统计推断在数模转换系统中的应用

第 10 组　张哲迪 5130309566 陆严 5130309111

摘要：本文结合统计方法，根据已知的少量数据实现对其余更多的未知数据的推断。文中分析了不同的拟合方法及其比较，选取了一种较优的拟合方式，并采用遗传算法，模拟一定种群大小的生物遗传进化方式。通过产生新解，不断比较优化，最终得到比较优化的解。
关键词：统计推断、占空比、多项式拟合，遗传算法，Matlab

Application of Statistical Inference in AD&DA Inverting System

group number:10

ABSTRACT:
This report sets up a mathematical model is combined with statistical methods to infer the large amount of unknown data based on a small amount of known data. This report analysis different fitting methods to select an optimum fit and uses Genetic Algorithm to simulate the process of solid cooling. By constantly generating new solutions and optimizing them, we ultimately get more optimal solution.
Key words: statistic Interference、duty cycle、polynomial fitting、Genetic Algorithm、Matlab

1. 引言

在工程实践中往往会设计许多的测量工具，这些工具主要由传感器和电子信号接收器组成，为了对这些批量生产的工具用一个科学而合理的方法来定标，使其在应用时能够达到一定的精度，但是由于工具的测量数据往往十分多，单靠人工一个一个的进行定标虽然会具有相当高的精度，但是在非精密仪器的情况下，人们希望能够找到一个省时省力的方法来对这些仪器进行定标，让其达到事前给的预期测量精度。显然对于每一单个“工具”都需要根据这个工具产生的某些数据点的反馈进行合理的模拟得出此工具的测量输入-输出关系式，在本文中对曲线插值、曲线拟合进行讨论，并选取一个较为合理的方法来计算输入-输出关系是，显然除了曲线的构造，在这种求解空间十分大的计算当中，启发式搜索总能起到十分重要的作用，本文中采用常见的启发式搜索：遗传算法对数据进行筛选。

2. 评价标准的构建

为评估和比较不同的校准方案，特制定以下成本计算规则。

2.1 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（2-1）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。

2.2 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。

2.3 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（2-2）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。

2.4 校准方案总体成本

按式（2-3）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. 寻找恰当的拟合函数

3.1 不分段拟合

设：

$$Y_i=\sum_{n=0}^{N}a_n X_i^{\,n}+\varepsilon_i \quad \text{其中 } i=1,2,\ldots,49,50 \quad (3\text{-}1)$$

设误差平方和为：

$$Q_1=\sum_{i=1}^{50}\varepsilon_i^2=\sum_{i=1}^{50}(Y_i-(\sum_{n=0}^{N}a_n X_i^{\,n}))^2 \quad (3\text{-}2)$$

欲使 Q_1 达到最小，等价于求 $Q_1(a_1,a_2,...,a_N)$ 的最小值点，故可直接对数据进行多项式拟合。544 组数据不同次数多项式拟合的计算结果见表 3-1。

表 3-1 不分段拟合对于不同次数的比较

	2 次多项式	3 次多项式	4 次多项式	5 次多项式	6 次多项式
Q_1 的总和	22.9211	5.8372	4.3316	3.4452	2.9883
Q_1 的最大值	1.1871	0.1544	0.1540	0.0892	0.0872

3.2 分为三段进行多项式曲线拟合

3.2.1 区间的划分

区间划分见表 3-2。

表 3-2 曲线拟合区间的划分

区间	电压范围	对应的采样点序号	多项式曲线类型
1	[5V，V_1]	1,2,…,N_1	n_1 次多项式
2	[V_1，V_2]	N_1,…,N_2	二次多项式
3	[V_2，10V]	N_3,46,…,53	n_2 次多项式

易见，表 3-2 中的电压范围、采样点序号和多项式曲线类型还未确定。

3.2.2 参数的确定

下面对表 3-2 中三类参数进行确定。

由于电压范围与采样点序号相对应，所以只要确定了采样点序号就可以确定电压范围。为了减小之后的拟合误差，我们采取定量分析的方法来划分拟合区间。由于计算二次差分时，仅对连续 3 点数据作业，相比之下，计算曲率或者弧弦距离可以对连续的更多点数据作业，依概率理论知识多点误差的作用有相互抵消的趋势，计算曲率或者弧弦距离更有优势。

