Optimisation Techniques and Algorithms Lab Assignment 7

Aim:

1) Implement Particle Swarm Optimization with the following parameters:
    a) Population size: 50
    b) Maximum number of iterations: 500
    c) Cognitive coefficient (c1) (alpha): 2
    d) Social coefficient (c2)(beta): 2

Optimize any 5 test functions.

2) Compare the results of GA and PSO for any 5 test functions on the following parameters for 500 iterations:
    a) Mean function values for 20 runs
    b) Best function values for 20 runs
    c) Standard deviation values for 20 runs

Compare the iteration-wise mean function values plots for both the values.

<u>Code:</u>

Code 1:

```
% Define parameters
nParticles = 50;              % Number of particles
nDimensions = 2;                % Number of dimensions (change as needed)
maxIterations = 500;          % Maximum number of iterations
cognitiveCoeff = 2;           % Cognitive coefficient (c1)
socialCoeff = 2;              % Social coefficient (c2)
inertiaWeight = 0.5;          % Inertia weight (w)
minBound = -10;                % Lower bound of search space
maxBound = 10;                 % Upper bound of search space

% Define the objective function
str = 'x1^2 + x2^2';  % Objective function as a string
syms x [1 nDimensions];
fitnessFunction = str2sym(str);
```

```
f = matlabFunction(fitnessFunction, 'vars', {x});

% Initialize particles' positions and velocities
positions = minBound + (maxBound - minBound) * rand(nParticles, nDimensions);
velocities = rand(nParticles, nDimensions);

% Evaluate initial fitness values
fitness = arrayfun(@(i) f(positions(i,:)), 1:nParticles, 'UniformOutput', false);
fitness = cell2mat(fitness);

% Initialize personal best positions and fitness
pBestPositions = positions;
pBestFitness = fitness;

% Find the initial global best
[globalBestFitness, globalBestIndex] = min(fitness);
gBestPosition = positions(globalBestIndex, :);

% Initialize array to store the best fitness values over iterations
bestFitnessOverTime = zeros(maxIterations, 1);

% PSO main loop
for iteration = 1:maxIterations
  % Update velocities
  r1 = rand(nParticles, nDimensions); % Random values for cognitive component
  r2 = rand(nParticles, nDimensions); % Random values for social component

  velocities = inertiaWeight * velocities + ...
    cognitiveCoeff * r1 .* (pBestPositions - positions) + ...
    socialCoeff * r2 .* (gBestPosition - positions);
```

```
    % Update positions
    positions = positions + velocities;

    % Apply boundary conditions
    positions = max(min(positions, maxBound), minBound);

    % Evaluate new fitness values
    fitness = arrayfun(@(i) f(positions(i,:)), 1:nParticles, 'UniformOutput', false);
    fitness = cell2mat(fitness);

    % Update personal bests
    improved = fitness < pBestFitness;
    pBestPositions(improved, :) = positions(improved, :);
    pBestFitness(improved) = fitness(improved);

    % Update global best
    [currentGlobalBestFitness, globalBestIndex] = min(fitness);
    if currentGlobalBestFitness < globalBestFitness
        globalBestFitness = currentGlobalBestFitness;
        gBestPosition = positions(globalBestIndex, :);
    end

    % Store the best fitness value for the current iteration
    bestFitnessOverTime(iteration) = globalBestFitness;

    % Display iteration information
    fprintf('Iteration %d: Best Fitness = %.4f\n', iteration, globalBestFitness);
end
```

```
% Output final results
fprintf('Final Global Best Fitness: %.4f\n', globalBestFitness);
fprintf('Final Global Best Position: [%s]\n', num2str(gBestPosition, '%.4f '));

% Plot convergence graph
figure;
plot(1:maxIterations, bestFitnessOverTime, 'LineWidth', 2);
xlabel('Iteration');
ylabel('Best Fitness Value');
title('Convergence Graph of PSO');
grid on;
```

