粒子群算法

1.本质——启发式算法的一种

启发式算法的定义：

一个基于直观或经验构造的算法，在**可接受的花费**下给出待解决**优化问题**的一个**可行解。**

可接受的花费： 计算时间和空间能接受

优化问题：在一定约束条件下，求解目标函数的最大值（或最小值）问题

可行解：得到的结果能用于工程实践

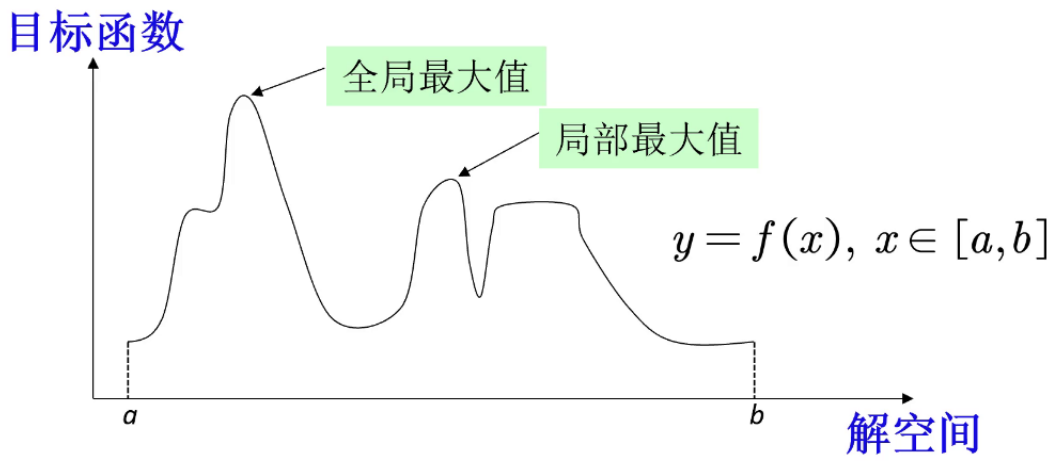
常见的启发式算法：

粒子群、模拟退火、遗传算法、蚁群算法、禁忌搜索算法等。

相反的算法：盲目搜索算法（枚举法、蒙特卡洛模拟）

（粒子群算法是通过模拟鸟类觅食行为而发展起来的一种基于群体写作的搜索算法，使整个群体的运动在问题求解空间中产生从无序到有序的演化过程，从而获得问题的可行解）

2.引例：爬山法求解目标函数全局最大值



步骤：

（1）在解空间随机生成许多初始解。

（2）根据初始解的位置，下一步有两种走法：向左走一步或者向右走一步（走得步长尽量小）。

（3）比较不同走法下目标函数的大小，舍弃掉小的那一部分。

（4）不断重复步骤，直到找到一个极大值点，此时结束搜索。

优点和缺点：

优点：效率高于盲目搜索

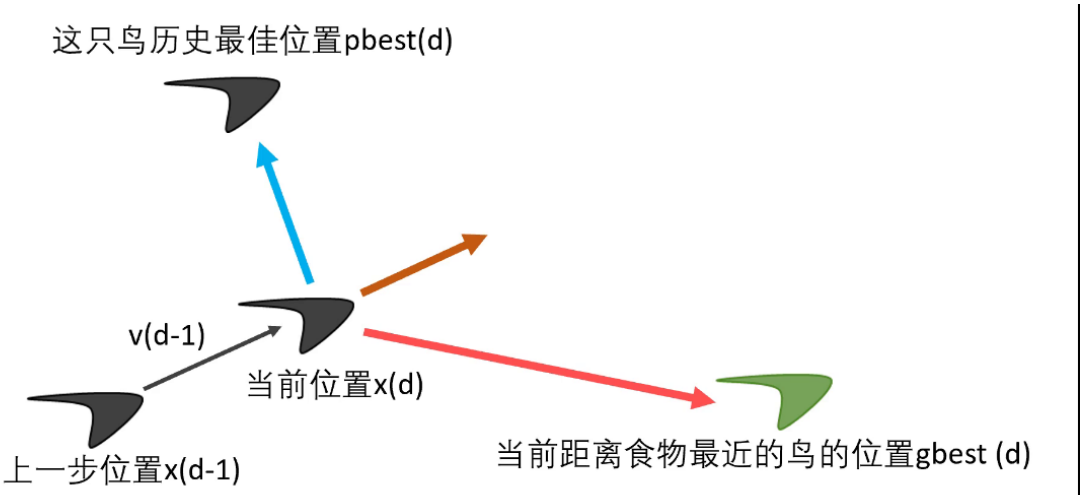
