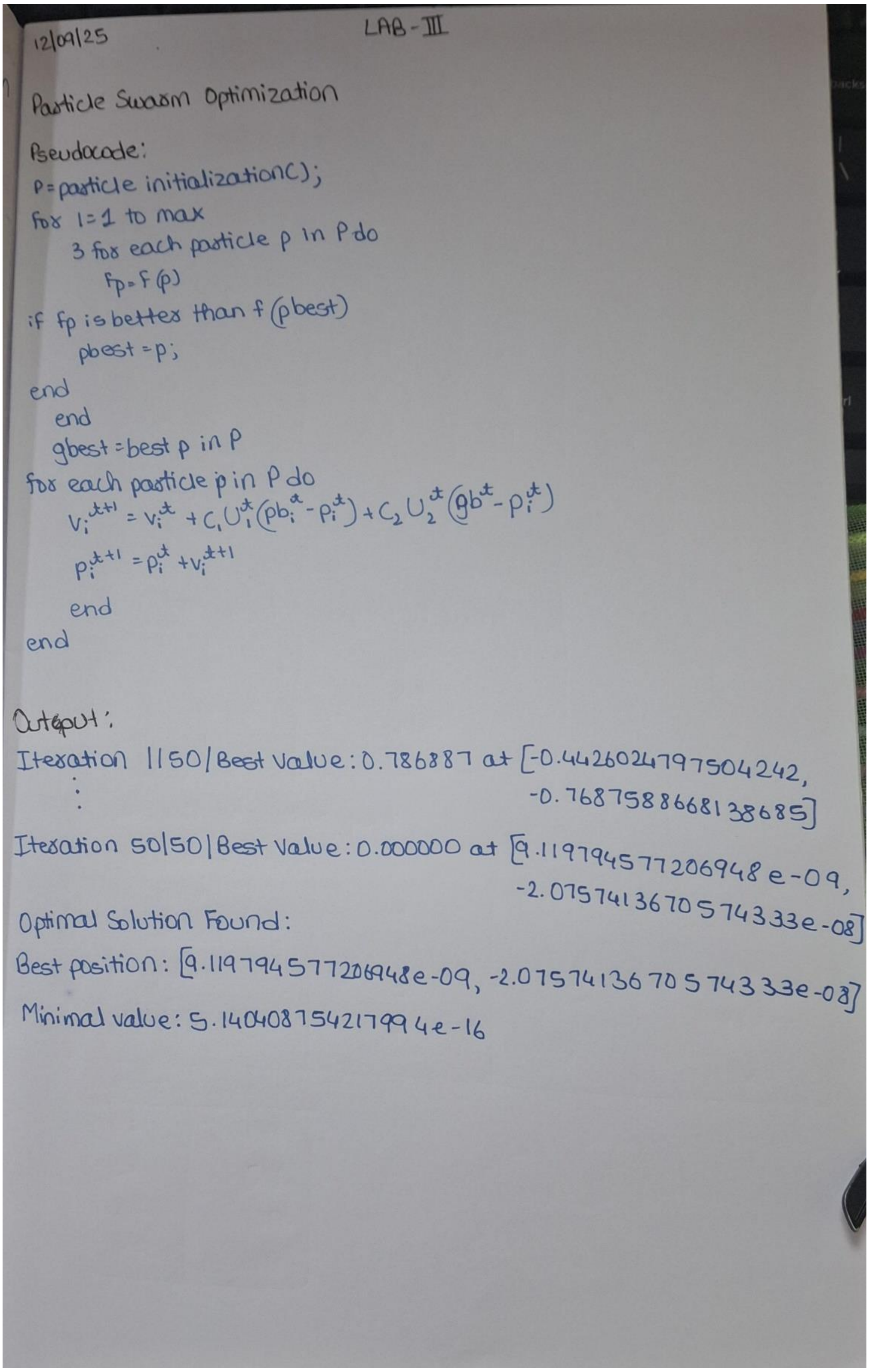
Lab 3

Particle Swarm Optimization



12/09/25

LAB - III

Particle Swarm Optimization

Pseudocode:

```
P=particle initialization();
for I=1 to max
      3 for each particle p in P do
            fp=f(p)
if fp is better than f(pbest)
      pbest=p;
end
   end
   gbest=best p in P
for each particle p in P do
      $v_i^{t+1} = v_i^t + c_1 U_1^t (pb_i^t - p_i^t) + c_2 U_2^t (gb^t - p_i^t)$
      $p_i^{t+1} = p_i^t + v_i^{t+1}$
