Experiment-3.3

Aim of the Experiment:

Implement a complex optimization problem such as banana function etc using particle swarm optimization (PSO).

Theory:

Particle Swarm Optimization (PSO) is a population-based stochastic optimization technique inspired by the social behavior of bird flocking or fish schooling. It was introduced by Kennedy and Eberhart in 1995. PSO aims to find the optimal solution to an optimization problem by iteratively updating a group of candidate solutions, called particles, in the search space.

Working of Particle Swarm Optimization (PSO):

1) Initialization: PSO begins by initializing a population of particles in the search space. Each particle represents a potential solution to the optimization problem. The position of a particle corresponds to a point in the search space, and its velocity determines the direction and magnitude of its movement in the search space.

2) Evaluation: Each particle's fitness or objective function value is evaluated based on its position in the search space. This function quantifies how good a particular solution is with respect to the optimization problem being solved.

3) Updating Personal Best (pbest): Each particle maintains its own best-known position in the search space, called the personal best (pbest). If the current position of a particle yields a better fitness value than its previous pbest, the particle updates its pbest to its current position.

4) Updating Global Best (gbest): Among all the personal best positions of the particles, the one with the best fitness value is identified as the global best (gbest). This gbest position represents the best solution found by any particle in the population.

5) Updating Velocity and Position: The velocity and position of each particle are updated based on its current velocity, position, personal best, and global best. The update equations for velocity and position are typically defined as follows:

Velocity update:

$$v_{id}(t+1) = w.v_{id}(t) + c_1.r_1.(pbest_{id} - x_{id}(t)) + c_2.r_2.(gbest_{id} - x_{id}(t))$$

Position update:

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$$

Where:

$v_{id}(t)$ is the velocity of the i-th particle along the d-th dimension at time t.

$x_{id}(t)$ is the position of the i-th particle along the d-th dimension at time t.

$pbest_{id}$ is the personal best position of the i-th particle along the d-th dimension.

$gbest_{id}$ is the global best position along the d-th dimension.

w is the inertia weight that controls the impact of the previous velocity.

c_1 and c_2 are acceleration coefficients.

r_1 and r_2 are random numbers sampled from a uniform distribution in the range [0, 1].

6) Termination: PSO iteratively updates the velocities and positions of particles until a termination criterion is met. This criterion could be a maximum number of iterations, reaching a satisfactory solution, or stagnation in the search process.

7) Output: The output of PSO is the global best solution found, i.e., the position in the search space corresponding to the global best fitness value.

PSO has several parameters that need to be set before running the algorithm, such as the number of particles, inertia weight, acceleration coefficients, and termination criteria. Tuning these parameters can significantly affect the performance and convergence behavior of the algorithm.

Rosenbrock Function:

Rosenbrock function (also known as the banana function) is a commonly used test function for optimization algorithms. It is defined as follows:

$$f(x,y) = (a - x)^2 + b(y - x^2)^2$$

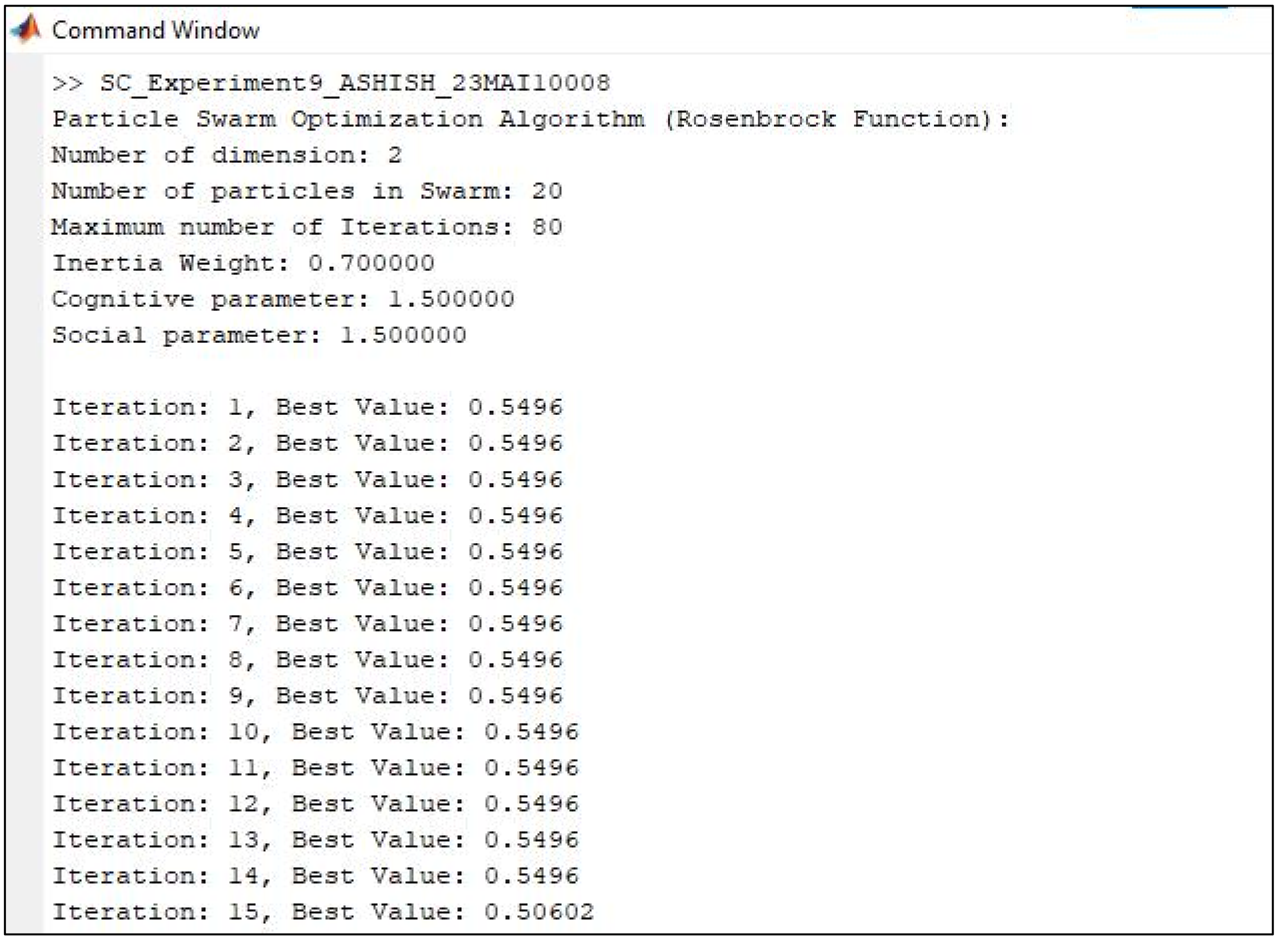
where $a = 1$ and $b = 100$.

Code for Experiment :