缺点：容易找到局部最优解、当处理多变量问题的时候效率很低

（爬山法是最简单的启发式算法）

3.粒子群算法的直观解释

假设：一群鸟在搜索食物

- 所有鸟都不知道食物在哪。
- 它们知道自己的当前位置。
- 它们知道离食物最近的鸟的位置。



三个参考：

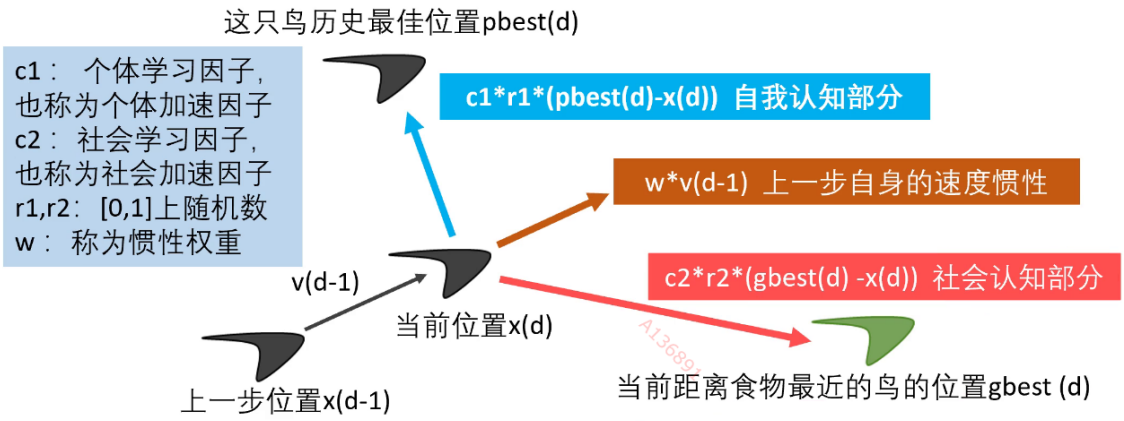
首先，**距离食物最近的鸟**会通知其他鸟往自己的方向来；

同时，每一只鸟的在搜索食物的过程中，**自己的位置也在不停变化**，因此每一只鸟也知道自己离食物最近的位置，这也是它们的一个参考；

最后，鸟在飞行中还需要考虑**惯性。**

（最后鸟下一步的位置取决于这三个方向的综合）

对图，我们可以进行如下分析：



这只鸟第d步所在的位置 = 第d-1步所在的位置 + 第d-1步的速度 * 运动的时间
x(d) = x(d-1) + v(d-1) * t (每一步运动的时间t一般取1)

这只鸟第d步的速度 =上一步自身的速度惯性 + 自我认知部分 +社会认知部分
v(d) = w*v(d-1) + c1*r1*(pbest(d)-x(d)) + c2*r2*(gbest(d)-x(d)) （三个部分的和）