方法一：对每组的每一个数据点计算其曲率，然后按数据点顺序取平均，比较得出划分拟合区间的数据点 N_1 和 N_2。

方法二：考查适中长度线段的弧弦距离。由于要将整条曲线分为三段，所以要求得两个弧弦距离最大的点，作为分点。故先将整组数据均分为两段，将每组数据的第 1 个数据点与第 25 个数据点相连，计算第 2 个点到第 24 个点的弧弦距离，第 26 个数据点与第 50 个数据点相连，计算第 27 个点到第 49 个点的弧弦距离，取两组的弧弦距离最大值点，计算 544 组数据，最大值点所落最频繁点作为划分拟合区间的数据点 N_1 和 N_2。

计算公式为：

$$d_i = \frac{|kX_i - Y_i + b|}{\sqrt{k^2+1}} \quad \text{其中} \; k = \frac{Y_m - Y_n}{X_m - X_n}, \; b = X_n - kY_n = X_m - kY_m,$$

$$m=50, n=25, i=2,3,...,24,25 \text{或} m=25, n=1, i=26,27,...,49,50 \qquad (3\text{-}3)$$

显然方法二的计算量比方法一小很多，所以我们采取方法二。

统计 469 组数据的 N_1 和 N_2，发现 N_1 和 N_2 落在第 18 个点和第 45 点最多，故我们将第 18 个点和第 45 个点作为分点。

3.2.3 多项式拟合

接着是确定区间 1 和区间 3 的多项式。不妨设：

区间 1：

$$Y_i = \sum_{m=0}^{n_1} a_m X_i^{\,m} + \varepsilon_i \qquad \text{其中 } i=1,2,\ldots,N_1 \qquad (3\text{-}4)$$

区间 2：

$$Y_j = b_2 X_j^2 + b_1 X_j + b_0 + \varepsilon_j \qquad \text{其中 } j=N_1,\ldots,N_2 \qquad (3\text{-}5)$$

区间 3：

$$Y_k = \sum_{n=0}^{n_2} c_n X_k{}^n + \varepsilon_k \qquad \text{其中 } k=N_2,\ldots,50 \tag{3-6}$$

以上各式中 $\varepsilon_h \sim N(0,\sigma^2)$ 且相互独立，其中 $h = 1,2,3,\ldots,50$

区间 1 与区间 2 衔接点两个方程连续且斜率相同：

$$\sum_{m=0}^{n_1} a_m X_{N_1}{}^m = b_2 X_{N_1}^2 + b_1 X_{N_1} + b_0 \tag{3-7}$$

解得：

$$a_0 = -\sum_{m=1}^{n_1} a_m X_{N_1}{}^m + b_2 X_{N_1}^2 + b_1 X_{N_1} + b_0 \tag{3-8}$$

同理可解得：

$$c_0 = -\sum_{n=1}^{n_2} c_n X_{N_2}{}^n + b_2 X_{N_2}^2 + b_1 X_{N_2} + b_0 \tag{3-9}$$

设误差平方和为：

$$\begin{aligned} Q_2 = \sum_{h=1}^{50} \varepsilon_h^2 = \sum_{i=1}^{N_1} (Y_i - (\sum_{\mathrm{m}=0}^{\mathrm{n}_1} a_m X_i{}^m))^2 + \\ \sum_{j=N_1}^{N_2} (Y_j - (b_2 X_j^2 + b_1 X_j + b_0))^2 + \sum_{k=N_2}^{50} (Y_k - (\sum_{n=0}^{n_2} c_n X_k{}^n))^2 \end{aligned} \tag{3-10}$$

欲使 Q_2 达到最小，等价于求 $Q_2(a_1,a_2,...,a_{n_1},b_0,b_1,b_2,c_1,c_2,...,c_{n_2})$ 的最小值点，故可直接对数据进行多项式拟合。例如：以下求取 $a_0,a_1,a_2,...,a_{n_1-1},a_{n_1}$：

$$\begin{pmatrix} 1 & X_1 & \cdots & X_1{}^{n_1-1} & X_1{}^{n_1} \\ 1 & X_2 & \cdots & X_2{}^{n_1-1} & X_2{}^{n_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_{N_1} & \cdots & X_{N_1}{}^{n_1-1} & X_{N_1}{}^{n_1} \\ 0 & 1 & \cdots (n_1-1) X_{N_1}{}^{n_1-2} & n_1 X_{N_1}{}^{n_1-1} & \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n_1-1} \\ a_{n_1} \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_{n_1} \\ b_1 \end{pmatrix} \tag{3-11}$$

解得：

$$\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n_1-1} \\ a_{n_1} \end{pmatrix} = \begin{pmatrix} 1 & X_1 & \cdots & X_1{}^{n_1-1} & X_1{}^{n_1} \\ 1 & X_2 & \cdots & X_2{}^{n_1-1} & X_2{}^{n_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_{N_1} & \cdots & X_{N_1}{}^{n_1-1} & X_{N_1}{}^{n_1} \\ 0 & 1 & \cdots (n_1-1) X_{N_1}{}^{n_1-2} & n_1 X_{N_1}{}^{n_1-1} & \end{pmatrix}^{-1} \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_{n_1} \\ b_1 \end{pmatrix} \tag{3-12}$$

469 组数据的计算结果见表 3-3.