      end
end
```

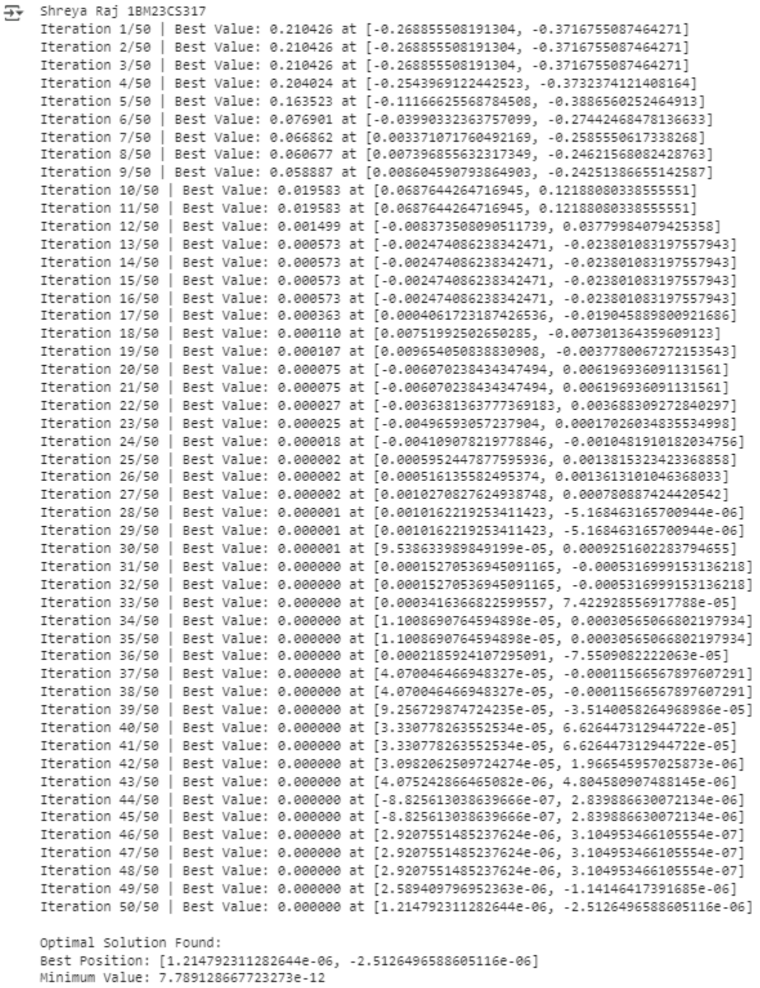
Output:

```
Iteration 1/50/Best Value:0.786887 at [-0.4426024797504242,
      :                                 -0.768758866813868 5]
      :
Iteration 50/50/Best Value:0.000000 at [9.119794577206948e-09,
                                        -2.075741367057433 3e-08]
Optimal Solution Found:
Best position: [9.119794577206948e-09, -2.0757413670574333e-08]
Minimal value: 5.140408754217994e-16
```

