Section 5PSO Optimization Methodology

December 25, 2025

1 PSO Optimization Methodology

This section describes the Particle Swarm Optimization (PSO) framework used to automatically tune controller gains for optimal performance. PSO enables data-driven gain selection, replacing manual tuning with systematic optimization across the full parameter space.

1.1 Particle Swarm Optimization Background

Algorithm Overview:

Particle Swarm Optimization is a population-based metaheuristic inspired by social behavior of bird flocking and fish schooling [ref37]. PSO maintains a swarm of candidate solutions (particles), each representing a controller gain vector, which explore the parameter space through velocity and position updates.

Algorithm Dynamics:

Each particle i has position $\mathbf{g}i$ (gain vector) and velocity $\mathbf{v}i$ that evolve according to:

where: - $\mathbf{g}i^{(k)}$ - position of particle i at iteration k (gain vector) - $\mathbf{v}i^{(k)}$ - velocity of particle i at iteration k - $\mathbf{p}i$ - personal best position (best gain vector found by particle i) - $\mathbf{g}$best - global best position (best gain vector found by entire swarm) - w - inertia weight (balances exploration vs exploitation) - $c1, c2$ - cognitive and social acceleration coefficients - $r1, r2$ - random numbers uniformly distributed in $[0, 1]$

Physical Interpretation:

- Inertia Term ($w\mathbf{v}i^{(k)}$): Maintains current search direction, enabling exploration of distant regions - Cognitive Term ($c1r1(\mathbf{p}i - \mathbf{g}i^{(k)})$): Attracts particle toward its own best-known solution (personal memory) - Social Term ($c2r2(\mathbf{g}\text{best} - \mathbf{g}i^{(k)})$): Attracts particle toward swarm's global best (collective knowledge)

Hyperparameter Selection:

Following standard PSO recommendations [ref38]: - Inertia weight: $w = 0.7$ (balanced exploration-exploitation) - Cognitive coefficient: $c1 = 2.0$ (standard value) - Social coefficient: $c2 = 2.0$ (balanced personal-global influence)

Rationale: The combination $w = 0.7$, $c1 = c2 = 2.0$ provides: - Sufficient exploration (w prevents premature convergence) - Balanced cognitive-social influence ($c1 \approx c2$) - Provable convergence guarantees [ref39]

—

1.2 Fitness Function Design

Multi-Objective Cost Function:

The fitness function balances four competing objectives: tracking accuracy, energy efficiency, control smoothness, and sliding mode stability. Given a gain vector $\mathbf{g}$, the PSO evaluates cost $J(\mathbf{g})$ by simulating the DIP system and integrating performance metrics.

Cost Components:

where:

- Integrated State Error (ISE):

Penalizes deviation from equilibrium across all 6 state variables (cart position, angles, velocities). Lower ISE indicates faster convergence and smaller transient errors.

- Control Effort:

Penalizes energy consumption. Minimizing U reduces actuator power requirements and battery drain.

- Control Rate (Slew):

Penalizes rapid control changes (chattering). High-frequency switching causes actuator wear, acoustic noise, and excites unmodeled dynamics. This term directly addresses chattering reduction objective.

- Sliding Variable Energy:

Penalizes deviation from sliding surface (recall $\sigma = \lambda 1\theta 1 + \lambda 2\theta 2 + k1\dot{\theta}1 + k2\dot{\theta}2$ from Section 3.1). Minimizing σ ensures system remains on or near sliding manifold, validating SMC design.

Cost Normalization:

Raw cost components span vastly different scales (e.g., ISE $\sim 10^{-2}$, $U \sim 10^{3}$), requiring normalization for balanced optimization:

where $(\text{ISE}0, U0, \Delta U0, \sigma 0)$ are normalization constants. Two strategies implemented:

- Fixed Normalization: Manual constants based on typical system behavior

- Baseline Normalization (Disabled by Default): Compute normalization from initial baseline controller simulation (avoided due to numerical instability when baseline performs poorly)

Cost Weights:

Rationale: - $w\text{state} = 1.0$ prioritizes settling time and overshoot (primary objectives) - $w\text{ctrl} = 0.1$ encourages energy efficiency without sacrificing performance - $w\text{rate} = 0.01$ penalizes chattering (small weight prevents excessive damping) - $w\text{stab} = 0.1$ enforces sliding mode constraint

Instability Penalty:

When simulation diverges (angles $|\theta i| > \pi/2$ or states $> 10^{6}$), particle fitness receives severe penalty:

where: - $t\text{fail}$ - time at which simulation became unstable - $P\text{penalty}$ - large penalty constant (typically 10^{6}) - Graded penalty: Earlier failures penalized more heavily than late-stage instability

This penalty guides PSO away from unstable gain regions, ensuring all converged solutions stabilize the system.