表 3-3 分段拟合对于不同次数的比较

	3+2+3 次多项式	4+2+3 次多项式	3+2+4 次多项式	4+2+4 次多项式
Q_2的总和	2.9493	2.7730	2.9047	2.7286
Q_2的最大值	0.1480	0.1454	0.1473	0.1096

3.3 结果分析与比较

观察表 3-2 可得：3 次多项式拟合比 2 次多项式拟合误差小很多，再提高次数，误差逐渐减小，但减小得很少，说明 3 次多项式已经能较好地描述该系统的特性。

观察表 3-3 可得：当两边的两段用 3 次多项式拟合时 Q_2 已经很小了，再提高次数对其几乎没有影响。

比较表 3-2 和表 3-3 可得：分段拟合比不分段拟合更有优势。

4. 方案选择

4.1 算法选择

4.1.1 暴力穷举

若用暴力穷举法，解决以上问题需要尝试约 1.54 亿种不同组合，显然这是不切实际的

4.1.2 遗传算法（GA）

1.思想概述

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。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。

优点：质量高，初值鲁棒性强，简单、通用、易实现。

缺点：(1)单一的遗传算法编码不能全面地将优化问题的约束表示出来。考虑约束的一个方法就是对不可行解采用阈值，这样，计算的时间必然增加。

(2)遗传算法通常的效率比其他传统的优化方法低。

(3)遗传算法容易过早收敛。

(4)遗传算法对算法的精度、可行度、计算复杂性等方面，还没有有效的定量分析方法。

4.2 算法实现

(1)建初始状态

初始种群是从解中随机选择出来的，将这些解比喻为染色体或基因，该种群被称为第一代，这和符号人工智能系统的情况不一样，在那里问题的初始状态已经给定了。

(2)评估适应度

对每一个解(染色体)指定一个适应度的值，根据问题求解的实际接近程度来指定(以便逼近求解问题的答案)。不要把这些“解”与问题的“答案”混为一谈，可以把它理解成为

要得到答案，系统可能需要利用的那些特性。

(3)繁殖

繁殖(包括子代突变)带有较高适应度值的那些染色体更可能产生后代(后代产生后也将发生突变)。后代是父母的产物，他们由来自父母的基因结合而成，这个过程被称为“杂交”。

(4)下一代

如果新的一代包含一个解，能产生一个充分接近或等于期望答案的输出，那么问题就已经解决了。如果情况并非如此，新的一代将重复他们父母所进行的繁衍过程，一代一代演化下去，直到达到期望的解为止。