论文中得到的结论：惯性权重取0.9-1.2合适，一般取0.9，最初的研究认为个体学习因子和社会学习因子取2比较合适，现在有所改变

不断地迭代，就能得到一个比较好的结果，上图只是一只鸟，如果要进一步取得最优解，需要拓展到整个鸟群。

很明显像这样是无法解决问题的，需要进一步将其转化为数学语言。

4.粒子群转化为数学语言：

粒子：优化问题的候选解。

位置：候选解所在的位置。

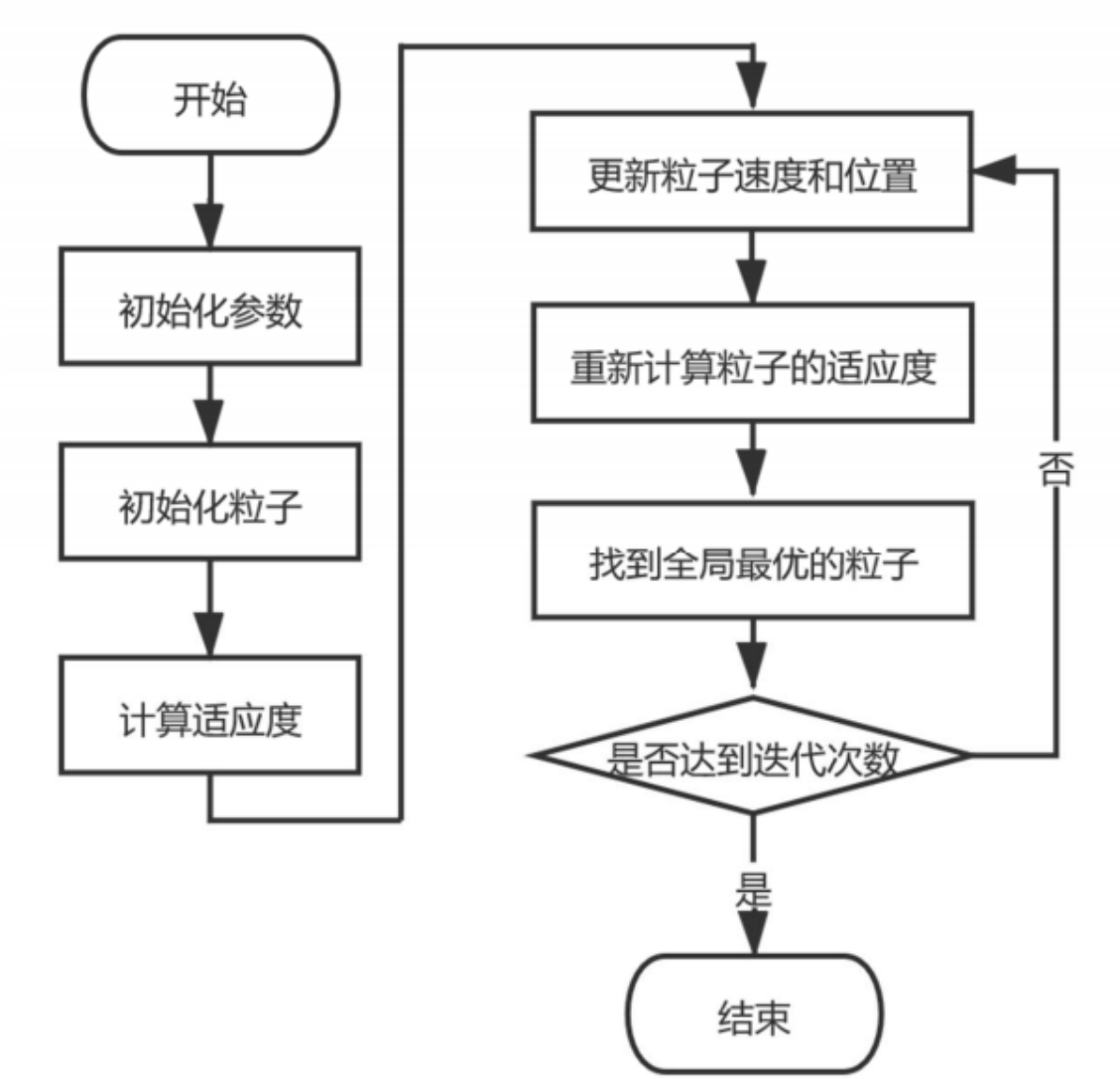
速度：候选解移动的速度。

适应度：候选解移动的速度。

个体最佳位置：单个粒子迄今为止找到的最佳位置。

群体最佳位置：所有粒子迄今为止找到的最佳位置。

5.粒子群算法的流程：



6.用Matlab自带函数求解粒子群：（以下步骤仅限理解）

首先要进行

Matlab自带函数（particleswarm函数，即为粒子群的英文名）优化的特别好，运行速度且找到的解也比较精准，内部实现比较复杂，

注意：这个函数只能求解最小值，如果目标函数为求最大值需要把原函数加负号

Matlab中particleswarm函数采用的是自适应的邻域模式，具体概念如下：

全局模式是指粒子群在搜索过程中将所有其他粒子都视为邻域粒子，它可以看做是邻域模式的极端情况，基本粒子群算法即属于全局模式。其优点是粒子邻域个体多，粒子群内的信息交流速度快，使得粒子群算法具有较快的收敛速度，但在一定程度上会降低粒子多样性，易陷入局部最优。

邻域模式是指粒子群在搜索过程中只将其周围部分粒子视为邻域粒子，这种模式使得粒子群可以被分割成多个不同的子群体，有利于在多个区域进行搜索，避免算法陷入局部最优。

关于自适应的理解：如果**适应度开始停滞时**，粒子群搜索会从邻域模式向全局模式转换，一旦**适应度开始下降**，又恢复到邻域模式，以避免陷入局部最优。当适应度的停滞次数足够大时，**惯性系数**开始逐渐减小，从而利于全局搜索。

预设参数的选取：

- **粒子个数 SwarmSize：**

  默认设置为： min{100,10*nvars}, nvars是变量个数。

  变量的个数间接反映了问题的复杂度，这是Matlab的一个默认的设置。

- **惯性权重 InertiaRange：**

  默认设置的范围为：[0.1,1.1]，注意，在迭代过程中惯性权重会采取自适应措施，

  随着迭代过程不断调整。

