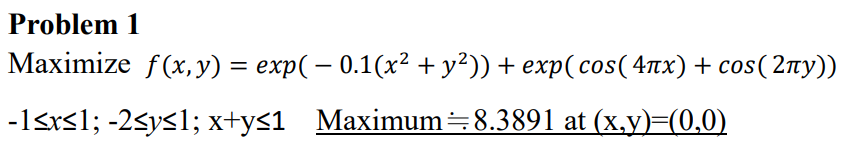
**計算智慧於工程上的應用**

**PSO Homework**

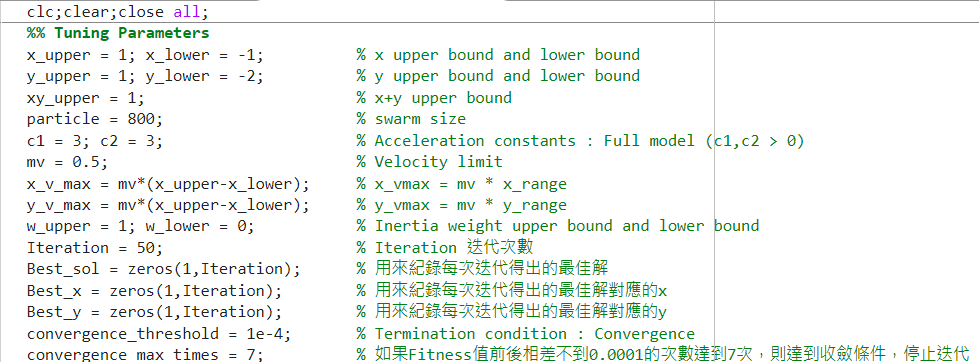
**M11205314**

**張原嘉**

Programming Language : Matlab



Code explanations



**Swarm size : 800**

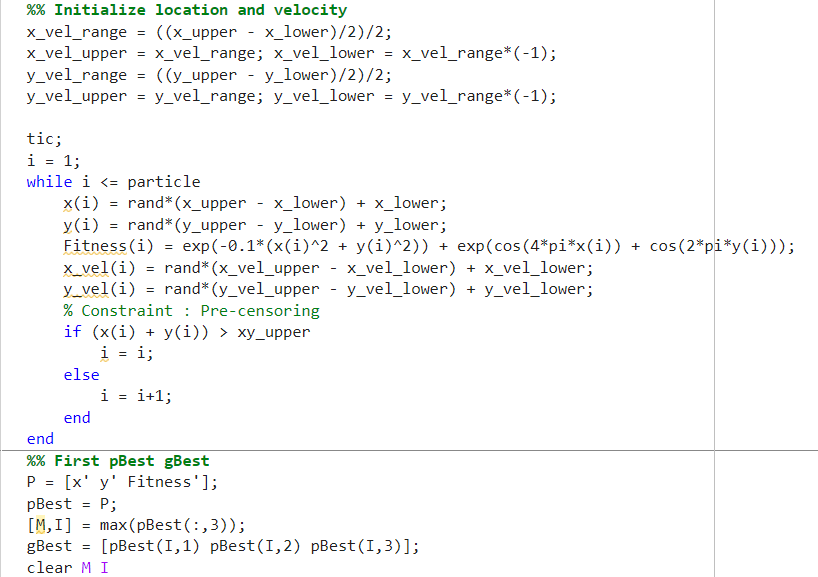
**迭代次數 : 50次**

**Termination condition : 當Fitness前後相差不到10^-4的次數超過7次時，達到收斂，停止迭代**

**Inertia weight : 使用從1遞減到0的w**

**Acceleration constants : Full model (c1 = c2 = 3)**

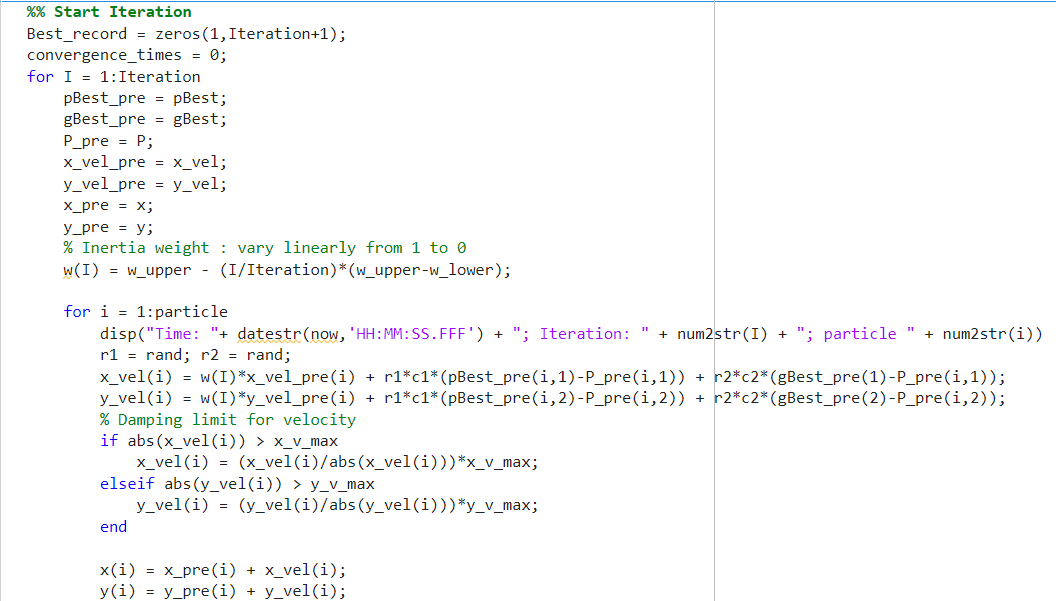