Code:

```
Print(“Shreya Raj 1BM23CS317”)

import random

# Objective (fitness) function: De Jong function
def fitness_function(position):
    x, y = position
    return x**2 + y**2 # minimize this function

# PSO parameters
num_particles = 10
num_iterations = 50
W = 0.3 # inertia weight (from PDF)
C1 = 2 # cognitive coefficient
C2 = 2 # social coefficient

# Initialize particles and velocities
particles = [[random.uniform(-10, 10), random.uniform(-10, 10)] for _ in
range(num_particles)]
velocities = [[0.0, 0.0] for _ in range(num_particles)]

# Initialize personal bests
pbest_positions = [p[:] for p in particles]
pbest_values = [fitness_function(p) for p in particles]

# Initialize global best
gbest_index = pbest_values.index(min(pbest_values))
gbest_position = pbest_positions[gbest_index][:]
gbest_value = pbest_values[gbest_index]

# PSO main loop
for iteration in range(num_iterations):
    for i in range(num_particles):
        r1, r2 = random.random(), random.random()

        # Update velocity
        velocities[i][0] = (W * velocities[i][0] +
                    C1 * r1 * (pbest_positions[i][0] - particles[i][0]) +
                    C2 * r2 * (gbest_position[0] - particles[i][0]))
        velocities[i][1] = (W * velocities[i][1] +
                    C1 * r1 * (pbest_positions[i][1] - particles[i][1]) +
                    C2 * r2 * (gbest_position[1] - particles[i][1]))

        # Update position
        particles[i][0] += velocities[i][0]
        particles[i][1] += velocities[i][1]
        # Evaluate fitness
        current_value = fitness_function(particles[i])

        # Update personal best
        if current_value < pbest_values[i]:
            pbest_positions[i] = particles[i][:]
            pbest_values[i] = current_value

            # Update global best
            if current_value < gbest_value:
                gbest_value = current_value
                gbest_position = particles[i][:]

        print(f"Iteration {iteration+1}/{num_iterations} | Best Value: {gbest_value:.6f} at {gbest_position}")

print("\nOptimal Solution Found:")
print(f"Best Position: {gbest_position}")
print(f"Minimum Value: {gbest_value}")
```