  由上文可知，惯性系数会自动地进行一个调整，有一个自适应的过程，所以是在一个范围之中。

- **个体学习因子 SelfAdjustmentWeight：**

  默认设置为：1.49 （和压缩因子的系数几乎相同）

  压缩因子法是一种手动编写粒子群算法的方法，本文没有赘述（同下）。

- **社会学习因子 SocialAdjustmentWeight：**

  默认设置为：1.49 （和压缩因子的系数几乎相同）

- **邻域内粒子的比例 MinNeighborsFraction：**

  默认设置为：0.25，由于采取的是邻域模式，因此定义了一个**“邻域最少粒子数目”**：minNeighborhoodSize = max{2,(粒子数目*邻域内粒子的比例)的整数部分}（一个粒子在自己的邻域内，算上自己应该至少由2个粒子），在迭代开始后，每个粒子会有一个邻域，初始时邻域内的粒子个数(记为Q)就等于“邻域最少粒子数目”，后续邻域内的粒子个数Q会自适应调整。

变量初始化和适应度的计算：

（1）速度初始化:

vmax = ub – lb（最大速度就是上界和下界的差额）； v = -vmax + 2*vmax* . rand(n,narvs);

粒子的最大速度，是为了保证下一步的位置离当前位置别太远，一般取自变量可行域的10%至20%，在后续更新粒子i的位置时有很大用处；

（2）位置初始化: 将每个粒子的位置均匀分布在上下界约束内

（3）计算每个粒子的适应度：

适应度设置为我们**要优化的目标函数**，由于该函数求解的是最小值问题，因此，最优解应为适应度最小即目标函数越小的解。

（4）初始化个体最优位置：

因为还没有开始循环，因此这里的个体最优位置就是我们初始化得到的位置。

（5）初始化所有粒子的最优位置：

因为每个粒子的适应度我们都已经求出来了，所以我们只需要找到适应度最低的那个粒子，并记其位置为所有粒子的最优位置。

更新粒子的速度和位置：

符号	含义
n	粒子个数
c_1	粒子的个体学习因子，也称为个体加速因子
c_2	粒子的社会学习因子，也称为社会加速因子
w	速度的惯性权重
v_i^d	第d次迭代时，第i个粒子的速度
x_i^d	第d次迭代时，第i个粒子所在的位置
$f(x)$	在位置x时的适应度值(一般取目标函数值)
$pbest_i^d$	到第d次迭代为止，第i个粒子经过的最好的位置
$gbest^d$	到第d次迭代为止，所有粒子经过的最好的位置

