particle swarm optimization

Algorithm:

step 1: Pick a mathematical function $f(x)$ to optimize.

step 2: set the parameters N, W, C_1, C_2 where C1 is weight of global best position of personal best position and C_2 is the weight of global Best position

step 3: Define the limits within which particle can move

step 4: Assign N with random velocity

step 5: For each particle calculate its fitness that is the best position

step 6: Update velocity based on the best velocity of its own based on the best velocity found by the swarm

step 7: It undergoes iterations to check the best solution found

step 8: Then in the final iteration it finds out the best value

Code:

```
import numpy as np
# Define the objective function
def f(x):
    return x**2

n_particles = 30
```

```
n_dims = 1
n_iter = 100
w = 0.7
c1 = 1.5
c2 = 1.5
pos = np.random.uniform(-10, 10, size = (n_
particles, n_dims))
vel = np.random.uniforms(-1, 1, size = (
 n_particles, n_dims))
Random velocities.

pbest_pos = pos.copy()
pbest_score = np.array([f(p) for p in pos])

gbest_pos = pbest_pos[np.argmin(pbest_
score)]
gbest_score = np.min(pbest_score)

for t in range(n_iter):
    for i in range(n_particles):
        fitness = f(pos[i])

        if fitness < pbest_score[i]:
            pbest_score[i] = fitness
            pbest_pos[i] = pos[i]

        if fitness < gbest_score:
            gbest_score = fitness
            gbest_pos = pos[i]
```

```
for i in range (n_particles):
    r1 = np.random.random(size = (n_dims))
    vel[i] = (w * vel[i] + c1 * r1 * (pbest_pos[i]
      - pos[i]) + (c2 * r2) * (gbest_pos - pos[i]))
    pos[i] = pos[i] + vel[i]

print (f"Iteration {i}/{n_iter}, Best
score: {gbest_score}")

print ("\nOptimization Complete.")
print (f"Best position (Global Best): {gbest_pos}")
print (f"Best score (Global Best fitness):
    {gbest_score}")
```

Sri