**Velocity maximum : mv \* range (mv = 0.5)**



**首先隨機產生初始的x和y值，這邊一開始就設定x和y的產出範圍會在上下界之內，所以不需再額外設定Constraint，但是x+y仍然需要使用Constraint，這邊使用的是Pre-censoring。**

**接著隨機產生x和y的初始速度，初始速度假設不超過range的一半，所以除以2，接著分成負到正，所以再除以2，舉例來說x是-1~1，所以x range等於(1-(-1))/2 = 1，接著再除以2，得到x velocity的範圍是(-0.5，0.5)，然後把x和y分別加上各自的velocity得到新的x和y並計算Fitness。**

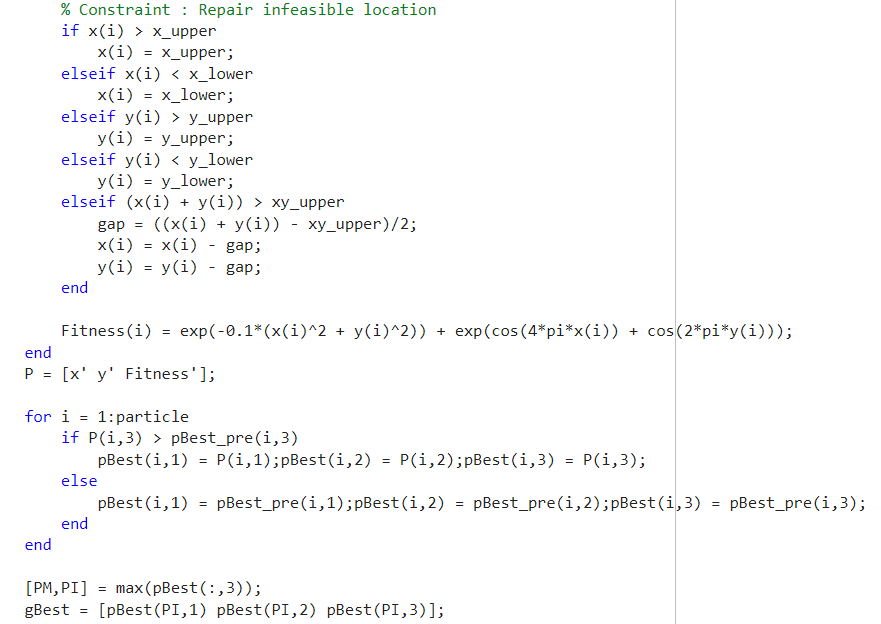
**計算pBest和gBest，初始的pBest就是一開始的結果，gBest就是pBest裡Fitness最好的那個particle。**



