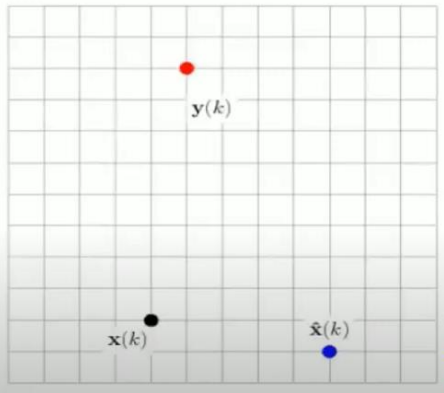
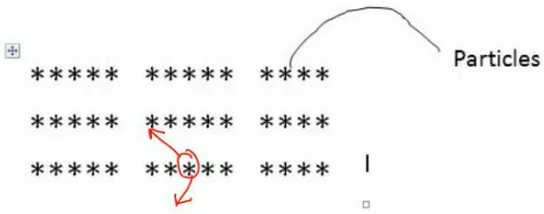
PSO solution update in 2D



feasible area ↕ Searching space

- x(k) - Current solution (4, 2)
- PBest - Particle's best solution (9, 1)
- GBest-Global best solution (5, 10)

Basic Overview/terms



No. of Particles ------ Swarm: (n)

Swarms move in a group with some certain speed ------ Inertia: (ω)

How they respond in group movement ------ Correction factor: (c_1, c_2)

VELOCITY & POSITION UPDATE EQUATIONS:

Mechanics of P.S.O.

$$v_{ij} = wv_{ij} + c_1 r_1 (pbest_{ij} - x_{ij}) + c_2 r_2 (gbest - x_{ij})$$

$$x_{ij} = v_{ij} + x_{ij}$$

rand. No. (0,1)

pbest → personal best
gbest → global

$$w = w_{max} - \left(\frac{iter}{maxiter}\right)(w_{max} - w_{min})$$

where

- w_{max}: final weight
- w_{min}: initial weight
- $maxiter$: maximum iteration number
- $iter$: current iteration number

min

	A	B
1	4	
2	-5	
3	3	
4	1	
5	9	

Test Optimization Problem

Minimize $f(x) = 10(x_1 - 1)^2 + 20x_1 x_2 + (x_3 - 3)^2$

Subject to

$$x_1 + x_2 + x_3 \leq 5$$
$$x_1^2 + x_2^2 - x_3 \leq 0$$
$$0 \leq x_1, x_2, x_3 \leq 10$$

PHASES OF THE PSO MATLAB CODE

Phase 1: Define Objective Function

Phase 2: PSO Parameters

Phase 3: Initialization of POSITION & VELOCITY

Phase 4: Function Evaluation

Phase 5: Compute pbest & gbest

Phase 6: Update Velocity & Position

Handling Boundary Constraints

Phase 7: Store Best Value

PHASES OF THE PSO MATLAB CODE

Phase 1: Define Objective Function

Phase 2: PSO Parameters

----- Loop for Maximum Run Start

Phase 3: Initialization of POSITION & VELOCITY

Phase 4: Function Evaluation

********* PSO Algorithm Start here

Phase 5: Compute pbest & gbest

Phase 6: Update Velocity & Position

Handling Boundary Constraints

********* PSO Algorithm End here

Phase 7: Store Best Value

----- Loop for Maximum Run End

Phase 1: DEFINING OBJECTIVE FUNCTION

Save the following code in MATLAB script file (*.m) and save as **fun.m**

```
function output = fun(X)

x1=X(:,1); x2=X(:,2); x3=X(:,3);
%%%%% defining Objective function here
fx= 10*(x1-1)^2 + 20.*x1.*x2 + (x3-3)^2;
%%%%% defining all constraints here
%%% All constraints must be converted into <= form
Con=[];
Con(1)=x1+x2+x3-5;
Con(2)=x1^2+x2^2-x3;
%%% defining Penalty for each constraint
for i=1:length(Con)
    If Con(i)>0
        Pen(i)=1;
    else
        Pen(i)=0;
    end
end
Penalty = 10000;                    %%Penalty on each constraint
output = fx + Penalty*sum(Pen);     %% fitness function
```