Robustness Enhancement (Optional):

For robust optimization, fitness evaluated across multiple physics realizations with parameter perturbations (±5percent in masses, lengths, inertias):

where $Jj(\mathbf{g})$ is cost under j-th perturbed model, and $(w\text{mean}, w\text{max}) = (0.7, 0.3)$ balances average performance against worst-case. This multi-scenario evaluation ensures gains generalize beyond nominal conditions.

—

1.3 Search Space and Constraints

Controller-Specific Parameter Bounds:

PSO searches over bounded hypercubes tailored to each controller type. Bounds derived from: - Physical constraints (positive gains, actuator limits) - Stability theory (Lyapunov gain conditions from Section 4) - Empirical experience (avoid degenerate gain combinations)

Classical SMC (6 parameters: $[k1, k2, \lambda 1, \lambda 2, K, kd]$):

Rationale: - Lower bounds prevent numerical singularities (e.g., $ki > 2.0$ ensures sliding surface well-defined) - Upper bounds prevent excessive control effort (e.g., $\lambda i \leq 50$ avoids actuator saturation) - Switching gain K range satisfies Theorem 4.1 condition $K > \bar{d}$ (disturbance bound $\bar{d} \approx 0.2$ for DIP)

STA SMC (6 parameters: $[K1, K2, k1, k2, \lambda 1, \lambda 2]$):

Constraint: $K1 > K2$ enforced by bounds ($K1 \geq 2.0$, $K2 \leq 29.0$). Theorem 4.2 requires:

For DIP system with $\bar{d} \approx 0.2$, $\beta \approx 1.0$ (from Section 2), conditions become $K1 > 0.6$, $K2 > 0.2$, easily satisfied by bounds.

Adaptive SMC (5 parameters: $[k1, k2, \lambda 1, \lambda 2, \gamma]$):

Note: Adaptive gain $K(t)$ not tuned by PSO; it adapts online starting from $K\text{init} = 10.0$ (fixed). PSO tunes adaptation rate γ and sliding surface parameters.

Hybrid Adaptive STA SMC (4 parameters: $[k1, k2, \lambda 1, \lambda 2]$):

Simplification: Hybrid controller mode-switching logic (Section 3.5) uses fixed internal gains; PSO tunes only sliding surface parameters shared by both STA and Adaptive modes.

Bound Justification - Issue num12 Resolution:

Original PSO implementation used wide bounds (e.g., $K \in [0.1, 100]$), causing frequent exploration of unstable regions. Analysis revealed: - 47percent of PSO iterations produced divergent simulations (instability penalty triggered) - Convergence slowed by wasted evaluations in infeasible regions

Solution: Narrowed bounds to conservative ranges around validated baseline gains $[5, 5, 5, 0.5, 0.5, 0.5]$, reducing unstable fraction to ¡10percent. This "safe exploration" strategy accelerates convergence without sacrificing optimality.

Physical Constraints:

All gain vectors must satisfy: - Positive gains: $ki, \lambda i, K, \gamma > 0$ (guaranteed by lower bounds) - Actuator limits: Resultant control $|u| \leq u\text{max} = 20$ N (enforced during simulation via saturation) - Real-time feasibility: Control law computation time ¡50 mus (validated post-optimization, Section 7.1)

—

1.4 Optimization Protocol

Swarm Configuration:

[TABLE - See Markdown version for details]

Initialization Strategy:

Particles initialized uniformly within bounds:

where $\mathcal{U}$ denotes uniform distribution, and $(\mathbf{g}\text{min}, \mathbf{g}\text{max})$ are controller-specific bounds from Section 5.3.

Velocity Clamping:

To prevent particles from escaping search space or exhibiting erratic behavior:

Velocity limited to 20percent of search space range per iteration, ensuring gradual exploration.

Termination Criteria:

PSO terminates when any of the following conditions met:

- Maximum iterations: $k = N\text{iter} = 200$ (primary criterion) - Convergence threshold: Global best cost change $< 10^{-6}$ for 20 consecutive iterations (early stopping) - Timeout: Wall-clock time exceeds 120 minutes (safety for computationally expensive fitness evaluations)

Note: In practice, criterion 1 (maximum iterations) always triggered first for DIP system (each fitness evaluation takes 0.5s for 10s simulation, total time = 40 particles x 200 iterations x 0.5s = 1.1 hours).