Code 2:

```
clc;
clear;

% Parameters
MaxIter = 100; % Total iterations
str = 'x1^2 + x2^2 + x3^2'; % Fitness function
D = 3; % Number of dimensions
Npop = 50; % Population size
CR = 0.6; % Crossover rate
MR = 0.05; % Mutation rate
n = 20; % Number of runs
LB = [-10, -10, -10]; % Lower bounds for each dimension
UB = [10, 10, 10]; % Upper bounds for each dimension

% Create symbolic variables and the fitness function
syms x [1 D];
fitnessFunction = str2sym(str);
```

```
f = matlabFunction(fitnessFunction, 'vars', {x});

% Initialize storage for fitness values
GA_bestFitness = zeros(MaxIter, n);
GA_meanFitness = zeros(MaxIter, 1);
GA_stddevFitness = zeros(MaxIter, 1);

PSO_bestFitness = zeros(MaxIter, n);
PSO_meanFitness = zeros(MaxIter, 1);
PSO_stddevFitness = zeros(MaxIter, 1);

% GA Algorithm
for run = 1:n
    % Initialize population
    X = LB + rand(Npop, D) .* (UB - LB);

    % Fitness evaluation
    fitness = zeros(Npop, 1);
    for i = 1:Npop
        fitness(i) = f(X(i, :));
    end

    for it = 1:MaxIter
        X_new = X; % Copy population for new individuals

        for i = 1:Npop
            % Selection of r1
            r1 = randi(Npop);
            while r1 == i
                r1 = randi(Npop);
```

```
        end

        % Crossover
        if rand < CR
            crossoverPoint = floor(D / 2); % Single-point crossover at midpoint
            X_new(i, 1:crossoverPoint) = X(i, 1:crossoverPoint);
            X_new(i, crossoverPoint+1:end) = X(r1, crossoverPoint+1:end);
        end

        % Mutation
        if rand < MR
            mutationPoint = randi(D);
            X_new(i, mutationPoint) = LB(mutationPoint) + rand * (UB(mutationPoint) -
LB(mutationPoint));
        end

        % Fitness evaluation
        fitness_new = f(X_new(i, :));
        if fitness_new < fitness(i)
            X(i, :) = X_new(i, :);
            fitness(i) = fitness_new;
        end
    end

    % Sort population by fitness
    [sortedFitness, ~] = sort(fitness, 'ascend');

    % Track best fitness for this iteration
    GA_bestFitness(it, run) = sortedFitness(1);
  end
end
```

```
% Compute GA statistics
GA_meanFitness = mean(GA_bestFitness, 2);
GA_stddevFitness = std(GA_bestFitness, 0, 2);

% PSO Algorithm
for run = 1:n
    [bestFitnessOverTime] = particleSwarmOptimization(f, Npop, MaxIter, D, LB, UB);
    PSO_bestFitness(:, run) = bestFitnessOverTime;
end

% Compute PSO statistics
PSO_meanFitness = mean(PSO_bestFitness, 2);
PSO_stddevFitness = std(PSO_bestFitness, 0, 2);

% Plot the results
figure;
plot(1:MaxIter, GA_meanFitness, 'r', 'LineWidth', 2);
hold on;
plot(1:MaxIter, PSO_meanFitness, 'b', 'LineWidth', 2);
xlabel('Iteration');
ylabel('Mean Fitness Value');
title('Mean Fitness Value Over Iterations');
legend('GA', 'PSO');
grid on;

figure;
plot(1:MaxIter, min(GA_bestFitness, [], 2), 'r', 'LineWidth', 2);
hold on;
plot(1:MaxIter, min(PSO_bestFitness, [], 2), 'b', 'LineWidth', 2);
```

```
xlabel('Iteration');
ylabel('Best Fitness Value');
title('Best Fitness Value Over Iterations');
legend('GA', 'PSO');
grid on;

figure;
plot(1:MaxIter, GA_stddevFitness, 'r', 'LineWidth', 2);
hold on;
plot(1:MaxIter, PSO_stddevFitness, 'b', 'LineWidth', 2);
xlabel('Iteration');
ylabel('Standard Deviation of Fitness Value');
title('Standard Deviation of Fitness Value Over Iterations');
legend('GA', 'PSO');
grid on;

% PSO function
function [bestFitnessOverTime] = particleSwarmOptimization(fitnessFunction, nParticles,
maxIterations, nDimensions, LB, UB)
  % PSO Parameters
  cognitiveCoeff = 2; % Cognitive coefficient
  socialCoeff = 2;    % Social coefficient
  inertiaWeight = 0.5; % Inertia weight

  % Initialize positions and velocities
  positions = LB + (UB - LB) .* rand(nParticles, nDimensions);
  velocities = rand(nParticles, nDimensions);

  % Evaluate initial fitness values
  fitness = arrayfun(@(i) fitnessFunction(positions(i,:)), 1:nParticles, 'UniformOutput',
false);
```

```
fitness = cell2mat(fitness);

% Initialize personal bests
pBestPositions = positions;
pBestFitness = fitness;

% Find the initial global best
[globalBestFitness, globalBestIndex] = min(fitness);
gBestPosition = positions(globalBestIndex, :);

% Initialize array to store best fitness values over iterations
bestFitnessOverTime = zeros(maxIterations, 1);

% PSO main loop
for iteration = 1:maxIterations
    % Update velocities
    r1 = rand(nParticles, nDimensions); % Random values for cognitive component
    r2 = rand(nParticles, nDimensions); % Random values for social component

    velocities = inertiaWeight * velocities + ...
        cognitiveCoeff * r1 .* (pBestPositions - positions) + ...
        socialCoeff * r2 .* (gBestPosition - positions);

    % Update positions
    positions = positions + velocities;

    % Apply boundary conditions
    positions = max(min(positions, UB), LB);

    % Evaluate new fitness values
```

```
        fitness = arrayfun(@(i) fitnessFunction(positions(i,:)), 1:nParticles, 'UniformOutput',
false);
        fitness = cell2mat(fitness);

        % Update personal bests
        improved = fitness < pBestFitness;
        pBestPositions(improved, :) = positions(improved, :);
        pBestFitness(improved) = fitness(improved);

        % Update global best
        [currentGlobalBestFitness, globalBestIndex] = min(fitness);
        if currentGlobalBestFitness < globalBestFitness
            globalBestFitness = currentGlobalBestFitness;
            gBestPosition = positions(globalBestIndex, :);
        end

        % Store the best fitness value for the current iteration
        bestFitnessOverTime(iteration) = globalBestFitness;
    end
end
```