Phase 2: PSO Parameters

```
LB = [0 0 0];          % Lower Bound of the variables
UB=[10 10 10];         % Upper Bound of the variables
m = 3;                 % Number of variables
n = 500;               % Population Size (Swarm Size)
wmax = 0.9;            % Inertia weight Maximum
wmin = 0.4;            % Inertia weight Minimum
c1 = 2.05;             % acceleration factor
c2 = 2.05;             % acceleration factor
Maxiter = 100;         % maximum number of iteration
```

Phase 3: Initialization of POSITION & VELOCITY

```
for i=1:n
    for j=1:m
        pos(i,j)=LB(i,j) + rand().*(UB(i,j) – LB(i,j));
    end
    end
vel=0.1.*pos;
```

Phase 4: Function Evaluation

```
for i = 1 : n
      out(i,1)=fun(pos(i,:)));
end
            %%%% initial pbest
pbestval = out;
pbest = pos;
                  %%%% initial gbest
[fminval, index]=min(out);
gbest=pbest(index,:);
```

Phase 5: COMPUTE PBEST & GBEST

Pso 5

```
X = pos;
Out = fun(X);

%%% UPDATE PBEST VALUES

Har = find(out<=pbestval);        % New min pbestvals
pbest(Har,:)=X(Har,:);            % update pbest positions
pbestval = out(Har);              % update pbestval

%%%% UPDATE GBEST VALUES

[fbestval, ind1]= min(pbestval);
if  fbestval<=fminval
      fminvalue = fbestval;
      gbest = pbest(ind1,:);
end
```

Phase 6: UPDATE VELOCITY & POSITION

```
for i = 1: n
for j = 1: m
%%% update velocity
    vel(i, j) = w.*vel(i, j) +
                c1.*rand().*(pbest(i,j) – pos(i,j)) +
                c2.*rand().*(gbest(1,j) – pos(i, j));
%%% update position
    pos(i, j) = vel(i, j) + pos(i, j);

%%% Handling Boundary constraints
    if pos(i,j)<LB(j)
        pos(i,j)=LB(j);
    elseif pos(i,j)>UB(j)
        pos(i,j)=UB(j)
    end
end          %%% end of j = 1: m  loop
end          %%% end of i = 1: n  loop
```

Phase 7: STORE BEST VALUE

```
F_ans(run)=fun(gbest);   % store best value in each run
F_gbest(run,:)=gbest;    % store best Position value
```

COMPLETE CODE

```
format short
clear all
clc

Phase 2: PSO Parameters

----- Loop for Maximum Run Start

Phase 3: Initialization of POSITION & VELOCITY

Phase 4: Function Evaluation

********* PSO Algorithm Start here
iter = 1;

while iter < = Maxiter
    w = wmax – (iter/ Maxiter).*(wmax-wmin);
```

```
while iter < = Maxiter
    w = wmax – (iter/ Maxiter).*(wmax-wmin);

Phase 5: Compute pbest & gbest

Phase 6: Update Velocity & Position

    Handling Boundary Constraints

iter = iter + 1

end                    % end of while loop

********* PSO Algorithm End here
Phase 7: Store Best Value

----- Loop for Maximum Run End
```

COMPLETE CODE - Cont

```
format short
clear all
clc

Phase 2: PSO Parameters

for run = 1: 500

Phase 3:

Phase 4:        Phase 5:        Phase 6:

Phase 7: Store Best Values

F_ans(run)=fun(gbest);   % store best value in each run
F_gbest(run,:)=gbest;    % store best Position value

end                      % end of RUN FOR LOOP

[bestFUN, bestRUN]=min(F_ans)  % find the optimal value
Best_X=F_gbest(bestRUN,:);     % find optimal solution
%%%%%% PLOTTING

plot(F_ans)
xlabel('Iteration')
ylabel('Fitness function value')
title('PSO Convergence Graph')
%%%%%% END OF THE CODE %%%%%%%%
```