**接著都是重複之前的步驟，所以用for迴圈進行迭代，將剛剛計算好的pBest、gBest、x y和velocity都用新變數取代，新變數會在名字後面加個pre(previous縮寫)，並用來計算下一批新的參數。**

**這邊的Inertia weight使用從1線性變化到0的遞減。**

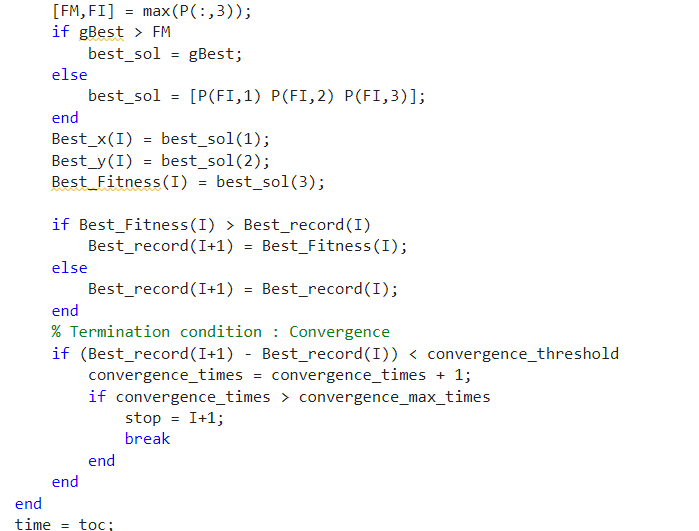
**會對計算出來的速度做限制，上限界不能超過指定的值Vmax。**

****

**接著會對計算出來的x和y進行限制，這裡使用的限制是Repair infeasible，將超過上下界的值直接修改成與上下界相同的值。**

**計算Fitness，接著拿計算出來的Fitness和上一批pBest做比較，若Fitness比之前的pBest大就取代掉，反之就維持原樣，最後得出新的一批pBest。**

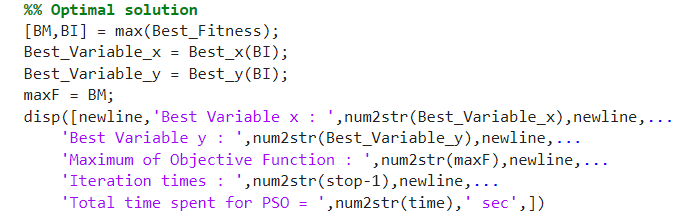
**gBest就是新的pBest裡最佳的那個解。**

****

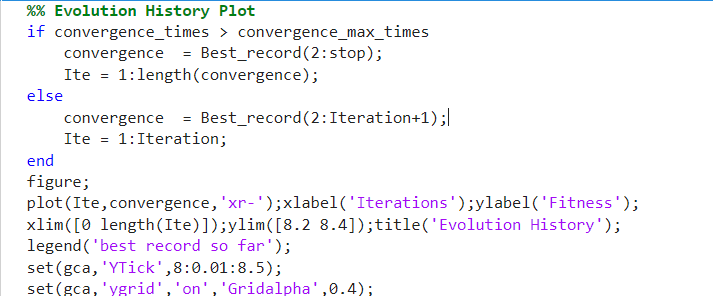
**接著會拿gBest和總群體得出的最佳Fitness做比較，比較誰才是最佳解，比較出來後會把最佳的那個解和xy定義為best\_sol，其中再把的x和y定義為Best\_x和Best\_y，最佳解定義為Best\_Fitness。**