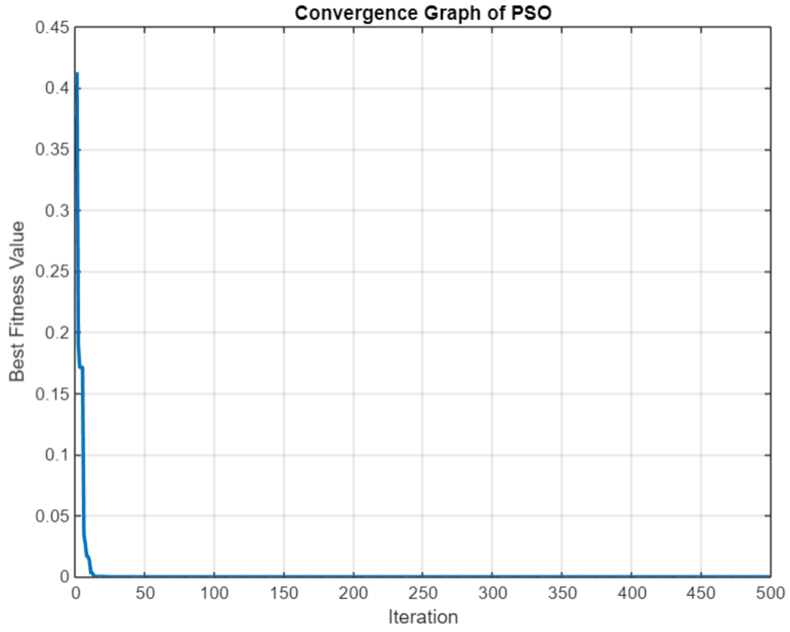
Output:

Function 1: Sphere function

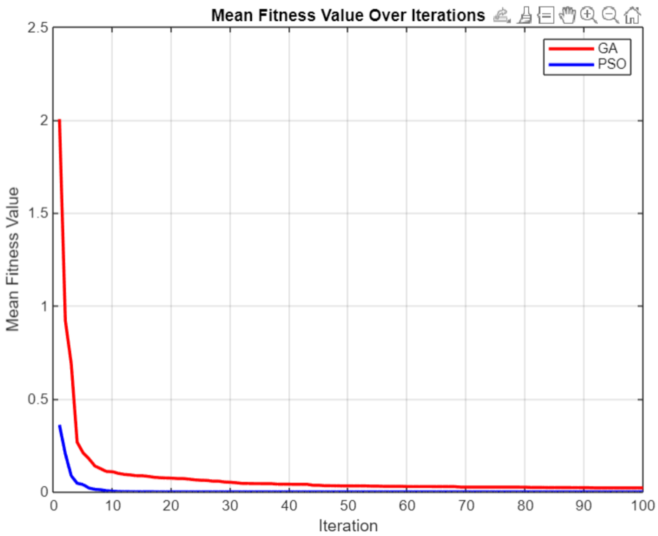
Dimensions=2

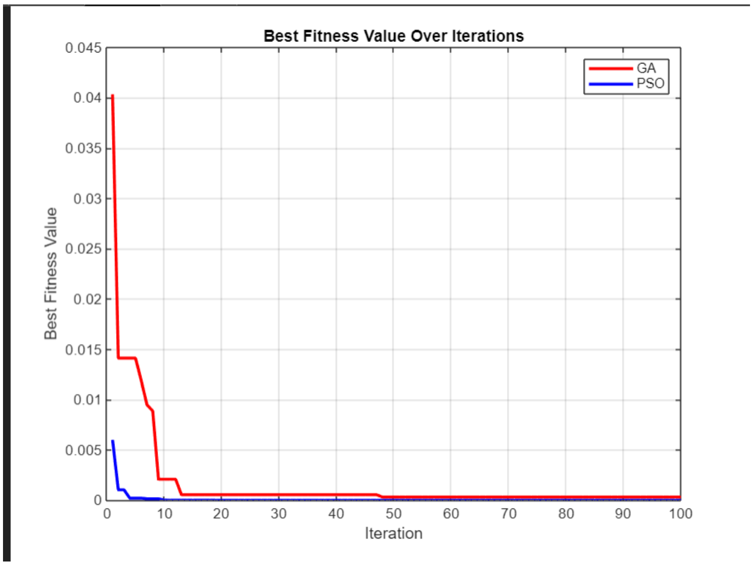
F='x1^2+x2^2'

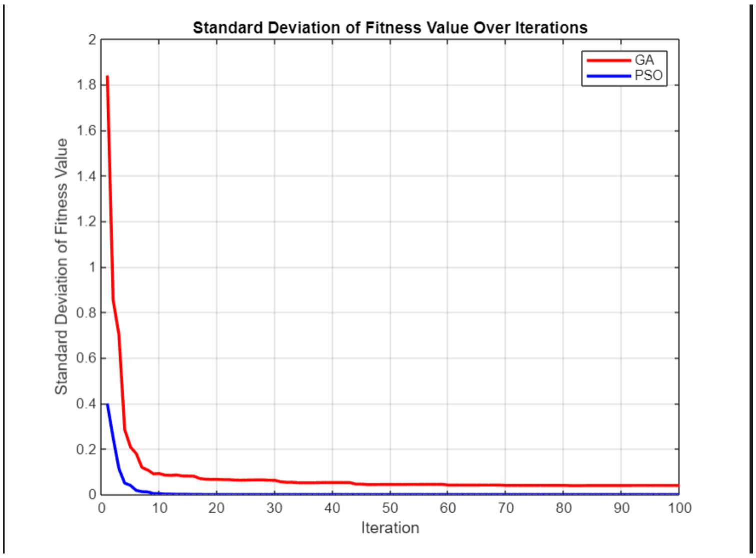
Code 1:



Code 2:





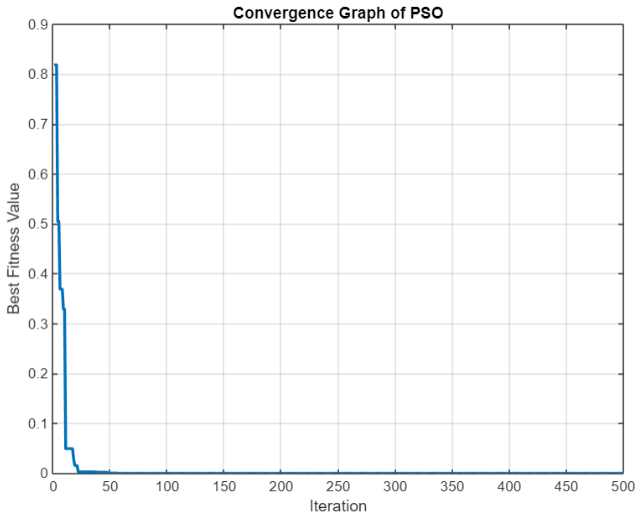


Function 2: Rosenbrock

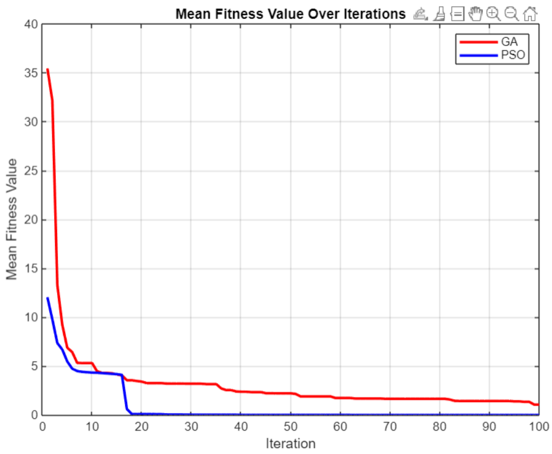
Dimensions=2

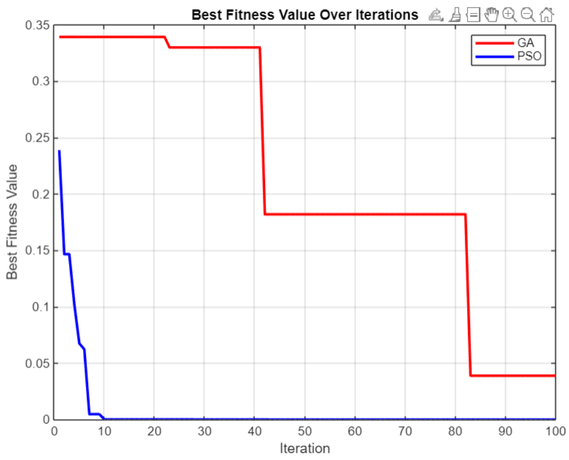
F='100*(x2^2 - x1^2)^2 + (1 - x1)^2'

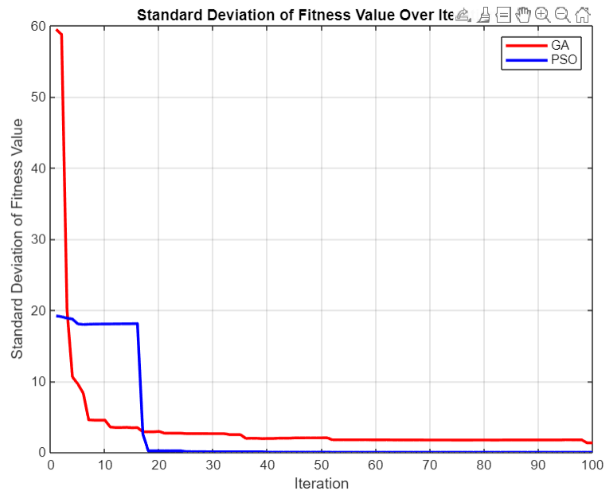
Code 1:


Convergence Graph of PSO
Best Fitness Value
Iteration

Code 2:


Mean Fitness Value Over Iterations
GA
PSO
Mean Fitness Value
Iteration


Best Fitness Value Over Iterations
GA
PSO
Best Fitness Value
Iteration

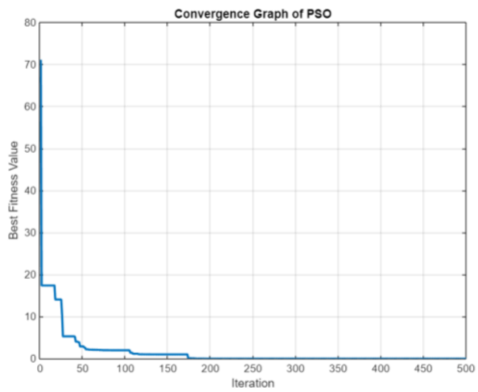


Function 3: Rastrigin Function

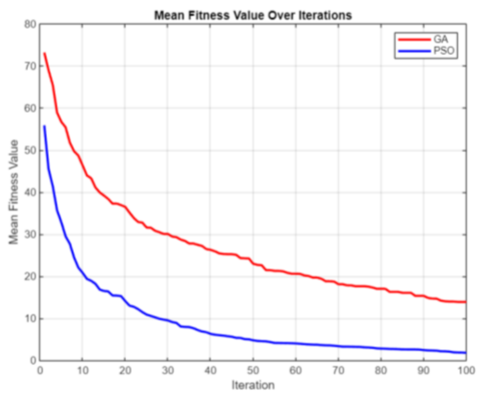
Dimensions:5

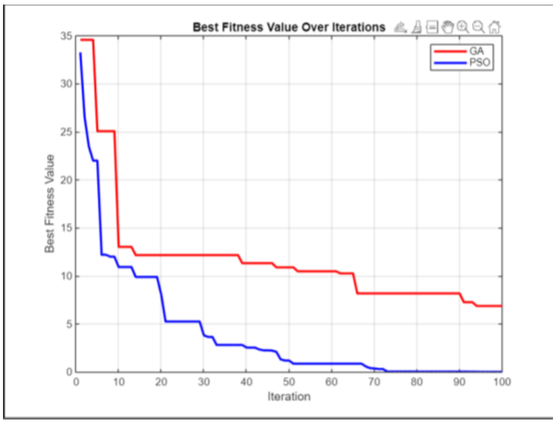
F='10*5 + (x1^2 - 10*cos(2*pi*x1)) + (x2^2 - 10*cos(2*pi*x2)) + (x3^2 - 10*cos(2*pi*x3)) + (x4^2 - 10*cos(2*pi*x4)) + (x5^2 - 10*cos(2*pi*x5))'

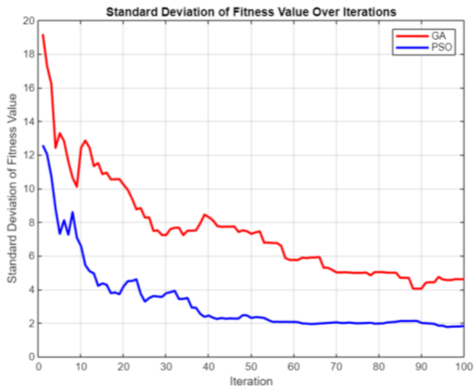
Code 1:



Code 2:





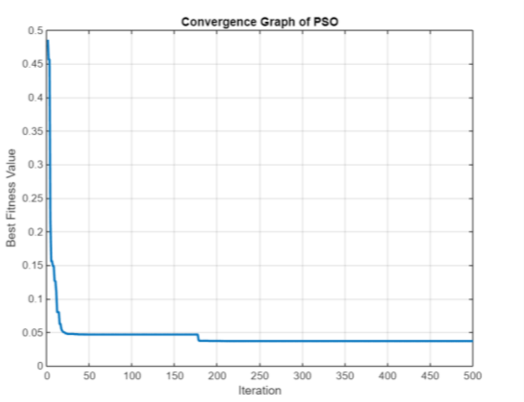


Function 4: Griewank Function:

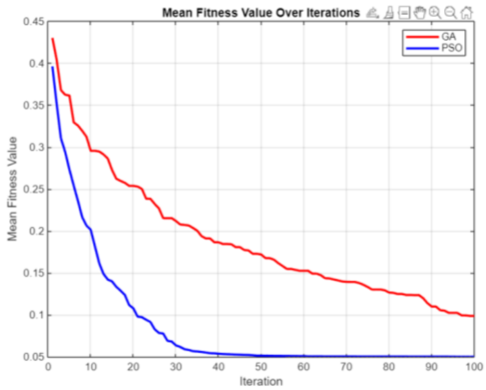
Dimensions: 5

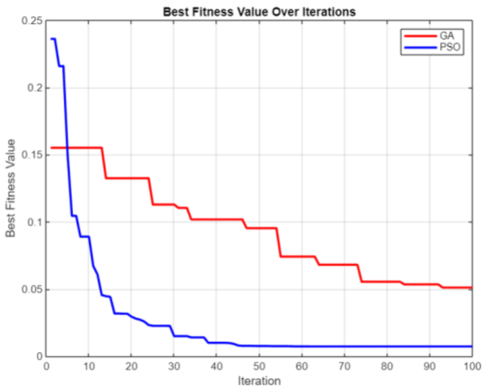
F='1 + (x1^2 + x2^2 + x3^2 + x4^2 + x5^2)/4000 - cos(x1)*cos(x2/sqrt(2))*cos(x3/sqrt(3))*cos(x4/sqrt(4))*cos(x5/sqrt(5))'

