2025-11-01

Repository: `https://github.com/theSadeQ/dip-smc-pso`
Documentation: academic/paper/presentations/podcasts/episodes/

Learning Objective: Understand Particle Swarm Optimization algorithm, cost function design, and how PSO achieves 6-21% performance improvements

The Million-Dollar Question

Common Pitfall

Problem: Controllers have 6-12 gain parameters ($\lambda_1, \lambda_2, k_1, \ldots$)
How do we pick those numbers? Can't just guess!
Search Space: If each gain has 10 possible values: 10^{12} combinations (1 trillion!)

Key Concept

Solution: Particle Swarm Optimization (PSO)
Nature-inspired algorithm that finds near-optimal solutions in **minutes**, not years!

PSO: The Bird Flocking Analogy

Biological Inspiration (Kennedy & Eberhart, 1995)

Scenario: Flock of birds searching for food in a huge field

Each bird knows:

enumiBest spot THEY found (personal memory)

0. enumiBest spot ANYONE found (social/global info)

0. enumiCurrent flight direction (momentum)

Every few seconds: Adjust direction using all three signals

0. "I found good food over there!" (cognitive)

- "Flock found great food that way!" (social)
- "I'm flying this direction" (inertia)

Translation to Optimization

Bird ↔ Optimization

Bird Flocking	PSO
Bird	Particle (can
Position in field	Controller gai
Food quality	Cost function value (
Flight direction	Velocity vector (para
Best food I found	Personal b
Best food anyone found	Global b

Concrete Example: Classical SMC (6 Gains)

Example

Particle 1 at Iteration 10:
Current Position: $[\lambda_1 = 15.2, \lambda_2 = 8.3, \lambda_3 = 12.1, \lambda_4 = 5.7, k_1 = 20.4, k_2 = 3.9]$
Velocity: $[\Delta\lambda_1 = 0.5, \Delta\lambda_2 = -0.3, \Delta\lambda_3 = 0.8, \ldots]$ (drifting right on λ_1)

Personal Best: $[14.8, 8.5, 12.0, 5.5, 21.0, 4.0]$ with cost $= 7.2$
Global Best: $[14.5, 8.2, 11.8, 5.4, 20.5, 3.8]$ with cost $= 6.9$
Particle thinks: "Move toward MY best (14.8) AND toward GLOBAL best (14.5), while keeping momentum!"

Why PSO for Control Systems?

- **Gradient-free**: SMC cost functions are non-smooth (no derivatives)
- **Global search**: Explores whole space, not just local hills
- **Parallelizable**: Evaluate 30 particles simultaneously
- **Robust**: Works with noisy cost evaluations

PSO Update Equations

Key Concept

Only **TWO equations** in PSO - both mirror the bird analogy!

Position Update (Simple Physics)

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$$

Interpretation: New position = old position + velocity
Just like physics: if at x moving with velocity v, next step you're at $x + v$

Velocity Update (The Heart of PSO)

$$\mathbf{v}_i(t+1) = w \cdot \mathbf{v}_i(t) + c_1 r_1(\mathbf{p}_i - \mathbf{x}_i) + c_2 r_2(\mathbf{g} - \mathbf{x}_i)$$

Parameters:

- w = inertia weight (0.4-0.9) - momentum damping
- c_1 = cognitive coefficient (≈ 2.0) - personal memory strength
- c_2 = social coefficient (≈ 2.0) - social attraction strength
- r_1, r_2 = random numbers $\in [0, 1]$ - stochasticity
- $\mathbf{p}_i$ = personal best of particle i
- $\mathbf{g}$ = global best (all particles)

Three Terms Explained

Term 1: Inertia $(w \cdot \mathbf{v}_i)$

- Fraction of current velocity
- Keep moving in current direction
- *Momentum* - bird doesn't instantly stop
- $w = 0.9$: High exploration
- $w = 0.4$: Focus on refinement