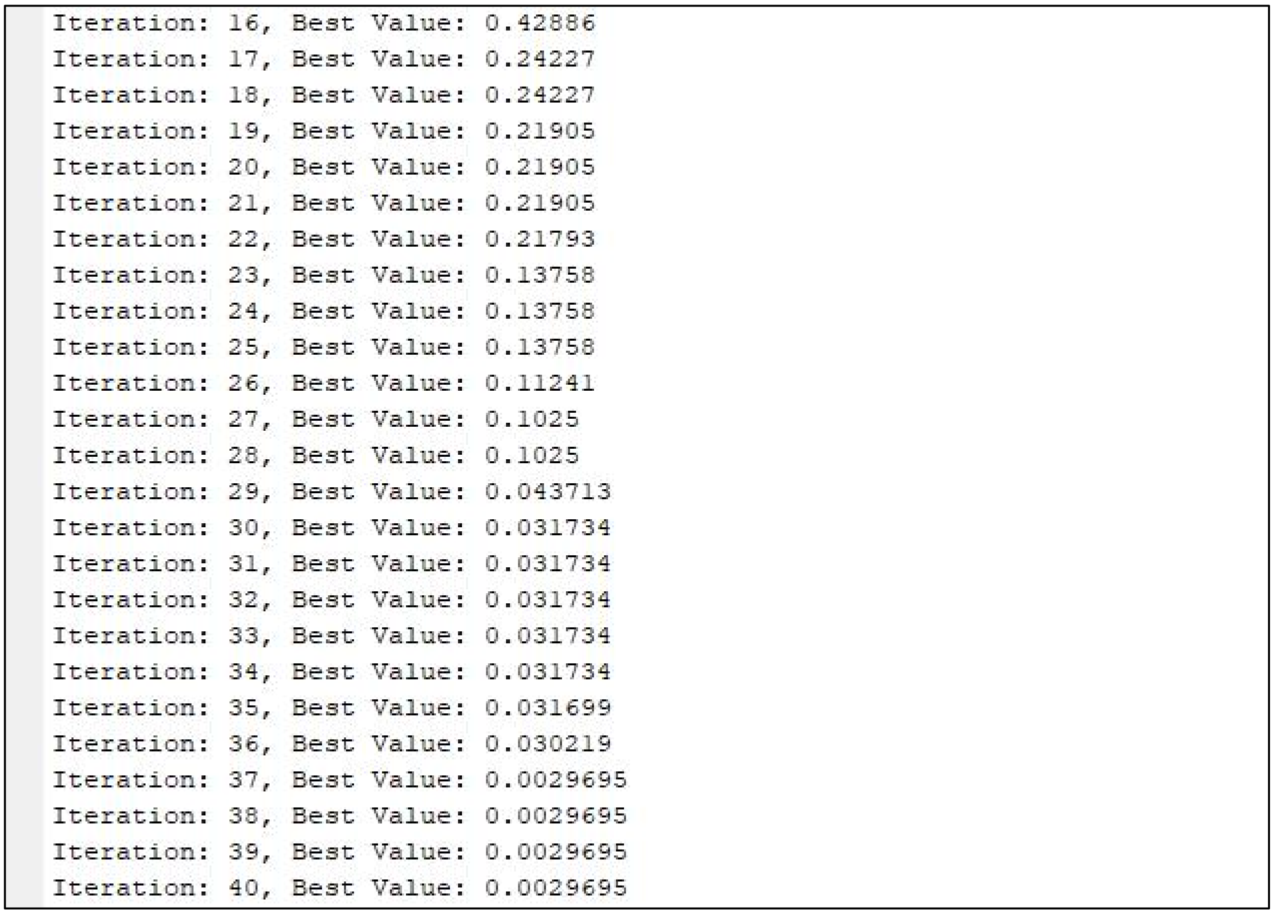
```
fprintf("Particle Swarm Optimization Algorithm (Rosenbrock Function): \n")

% Define the Objective function (Rosenbrock Function)
rosenbrock = @(x) (1 - x(1))^2 + 100*(x(2) - x(1)^2)^2;

% PSO parameters
dimension = 2; % Number of dimensions
numParticles = 20; % Number of particles in the swarm
maxIterations = 80; % Maximum number of iterations
w = 0.7; % inertia weight
c1 = 1.5; % cognitive parameter
c2 = 1.5; % social parameter

fprintf("Number of dimension: %d\n",dimension);
fprintf("Number of particles in Swarm: %d\n",numParticles);
fprintf("Maximum number of Iterations: %d\n",maxIterations);
fprintf("Inertia Weight: %f\n",w);
fprintf("Cognitive parameter: %f\n",c1);
fprintf("Social parameter: %f\n\n",c2);


% Initialize particles
% Random initialization between -5 and 5 for particle position
particles.position = rand(numParticles, dimension) * 10 - 5;
particles.velocity = zeros(numParticles, dimension);

particles.bestPosition = particles.position;
particles.bestValue = inf(1, numParticles);
globalBestPosition = zeros(1, dimension);
globalBestValue = inf;


% PSO optimization loop
for iter = 1:maxIterations
    % Update particle positions and velocities
    r1 = rand(numParticles, dimension);
    r2 = rand(numParticles, dimension);

    particles.velocity = w * particles.velocity + c1 * r1 .* (particles.bestPosition -
                         particles.position) + c2 * r2 .* (repmat(globalBestPosition,
                          numParticles, 1) - particles.position);
    particles.position = particles.position + particles.velocity;

    % Update personal best positions
    currentValues = zeros(1, numParticles);
    for i = 1:numParticles
          currentValues(i) = rosenbrock(particles.position(i,:));
    end
```

```
        improvedParticles = currentValues < particles.bestValue;
        particles.bestPosition(improvedParticles, :) = particles.position(improvedParticles, :);
        particles.bestValue(improvedParticles) = currentValues(improvedParticles);

        % Update global best position
        [minValue, minIndex] = min(particles.bestValue);
        if minValue < globalBestValue
              globalBestValue = minValue;
              globalBestPosition = particles.bestPosition(minIndex, :);
        end

        % Display current iteration and best value found
        disp(['Iteration: ', num2str(iter), ', Best Value: ', num2str(globalBestValue)]);
    end

    fprintf('Optimization complete.\n\n');
    disp(['Global Minimum found at: ', num2str(globalBestPosition)]);
    disp(['Minimum Value: ', num2str(globalBestValue)]);
```

