

# Particle Swarm Optimization Algorithm

Prof. DSc. BARSEKH-ONJI Aboud

Universidad Anáhuac México  
Facultad de Ingeniería

November 5, 2025

# Contenido

- 1 Introduction
- 2 Basic PSO Algorithm
- 3 PSO Dynamics
- 4 PSO Parameters and Termination
- 5 PSO Variants and Enhancements
- 6 PSO in Matlab
- 7 Conclusion

# What is PSO?

- PSO is a population-based stochastic optimization algorithm inspired by social behavior of bird flocking.
- Each individual (called a particle) represents a potential solution.
- Particles adjust their positions in the search space by learning from their own experience and that of their neighbors.
- Developed originally to simulate the choreography of birds.

# Particle Representation and Movement

- A swarm is a collection of particles.
- Each particle has a position  $x_i$  and velocity  $v_i$ .
- Position update:

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

# Velocity Update (gbest model)

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_1 (y_{ij}(t) - x_{ij}(t)) + c_2 r_2 (\hat{y}_j(t) - x_{ij}(t))$$

- $y_{ij}(t)$ : personal best
- $\hat{y}_j(t)$ : global best
- $r_1, r_2$ : random numbers in  $[0, 1]$

# Global vs. Local Best

- **gbest:** global best PSO, fast convergence, less diversity
- **lbest:** local best PSO, slower convergence, more robust
- Neighborhoods overlap to share information

# Inertia Weight in PSO

- Introduced by Shi and Eberhart to control the balance between exploration and exploitation.
- Modifies the velocity equation:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1r_1(y_{ij}(t) - x_{ij}(t)) + c_2r_2(\hat{y}_j(t) - x_{ij}(t))$$

- $w$  is the inertia weight. High  $w$  encourages exploration; low  $w$  promotes exploitation.

# Effects and Tuning of Inertia Weight

- If  $w \geq 1$ : velocity may diverge, particles can overshoot.
- If  $w < 1$ : velocities decay over time, enabling convergence.
- Typically  $w \in [0.4, 0.9]$ .
- Must be chosen in conjunction with  $c_1$ ,  $c_2$  to avoid divergence:

$$w > \frac{1}{2}(c_1 + c_2) - 1$$



# Dynamic Inertia Weight Strategies

- **Linear Decrease:**

$$w(t) = w_{\max} - \left( \frac{w_{\max} - w_{\min}}{t_{\max}} \right) t$$

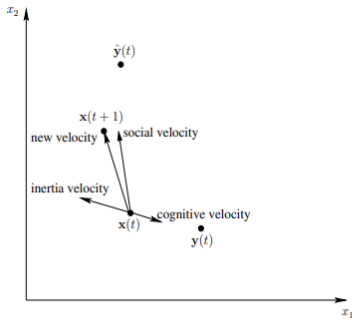
- **Nonlinear Decay** (e.g., exponential):

$$w(t) = \alpha w(t-1) \quad \text{with } \alpha \in (0, 1)$$

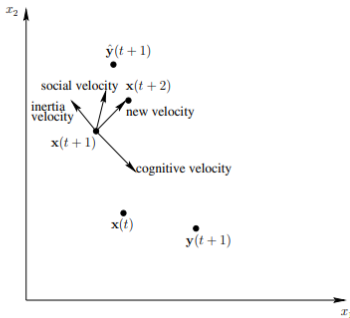
- **Fuzzy Adaptive** and **Stochastic** adjustments also explored.
- Goal: allow wide exploration initially, focus on refinement in later iterations.

# Velocity Components

- **Inertia:** memory of previous direction
- **Cognitive:** tendency to return to personal best
- **Social:** tendency to follow best in neighborhood



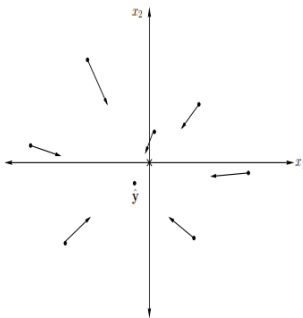
(a) Time Step  $t$



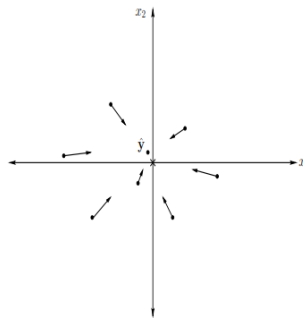
(b) Time Step  $t + 1$

# Geometric Interpretation-gbest

- Motion guided by vector components
- Balance between exploration and exploitation



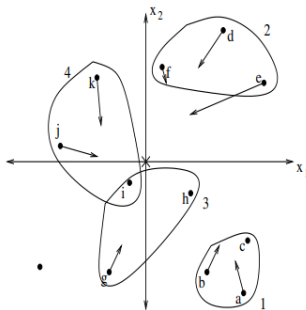
(a) At time  $t = 0$



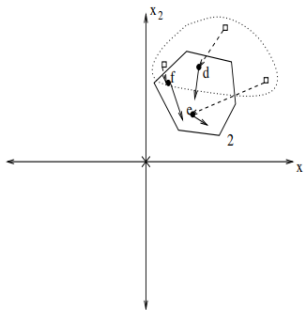
(b) At time  $t = 1$

# Geometric Interpretation-lbest

- Motion guided by vector components
- Balance between exploration and exploitation