**因為會有好幾次迭代，所以每一次都會拿最佳解Best\_Fitness去和上一次的做比較，大者取代，小者維持原樣，最後把值輸進Best\_record裡(代表最好的歷史紀錄)。**

**接著對每次的最佳紀錄Best\_record去做比較，如果最佳紀錄沒有明顯的增長，這裡定義沒有明顯的增長是指前面定義的convergence\_threshold = 1e-4 (0.0001)的次數達到convergence\_max\_times(設定7次)就停止迭代。**

****

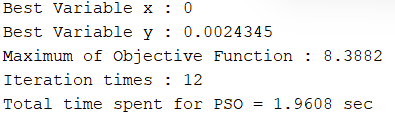
**找出最佳解和對應的xy並顯示出來，並顯示迭代次數和Code執行的時長。**

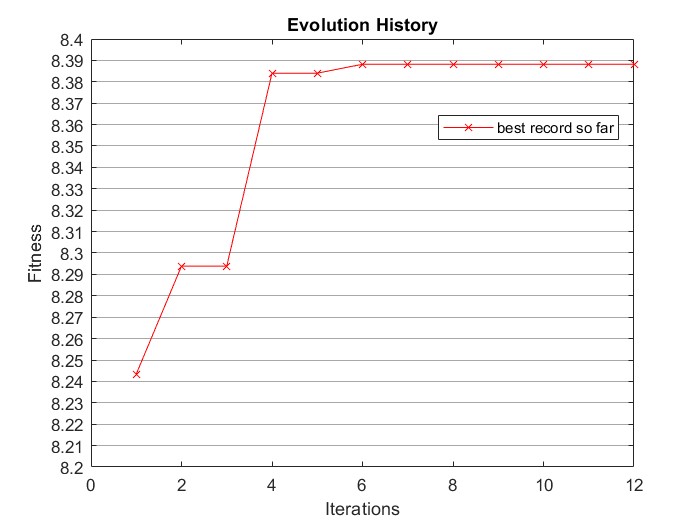
****

**畫出每次迭代的最佳紀錄(Best record so far)**

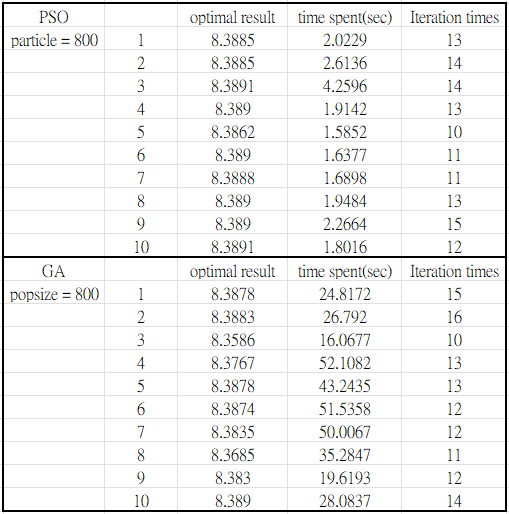
**這邊的if是指假如迭代因為收斂而提前停止時畫出來的圖和假如迭代沒有提前停止(迭代次數不夠長或遲遲沒有收斂)而畫出來的圖。**