Term 2: Cognitive $(c_1 r_1(\mathbf{p}_i - \mathbf{x}_i))$

- Vector toward personal best
- Pull toward own best discovery
- *"I found good food at* $\mathbf{p}_i$*!"*
- $c_1 = 0$: Ignore personal history
- $c_1 = 3$: Strongly trust yourself

Term 3: Social $(c_2 r_2(\mathbf{g} - \mathbf{x}_i))$

- Vector toward global best
- Pull toward swarm's best
- *"Someone found amazing food at* $\mathbf{g}$*!"*
- $c_2 = 0$: No cooperation
- $c_2 = 3$: Strongly trust swarm

Pro Tip

Random numbers (r_1, r_2)**:** Crucial for diversity! Without randomness, particles deterministically converge to same point - no exploration!

Worked Example: The Force Battle

Example

Particle 3, Iteration 5, First Gain (λ_1):
Setup:

- Current position: $\lambda_1 = 15.2$ (particle is HERE)
- Current velocity: $\Delta\lambda_1 = +0.5$ (drifting RIGHT, toward higher values)
- Personal best: $\lambda_1 = 14.8$ (own best is LOWER, to the LEFT)
- Global best: $\lambda_1 = 14.5$ (swarm best is even LOWER, further LEFT)

Three Forces:

enumi**Inertia:** Small push RIGHT (+0.35) - 70% of current momentum

0. enumi**Cognitive:** Moderate pull LEFT (toward 14.8) - random factor 0.42

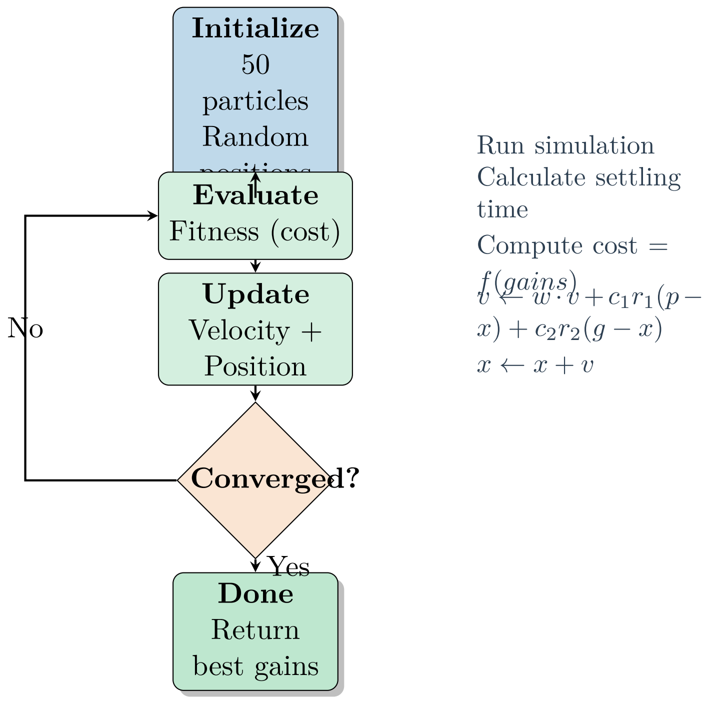
0. enumi**Social:** STRONG pull LEFT (toward 14.5) - random factor 0.78 (nearly max!)

