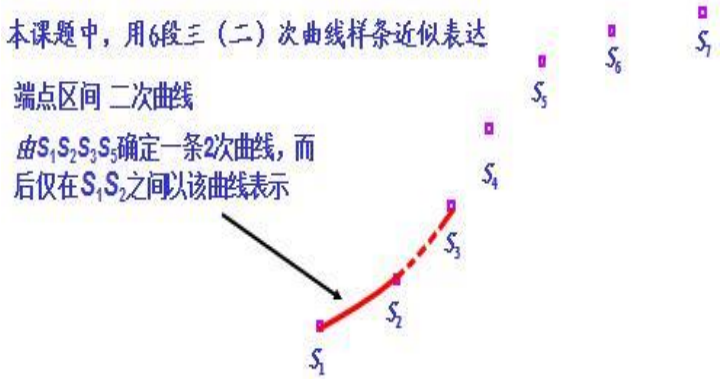


图 2.3 两端拟合方式[1]

我们对随机产生的七个点，按照一下方式进行拟合：

用 S_1,S_2,S_3 进行二次拟合，应用于 S_1S_2 段

用 S_1,S_2,S_3,S_4 进行三次拟合，用到 S_2S_3 段

用 S_2,S_3,S_4,S_5 进行三次拟合，用到 S_3S_4 段

用 S_3,S_4,S_5,S_6 进行三次拟合，用到 S_4S_5 段

用 S_4,S_5,S_6,S_7 进行三次拟合，用到 S_5S_6 段

用 S_5,S_6,S_7 进行二次拟合，用到 S_6S_7 段

3. 数据分析

表 1 遗传优化代码运行结果成本数据表

繁衍代数	成本	取点
10 代	116.6725	1 10 23 25 38 44 47 51
30 代	112.2230	6 10 24 25 37 44 49
30 代优化	107.9628	4 9 15 22 29 31 44 49
70 代	98.4165	3 7 20 30 42 49
70 代优化	95.2387	3 11 22 31 43 49
100 代	106.5318	1 10 23 25 38 44 49
100 代优化	96.4587	1 9 20 27 34 44 49

上表中最优解为优化后的 95.2387，此时取点为[3　11　22　31　43　49]

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。

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('20151010dataform.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遗传算法优化算法Matlab代码

```
my_answer=[ 1  10  23  25  38  44  47  51];%此处输入GA算法所得最优解
minput=dlmread('20151010dataform.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<=0.4);
le1_0=(errabs<=0.6);
le2_0=(errabs<=0.8);
le3_0=(errabs<=1);
le5_0=(errabs<=2);
le6_0=(errabs<=3);
le7_0=(errabs<=5);
g5_0=(errabs>5);
sij=0.1*(le1_0-le0_5)+0.7*(le2_0-le1_0)+0.9*(le3_0-le2_0)+1.5*(le5_0-
le3_0)+6*(le6_0-le5_0)+12*(le7_0-le6_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);
```