(a) Local Best Illustrated – Initial Swarm



(b) Local Best – Second Swarm

# Key Parameters

- Swarm size  $n_s$
- Cognitive and social coefficients  $c_1, c_2$
- Inertia weight  $w$
- Maximum velocity  $v_{max}$

# Termination Criteria

- Maximum number of iterations
- Acceptable fitness threshold
- No improvement over iterations
- Normalized swarm radius
- Objective function slope  $\sim 0$

# Common Variants

- Inertia weight PSO
- Constriction factor PSO
- Barebones PSO
- Fully Informed PSO (FIPS)
- Charged PSO

# Hybrid PSO Models

- PSO with mutation (Gaussian, Cauchy)
- PSO with DE (Differential Evolution) crossover
- PSO + local search (GCPSO)
- PSO with genetic selection pressure



# Multi-Swarm and Niching PSOs

- Multi-swarm: cooperative or competitive subgroups
- Predator-prey PSO
- Self-organizing and adaptive networks
- Split dimension optimization

# PSO in Matlab

- `particleswarm` is a solver in MATLAB's Global Optimization Toolbox.
- It implements Particle Swarm Optimization (PSO) for solving bound-constrained optimization problems.
- Useful for problems where the objective function is non-differentiable, noisy, or has multiple local minima.

# Basic Syntax

- `x = particleswarm(fun, nvars)`  
Minimizes the objective function `fun` with `nvars` variables.
- `x = particleswarm(fun, nvars, lb, ub)`  
Adds lower (`lb`) and upper (`ub`) bounds to the variables.
- `x = particleswarm(fun, nvars, lb, ub, options)`  
Allows customization through the `options` structure.

# Key Parameters (1/2)

## ■ **SwarmSize**

Number of particles in the swarm. Default is 100.

## ■ **MaxIterations**

Maximum number of iterations allowed. Default is 100.

## ■ **FunctionTolerance**

Termination tolerance on the function value. Default is  $1e-6$ .

## ■ **InertiaRange**

Range for inertia weight. Controls exploration vs. exploitation.

## Key Parameters (2/2)

- **SelfAdjustmentWeight**

Weight for the particle's own best position influence.

- **SocialAdjustmentWeight**

Weight for the swarm's best position influence.

- **HybridFcn**

Function handle for a hybrid function to refine the solution.

- **Display**

Level of display output: 'off', 'iter', or 'final'.

# Velocity Update Equation

$$v = W \cdot v + y_1 \cdot u_1 \cdot (p - x) + y_2 \cdot u_2 \cdot (g - x)$$

- $v$ : Current velocity
- $W$ : Inertia weight
- $y_1$ : SelfAdjustmentWeight
- $y_2$ : SocialAdjustmentWeight
- $u_1, u_2$ : Random vectors in  $[0,1]$
- $p$ : Particle's best-known position
- $g$ : Global best-known position

# Termination Criteria

- Maximum number of iterations reached.
- Change in the best function value is less than `FunctionTolerance`.
- No improvement in the best function value for `MaxStallIterations`.

# PSO Parameters: Exploration vs. Exploitation

Parameter	Typical Range	Exploration	Exploitation
$w$	0.4 – 0.9	High $w$ (e.g., 0.9)	Low $w$ (e.g., 0.4)
$c_1$	1.5 – 2.5	High $c_1$	Low $c_1$
$c_2$	1.5 – 2.5	Low $c_2$	High $c_2$

**Table:** Impact of PSO parameters on exploration and exploitation



# PSO Parameters: Exploration vs. Exploitation

- **Inertia Weight ( $w$ ):** Controls the momentum of particles. Higher values encourage global exploration, while lower values promote local exploitation.
- **Cognitive Coefficient ( $c_1$ ):** Represents the particle's tendency to return to its own best position. Higher values increase individual exploration.
- **Social Coefficient ( $c_2$ ):** Represents the particle's tendency to move towards the swarm's best position. Higher values increase convergence and exploitation.

# Best Practices

- Start with default parameters; adjust based on problem complexity.
- Use bounds (lb, ub) to confine the search space.
- Consider using a hybrid function for fine-tuning the solution.
- Monitor convergence using the Display option.

# Conclusion

- PSO is an efficient population-based optimization algorithm
- Many variants have improved its robustness and flexibility
- Understanding social structure and velocity dynamics is key
- Suitable for continuous, discrete, and dynamic environments

# References

- MATLAB Documentation: particleswarm
- MATLAB Documentation: Particle Swarm Options
- Engelbrecht, A. P. (2007). *Computational Intelligence: An Introduction*. John Wiley & Sons