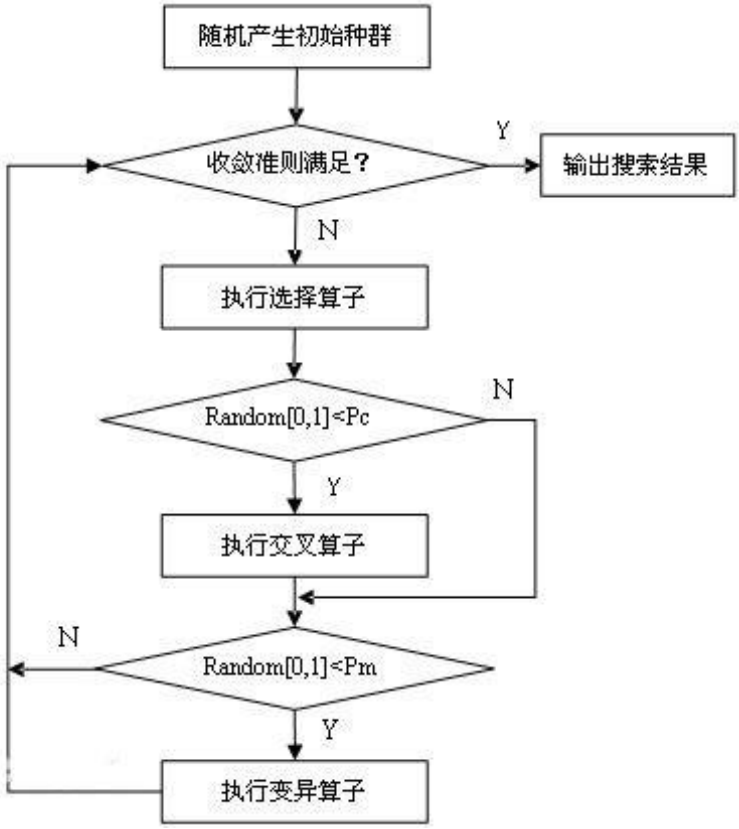

图 4-1 遗传算法流程图

4.3 算法详解

(1)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)。

(2)个体评价：计算群体 P(t)中各个个体的适应度。

遗传算法

(3)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。

(4)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。

(5)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)。

(6)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。

5. 拓展研究

5.1 计算方式的改进

起初，我们的程序运行时间极长，在参考了一些文献，并在面谈中向老师咨询后，我们发现，matlab 在对 for 循环语句进行计算时，耗时特别多，经过查阅资料知道可以通过改写语句的方式来提高运行速度，有以下两种方式：

（1） 将for循环语句改写为向量形式

（2） 利用并行计算 parfor 语句，开启多线程并行计算,如图 5-1

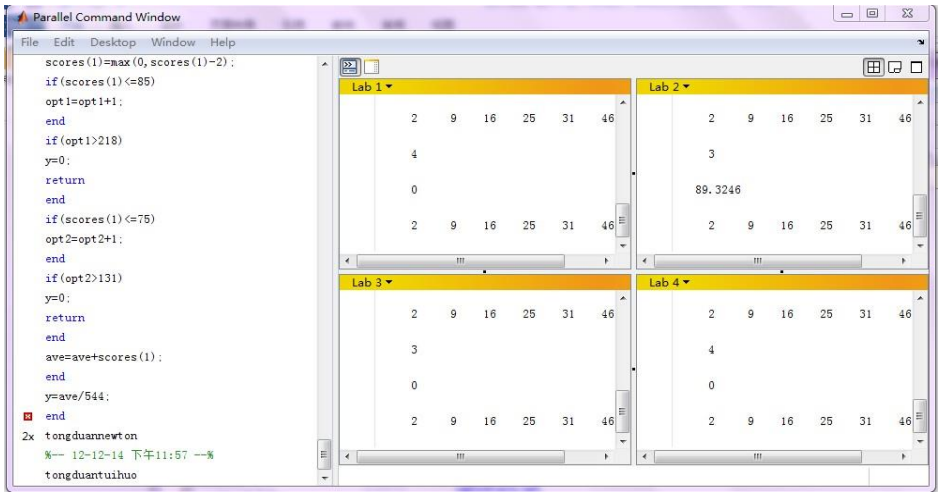

图 5-1 pmode 模式下的多窗口并行运行

（3） 利用 matlab 自配的分布式计算工具箱（parallel computing）进行分布式计算[3]

5.2 寻找最少的检测点

六个检测点可以满足课题要求的情况下，可以认为更少的点也可以满足课题要求。但考虑到数据本身呈现出来的分段以及高次曲线性质，因此可以认为检测点最少不能少于 4 个。基于以上分析，分别 4，5，6 个检测点的做出尝试。减少检测点后，使用遗传算法进行尝试。

6. 最终结论

综上所述，利用遗传算法，并使用三次样条插值拟合进行拟合后，可以找到六个点分别为 3,12,22,31,41,50 使得成本最低，为 92.96，满足评价条件。同时，减少一个点，也可以找到五个点分别为 3,14,26,39,49 使得成本为 103.56，明显高于前一个选点的成本。而四个点则是成本过大，因此，六个点为宜。从中选择成本最低的点可以在工程上减少检测成本。

选择的点：3,12,22,31,41,50

7. 致谢

感谢袁焱老师和李老师的耐心指导与细致讲解，并为我们小组的论文提出宝贵的指导意见！

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] 百度百科 词条“三次样条插值，遗传算法，模拟退火算法 ”

