PARTICLE SWARM OPTIMIZATION

```
import numpy as np

def f(x):
    return np.sum(x**2)

def particle_swarm_optimization(objective_func, bounds, num_particles,
max_iterations):
    num_dimensions = len(bounds)

    particles_position = np.random.uniform(low=[b[0] for b in bounds],
                                           high=[b[1] for b in bounds],
                                           size=(num_particles,
num_dimensions))
    particles_velocity = np.random.uniform(low=-1, high=1,
size=(num_particles, num_dimensions))

    personal_best_position = np.copy(particles_position)
    personal_best_value = np.array([objective_func(p) for p in
personal_best_position])

    global_best_position =
personal_best_position[np.argmin(personal_best_value)]
    global_best_value = np.min(personal_best_value)

    print(f"Iteration 0: Best objective function value =
{global_best_value}")

    for iteration in range(max_iterations):
        r1 = np.random.random((num_particles, num_dimensions))
        r2 = np.random.random((num_particles, num_dimensions))

        cognitive_velocity = 2.0 * r1 * (personal_best_position -
particles_position)
        social_velocity = 2.0 * r2 * (global_best_position -
particles_position)
        particles_velocity = particles_velocity + cognitive_velocity +
social_velocity
        particles_position = particles_position + particles_velocity

        for i in range(num_dimensions):
            particles_position[:, i] = np.clip(particles_position[:,
i], bounds[i][0], bounds[i][1])

        current_value = np.array([objective_func(p) for p in
particles_position])
        update_mask = current_value < personal_best_value
```

```
        personal_best_position[update_mask] =
particles_position[update_mask]
        personal_best_value[update_mask] = current_value[update_mask]

        if np.min(current_value) < global_best_value:
            global_best_value = np.min(current_value)
            global_best_position =
particles_position[np.argmin(current_value)]

        print(f"Iteration {iteration + 1}: Best objective function
value = {global_best_value}")


    return global_best_position, global_best_value

bounds = [(-10, 10), (-10, 10)]

best_position, best_value = particle_swarm_optimization(f, bounds,
num_particles=30, max_iterations=10)

print("\nFinal Best position found:", best_position)
print("Final Best objective function value:", best_value)
```

OUTPUT:

```
Iteration 0: Best objective function value = 7.87512466951071
Iteration 1: Best objective function value = 1.3547762769002425
Iteration 2: Best objective function value = 0.09631496754081659
Iteration 3: Best objective function value = 0.09631496754081659
Iteration 4: Best objective function value = 0.09631496754081659
Iteration 5: Best objective function value = 0.09631496754081659
Iteration 6: Best objective function value = 0.09631496754081659
Iteration 7: Best objective function value = 0.09631496754081659
Iteration 8: Best objective function value = 0.09631496754081659
Iteration 9: Best objective function value = 0.09631496754081659
Iteration 10: Best objective function value = 0.09631496754081659

Final Best position found: [-0.303588   -0.06441502]
Final Best objective function value: 0.09631496754081659
```