Find optimal solution(x and y) and convergence history

****

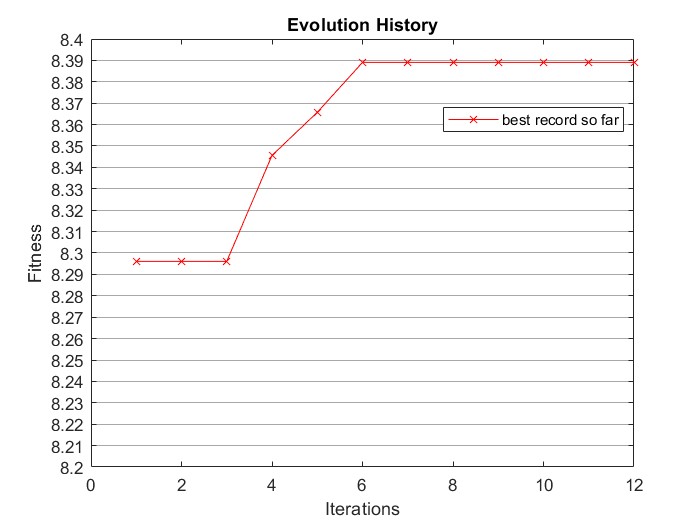
****

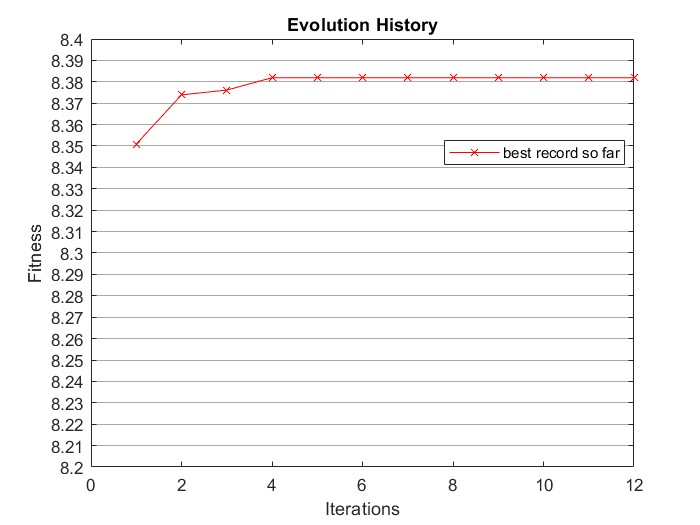
Comparison with GA



將PSO和GA分別執行十次的結果，兩個終止條件一樣，都是收斂沒有明顯變化時就停止迭代，可以看出兩種方法得到的最佳解都差不多相近，但GA耗時卻比較久一點。

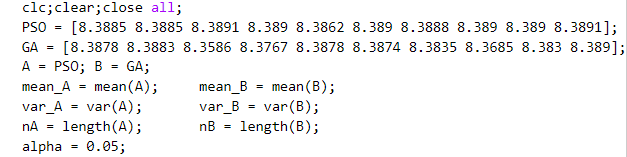
Compare the convergence histories of PSO and GA



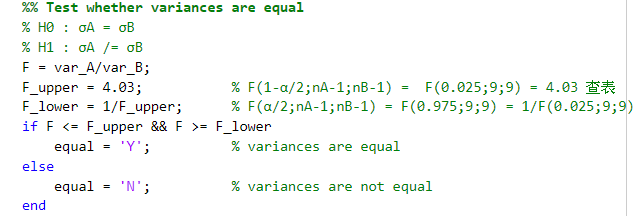


此為其中一次PSO和GA的Evolution histories執行結果

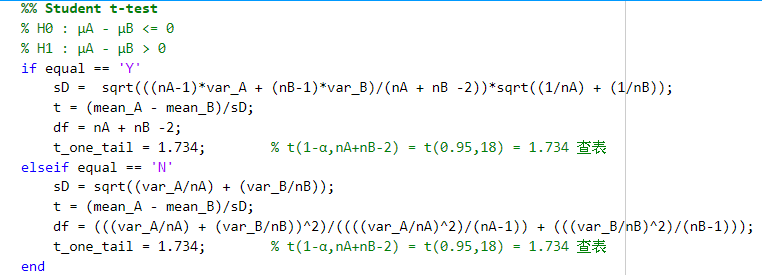
Draw a conclusion on whether one algorithm is significantly superior to the other



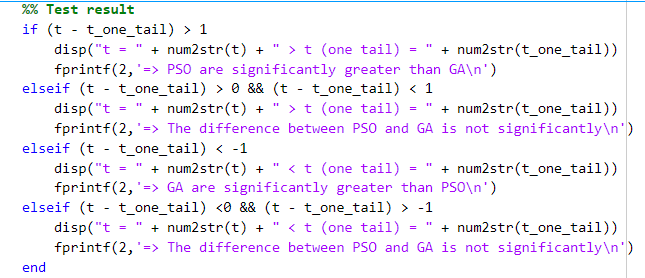
將PSO十次的結果與GA十次的結果分別計算平均數和變異數，PSO令為A，GA令為B，α值設為0.05。