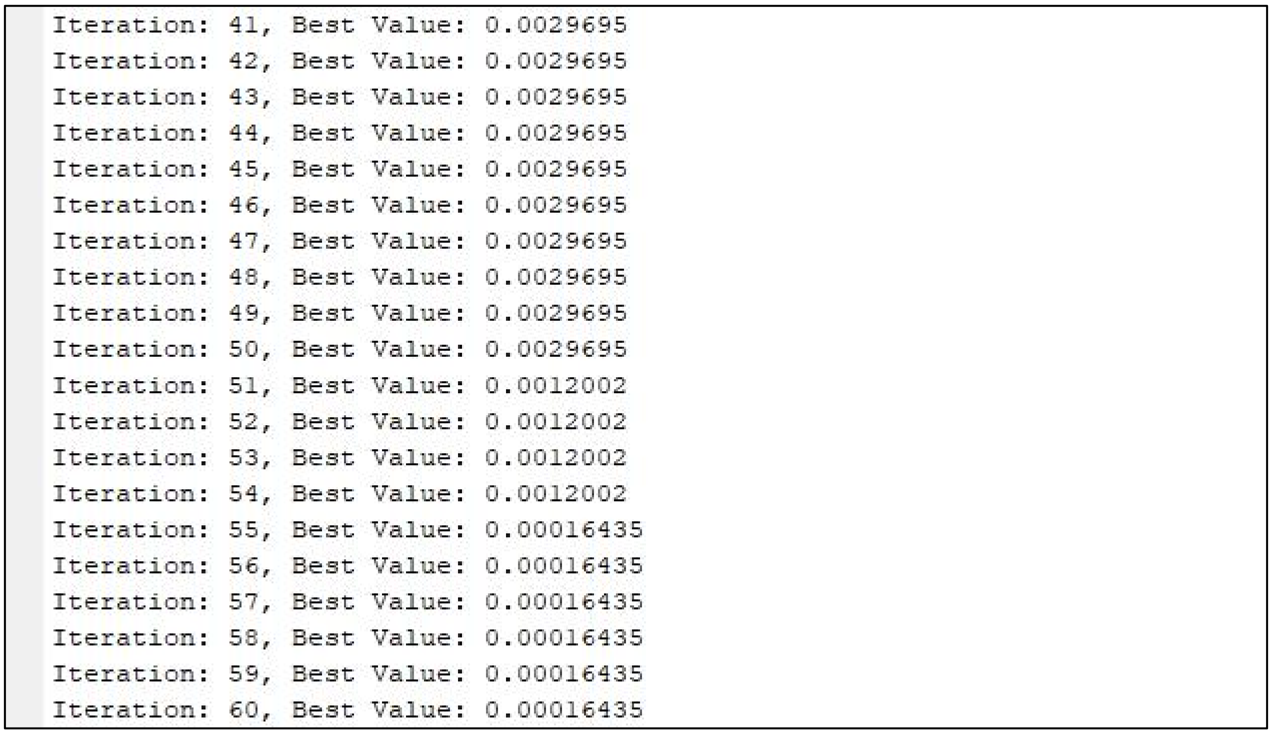
Result/Output :



