UNIVERSITE Ahmed zabana RELIZANE
Institut des sciences et technologies Département de Génie Electrique
2 ème année Master : Electrotechnique Industrielle

TP 03 : technique d'optimisation PSO

1. Objectif du Tp :

L'objectif de ce TP est d'implémenter un tel système d'optimisation PSO sous matlab en tirant du caractère matriciel de cet environnement.

2. Algorithme PSO

Particle Swarm Optimization Function File: Sphere(x)

```
function F1 = Sphere(x)
F1 = sum(x.^2);
end
```

```
%%Particle Swarm Optimization File Name Save as: pso.m
clear;
close all;
%% Fitness Function Calling
FitnessFunction=@(x) Sphere(x);        % Fitness Function Calling

% Total Number of Decision Variables Used
nVar=10;

% Size of Decision Variables Matrix
VarSize=[1 nVar];

% Lower Bound
LowerBound =-10;

% Upper Bound
UpperBound = 10;

%% Parameters Initialization Phase

% Maximum Number of Iterations used.
MaxT=100;

% Total Number of Search Agents used.
PopulationSize = 10;

% Initialize PSO Parameters
% Inertia Weight
w=1;

% Inertia Weight Damping Ratio
wdamp=0.99;
```

```
% Personal Learning Coefficient
c1=1.5;

% Global Learning Coefficient
c2=2.0;

% Velocity Limits
VelMax=0.1*(UpperBound-LowerBound);
VelMin=-VelMax;

%% Initialization Position, Cost, Velocity, Best_Position, Best_Cost
empty_particle.Position=[];
empty_particle.Cost=[];
empty_particle.Velocity=[];
empty_particle.Best.Position=[];
empty_particle.Best.Cost=[];
particle=repmat(empty_particle,PopulationSize ,1);
GlobalBest.Cost=inf;

for i=1:PopulationSize
% Initialize Position for each search Agent in the search space
    particle(i).Position=unifrnd(LowerBound,UpperBound,VarSize);
% Initialize Velocity for each search Agent in the search space
    particle(i).Velocity=zeros(VarSize);
 % Fitness Values Calculation for each search Agent in the search space
    particle(i).Cost=FitnessFunction(particle(i).Position);
% Update Personal Best Position for the particles
    particle(i).Best.Position=particle(i).Position;
    particle(i).Best.Cost=particle(i).Cost;
 % Update Global Best Position for each search Agent in the search space
    if particle(i).Best.Cost<GlobalBest.Cost
        GlobalBest=particle(i).Best;
    end
end
BestCost=zeros(MaxT,1);
%% PSO Main Loop
for CurrentIteration=1:MaxT
    for i=1:PopulationSize
 % Update Velocity for each search Agent in the search space
        particle(i).Velocity = w*particle(i).Velocity
+c1*rand(VarSize).*(particle(i).Best.Position-particle(i).Position)
+c2*rand(VarSize).*(GlobalBest.Position-particle(i).Position);
 % Apply Velocity Limits
        particle(i).Velocity = max(particle(i).Velocity,VelMin);
        particle(i).Velocity = min(particle(i).Velocity,VelMax);
 % Update Position for Each Particle
        particle(i).Position = particle(i).Position + particle(i).Velocity;
 %  % Check Boundries [-10, 10]
        Outside=(particle(i).Position<LowerBound |
particle(i).Position>UpperBound);
        particle(i).Velocity(Outside)=-particle(i).Velocity(Outside);
        particle(i).Position = max(particle(i).Position,LowerBound);
        particle(i).Position = min(particle(i).Position,UpperBound);
 % Fitness Values Calculation
        particle(i).Cost = FitnessFunction(particle(i).Position);
% Update Personal Best
```

```
        if particle(i).Cost<particle(i).Best.Cost
            particle(i).Best.Position=particle(i).Position;
            particle(i).Best.Cost=particle(i).Cost;
 % Update Global Best
            if particle(i).Best.Cost<GlobalBest.Cost
                GlobalBest=particle(i).Best;
            end
        end
end

BestCost(CurrentIteration)=GlobalBest.Cost;
disp(['Current Iteration Number = ' num2str(CurrentIteration) ': Best Cost
Found = ' num2str(BestCost(CurrentIteration))]);

 w=w*wdamp;

end
BestSol = GlobalBest;
%% Results
figure;
%plot(BestCost,'LineWidth',2);
semilogy(BestCost,'LineWidth',2);
xlabel('Iteration Numbers');
ylabel('Best Cost Found');
grid on;
```

3. Travail demandé

1. Exécuter ce programme
2. Expliquer les étapes de cet algorithme
3. Donner le critère d'arrêt
4. Donner le résultat final ainsi la courbe .