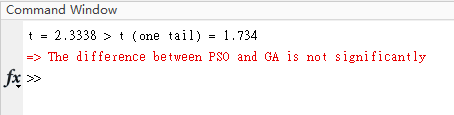


首先確認兩者的標準差是否相同，F的上下限分別使用講義的P22的表格查詢與計算。



這裡因為取樣次數比較少(只有十次)，使用Student t-test來檢驗，根據上面得出的結果得知標準差相同與否，會分別對應不同的sD和df的計算方式。





得出t和t(one tail)相差只有0.6，兩者差異不大，PSO和GA兩種演算法沒有明顯的差距

Matlab Code:

clc;clear;close all;

%% Tuning Parameters

x\_upper = 1; x\_lower = -1; % x upper bound and lower bound

y\_upper = 1; y\_lower = -2; % y upper bound and lower bound

xy\_upper = 1; % x+y upper bound

particle = 800; % swarm size

c1 = 3; c2 = 3; % Acceleration constants : Full model (c1,c2 > 0)

mv = 0.5; % Velocity limit

x\_v\_max = mv\*(x\_upper-x\_lower); % x\_vmax = mv \* x\_range

y\_v\_max = mv\*(x\_upper-x\_lower); % y\_vmax = mv \* y\_range

w\_upper = 1; w\_lower = 0; % Inertia weight upper bound and lower bound

Iteration = 50; % Iteration 迭代次數

Best\_sol = zeros(1,Iteration); % 用來紀錄每次迭代得出的最佳解

Best\_x = zeros(1,Iteration); % 用來紀錄每次迭代得出的最佳解對應的x

Best\_y = zeros(1,Iteration); % 用來紀錄每次迭代得出的最佳解對應的y

convergence\_threshold = 1e-4; % Termination condition : Convergence

convergence\_max\_times = 7; % 如果Fitness值前後相差不到0.0001的次數達到7次，則達到收斂條件，停止迭代

%% Initialize location and velocity

x\_vel\_range = ((x\_upper - x\_lower)/2)/2;

x\_vel\_upper = x\_vel\_range; x\_vel\_lower = x\_vel\_range\*(-1);

y\_vel\_range = ((y\_upper - y\_lower)/2)/2;

y\_vel\_upper = y\_vel\_range; y\_vel\_lower = y\_vel\_range\*(-1);

tic;

i = 1;

while i <= particle

x(i) = rand\*(x\_upper - x\_lower) + x\_lower;

y(i) = rand\*(y\_upper - y\_lower) + y\_lower;

Fitness(i) = exp(-0.1\*(x(i)^2 + y(i)^2)) + exp(cos(4\*pi\*x(i)) + cos(2\*pi\*y(i)));

x\_vel(i) = rand\*(x\_vel\_upper - x\_vel\_lower) + x\_vel\_lower;

y\_vel(i) = rand\*(y\_vel\_upper - y\_vel\_lower) + y\_vel\_lower;

% Constraint : Pre-censoring

if (x(i) + y(i)) > xy\_upper

i = i;

else

i = i+1;

end

end

%% First pBest gBest

P = [x' y' Fitness'];

pBest = P;

[M,I] = max(pBest(:,3));

gBest = [pBest(I,1) pBest(I,2) pBest(I,3)];

clear M I

%% Start Iteration

Best\_record = zeros(1,Iteration+1);

convergence\_times = 0;

for I = 1:Iteration

pBest\_pre = pBest;

gBest\_pre = gBest;

P\_pre = P;

x\_vel\_pre = x\_vel;

y\_vel\_pre = y\_vel;

x\_pre = x;

y\_pre = y;

