USN : 1BM22CS235

LAB-2 : Partcle Swarm Optmizaton for Functon Optmizaton:

CODE:

```
#lab-3: pso import numpy as np import random




# Define the optmizaton problem (Rastrigin Functon)
def rastrigin(x):
    A = 10
    return A * len(x) + sum([(xi**2 - A * np.cos(2 * np.pi * xi)) for xi in x])


# Partcle Swarm Optmizaton (PSO) implementaton
class Partcle:
    def __init__(self, dimension, lower_bound, upper_bound):
        # Initalize the partcle positon and velocity randomly
        self.positon = np.random.uniform(lower_bound, upper_bound, dimension)
        self.velocity = np.random.uniform(-1, 1, dimension)
        self.best_positon = np.copy(self.positon)
        self.best_value = rastrigin(self.positon)


    def update_velocity(self, global_best_positon, w, c1,
    c2):
        # Update the velocity of the partcle
        r1 = np.random.rand(len(self.positon))
        r2 = np.random.rand(len(self.positon))


        # Inerta term
        inerta = w * self.velocity


        # Cognitve term (individual best)
```

```
        cognitve = c1 * r1 * (self.best_positon - self.positon)

        # Social term (global best)
        social = c2 * r2 * (global_best_positon - self.positon)

        # Update velocity
        self.velocity = inerta + cognitve + social

    def update_positon(self, lower_bound, upper_bound):
        # Update the positon of the partcle
        self.positon = self.positon + self.velocity

        # Ensure the partcle stays within the bounds
        self.positon = np.clip(self.positon, lower_bound, upper_bound)

    def evaluate(self):
        # Evaluate the fitness of the partcle
        fitness = rastrigin(self.positon)

        # Update the partcle's best positon if necessary
        if fitness < self.best_value:
            self.best_value = fitness
            self.best_positon = np.copy(self.positon)

def partcle_swarm_optmizaton(dim, lower_bound, upper_bound, num_partcles=30,
max_iter=100, w=0.5, c1=1.5, c2=1.5):
    # Initalize partcles
    partcles = [Partcle(dim, lower_bound, upper_bound) for _ in range(num_partcles)]

    # Initalize the global best positon and value
    global_best_positon = partcles[0].best_positon
```

```
    global_best_value = partcles[0].best_value

    for i in range(max_iter):
        # Update each partcle
        for partcle in partcles:
            partcle.update_velocity(global_best_positon, w, c1, c2)
            partcle.update_positon(lower_bound, upper_bound)
            partcle.evaluate()

            # Update global best positon if needed
            if partcle.best_value < global_best_value:
                global_best_value = partcle.best_value
                global_best_positon = np.copy(partcle.best_positon)

        # Optonally print the progress
        if (i+1 ) % 10 == 0:
            print(f"Iteraton {i+1 }/{max_iter} - Best Fitness:
            {global_best_value}")
    return global_best_positon, global_best_value

# Set the parameters for the PSO algorithm
dim = 2              # Number of dimensions for the functon
lower_bound = -5.12 # Lower bound of the search space
upper_bound = 5.12    # Upper bound of the search space
num_partcles = 30     # Number of partcles in the swarm
max_iter = 100       # Number of iteratons

# Run the PSO
best_positon, best_value = partcle_swarm_optmizaton(dim, lower_bound, upper_bound,
num_partcles, max_iter)
```

Output the best soluton found

print("\nBest Soluton Found:")

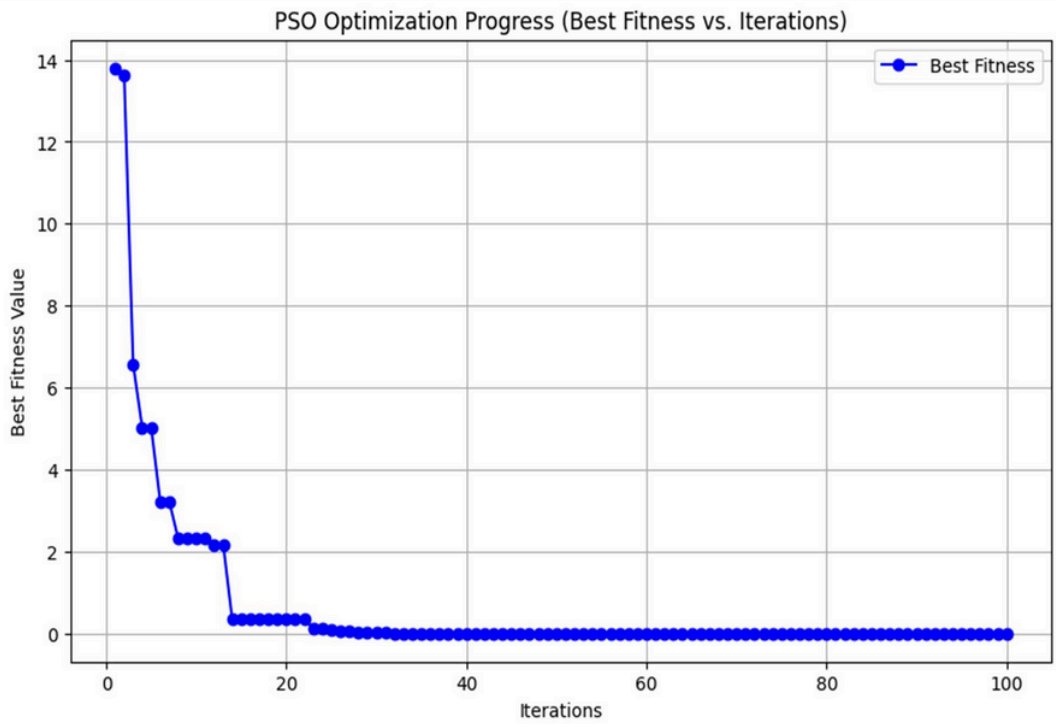
print("Positon:", best_positon)