Who Wins? Peer pressure dominates! Combined leftward forces (cognitive + social) overpower rightward momentum.
Outcome:

0. New velocity: ≈ -1.0 (was drifting right +0.5, now speeding left -1.0)

- New position: Moves from 15.2 $\rightarrow$ 14.1 (jumped left by 1.0)
- **Particle reversed direction!** Listened to collective wisdom: "Smaller λ_1 works better"

Convergence Behavior Over Time



Quick Summary

Early iterations (1-10): Large velocities, particles spread out (*exploration*)
Middle iterations (10-30): Velocities moderate, converge toward promising regions (*convergence*)
Late iterations (30-50): Small velocities, fine-tune around optimum (*exploitation*)

Cost Function: The Weighted Report Card

Key Concept

Four objectives to balance:

enumi**Accuracy**: Keep angles at zero (tracking grade)

0. enumi**Efficiency**: Minimize force/energy (energy grade)
0. enumi**Smoothness**: Avoid chattering (smoothness grade)
0. enumi**Stability**: Don't blow up! (Pass/Fail test)

The Report Card Formula

Total Cost

$$J_{\text{total}} = 0.6 \cdot J_{\text{state}} + 0.3 \cdot J_{\text{control}} + 0.1 \cdot J_{\text{rate}} + J_{\text{stability}}$$

Weights: 60% accuracy, 30% efficiency, 10% smoothness, + death penalty

Component 1: State Error (60%)

Measures: Tracking performance - how well pendulum stays upright

Math:

$$J_{\text{state}} = \sum_{i=1}^{N} (\theta_1^2 + \theta_2^2 + x^2)\Delta t$$

Why square?

0. Large errors hurt more

- $\theta = 10ř \Rightarrow \text{cost} = 100$
- $\theta = 20ř \Rightarrow \text{cost} = 400$

Example Values:

- Good: $J_{\text{state}} = 2.3$ (angles $< 2ř$)
- Mediocre: $J_{\text{state}} = 15.7$ (oscillates to 5ř)
- Bad: $J_{\text{state}} = 89.2$ (large swings)

Pro Tip

Squaring ensures positive/negative errors both count: $\theta = -5ř$ same as $\theta = +5ř$

Component 2: Control Effort (30%)

Measures: How much force/energy used

Math:

$$J_{\text{control}} = \sum_{i=1}^{N} u_i^2 \Delta t$$

Why minimize?

enumiHardware limits (motors have max torque)

0. enumiEnergy efficiency (battery life)
0. enumiSafety (aggressive control damages equipment)
0. enumiOverfitting prevention

Example Values:

0. Efficient: $J_{\text{control}} = 45.2$ (smooth, moderate)

- Aggressive: $J_{\text{control}} = 180.3$ (wasteful)

Example

Don't use sledgehammer when feather will do!

Component 3: Chattering (10%)

Measures: High-frequency oscillations in control signal (SMC's Achilles' heel)

Math:

$$J_{\text{rate}} = \sum_{i=1}^{N} \left(\frac{u_i - u_{i-1}}{\Delta t} \right)^2 \Delta t$$

Chattering Causes:

- Mechanical wear
- Heat generation
- Acoustic noise (high-pitched whine)
- Measurement noise amplification

Example:

Good (smooth):

$$u = [10.0, 10.2, 10.1, 9.9, 10.0]$$

$J_{\text{rate}} = 0.02$

Chattering (bad):

$$u = [10.0, -8.0, 12.0, -9.0, 11.0]$$

$J_{\text{rate}} = 284.0$

Component 4: The Death Penalty (Stability)

Common Pitfall

Hard Constraint: Pass/Fail test

$$J_{\text{stability}} = \begin{cases} 0 & \text{if } |\theta_1| < 45\text{ř AND } |\theta_2| < 45\text{ř for all } t \\ 1000 & \text{otherwise (system fell/diverged)} \end{cases}$$

Why 1000? Much larger than typical $J_{\text{state}} + J_{\text{control}} + J_{\text{rate}}$ (5-50)
PSO immediately rejects unstable controllers!

Example Report Cards

Good Controller:

- Accuracy: $2.3 \times 0.6 = 1.38$
- Efficiency: $45.2 \times 0.3 = 13.56$
- Smoothness: $0.02 \times 0.1 = 0.002$
- Death Penalty: 0 (stable!)
- **Total: 14.94** (Low is good!)