[3] 贺才兴等 概率论与数理统计 科学出版社 2007

[4] 薛定宇, 陈阳泉 高等应用数学问题的 MATLAB 求解 北京: 清华大学出版社, 2004

[5] matlab 官方网站—分布式计算 http://www.mathworks.cn/programs/ad/distcomp

9. 附录

附录 1：遗传算法主函数（matlab）

```
function [m,n,p,q] = GeneticAlgorithm(pop_size, chromo_size,
generation_size, cross_rate, mutate_rate, elitism)

global G ;
global fitness_value;
global best_fitness;
global fitness_avg;
global best_individual;
global best_generation;

fitness_avg = zeros(generation_size,1);

fitness_value(pop_size) = 0;
best_fitness = 0;
best_generation = 0;
initialize(pop_size, chromo_size);

for G=1:generation_size
    fitness(pop_size, chromo_size);
    rank(pop_size, chromo_size);
    selection(pop_size, chromo_size, elitism);
    crossover(pop_size, chromo_size, cross_rate);
    mutation(pop_size, chromo_size, mutate_rate);
end

plotGA(generation_size);
m = best_individual;
n = best_fitness;
p = best_generation;

q = [];
for j=1:chromo_size
    if best_individual(j) == 1
          q = [q, j];
```

```
    end
end

m
n
p
q

clear i;
clear j;
```

附录2：遗传算法主要函数实现

（1）计算种群适应度

```
function fitness(pop_size, chromo_size)
global fitness_value;
global pop;
global G;

for i=1:pop_size
    fitness_value(i) = 0;
end

data=csvread('20141010dataform.csv');
a=1:2:(length(data)-1);
b=2:2:length(data);
X=data(a,:);
Y=data(b,:);

for i=1:pop_size
    point = [];
    for j=1:chromo_size
        if pop(i, j) == 1
            fitness_value(i) = fitness_value(i) + 12 * length(a);
            point = [point, j];
        end
    end

    if length(point) < 3
        fitness_value(i) = fitness_value(i) + 10000;
    end

    point_x = X(:,point);
    point_y = Y(:,point);
```

```
    for q=1:length(a)
        point_x_q = point_x(q,:);
        point_y_q = point_y(q,:);


        point_x_q_all = X(q,:);
        point_y_q_all = Y(q,:);

        y_current = spline(point_x_q, point_y_q,point_x_q_all);

        Q = abs(point_y_q_all - y_current);

        for j=1:51
            if Q(j) > 5
                fitness_value(i) = fitness_value(i) + 25;
                else if Q(j) > 3
                       fitness_value(i) = fitness_value(i) + 12;
                       else if Q(j) > 2
                              fitness_value(i) = fitness_value(i) + 6;
                              else if Q(j) > 1
                                     fitness_value(i) = fitness_value(i)
+ 1.5;
                                     else if Q(j) > 0.5
                                            fitness_value(i) =
fitness_value(i) + 0.5;
                                            else if Q(j) >= 0
                                                   fitness_value(i) =
fitness_value(i) + 0;
                                               end
                                        end
                                 end
                          end
                   end
            end
        end
    end

    fitness_value(i) = fitness_value(i) / 469;
    fitness_value(i) = 100000 - fitness_value(i);
end

clear i;
clear j;
```

（2）选择操作

```
function selection(pop_size, chromo_size, elitism)
global pop;
global fitness_table;

for i=1:pop_size
    r = rand * fitness_table(pop_size);
    first = 1;
    last = pop_size;
    mid = round((last+first)/2);
    idx = -1;
    while (first <= last) && (idx == -1)
        if r > fitness_table(mid)
            first = mid;
        elseif r < fitness_table(mid)
            last = mid;
        else
            idx = mid;
            break;
        end
        mid = round((last+first)/2);
        if (last - first) == 1
            idx = last;
            break;
        end
   end

   for j=1:chromo_size
        pop_new(i,j)=pop(idx,j);
   end
end

if elitism
    p = pop_size-1;
else
    p = pop_size;
end
for i=1:p
   for j=1:chromo_size
        pop(i,j) = pop_new(i,j);
   end
end

clear i;
```

```
clear j;
clear pop_new;
clear first;
clear last;
clear idx;
clear mid;
```

(3)单点变异

```
function mutation(pop_size, chromo_size, mutate_rate)
global pop;

for i=1:pop_size
    if rand < mutate_rate
        mutate_pos = round(rand*chromo_size);
        if mutate_pos == 0
            continue;
        end
        pop(i,mutate_pos) = 1 - pop(i, mutate_pos);
    end
end

clear i;
clear mutate_pos;
```

(4)单点交叉

```
function crossover(pop_size, chromo_size, cross_rate)
global pop;
for i=1:2:pop_size
    if(rand < cross_rate)
        cross_pos = round(rand * chromo_size);
        if or (cross_pos == 0, cross_pos == 1)
            continue;
        end
        for j=cross_pos:chromo_size
            temp = pop(i,j);
            pop(i,j) = pop(i+1,j);
            pop(i+1,j) = temp;
        end
    end
end


clear i;
clear j;
```

```
clear temp;
clear cross_pos;
```