% Inertia weight : vary linearly from 1 to 0

w(I) = w\_upper - (I/Iteration)\*(w\_upper-w\_lower);

for i = 1:particle

disp("Time: "+ datestr(now,'HH:MM:SS.FFF') + "; Iteration: " + num2str(I) + "; particle " + num2str(i))

r1 = rand; r2 = rand;

x\_vel(i) = w(I)\*x\_vel\_pre(i) + r1\*c1\*(pBest\_pre(i,1)-P\_pre(i,1)) + r2\*c2\*(gBest\_pre(1)-P\_pre(i,1));

y\_vel(i) = w(I)\*y\_vel\_pre(i) + r1\*c1\*(pBest\_pre(i,2)-P\_pre(i,2)) + r2\*c2\*(gBest\_pre(2)-P\_pre(i,2));

% Damping limit for velocity

if abs(x\_vel(i)) > x\_v\_max

x\_vel(i) = (x\_vel(i)/abs(x\_vel(i)))\*x\_v\_max;

elseif abs(y\_vel(i)) > y\_v\_max

y\_vel(i) = (y\_vel(i)/abs(y\_vel(i)))\*y\_v\_max;

end

x(i) = x\_pre(i) + x\_vel(i);

y(i) = y\_pre(i) + y\_vel(i);

% Constraint : Repair infeasible location

if x(i) > x\_upper

x(i) = x\_upper;

elseif x(i) < x\_lower

x(i) = x\_lower;

elseif y(i) > y\_upper

y(i) = y\_upper;

elseif y(i) < y\_lower

y(i) = y\_lower;

elseif (x(i) + y(i)) > xy\_upper

gap = ((x(i) + y(i)) - xy\_upper)/2;

x(i) = x(i) - gap;

y(i) = y(i) - gap;

end

Fitness(i) = exp(-0.1\*(x(i)^2 + y(i)^2)) + exp(cos(4\*pi\*x(i)) + cos(2\*pi\*y(i)));

end

P = [x' y' Fitness'];

for i = 1:particle

if P(i,3) > pBest\_pre(i,3)

pBest(i,1) = P(i,1);pBest(i,2) = P(i,2);pBest(i,3) = P(i,3);

else

pBest(i,1) = pBest\_pre(i,1);pBest(i,2) = pBest\_pre(i,2);pBest(i,3) = pBest\_pre(i,3);

end

end

[PM,PI] = max(pBest(:,3));

gBest = [pBest(PI,1) pBest(PI,2) pBest(PI,3)];

[FM,FI] = max(P(:,3));

if gBest > FM

best\_sol = gBest;

else

best\_sol = [P(FI,1) P(FI,2) P(FI,3)];

end

Best\_x(I) = best\_sol(1);

Best\_y(I) = best\_sol(2);

Best\_Fitness(I) = best\_sol(3);

if Best\_Fitness(I) > Best\_record(I)

Best\_record(I+1) = Best\_Fitness(I);

else

Best\_record(I+1) = Best\_record(I);

end

% Termination condition : Convergence

if (Best\_record(I+1) - Best\_record(I)) < convergence\_threshold

convergence\_times = convergence\_times + 1;

if convergence\_times > convergence\_max\_times

stop = I+1;

break

end

end

end

time = toc;

%% Optimal solution

[BM,BI] = max(Best\_Fitness);

Best\_Variable\_x = Best\_x(BI);

Best\_Variable\_y = Best\_y(BI);

maxF = BM;

disp([newline,'Best Variable x : ',num2str(Best\_Variable\_x),newline,...

'Best Variable y : ',num2str(Best\_Variable\_y),newline,...

'Maximum of Objective Function : ',num2str(maxF),newline,...

'Iteration times : ',num2str(stop-1),newline,...

'Total time spent for PSO = ',num2str(time),' sec',])

%% Evolution History Plot

if convergence\_times > convergence\_max\_times

convergence = Best\_record(2:stop);

Ite = 1:length(convergence);

else

convergence = Best\_record(2:Iteration+1);

Ite = 1:Iteration;

end

figure;

plot(Ite,convergence,'xr-');xlabel('Iterations');ylabel('Fitness');

xlim([0 length(Ite)]);ylim([8.2 8.4]);title('Evolution History');

legend('best record so far');

set(gca,'YTick',8:0.01:8.5);

set(gca,'ygrid','on','Gridalpha',0.4);