function [pso F] = pso\_2D()

% FUNCTION PSO  --------USE Particle Swarm Optimization Algorithm

% global present;

% close all;

clc;

clear all;

pop\_size = 10;                  %   pop\_size 种群大小 ///粒子数量

part\_size = 2;                 %   part\_size 粒子大小 ///粒子的维数

gbest = zeros(1,part\_size+1);   %   gbest 当前搜索到的最小的值

max\_gen = 200;                  %   max\_gen 最大迭代次数

%best=zeros(part\_size,pop\_size\*part\_size);%xuan

region=zeros(part\_size,2);      % 设定搜索空间范围->解空间

region=10\*[-3,3;-3,3;-3,3;-3,3;-3,3;-3,3;-3,3;-3,3;-3,3;-3,3]; % 每一维设定不同范围(称之为解空间，不是可行域空间）

rand('state',sum(100\*clock));   % 重置随机数发生器状态

%当前种群的信息矩阵，逐代进化的群体 % 当前位置,随机初始化

% 一个10\*3的随机的矩阵（初始化所有粒子的所有维数的位置值），其中最后一列为

arr\_present = ini\_pos(pop\_size,part\_size);

% 初始化当前速度

% 一个10\*2的随机的矩阵（初始化所有粒子的所有维数的速度值）

v=ini\_v(pop\_size,part\_size);

%不是当前种群，可看作是一个外部的记忆体，存储每个粒子历史最优值（2维数值）：根据适应度更新！

%注意：pbest数组10\*3  最后一列保存的是适应度

pbest = zeros(pop\_size,part\_size+1);    % pbest：粒子以前搜索到的最优值，最后一列包括这些值的适应度

% 1\*80 保存每代的最优值

best\_record = zeros(part\_size+1,max\_gen);         % best\_record数组：记录每一代的最好的粒子的适应度

w\_max = 0.9;          %  w\_max权系数最大值

w\_min = 0.2;          %  w\_min权系数最小值

v\_max = 2;            %  最大速度,为粒子的范围宽度

c1 = 2;               %  学习因子1

c2 = 2;               %  学习因子2

% ————————————————————————

%   计算原始种群的适应度,及初始化

% ————————————————————————

% 注意：传入的第一个参数是当前的粒子群体 ，ini\_fit函数计算每个粒子的适应度

% arr\_present(:,end)是最后一列 ，保存每个粒子的适应值，是这样的！xuan

arr\_present(:,end)= ini\_fit( arr\_present, pop\_size, part\_size );

% 数组赋值，初始化每个粒子个体的历史最优值，以后会更新的

pbest = arr\_present;         % 初始化各个粒子最优值

% 找到当前群体中适应度最小的（在最后一列中寻找），best\_value

% 改为max，表示关联度最大

[best\_value best\_index] = max(arr\_present(:,end)); %初始化全局最优，即适应度为全局最小的值，根据需要也可以选取为最大值

% 唯一的全局最优值，是当前代所有粒子中最好的一个

gbest = arr\_present(best\_index,:);

% 因为是多目标，因此这个-----------------

% 只是示意性的画出3维的

%x=[-3:0.01:3];

%y=[-3:0.01:3];

%[X,Y]=meshgrid(x,y);

%Z1=(-10)\*exp((-0.2)\*sqrt(X^2+Y^2));

%Z2=(abs(X))^0.8+abs(Y)^0.8+5\*sin(X^3)+5\*sin(Y^3);

%z1=@(x,y)(-10)\*exp((-0.2)\*sqrt(x^2+y^2));

%z2=@(x,y)(abs(x))^0.8+abs(y)^0.8+5\*sin(x^3)+5\*sin(y^3);

%ezmeshc(z1);grid on;

%ezmeshc(z2);grid on;

%开始进化，直到最大代数截至

for i=1:max\_gen

    %grid on;

    %三维图象 %多维图象是画不出来的

    %ezmesh(z),hold on,grid on;

    %画出粒子群

    %plot3(arr\_present(:,1),arr\_present(:,2),arr\_present(:,3),'\*'),hold off;

    %drawnow

    %flush

    %pause(0.01);

    w = w\_max-(w\_max-w\_min)\*i/max\_gen; % 线形递减权重

    % 当前进化代数：对于每个粒子进行更新和评价----->>>>>>>

    for j=1:pop\_size

        v(j,:) = w.\*v(j,:)+c1.\*rand.\*(pbest(j,1:part\_size)-arr\_present(j,1:part\_size))...

            +c2.\*rand.\*(gbest(1:part\_size)-arr\_present(j,1:part\_size)); %  粒子速度更新 (a)

        % 判断v的大小，限制v的绝对值小于20———————————————————

        for k=1:part\_size

            if abs(v(j,k))>20

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

                v(j,k)=20\*rand();

            end

        end

        %前几列是位置信息

        arr\_present(j,1:part\_size) = arr\_present(j,1:part\_size)+v(j,1:part\_size);% 粒子位置更新 (b)

        %最后一列是适应度

        arr\_present(j,end) = fitness(part\_size,arr\_present(j,1:part\_size)); % 适应度更新 （保存至最后一列）