在每次迭代中，我们要分别更新每一个粒子的信息。**例如: 对于现在要更新的粒子i ,我们要进行以下几个操作:**

（1）**随机生成粒子i的邻域（与上文的定义略有出入）**，邻域内一共Q个粒子(包含粒子i本身)，并**找到这Q个粒子中位置最佳的那个粒子**，此时它的目标函数值最小，记其位置为lbest；

（2）**更新粒子i的速度**，公式为：

$$v_i^d = wv_i^{d-1} + c_1 r_1 (pbest_i^d - x_i^d) + c_2 r_2 (gbest^d - x_i^d)$$

$$其中r_1, r_2是[0,1]上的随机数$$

这个速度公式和基本粒子群算法最大的不同在于：这里的群体信息交流部分使用的是邻域内的最优位置，而不是整个粒子群的最优位置，**这是粒子群算法的核心公式**；

（3）**更新粒子i的位置**，公式为：

$$x_i^{d+1} = x_i^d + v_i^d$$

每一步运动的时间t记为1，方便计算。

（4）**修正位置和速度**：如果粒子i的位置超过了约束，就将其位置修改到边界处；另外如果这个粒子的位置在边界处，我们还需要查看其速度是否超过了最大速度，如果超过的话将这个速度变为0。(注意，如果是多元函数的话可能只有某个分量超过了约束，我们的修改只需要针对这个分量即可)；

（5）**计算粒子i的适应度**，如果小于其历史最佳的适应度，就更新粒子i的历史最佳位置为现在的位置；另外还需要判断粒子i的适应度是否要小于所有粒子迄今为止找到的最小适应度，如果小的话需要更新所有的粒子的最佳位置为粒子i的位置。

自适应调整参数：（直接翻译Matlab官方pdf文件）：

假设在第d次迭代过程中，所有的粒子的信息都已经更新好了，那么在开始下一次的迭代之前，需要更新模型中的参数，这里就体现了自适应过程。

规则如下：记此时所有粒子的最小适应度为a，上一次迭代完成后所有粒子的最小适应度为b。比较a和b的相对大小，如果a<b，则记flag = 1; 否则记flag = 0.

如果：flag = 0，那么做下面的操作：

（1）更新 c = c + 1 ； 这里的c表示“停滞次数计数器”，在开始迭代前就初始为0

（2）更新 Q = min{Q+ minNeighborhoodSize, SwarmSize} ； Q: 邻域内的粒子个数; minNeighborhoodSize: 邻域最少粒子数目； SwarmSize: 粒子的总数

