BIS LAB

LAB 2

Particle Swarm Optimization for Function Optimization:

Particle Swarm Optimization (PSO) is inspired by the social behavior of birds flocking or fish schooling. PSO is used to find optimal solutions by iteratively improving a candidate solution with regard to a given measure of quality. Implement the PSO algorithm using Python to optimize a mathematical function.

PYTHON CODE:

This version will **maximize** a mathematical function — for example:

f(x)=xsin⁡(10πx)+1.0,for x∈[0,1]f(x) = x \sin(10\pi x) + 1.0, \quad \text{for } x \in [0, 1]f(x)=xsin(10πx)+1.0,for x∈[0,1]

```
import random
import math

# 1. Define the Problem
def objective_function(x):
    """Example mathematical function to maximize."""
    return x * math.sin(10 * math.pi * x) + 1.0


# 2. Initialize Parameters
NUM_PARTICLES = 30      # Number of particles in the swarm
MAX_ITER = 50           # Number of iterations (generations)
W = 0.7                 # Inertia weight
C1 = 1.5                # Cognitive (particle's own experience)
C2 = 1.5                # Social (swarm experience)
X_MIN, X_MAX = 0, 1     # Search space boundaries
V_MIN, V_MAX = -0.1, 0.1  # Velocity limits


# 3. Initialize Particles
particles = []  # Each particle: position, velocity, personal_best_pos,
personal_best_val

for _ in range(NUM_PARTICLES):
    x = random.uniform(X_MIN, X_MAX)
    v = random.uniform(V_MIN, V_MAX)
    fitness = objective_function(x)
    particles.append({
        "position": x,
```

```
        "velocity": v,
        "best_position": x,
        "best_value": fitness
    })

# Initialize global best
global_best_position = max(particles, key=lambda p:
p["best_value"])["best_position"]
global_best_value = objective_function(global_best_position)


# 4-6. Main PSO Loop
for iteration in range(MAX_ITER):
    for particle in particles:
        x = particle["position"]
        v = particle["velocity"]
        fitness = objective_function(x)

        # Update personal best
        if fitness > particle["best_value"]:
            particle["best_value"] = fitness
            particle["best_position"] = x

        # Update global best
        if fitness > global_best_value:
            global_best_value = fitness
            global_best_position = x

    # Update velocities and positions
    for particle in particles:
        r1, r2 = random.random(), random.random()
        cognitive = C1 * r1 * (particle["best_position"] -
particle["position"])
        social = C2 * r2 * (global_best_position -
particle["position"])
        new_velocity = W * particle["velocity"] + cognitive + social
        new_velocity = max(min(new_velocity, V_MAX), V_MIN)  # Clamp
velocity

        new_position = particle["position"] + new_velocity
        new_position = max(min(new_position, X_MAX), X_MIN)  # Keep
within bounds

        particle["velocity"] = new_velocity
        particle["position"] = new_position

    # Print progress
    print(f"Iteration {iteration + 1} | Best Fitness:
{global_best_value:.5f}")
```

```python
# 7. Output Best Solution
print("\n Best Solution Found:")
print(f"x = {global_best_position:.5f}")
print(f"Fitness = {global_best_value:.5f}")
```

OUTPUT:

```
Iteration 1 | Best Fitness: 1.81892
Iteration 2 | Best Fitness: 1.81892
Iteration 3 | Best Fitness: 1.81892
Iteration 4 | Best Fitness: 1.85036
Iteration 5 | Best Fitness: 1.85036
Iteration 6 | Best Fitness: 1.85036
Iteration 7 | Best Fitness: 1.85036
Iteration 8 | Best Fitness: 1.85036
Iteration 9 | Best Fitness: 1.85051
Iteration 10 | Best Fitness: 1.85051
Iteration 11 | Best Fitness: 1.85051
Iteration 12 | Best Fitness: 1.85051
Iteration 13 | Best Fitness: 1.85051
Iteration 14 | Best Fitness: 1.85052
Iteration 15 | Best Fitness: 1.85052
Iteration 16 | Best Fitness: 1.85052
Iteration 17 | Best Fitness: 1.85056
Iteration 18 | Best Fitness: 1.85058
Iteration 19 | Best Fitness: 1.85058
Iteration 20 | Best Fitness: 1.85058
Iteration 21 | Best Fitness: 1.85058
Iteration 22 | Best Fitness: 1.85058
Iteration 23 | Best Fitness: 1.85060
Iteration 24 | Best Fitness: 1.85060
Iteration 25 | Best Fitness: 1.85060
Iteration 26 | Best Fitness: 1.85060
Iteration 27 | Best Fitness: 1.85060
Iteration 28 | Best Fitness: 1.85060
Iteration 29 | Best Fitness: 1.85060
Iteration 30 | Best Fitness: 1.85060
Iteration 31 | Best Fitness: 1.85060
Iteration 32 | Best Fitness: 1.85060
Iteration 33 | Best Fitness: 1.85060
Iteration 34 | Best Fitness: 1.85060
Iteration 35 | Best Fitness: 1.85060
Iteration 36 | Best Fitness: 1.85060
Iteration 37 | Best Fitness: 1.85060
Iteration 38 | Best Fitness: 1.85060
Iteration 39 | Best Fitness: 1.85060
Iteration 40 | Best Fitness: 1.85060
Iteration 41 | Best Fitness: 1.85060
Iteration 42 | Best Fitness: 1.85060
Iteration 43 | Best Fitness: 1.85060
Iteration 44 | Best Fitness: 1.85060
Iteration 45 | Best Fitness: 1.85060
Iteration 46 | Best Fitness: 1.85060
Iteration 47 | Best Fitness: 1.85060
Iteration 48 | Best Fitness: 1.85060
Iteration 49 | Best Fitness: 1.85060
Iteration 50 | Best Fitness: 1.85060

 Best Solution Found:
x = 0.85119
Fitness = 1.85060
```