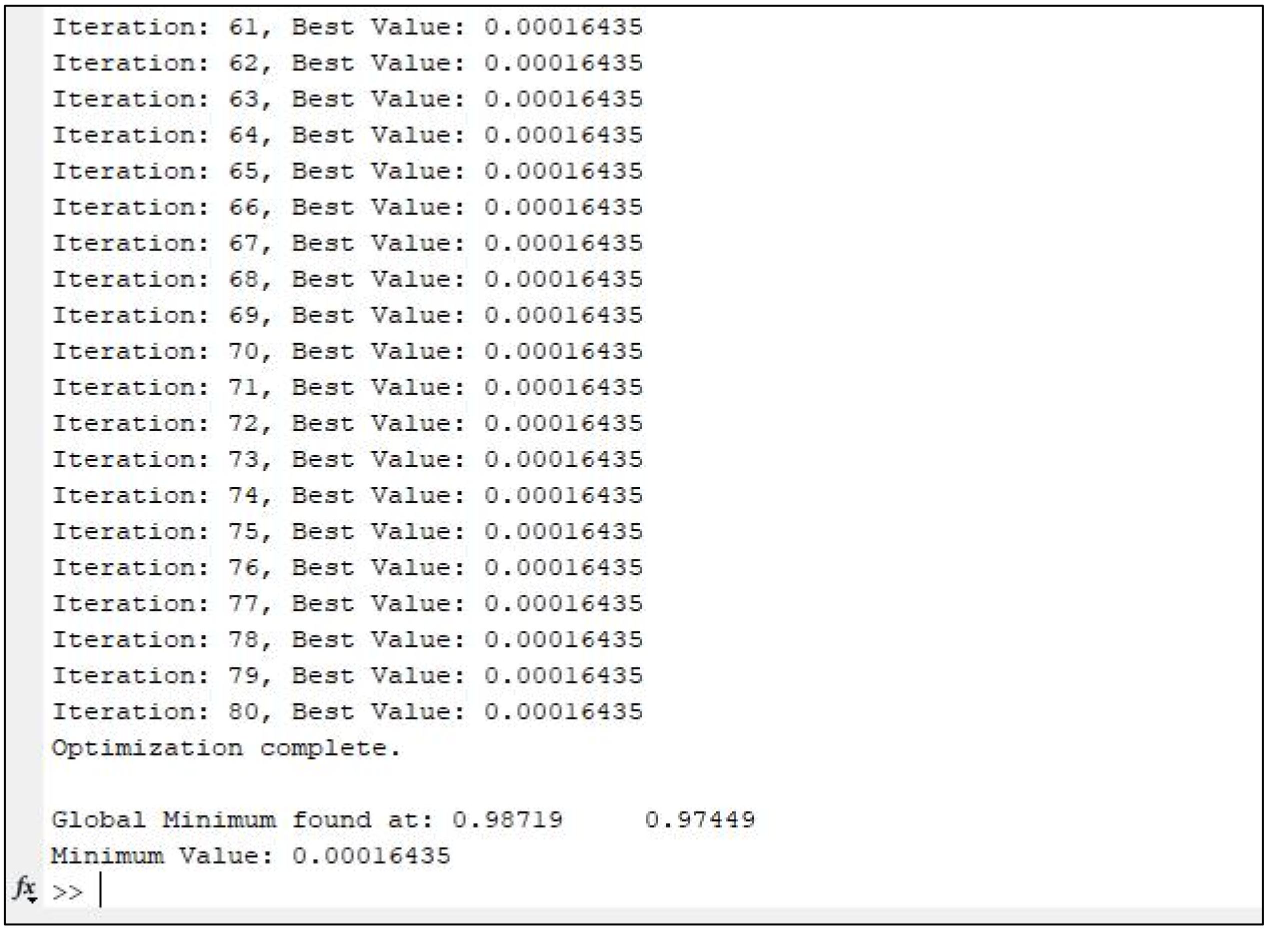
```
>> SC_Experiment9_ASHISH_23MAI10008
Particle Swarm Optimization Algorithm (Rosenbrock Function):
Number of dimension: 2
Number of particles in Swarm: 20
Maximum number of Iterations: 80
Inertia Weight: 0.700000
Cognitive parameter: 1.500000
Social parameter: 1.500000

Iteration: 1, Best Value: 0.5496
Iteration: 2, Best Value: 0.5496
Iteration: 3, Best Value: 0.5496
Iteration: 4, Best Value: 0.5496
Iteration: 5, Best Value: 0.5496
Iteration: 6, Best Value: 0.5496
Iteration: 7, Best Value: 0.5496
Iteration: 8, Best Value: 0.5496
Iteration: 9, Best Value: 0.5496
Iteration: 10, Best Value: 0.5496
Iteration: 11, Best Value: 0.5496
Iteration: 12, Best Value: 0.5496
Iteration: 13, Best Value: 0.5496
Iteration: 14, Best Value: 0.5496
Iteration: 15, Best Value: 0.50602
```



```
Iteration: 16, Best Value: 0.42886
Iteration: 17, Best Value: 0.24227
Iteration: 18, Best Value: 0.24227
Iteration: 19, Best Value: 0.21905
Iteration: 20, Best Value: 0.21905
Iteration: 21, Best Value: 0.21905
Iteration: 22, Best Value: 0.21793
Iteration: 23, Best Value: 0.13758
Iteration: 24, Best Value: 0.13758
Iteration: 25, Best Value: 0.13758
Iteration: 26, Best Value: 0.11241
Iteration: 27, Best Value: 0.1025
Iteration: 28, Best Value: 0.1025
Iteration: 29, Best Value: 0.043713
Iteration: 30, Best Value: 0.031734
Iteration: 31, Best Value: 0.031734
Iteration: 32, Best Value: 0.031734
Iteration: 33, Best Value: 0.031734
Iteration: 34, Best Value: 0.031734
Iteration: 35, Best Value: 0.031699
Iteration: 36, Best Value: 0.030219
Iteration: 37, Best Value: 0.0029695
Iteration: 38, Best Value: 0.0029695
Iteration: 39, Best Value: 0.0029695
Iteration: 40, Best Value: 0.0029695
```



```
Iteration: 41, Best Value: 0.0029695
Iteration: 42, Best Value: 0.0029695
Iteration: 43, Best Value: 0.0029695