如果：flag = 1，那么做下面的操作：

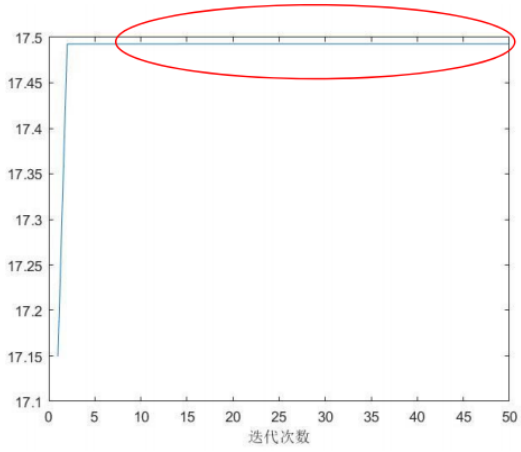
（1）更新 Q = minNeighborhoodSize

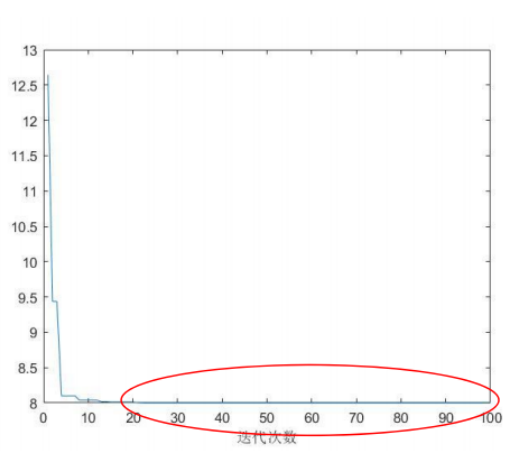
（2）更新 c = max{c-1, 0}

（3）判断c的大小，如果c < 2，则更新w = 2*w； 如果c > 5，则更新w = w/2；这里的w是惯性权重，如果计算的结果超过了惯性权重的上界或低于下界都需要将其进行调整到边界处。

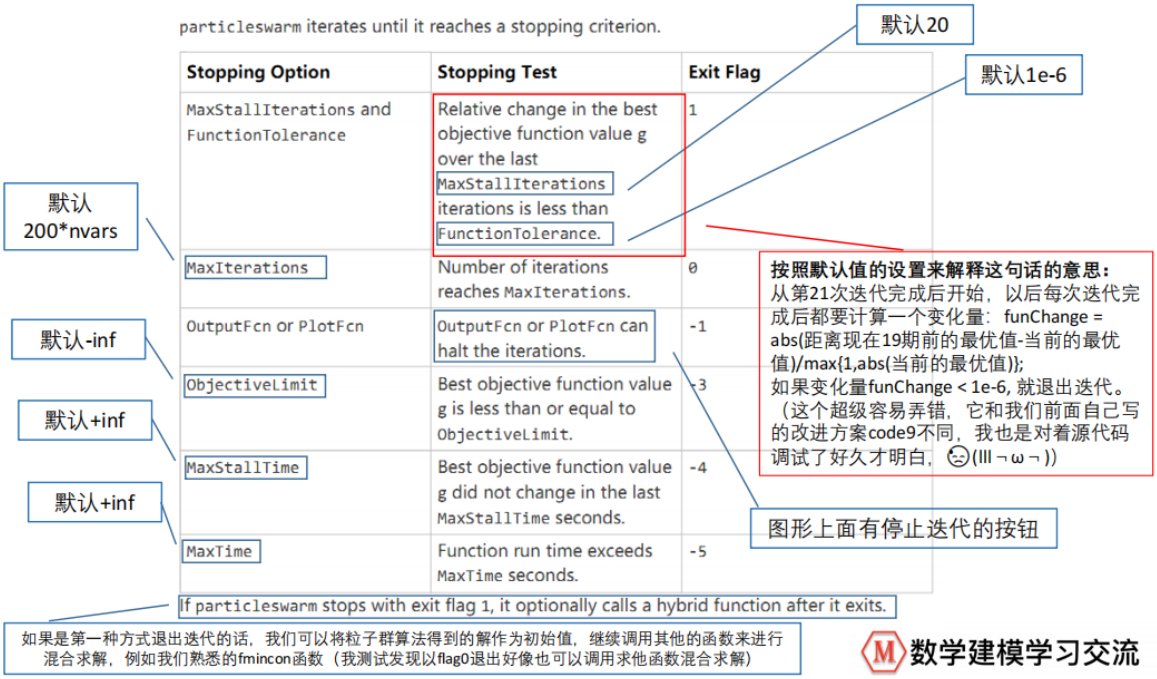
自动退出迭代循环：

当一个迭代循环到如此位置的时候，就需要跳出循环。





Matlab自带的粒子群函数可以设置几种自动退出迭代循环的方法：



运用的最多的是前两个：最大的停滞迭代次数和函数容忍量、最大迭代次数

具体例题：

1.求解函数y = x1^2+x2^2-x1x2-10x1-4x2+60在[-15,15]内的最小值（最小值为8）

```
narvs = 2; % 变量个数
x_lb = [-15 -15]; % x的下界(长度等于变量的个数，每个变量对应一个下界约束)
x_ub = [15 15]; % x的上界
[x,fval,exitflag,output] = particleswarm(@Obj_fun2, narvs, x_lb, x_ub)
```

