LAB 2
PARTICLE SWAM PROBLEM

1BM22CS263
SHREE VARNA M

CODE:

```
import random

def particle_swarm_optimization(fitness_function, num_particles=30,
max_iterations=100,
                                inertia_weight=0.5,
cognitive_coef=1.5, social_coef=2.0,
                                value_range=(-10, 10)):
    # Initialize particles
    particles = []
    for _ in range(num_particles):
        # Random position and velocity within the specified range
        position = [random.uniform(value_range[0], value_range[1])
for _ in range(2)]  # 2D example
        velocity = [random.uniform(-1, 1) for _ in range(2)]
        best_position = position[:]
        best_fitness = fitness_function(position)
        particles.append({
            "position": position,
            "velocity": velocity,
            "best_position": best_position,
            "best_fitness": best_fitness
        })

    # Initialize global best
    global_best_position = None
    global_best_fitness = float("inf")

    # Main optimization loop
    for _ in range(max_iterations):
        for particle in particles:
            # Calculate fitness
            fitness = fitness_function(particle["position"])

            # Update personal best
            if fitness < particle["best_fitness"]:
                particle["best_position"] = particle["position"][:]
                particle["best_fitness"] = fitness

            # Update global best
```

```
            if fitness < global_best_fitness:
                global_best_position = particle["position"][:]
                global_best_fitness = fitness

        # Update velocities and positions
        for particle in particles:
            new_velocity = []
            for i in range(len(particle["position"])):
                inertia = inertia_weight * particle["velocity"][i]
                cognitive = cognitive_coef * random.random() *
(particle["best_position"][i] - particle["position"][i])
                social = social_coef * random.random() *
(global_best_position[i] - particle["position"][i])
                new_velocity.append(inertia + cognitive + social)

            particle["velocity"] = new_velocity
            particle["position"] = [
                max(min(p + v, value_range[1]), value_range[0])  #
Clip to value_range
                for p, v in zip(particle["position"],
particle["velocity"])
            ]

    return global_best_position, global_best_fitness

# Example usage with a function to optimize (Sphere function)
def sphere_function(position):
    return sum(x**2 for x in position)

best_position, best_fitness =
particle_swarm_optimization(sphere_function)
print("Best position:", best_position)
print("Best fitness:", best_fitness)
```

OUTPUT :

```
Best position: [3.082848116055915e-11, 6.52701773519799e-11]
Best fitness: 5.2105913022258594e-21
```