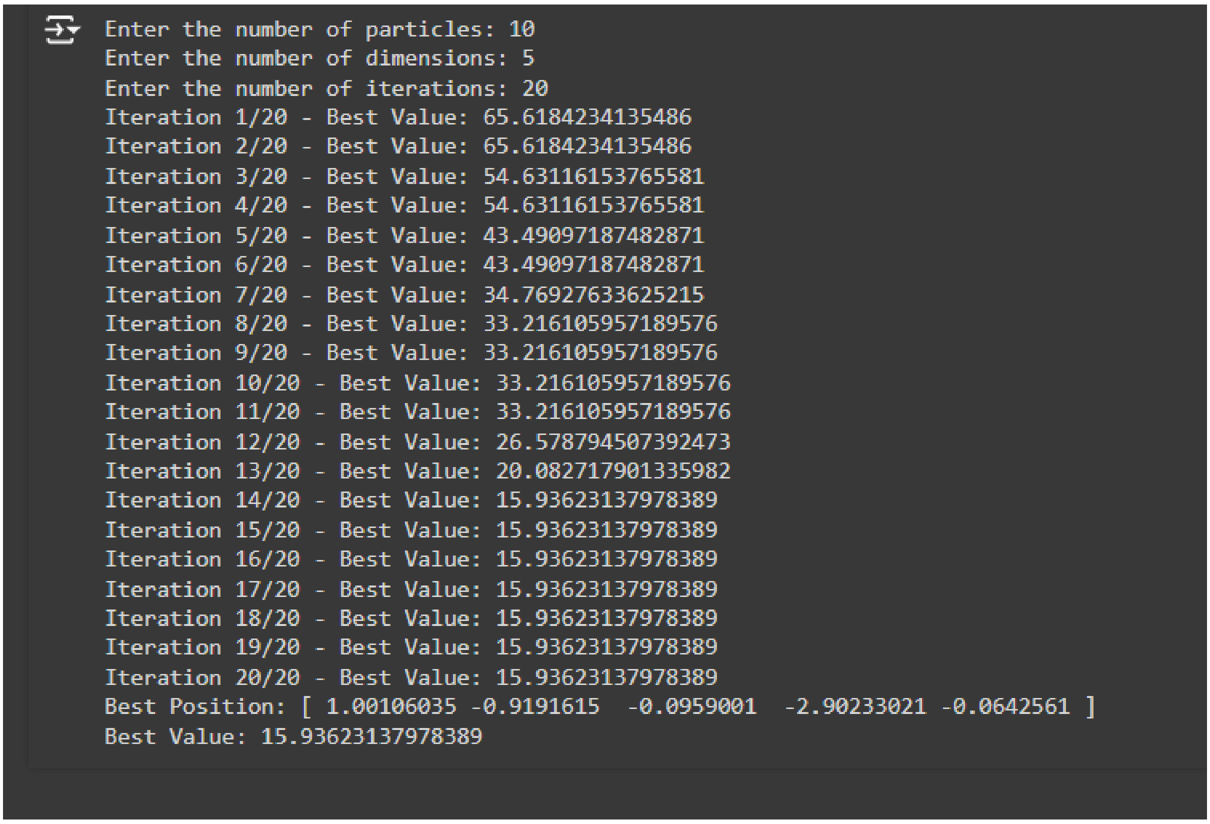
Lab 2:

Particle swarm

```
import numpy as np
def rastrigin(x):
    A = 10
    return A * len(x) + sum([xi**2 - A * np.cos(2 * np.pi * xi) for xi in x])
class Particle:
    def __init__(self, dim):
        self.position = np.random.uniform(-5.12, 5.12, dim)
        self.velocity = np.random.uniform(-1, 1, dim)
        self.best_position = np.copy(self.position)
        self.best_value = rastrigin(self.position)
    def update_velocity(self, global_best_position, inertia_weight, cognitive_coef, social_coef):
        r1, r2 = np.random.rand(2)
        cognitive_velocity = cognitive_coef * r1 * (self.best_position - self.position)
        social_velocity = social_coef * r2 * (global_best_position - self.position)
        self.velocity = inertia_weight * self.velocity + cognitive_velocity + social_velocity
    def update_position(self):
        self.position += self.velocity
        self.position = np.clip(self.position, -5.12, 5.12)
        current_value = rastrigin(self.position)
        if current_value < self.best_value:
            self.best_value = current_value
            self.best_position = np.copy(self.position)
def pso(num_particles, dim, num_iterations):
    inertia_weight = 0.7
    cognitive_coef = 1.5
    social_coef = 1.5

    particles = [Particle(dim) for _ in range(num_particles)]
    global_best_position = particles[0].best_position
    global_best_value = particles[0].best_value
    for iteration in range(num_iterations):
        for particle in particles:
            particle.update_velocity(global_best_position, inertia_weight, cognitive_coef, social_coef)
            particle.update_position()

            if particle.best_value < global_best_value:
                global_best_value = particle.best_value
                global_best_position = particle.best_position

        print(f"Iteration {iteration+1}/{num_iterations} - Best Value: {global_best_value}")
    return global_best_position, global_best_value
num_particles = int(input("Enter the number of particles: "))
dim = int(input("Enter the number of dimensions: "))
num_iterations = int(input("Enter the number of iterations: "))
best_position, best_value = pso(num_particles, dim, num_iterations)

print(f"Best Position: {best_position}")
print(f"Best Value: {best_value}")
```

Ouput:-



```
Enter the number of particles: 10
Enter the number of dimensions: 5
Enter the number of iterations: 20
Iteration 1/20 - Best Value: 65.6184234135486
Iteration 2/20 - Best Value: 65.6184234135486
Iteration 3/20 - Best Value: 54.63116153765581
Iteration 4/20 - Best Value: 54.63116153765581
Iteration 5/20 - Best Value: 43.49097187482871
Iteration 6/20 - Best Value: 43.49097187482871
Iteration 7/20 - Best Value: 34.76927633625215
Iteration 8/20 - Best Value: 33.216105957189576
Iteration 9/20 - Best Value: 33.216105957189576
Iteration 10/20 - Best Value: 33.216105957189576
Iteration 11/20 - Best Value: 33.216105957189576
Iteration 12/20 - Best Value: 26.578794507392473
Iteration 13/20 - Best Value: 20.082717901335982
Iteration 14/20 - Best Value: 15.93623137978389
Iteration 15/20 - Best Value: 15.93623137978389
Iteration 16/20 - Best Value: 15.93623137978389
Iteration 17/20 - Best Value: 15.93623137978389
Iteration 18/20 - Best Value: 15.93623137978389
Iteration 19/20 - Best Value: 15.93623137978389
Iteration 20/20 - Best Value: 15.93623137978389
Best Position: [ 1.00106035 -0.9191615  -0.0959001  -2.90233021 -0.0642561 ]
Best Value: 15.93623137978389
```