```
import numpy as np
import matplotlib.pyplot as plt

# Sphere function (fitness function)
def sphere_function(position):
    return np.sum(position**2)

# Parameters
num_particles = 30  # Number of particles
num_iterations = 100  # Number of iterations
dim = 2  # Dimensionality of the problem
bounds = [-10, 10]  # Search space bounds
inertia_weight = 0.7  # w
cognitive_coefficient = 1.5  # c1
social_coefficient = 1.5  # c2
tolerance = 1e-6  # Stopping tolerance for fitness

# Initialize particles
class Particle:
    def __init__(self):
        self.position = np.random.uniform(bounds[0], bounds[1], dim)  # Random
position
        self.velocity = np.random.uniform(-1, 1, dim)  # Random velocity
        self.best_position = np.copy(self.position)  # Personal best position
        self.best_fitness = sphere_function(self.position)  # Personal best fitness
        self.fitness = self.best_fitness  # Current fitness

    def update_velocity(self, global_best_position):
        r1, r2 = np.random.rand(dim), np.random.rand(dim)
        cognitive_term = cognitive_coefficient * r1 * (self.best_position -
self.position)
        social_term = social_coefficient * r2 * (global_best_position -
self.position)
        self.velocity = inertia_weight * self.velocity + cognitive_term +
social_term

    def update_position(self):
        self.position += self.velocity
        self.position = np.clip(self.position, bounds[0], bounds[1])  # Keep within
bounds
        self.fitness = sphere_function(self.position)
        if self.fitness < self.best_fitness:  # Update personal best
            self.best_fitness = self.fitness
            self.best_position = np.copy(self.position)

# PSO implementation
def particle_swarm_optimization():
    particles = [Particle() for _ in range(num_particles)]
    global_best_position = particles[0].best_position
    global_best_fitness = particles[0].best_fitness
```

```
    fitness_history = []

    # Update global best from the initial population
    for particle in particles:
        if particle.best_fitness < global_best_fitness:
            global_best_fitness = particle.best_fitness
            global_best_position = np.copy(particle.best_position)

    for iteration in range(num_iterations):
        for particle in particles:
            particle.update_velocity(global_best_position)
            particle.update_position()

            # Update global best
            if particle.best_fitness < global_best_fitness:
                global_best_fitness = particle.best_fitness
                global_best_position = np.copy(particle.best_position)

        fitness_history.append(global_best_fitness)  # Track global best fitness

        # Early stopping if fitness reaches tolerance
        if global_best_fitness <= tolerance:
            print(f"Converged at iteration {iteration}")
            break

    return global_best_position, global_best_fitness, fitness_history

# Run the PSO algorithm
best_position, best_fitness, fitness_history = particle_swarm_optimization()

# Print the results
print("Best position found:", best_position)
print("Best fitness achieved:", best_fitness)

# Plot fitness over iterations
plt.plot(fitness_history)
plt.title("Fitness Over Iterations (PSO on Sphere Function)")
plt.xlabel("Iteration")
plt.ylabel("Fitness (Objective Function Value)")
plt.grid()
plt.show()
```