Output:



```
Shreya Raj 1BM23CS317
Iteration 1/50 | Best Value: 0.210426 at [-0.268855508191304, -0.3716755087464271]
Iteration 2/50 | Best Value: 0.210426 at [-0.268855508191304, -0.3716755087464271]
Iteration 3/50 | Best Value: 0.210426 at [-0.268855508191304, -0.3716755087464271]
Iteration 4/50 | Best Value: 0.204024 at [-0.2543969122442523, -0.3732374121408164]
Iteration 5/50 | Best Value: 0.163523 at [-0.11166625568784508, -0.3886560252464913]
Iteration 6/50 | Best Value: 0.076901 at [-0.03990332363757099, -0.27442468478136633]
Iteration 7/50 | Best Value: 0.066862 at [0.003371071760492169, -0.2585550617338268]
Iteration 8/50 | Best Value: 0.060677 at [0.007396855632317349, -0.24621568082428763]
Iteration 9/50 | Best Value: 0.058887 at [0.008604590793864903, -0.24251386655142587]
Iteration 10/50 | Best Value: 0.019583 at [0.0687644264716945, 0.12188080338555551]
Iteration 11/50 | Best Value: 0.019583 at [0.0687644264716945, 0.12188080338555551]
Iteration 12/50 | Best Value: 0.001499 at [-0.008373508090511739, 0.03779984079425358]
Iteration 13/50 | Best Value: 0.000573 at [-0.002474086238342471, -0.023801083197557943]
Iteration 14/50 | Best Value: 0.000573 at [-0.002474086238342471, -0.023801083197557943]
Iteration 15/50 | Best Value: 0.000573 at [-0.002474086238342471, -0.023801083197557943]
Iteration 16/50 | Best Value: 0.000573 at [-0.002474086238342471, -0.023801083197557943]
Iteration 17/50 | Best Value: 0.000363 at [0.0004061723187426536, -0.019045889800921686]
Iteration 18/50 | Best Value: 0.000110 at [0.00751992502650285, -0.007301364359609123]
Iteration 19/50 | Best Value: 0.000107 at [0.009654050838830908, -0.0037780067272153543]
Iteration 20/50 | Best Value: 0.000075 at [-0.006070238434347494, 0.006196936091131561]
Iteration 21/50 | Best Value: 0.000075 at [-0.006070238434347494, 0.006196936091131561]
Iteration 22/50 | Best Value: 0.000027 at [-0.0036381363777369183, 0.003688309272840297]
Iteration 23/50 | Best Value: 0.000025 at [-0.00496593057237904, 0.00017026034835534998]
Iteration 24/50 | Best Value: 0.000018 at [-0.004109078219778846, -0.0010481910182034756]
Iteration 25/50 | Best Value: 0.000002 at [0.0005952447877595936, 0.0013815323423368858]
Iteration 26/50 | Best Value: 0.000002 at [0.000516135582495374, 0.0013613101046368033]
Iteration 27/50 | Best Value: 0.000002 at [0.0010270827624938748, 0.000780887424420542]
Iteration 28/50 | Best Value: 0.000001 at [0.0010162219253411423, -5.168463165700944e-06]
Iteration 29/50 | Best Value: 0.000001 at [0.0010162219253411423, -5.168463165700944e-06]
Iteration 30/50 | Best Value: 0.000001 at [9.538633989849199e-05, 0.0009251602283794655]
Iteration 31/50 | Best Value: 0.000000 at [0.00015270536945091165, -0.0005316999153136218]
Iteration 32/50 | Best Value: 0.000000 at [0.00015270536945091165, -0.0005316999153136218]
Iteration 33/50 | Best Value: 0.000000 at [0.0003416366822599557, 7.422928556917788e-05]
Iteration 34/50 | Best Value: 0.000000 at [1.1008690764594898e-05, 0.00030565066802197934]
Iteration 35/50 | Best Value: 0.000000 at [1.1008690764594898e-05, 0.00030565066802197934]
Iteration 36/50 | Best Value: 0.000000 at [0.0002185924107295091, -7.5509082222063e-05]
Iteration 37/50 | Best Value: 0.000000 at [4.070046466948327e-05, -0.00011566567897607291]
Iteration 38/50 | Best Value: 0.000000 at [4.070046466948327e-05, -0.00011566567897607291]
Iteration 39/50 | Best Value: 0.000000 at [9.256729874724235e-05, -3.5140058264968986e-05]
Iteration 40/50 | Best Value: 0.000000 at [3.330778263552534e-05, 6.626447312944722e-05]
Iteration 41/50 | Best Value: 0.000000 at [3.330778263552534e-05, 6.626447312944722e-05]
Iteration 42/50 | Best Value: 0.000000 at [3.0982062509724274e-05, 1.966545957025873e-06]
Iteration 43/50 | Best Value: 0.000000 at [4.075242866465082e-06, 4.804580907488145e-06]
Iteration 44/50 | Best Value: 0.000000 at [-8.825613038639666e-07, 2.839886630072134e-06]
Iteration 45/50 | Best Value: 0.000000 at [-8.825613038639666e-07, 2.839886630072134e-06]
Iteration 46/50 | Best Value: 0.000000 at [2.9207551485237624e-06, 3.104953466105554e-07]
Iteration 47/50 | Best Value: 0.000000 at [2.9207551485237624e-06, 3.104953466105554e-07]
Iteration 48/50 | Best Value: 0.000000 at [2.9207551485237624e-06, 3.104953466105554e-07]
Iteration 49/50 | Best Value: 0.000000 at [2.589409796952363e-06, -1.14146417391685e-06]
Iteration 50/50 | Best Value: 0.000000 at [1.214792311282644e-06, -2.5126496588605116e-06]

Optimal Solution Found:
Best Position: [1.214792311282644e-06, -2.5126496588605116e-06]
Minimum Value: 7.789128667723273e-12
```