以下是@Obj_fun2的具体函数：

```
function y = Obj_fun2(x)
% y = x1^2+x2^2-x1*x2-10*x1-4*x2+60
% x是一个向量
    y = x(1)^2 + x(2)^2 - x(1)*x(2) - 10*x(1) - 4*x(2) + 60;
end
```

这里对结果进行解释：

```
Optimization ended: relative change in the objective value
over the last OPTIONS.MaxStallIterations iterations is less than
OPTIONS.FunctionTolerance.

x =

    8.0000    6.0000


fval =

    8.0000
```

```
exitflag =

     1


output =

  包含以下字段的 struct:

      rngstate: [1x1 struct]
    iterations: 43
     funccount: 880
       message: 'Optimization ended: relative change in the objective value ↵over
the last OPTIONS.MaxStallIterations iterations is less than
OPTIONS.FunctionTolerance.'
    hybridflag: []
```

- **message: 'Optimization ended: relative change in the objective value ↵over the last OPTIONS.MaxStallIterations iterations is less than OPTIONS.FunctionTolerance.'**为退出循环的原因，具体原因参照下表：

  Matlab自带的粒子群函数可以设置几种自动退出迭代循环的方法：

particleswarm iterates until it reaches a stopping criterion.

Stopping Option	Stopping Test	Exit Flag
MaxStallIterations and FunctionTolerance	Relative change in the best objective function value g over the last MaxStallIterations iterations is less than FunctionTolerance.	1
MaxIterations	Number of iterations reaches MaxIterations.	0
OutputFcn or PlotFcn	OutputFcn or PlotFcn can halt the iterations.	-1
ObjectiveLimit	Best objective function value g is less than or equal to ObjectiveLimit.	-3
MaxStallTime	Best objective function value g did not change in the last MaxStallTime seconds.	-4
MaxTime	Function run time exceeds MaxTime seconds.	-5

If particleswarm stops with exit flag 1, it optionally calls a hybrid function after it exits.

默认20

默认1e-6

默认 200*nvars

默认-inf

默认+inf

默认+inf

按照默认值的设置来解释这句话的意思：
从第21次迭代完成后开始，以后每次迭代完成后都要计算一个变化量：funChange = abs(距离现在19期前的最优值-当前的最优值)/max{1,abs(当前的最优值)};
如果变化量funChange < 1e-6, 就退出迭代。
（这个超级容易弄错，它和我们前面自己写的改进方案code9不同，我也是对着源代码调试了好久才明白，😓(lll ¬ ω ¬))

图形上面有停止迭代的按钮

如果是第一种方式退出迭代的话，我们可以将粒子群算法得到的解作为初始值，继续调用其他的函数来进行混合求解，例如我们熟悉的fmincon函数（我测试发现以flag0退出好像也可以调用求他函数混合求解）