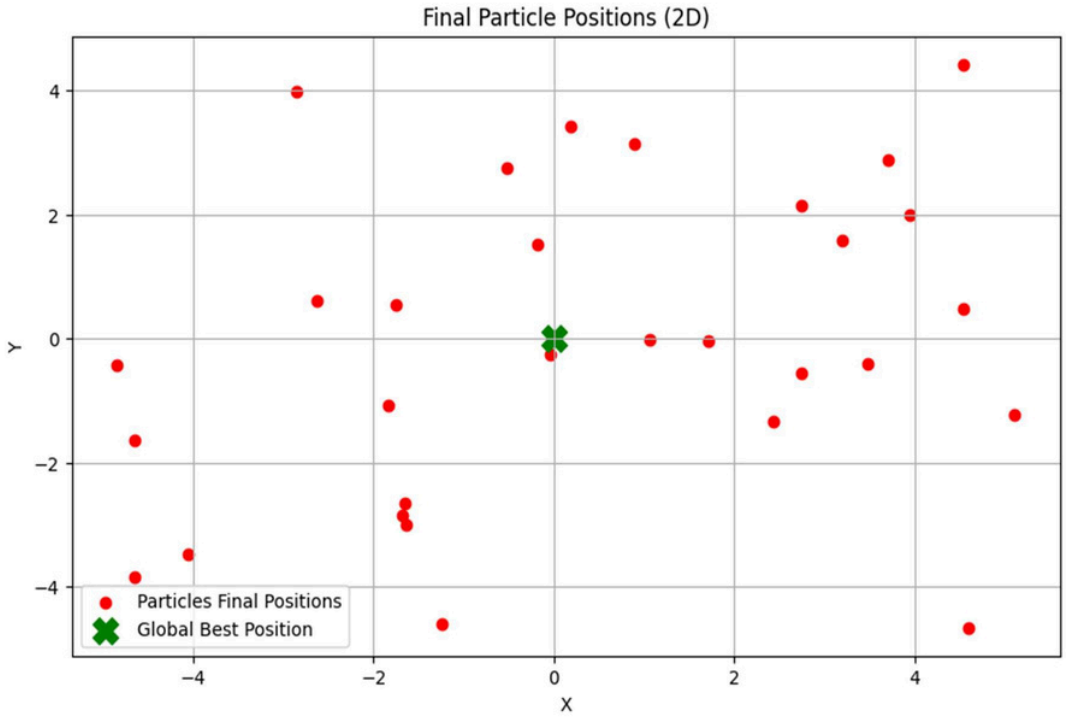
print("Fitness:", best_value)

OUTPUT:

```
Iteration 10/100 - Best Fitness: 2.3145203625443997
Iteration 20/100 - Best Fitness: 0.34026142761705813
Iteration 30/100 - Best Fitness: 0.0158886712260653
Iteration 40/100 - Best Fitness: 5.572809527620848e-06
Iteration 50/100 - Best Fitness: 3.493363465167931e-08
Iteration 60/100 - Best Fitness: 2.8475000135586015e-11
Iteration 70/100 - Best Fitness: 1.4210854715202004e-14
Iteration 80/100 - Best Fitness: 0.0
Iteration 90/100 - Best Fitness: 0.0
Iteration 100/100 - Best Fitness: 0.0

Best Solution Found:
Position: [ 1.64289135e-09 -1.88899730e-09]
Fitness: 0.0
```




Final Particle Positions (2D)
Particles Final Positions
Global Best Position
X
Y
4
2
0
−2
−4
−4
−2
0
2
4