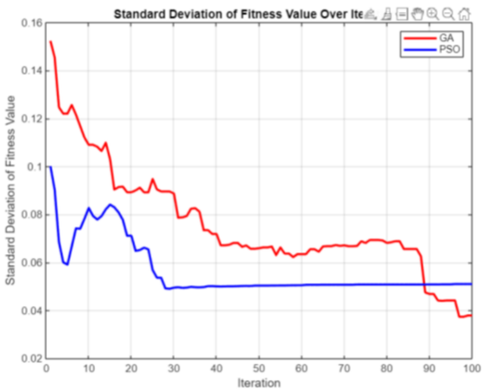
Code 1:



Code 2:





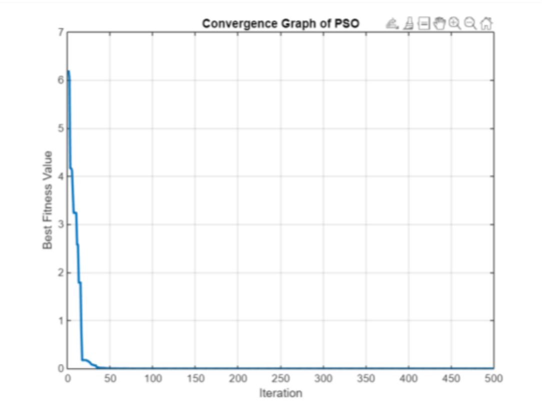


Function 5: Ackley Function:

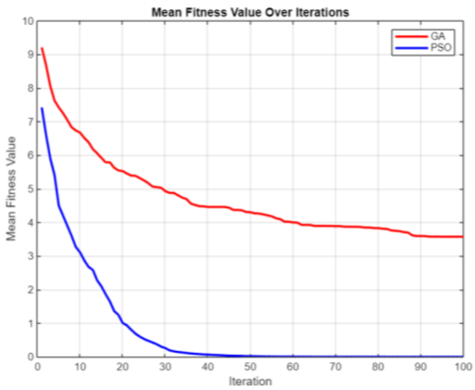
Dimensions: 5

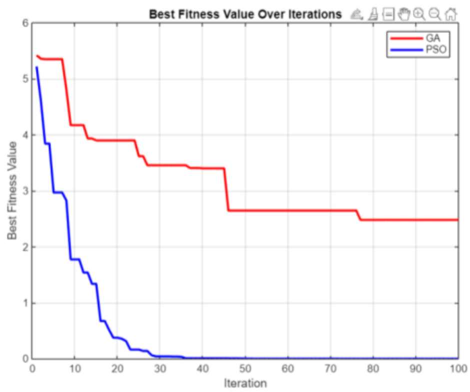
F='-20*exp(-0.2*sqrt((x1^2 + x2^2 + x3^2 + x4^2 + x5^2)/5)) - exp((cos(2*pi*x1) + cos(2*pi*x2) + cos(2*pi*x3) + cos(2*pi*x4) + cos(2*pi*x5))/5) + 20 + exp(1)'

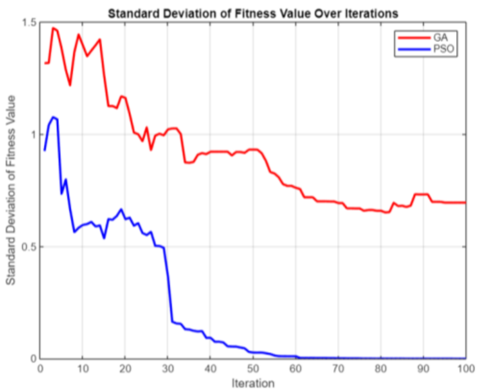
Code 1:


Convergence Graph of PSO
Best Fitness Value
Iteration

Code 2:


Mean Fitness Value Over Iterations
GA
PSO
Mean Fitness Value
Iteration


Best Fitness Value Over Iterations
GA
PSO
Best Fitness Value
Iteration


Standard Deviation of Fitness Value Over Iterations
GA
PSO
Standard Deviation of Fitness Value
Iteration