Unstable Controller:

- Accuracy: $0.5 \times 0.6 = 0.30$
- Efficiency: $30.0 \times 0.3 = 9.00$
- Smoothness: $0.01 \times 0.1 = 0.001$
- Death Penalty: **1000** (UNSTABLE!)
- **Total: 1009.30** (Terrible!)

Real Performance Improvements (MT-8 Benchmark)

PSO Results

Test Setup:

- 50 particles, 50 iterations (2500 evaluations per controller)
- Duration: 2-4 hours per controller
- Controllers tested: All 7 variants

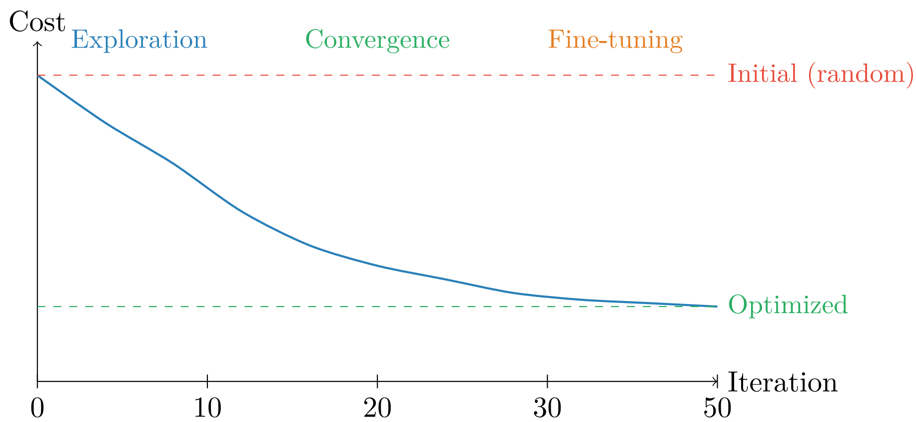
Performance Gains:

Controller	Before PSO
Classical SMC	Cost = 47.2
STA-SMC	Cost = 18.5
Hybrid Adaptive STA	Cost = 15.3
Average (all 7)	

Key Concept

Real-world impact: These percentage improvements can be the difference between successful SpaceX rocket landing and expensive fireball!

Convergence Timeline



Quick Reference: PSO Parameters

Standard PSO Configuration

- **Swarm size:** 30-50 particles (typical: 30)
- **Max iterations:** 50-100 (typical: 50)
- **Inertia weight:** $w = 0.7$ (balance exploration/exploitation)
- **Cognitive coeff:** $c_1 = 2.0$ (personal memory)
- **Social coeff:** $c_2 = 2.0$ (swarm cooperation)
- **Bounds:** Controller-specific (e.g., $\lambda \in [1, 50]$)

Cost Function Weights

- State error: $w_1 = 0.6$ (60% - tracking priority)
- Control effort: $w_2 = 0.3$ (30% - efficiency)
- Chattering: $w_3 = 0.1$ (10% - smoothness)
- Stability penalty: 1000 (death penalty)

Implementation Tips

Quick Summary

Performance:

enumiUse vectorized simulation for particle evaluation (10-100x speedup)

0. enumiParallelize across CPU cores (near-linear scaling)

0. enumiUse Simplified model for PSO, validate with Full model

Convergence:

0. Monitor global best cost - should decrease smoothly

- If stagnant after 20 iterations: Increase w or swarm diversity
- If oscillating: Decrease w (more exploitation)

Validation:

- ALWAYS re-test optimized gains with Full Nonlinear model
- Test robustness: Add 10% parameter uncertainty
- Check multiple initial conditions (Monte Carlo)

What's Next?

Key Concept

E005: Simulation Engine - The runner, vectorized simulators, and how we achieve 100x speedups for PSO
E006+: Advanced Topics - Analysis tools, benchmarks, Lyapunov proofs, research outputs
Remember: PSO gave us the gains. Now we need the engine to run 2500 simulations efficiently!