Iteration: 44, Best Value: 0.0029695
Iteration: 45, Best Value: 0.0029695
Iteration: 46, Best Value: 0.0029695
Iteration: 47, Best Value: 0.0029695
Iteration: 48, Best Value: 0.0029695
Iteration: 49, Best Value: 0.0029695
Iteration: 50, Best Value: 0.0029695
Iteration: 51, Best Value: 0.0012002
Iteration: 52, Best Value: 0.0012002
Iteration: 53, Best Value: 0.0012002
Iteration: 54, Best Value: 0.0012002
Iteration: 55, Best Value: 0.00016435
Iteration: 56, Best Value: 0.00016435
Iteration: 57, Best Value: 0.00016435
Iteration: 58, Best Value: 0.00016435
Iteration: 59, Best Value: 0.00016435
Iteration: 60, Best Value: 0.00016435
```



```
Iteration: 61, Best Value: 0.00016435
Iteration: 62, Best Value: 0.00016435
Iteration: 63, Best Value: 0.00016435
Iteration: 64, Best Value: 0.00016435
Iteration: 65, Best Value: 0.00016435
Iteration: 66, Best Value: 0.00016435
Iteration: 67, Best Value: 0.00016435
Iteration: 68, Best Value: 0.00016435
Iteration: 69, Best Value: 0.00016435
Iteration: 70, Best Value: 0.00016435
Iteration: 71, Best Value: 0.00016435
Iteration: 72, Best Value: 0.00016435
Iteration: 73, Best Value: 0.00016435
Iteration: 74, Best Value: 0.00016435
Iteration: 75, Best Value: 0.00016435
Iteration: 76, Best Value: 0.00016435
Iteration: 77, Best Value: 0.00016435
Iteration: 78, Best Value: 0.00016435
Iteration: 79, Best Value: 0.00016435
Iteration: 80, Best Value: 0.00016435
Optimization complete.

Global Minimum found at: 0.98719     0.97449
Minimum Value: 0.00016435
fx >>
```

Learning outcomes :

1. Learnt about the concept of Particle Swarm Optimization(PSO).
2. Learnt about the Rosenbrock function used in PSO algorithm.
3. Learnt about the applications of Particle Swarm Optimization algorithm.
4. Learnt about cognitive and social parameter in PSO algorithm.
5. Learnt about how to apply PSO on complex functions.