Reproducibility:

All PSO runs seeded with fixed random seed (seed = 42) for deterministic results:

Computational Cost:

Total function evaluations per PSO run:

Each simulation: 10s duration, dt=0.01s -¿ 1000 time steps Total compute time: 1-2 hours on standard workstation (Intel i7, 16GB RAM, no GPU)

Vectorized Simulation Acceleration:

To reduce wall-clock time, particle evaluations vectorized using NumPy broadcasting: - Batch size: 40 particles simulated simultaneously - Speedup: 15x vs sequential evaluation (due to NumPy BLAS/LAPACK acceleration) - Memory: 200 MB for batch storage (40 particles x 1000 steps x 6 states x 8 bytes)

Post-Optimization Validation:

Best gain vector **g**best validated via: - Monte Carlo robustness test: 100 runs with random initial conditions (±0.05 rad range) - Model uncertainty sweep: ±10percent and ±20percent parameter perturbations (masses, lengths, inertias) - Compute time measurement: Verify control law meets ¡50 mus real-time constraint

Only if all validation tests pass, **g**best accepted as tuned gains. Otherwise, PSO re-run with adjusted bounds or fitness function weights.

[FIGURE - See Markdown version]

Figure 5.1: PSO Convergence Curves for **Classical SMC** Gain Optimization. Plot displays global best fitness (cost function value, Equation 5.2) evolution over 200 PSO iterations for four SMC controller variants, demonstrating typical particle swarm optimization convergence behavior on multi-modal control landscapes. **Classical SMC** (blue curve) exhibits fastest convergence, reaching fitness plateau 5.0 by iteration 60 due to simple 6-parameter space. **STA-SMC** (green curve) shows moderate convergence rate, achieving final fitness 4.0 with logarithmic improvement pattern characteristic of gradient-free optimization. **Adaptive SMC** (red curve) displays slowest convergence due to higher-dimensional search space (8 parameters including adaptation rates), settling at 6.0 after 150 iterations. Hybrid Adaptive STA (orange curve) demonstrates intermediate behavior, converging to 4.5 with two-phase pattern: rapid exploration (iterations 0-50, -40percent cost reduction) followed by gradual exploitation (iterations 50-200, diminishing returns). Early exploration phase shows high fitness variance as swarm explores parameter space; later exploitation exhibits smooth monotonic decrease as particles cluster around global optimum. All curves validate PSO termination criterion 1 (maximum 200 iterations) as primary stopping condition, with convergence threshold criterion 2 never triggered (cost changes remain ¿$10^-6 throughout).Total computational cost: 8,000 function evaluations per controller (40 particles x 200 iterations),$
$2 hours wall-clock time on standard workstation with NumPy vectorization achieving 15x speedup over sequential eva$
$off between parameter space dimensionality and convergence speed: simpler controllers (Classical) optimize faster$

—

1.5 Robust Multi-Scenario PSO Optimization (Addressing Overfitting)

Single-Scenario Optimization Pitfall:

Standard PSO protocol (Sections 5.2-5.4) optimizes gains for specific initial conditions (e.g., $[\theta 1, \theta 2] = [0.05, -0.03]$ rad). While this produces excellent performance for training scenarios, it suffers from severe generalization failure when tested on realistic disturbances: - 144.59x chattering degradation when testing on larger perturbations (± 0.3 rad vs ± 0.05 rad training) - Gains specialized for narrow operating envelope fail catastrophically outside training conditions

Root Cause: PSO converges to local minimum specialized for training conditions. The fitness function never encounters challenging scenarios, resulting in overfitted solutions analogous to machine learning models that memorize training data rather than learning generalizable patterns.

Robust PSO Solution:

To address this overfitting problem, we implemented a multi-scenario robust PSO approach that evaluates candidate gains across diverse initial condition sets spanning the operational envelope.

Multi-Scenario Fitness Function:

where: - ICj - j-th initial condition from scenario distribution - Nscenarios = 15 - number of evaluation scenarios per fitness call - $\alpha = 0.3$ - worst-case penalty weight (balances mean vs worst-case performance) - $J(\mathbf{g}; \text{IC}j)$ - standard cost function (Eq. 5.2) evaluated on scenario j

Scenario Distribution Strategy:

The 15 scenarios are distributed to emphasize real-world robustness while maintaining baseline performance:

[TABLE - See Markdown version for details]

Design Rationale: - 50percent weight on large disturbances reflects operational emphasis on robustness - 20percent nominal weight prevents complete sacrifice of baseline performance - Worst-case penalty ($\alpha = 0.3$) prevents gains that excel on some scenarios but catastrophically fail on others

Validation Results (MT-7 Protocol):

Validated on **Classical SMC** with 2,000 simulations (500 runs x 4 conditions):

