%PARTICLE SWARM OPTIMIZATION

clc;

clear all;

%% Problem Definition

CostFunction=@(x) (1/fobjco2egr(x)); % Cost Function

nVar=3; % Number of Decision Variables

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

VarMin=[0.23 1071 31]; % Lower Bound of Variables

VarMax=[0.625 1300 40]; % Upper Bound of Variables

%% PSO Parameters

MaxIt=50; % Maximum Number of Iterations

nPop=200; % Population Size (Swarm Size)

% PSO Parameters

w=1; % Inertia Weight

wdamp=0.99; % Inertia Weight Damping Ratio

c1=1.5; % Personal Learning Coefficient

c2=2; % Global Learning Coefficient

% If you would like to use Constriction Coefficients for PSO,

% uncomment the following block and comment the above set of parameters.

% % Constriction Coefficients

% phi1=2.05;

% phi2=2.05;

% phi=phi1+phi2;

% chi=2/(phi-2+sqrt(phi^2-4\*phi));

% w=chi; % Inertia Weight

% wdamp=1; % Inertia Weight Damping Ratio

% c1=chi\*phi1; % Personal Learning Coefficient

% c2=chi\*phi2; % Global Learning Coefficient

% Velocity Limits

VelMax=0.1\*(VarMax-VarMin);

VelMin=-VelMax;

%% Initialization

empty\_particle.Position=[];

empty\_particle.Cost=[];

empty\_particle.Velocity=[];

empty\_particle.Best.Position=[];

empty\_particle.Best.Cost=[];

particle=repmat(empty\_particle,nPop,1);

GlobalBest.Cost=inf;

for i=1:nPop

% Initialize Position

particle(i).Position=unifrnd(VarMin,VarMax,VarSize);

% Initialize Velocity

particle(i).Velocity=zeros(VarSize);

% Evaluation

particle(i).Cost=CostFunction(particle(i).Position);

% Update Personal Best

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

BestCost=zeros(MaxIt,1);

%% PSO Main Loop

for it=1:MaxIt

for i=1:nPop

% Update Velocity

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

particle(i).Position = particle(i).Position + particle(i).Velocity;

% Velocity Mirror Effect

IsOutside=(particle(i).Position<VarMin | particle(i).Position>VarMax);

particle(i).Velocity(IsOutside)=-particle(i).Velocity(IsOutside);

% Apply Position Limits

particle(i).Position = max(particle(i).Position,VarMin);

particle(i).Position = min(particle(i).Position,VarMax);

% Evaluation

particle(i).Cost = CostFunction(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(it)=1/GlobalBest.Cost;

disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(BestCost(it))]);

w=w\*wdamp;

end

BestSol = GlobalBest

%% Results

figure(gcf)

title('Grafik Nilai Maksimum PSO','color','b')

xlabel('Jumlah Iterasi')

ylabel('Nilai Fungsi Obyektif')

hold on

grid on

plot(BestCost,'LineWidth',2);

hold on

% plot(BestCost,'LineWidth',2);

% semilogy(BestCost,'LineWidth',2);

% xlabel('Jumlah iterasi');

% ylabel('Nilai fungsi obyektif');

% grid on;

% save ('PSOalone.mat')