**LAB – 3**

**PARTICLE SWARM ALGORITHM**

**(APPLICATION USED LINEAR MODEL TRAINING)**

**CODE :**

import numpy as np

def fitness\_function(x):

    return np.sum(x\*\*2)

num\_particles = 10

num\_dimensions = 2

num\_iterations = 5

w = 5

c1 = 20

c2 = 25

positions = np.random.uniform(1, 5, (num\_particles, num\_dimensions))

velocities = np.zeros((num\_particles, num\_dimensions))

personal\_best\_positions = positions.copy()

personal\_best\_fitness = np.array([fitness\_function(p) for p in positions])

global\_best\_index = np.argmin(personal\_best\_fitness)

global\_best\_position = personal\_best\_positions[global\_best\_index]

global\_best\_fitness = personal\_best\_fitness[global\_best\_index]

for iteration in range(num\_iterations):

    for i in range(num\_particles):

        r1 = np.random.rand(num\_dimensions)

        r2 = np.random.rand(num\_dimensions)

        cognitive\_velocity = c1 \* r1 \* (personal\_best\_positions[i] - positions[i])

        social\_velocity = c2 \* r2 \* (global\_best\_position - positions[i])

        velocities[i] = w \* velocities[i] + cognitive\_velocity + social\_velocity

        positions[i] += velocities[i]

        fitness = fitness\_function(positions[i])

        if fitness < personal\_best\_fitness[i]:

            personal\_best\_fitness[i] = fitness

            personal\_best\_positions[i] = positions[i].copy()

        if fitness < global\_best\_fitness:

            global\_best\_fitness = fitness

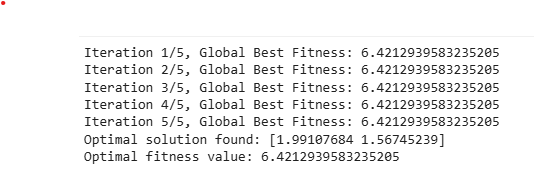
            global\_best\_position = positions[i].copy()

    print(f"Iteration {iteration+1}/{num\_iterations}, Global Best Fitness: {global\_best\_fitness}")

print("Optimal solution found:", global\_best\_position)

print("Optimal fitness value:", global\_best\_fitness)

**OUTPUT :**

****