[TABLE - See Markdown version for details]

Statistical Significance: - Welch's t-test: t = 5.34, p ¡ 0.001 (highly significant) - Effect size: Cohen's d = 0.53 (medium-large practical difference) - Conclusion: Improvement is statistically robust, not due to random variation

Key Findings: - Substantial Overfitting Reduction: 7.5x improvement in generalization (144.59x -¿ 19.28x degradation) - Absolute Performance: 94percent chattering reduction on realistic conditions (115k -¿ 6.9k) - Consistency: Tighter confidence intervals indicate more predictable behavior - Target Status: Partially achieved (19.28x degradation vs ¡5x target)

Computational Cost: - Overhead: 15x increase in fitness evaluation time (Nscenarios = 15) - Total PSO time: 6-8 hours (vs 1-2 hours for single-scenario) - Mitigation: Batch simulation vectorization evaluates multiple scenarios in parallel - Practical feasibility: Validated on standard workstation hardware (8-core CPU)

Implementation:

Robust PSO available via CLI flag:

Configuration parameters in ‘config.yaml‘:

Critical Insight: Any PSO-tuned controller intended for real-world deployment must undergo multi-scenario optimization and validation across the full expected operating range. Single-scenario optimization is suitable only for highly constrained laboratory environments where initial conditions remain within narrow bounds. The 7.5x generalization improvement demonstrates that robust PSO is essential for bridging the lab-to-deployment gap.

[FIGURE - See Markdown version]

Figure 5.2: MT-6 Adaptive Boundary Layer PSO Convergence Analysis. Dual-panel visualization comparing optimization trajectories for **Classical SMC** adaptive boundary layer tuning

(MT-6 benchmark). Left panel shows fitness evolution over 200 PSO iterations with multi-start validation: 5 independent PSO runs (different colors) demonstrate algorithm consistency, all converging to similar final cost values (6.2-6.5) despite different initialization, validating global optimum discovery rather than local minimum trapping. Fitness computed via Equation 5.2 multi-objective cost function (state error + control effort + smoothness penalty). Right panel presents particle diversity metric (swarm spread in parameter space) declining from initial uniform distribution (diversity 0.8) to tight clustering around optimum (diversity 0.1 by iteration 150), illustrating classic explore-exploit transition characteristic of PSO. Rapid diversity collapse (iterations 50-100) indicates premature convergence risk mitigated by inertia weight scheduling (w linearly decreasing 0.9 -¿ 0.4). Dashed vertical line marks iteration 120 where global best improvement stalls (¡10$^-6 change for 20 iterations), though termination criterion 2 (early stopping) never triggered, with algorithm runn$
$6 protocol optimizing two boundary layer parameters (\epsilon$min, α) for classical SMC chattering reduction. Note: Follow-up validation with unbiased frequency-domain metrics revealed that adaptive boundary layer achieves only marginal chattering reduction (3.7percent) vs fixed boundary layer, below the 30percent target. The fixed boundary layer (epsilon=0.02) was found to be near-optimal for this DIP system. This figure demonstrates PSO convergence characteristics rather than optimality of the resulting parameters. Demonstrates PSO robustness to initialization and convergence reliability for moderate-dimensional spaces (2-8 parameters typical for SMC gain tuning).

1.6 PSO Optimization Example: Classical SMC Gain Tuning

This section presents a concrete walkthrough of PSO gain optimization for **Classical SMC**, demonstrating the algorithm's convergence behavior with real numerical data.

—

Example 5.1: **Classical SMC** PSO Run (40 particles, 200 iterations)

Objective: Optimize 6 gains [k, k, lambda, lambda, K, k d] for **Classical SMC** to minimize multi-objective cost (Eq. 5.2).

Initial Swarm (Iteration 0):

40 particles initialized uniformly within bounds (Section 5.3):

Convergence Trajectory (Selected Iterations):

[TABLE - See Markdown version for details]

Convergence Analysis:

- Exploration Phase (Iterations 0-60): - Cost drops rapidly: 12.8 -¿ 4.82 (-62percent in 60 iterations) - Swarm diversity high (particles spread across parameter space) - Large velocity updates as particles discover promising regions - 8percent of particles trigger instability penalty (outside stable bounds)

- Exploitation Phase (Iterations 60-200): - Cost improves gradually: 4.82 -¿ 4.21 (-13percent in 140 iterations) - Swarm converges around global optimum (diversity-¿0) - Velocity decreases (particles fine-tune near best solution) - ¡1percent instability fraction (swarm clustered in stable region)

- Termination: - Maximum iterations (200) criterion triggered - Convergence threshold NOT met (cost still changing ¿10) - Final cost change (iter 190-200): 4.23 -¿ 4.21 (Delta = 0.02)