  与上文文exitflag = 1相对应，**这个数据我们一般不用考虑。**

- x =

```
    8.0000    6.0000
```

fval =

```
    8.0000
```

这是最终取得的x最优解以及所对应的f（x）的取值。

- **iterations: 43** 为迭代的次数，这个随着不同的运行数值不相同。

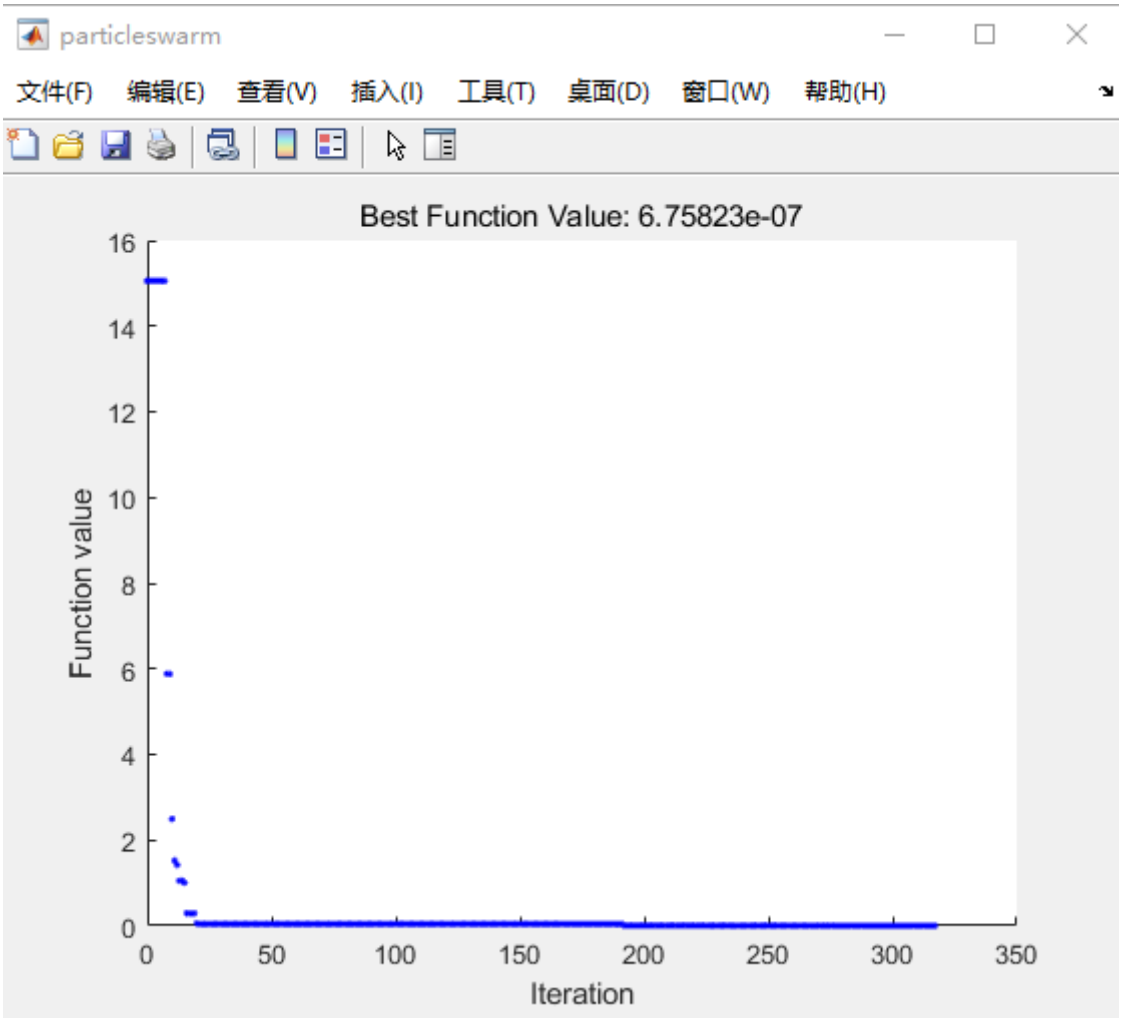
画迭代图的指令：

```
options = optimoptions('particleswarm','PlotFcn','pswplotbestf')
[x,fval] = particleswarm(@Obj_fun3,narvs,x_lb,x_ub,options)
```

以下是@Obj_fun3函数：

```
function y = Obj_fun3(x)
% 注意，x是一个向量
    tem1 = x(1:end-1);
    tem2 = x(2:end);
    y = sum(100 * (tem2-tem1.^2).^2 + (tem1-1).^2);
end
```

具体图像如下：



7.总结：

粒子群算法作为启发式算法的一种，自身具有特别特别深的学问，也是目前研究优化方法的一种常用的模式。如果要深入研究粒子群算法的话篇幅远远不够，本文只是基于建模比赛的程度上运用Matlab自带的工具箱进行求解。