        %  适应度评价与可行域限制

        if (arr\_present(j,end)>pbest(j,end))&(Region\_in(arr\_present(j,:),region)) % 根据条件更新pbest,如果是最小的值为小于号,相反则为大于号

            pbest(j,:) = arr\_present(j,:);  % 更新个体的历史极值

        end

    end

    % 以下更新全局的极值

    [best best\_index] = max(arr\_present(:,end));                     % 如果是最小的值为min,相反则为max

    if best>gbest(end) & ( Region\_in(arr\_present(best\_index,:),region) ) % 如果当前最好的结果比以前的好，则更新最优值gbest,如果是最小的值为小于号,相反则为大于号

        gbest = arr\_present(best\_index,:); % 全局的极值

    end

    %------------混沌---------------------------------

    xlhd = gbest(1:part\_size);

    if(1)

       for p=1:25 %次数

        %1生成

              cxl=rand(1,part\_size);

              for j=1:part\_size

                  if cxl(j)==0

                      cxl(j)=0.1;

                  end

                  if cxl(j)==0.25

                      cxl(j)=0.26;

                  end

                  if cxl(j)==0.5

                      cxl(j)=0.51;

                  end

                  if cxl(j)==0.75

                      cxl(j)=0.76;

                  end

                  if cxl(j)==1

                      cxl(j)=0.9;

                  end

              end

        %2映射

              al=-30;bl=30;

              rxl=al+(bl-al)\*cxl;

        %3搜索

              bate = 0.1;

              xlhd=xlhd+bate\*rxl;

              if fitness(part\_size,xlhd)>gbest(end)

                  gbest(1:part\_size)=xlhd;

                  gbest(end)=fitness(part\_size,xlhd);

              end

        %4更新

             for j=1:part\_size

                  cxl(j)=4\*cxl(j)\*(1-cxl(j));

             end

       end

    end

    %-------------混沌--------------------------------

    %当前代的最优粒子的适应度（取自）保存

    best\_record(:,i) = gbest; % gbest：一个行向量

end

pso = gbest; % 最优个体

display(gbest);

figure;

plot(best\_record(end,:));% 最优解与代数的进化关系图

best=zeros(part\_size,max\_gen);

for i=1:part\_size-1

    best(i,:)=best\_record(i,:);

end

pareto1= zeros(1,max\_gen);

pareto2= zeros(1,max\_gen);

for i=1:max\_gen

    pareto1(i)=f1(part\_size, best(:,i) );

    pareto2(i)=f2(part\_size, best(:,i) );

end

figure;

i=1:max\_gen;

%plot(i,pareto1(i),'r\*',i,pareto2(i),'g\*');

plot(pareto1(i),pareto2(i),'r+');

xlabel('f1');ylabel('f2');

title('Pareto曲线');

%figure;

%plot(,f2(best\_record),);

% movie2avi(F,'pso\_2D1.avi','compression','MSVC');

%子函数

%-------------------------------------------------------------------------

%-------------------------------------------------------------------------

%返回随机的位置

function ini\_present=ini\_pos(pop\_size,part\_size)

ini\_present = 10\*3\*rand(pop\_size,part\_size+1);       %初始化当前粒子位置,使其随机的分布在工作空间

%返回一个随机的矩阵，10\*（2+1），最后一列将用来保存适应度

%返回随机的速度

function ini\_velocity=ini\_v(pop\_size,part\_size)

ini\_velocity =20\*(rand(pop\_size,part\_size));     %初始化当前粒子速度,使其随机的分布在速度范围内

%判断是否处于范围内

function flag = Region\_in(pos\_present,region)

[m n]=size(pos\_present); % 1\*11  n返回解的维数10

flag=1;

for j=1:n-1

    flag = flag & ( pos\_present(1,j)>=region(j,1) ) & ( pos\_present(1,j)<=region(j,2) );

end

%初始化适应度

function arr\_fitness = ini\_fit(pos\_present,pop\_size,part\_size)

for k=1:pop\_size

    arr\_fitness(k,1) = fitness(part\_size,pos\_present(k,1:part\_size));  %计算原始种群的适应度

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%    计算适应度

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function fit = fitness(n,xp)

%需要求极值的函数,本例即peaks函数

%y0=[-85.4974,-29.9217]; % 注意：这是基准序列，也就是单个最优的极值

y0=[-9.9907,-7.7507];

%y0=[-39.6162,-18.4561];

% y0=[-86.8312,-29.9217];

y1=[f1(n,xp),f2(n,xp)];  % n为粒子维数

fit=graydegree(2,y0,y1); % 关联度在某种意义上就是适应度

%目标函数1

function r=f1(n,x)

r=0;

for i=1:n-1

    r=r+(-10)\*exp((-0.2)\*sqrt(x(i)^2+x(i+1)^2));

end

%目标函数2

function r=f2(n,x)

r=0;

for i=1:n

    r=r+(abs(x(i)))^0.8+5\*sin(x(i)^3);

end

%约束函数1

function r=g1(n,x)

r=0;

for i=1:n

    r=0;

end

%约束函数2

function r=g2(n,x)

r=0;

for i=1:n

    r=0;

end

%  灰色关联度计算函数 ( 越大相似性越好 )

%  tn目标函数个数   x0基准序列（一组值）   x1贷检（一组值）

function gama = graydegree( tn,y0,y1 )

gama=0;

rou =0.5;

kesa= zeros(tn,1);

m1= abs(y0(1)-y1(1)) ;

m2= abs(y0(1)-y1(1)) ;

for i=1:tn

    if( abs(y0(i)-y1(i))>m2 ) %------------------应该取大于呢还是小于

        m2= abs(y0(i)-y1(i));

    end

end

for i=1:tn

    kesa(i) = ( m1+rou\*m2)/( abs(y0(i)-y1(i)) +rou\*m2 );

    gama = gama + kesa(i);

end

gama = gama/tn;

% 可行解的判决函数  gn为约束条件的个数(暂时未用)   n为解（粒子）的维数

function bool = feasible( x,n )

r=0;

%for i=1:gn

    r=max( 0, g1(n,x), g2(n,x) );%判断约束条件

%end

if(r>0)

    bool=0; %不可行解

else

    bool=1; %可行解

end