OPTIMIZATION PROJECT REVIEW

Soumya Ranjan
17BEE0134
A1

Code:

Sphere function:

```
function z=Sphere(x)

    z=sum(x.^2);

end
```

Harmony Search Algorithm:

```
%R = unifrnd(A,B) returns an array R of random
numbers generated from the continuous uniform
distributions with lower and upper endpoints
specified by A and B ,
%B = repmat(A,m,n) creates a large matrix B
consisting of an m -by- n tiling of copies of A .
The statement repmat(A,n ) creates an n -by- n
tiling. B = repmat(A,[m n]) accomplishes the same
result as repmat(A,m,n) .
clc;
clear;
close all;

%% Problem Definition
CostFunction=@(x) Sphere(x);        % Cost Function
nVar=5;              % Number of Deciison Variables
VarSize=[1 nVar];    % Decision Variables Matrix
Size
VarMin=-10;          % Decision Variables Lower
Bound
VarMax= 10;          % Decision Variables Upper
Bound
%% Harmony Search Parameters
```

```
MaxIt=5000;      % Maximum Number of Iterations
HMS=25;          % Harmony Memory Size
nNew=20;         % Number of New Harmonies
HMCR=0.9;        % Harmony Memory Consideration Rate
PAR=0.1;         % Pitch Adjustment Rate
FW=0.02*(VarMax-VarMin);     % Fret Width
(Bandwidth)
FW_damp=0.995;               % Fret Width Damp Ratio
%% Initialization

% Empty Harmony Structure
empty_harmony.Position=[];
empty_harmony.Cost=[];

% Initialize Harmony Memory
HM=repmat(empty_harmony,HMS,1);
% Create Initial Harmonies
for i=1:HMS
    HM(i).Position=unifrnd(VarMin,VarMax,VarSize);
    HM(i).Cost=CostFunction(HM(i).Position);
end

% Sort Harmony Memory
[~, SortOrder]=sort([HM.Cost]);
HM=HM(SortOrder);

% Update Best Solution Ever Found
BestSol=HM(1);

% Array to Hold Best Cost Values
BestCost=zeros(MaxIt,1);

%% Harmony Search Main Loop

for it=1:MaxIt

    % Initialize Array for New Harmonies
    NEW=repmat(empty_harmony,nNew,1);
```

```
    % Create New Harmonies
    for k=1:nNew

        % Create New Harmony Position

NEW(k).Position=unifrnd(VarMin,VarMax,VarSize);
        for j=1:nVar
            if rand<=HMCR
                % Use Harmony Memory
                i=randi([1 HMS]);

NEW(k).Position(j)=HM(i).Position(j);
            end

            % Pitch Adjustment
            if rand<=PAR
                %DELTA=FW*unifrnd(-1,+1);    %
Uniform
                DELTA=FW*randn();           %
Gaussian (Normal)

NEW(k).Position(j)=NEW(k).Position(j)+DELTA;
            end

        end

        % Apply Variable Limits

NEW(k).Position=max(NEW(k).Position,VarMin);

NEW(k).Position=min(NEW(k).Position,VarMax);

        % Evaluation
        NEW(k).Cost=CostFunction(NEW(k).Position);

    end

    % Merge Harmony Memory and New Harmonies
    HM=[HM
```

```
            NEW]; %#ok

        % Sort Harmony Memory
        [~, SortOrder]=sort([HM.Cost]);
        HM=HM(SortOrder);

        % Truncate Extra Harmonies
        HM=HM(1:HMS);

        % Update Best Solution Ever Found
        BestSol=HM(1);

        % Store Best Cost Ever Found
        BestCost(it)=BestSol.Cost;

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

        % Damp Fret Width
        FW=FW*FW_damp;

end

%% Results

figure;
%plot(BestCost,'LineWidth',2);
semilogy(BestCost,'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
```