Performance Improvement:

Baseline gains (manual tuning): [5.0, 5.0, 5.0, 5.0, 0.5, 0.5] - Settling time: 2.50s - Overshoot: 8.0percent - Chattering index: 12.4 - Cost: 18.5

PSO-optimized gains: [5.2, 3.1, 10.5, 8.3, 1.5, 0.91] - Settling time: 1.82s (-27percent improvement) - Overshoot: 2.3percent (-71percent reduction) - Chattering index: 7.1 (-43percent reduction) - Cost: 4.21 (-77percent reduction)

Key Observations:

- Multi-objective trade-off: PSO balances settling time, overshoot, and chattering automatically (weights: 1.0, 0.1, 0.01 from Section 5.2) - Gain interpretation: - Increased lambda, lambda (5.0-¿10.5, 5.0-¿8.3): Faster convergence rates - Increased K (0.5-¿1.5): Stronger switching action (robustness) - Decreased k, k (5.0-¿5.2, 5.0-¿3.1): Gentler sliding surface (less aggressive) - Increased k d (0.5-¿0.91): More damping (reduced overshoot) - Computational cost: 8,000 simulations (40 particles x 200 iterations) @ 0.5s each = 1.1 hours - Reproducibility: Seeded with np.random.seed(42) -¿ deterministic results

Visual Interpretation (Figure 5.1):

The convergence curve for **Classical SMC** (blue line in Figure 5.1) shows logarithmic decay characteristic of PSO: - Steep initial drop (iterations 0-60): exploration discovers good regions - Gradual tail (iterations 60-200): exploitation refines solution - No premature convergence: cost continues improving throughout

Comparison with Other Controllers:

- **STA-SMC** (green): Similar convergence pattern but slower due to Lyapunov constraint checks (final cost 4.0 vs 4.21) - **Adaptive SMC** (red): Slowest convergence (8 parameters vs 6) but achieves comparable final cost (6.0) - Hybrid STA (orange): Two-phase convergence (rapid STA tuning -¿ slower Adaptive refinement, final cost 4.5)

1.7 Hyperparameter Sensitivity Analysis

While Section 5.4 specifies standard PSO hyperparameters (w=0.7, c=c=2.0), this section quantifies the impact of hyperparameter variations on optimization performance.

Table 5.1: PSO Hyperparameter Sensitivity Study (**Classical SMC**, 6 parameters)

[TABLE - See Markdown version for details]

Key Findings:

- Inertia Weight (w) - High Sensitivity: - Optimal range: w [0.6, 0.8] - w too high (0.9): Excessive exploration -¿ slow convergence (+30 iterations) - w too low (0.5): Premature convergence -¿ 14.5percent worse cost (trapped in local min) - Impact: ±20percent change in w -¿ ±10percent change in final cost

- Cognitive Coefficient (c) - Low Sensitivity: - Optimal range: c [1.5, 2.5] - Impact moderate: ±50percent change in c -¿ ±2percent change in cost - Personal memory less critical than social learning for this problem

- Social Coefficient (c) - Moderate Sensitivity: - Optimal range: c [1.5, 2.5] - Higher c (3.0) slightly beneficial (-2.1percent cost) but risks premature convergence - Impact: ±50percent change in c -¿ ±5percent change in cost

- Balance Critical: - Unbalanced c=1.0, c=3.0 causes swarm collapse (+23percent cost degradation) - Recommendation: maintain c = c (equal cognitive-social influence)

Sensitivity Ranking (from highest to lowest impact):

- Inertia w: ±20percent -¿ ±10percent cost change (High sensitivity) - Social c: ±50percent -¿ ±5percent cost change (Moderate sensitivity) - Cognitive c: ±50percent -¿ ±2percent cost change (Low sensitivity)

Practical Recommendation:

Stick with standard values (w=0.7, c=c=2.0) for SMC gain tuning: - Validated across multiple controller types (Classical, STA, Adaptive, Hybrid) - Robust to problem variations (different fitness landscapes) - No evidence that custom tuning provides significant benefit (¡3percent improvement) - Adaptive inertia scheduling (0.9-¿0.4) showed marginal gains (-0.7percent) not worth implementation complexity

When to Customize:

- Convergence too slow (¿200 iterations): Decrease w to 0.6, increase c to 2.5 - Premature convergence (¡50 iterations): Increase w to 0.8-0.9 - High-dimensional problems (¿10 parameters): Increase N p to 60-80, decrease w to 0.5-0.6

—

1.8 Algorithm Selection Rationale: Why PSO for SMC Gain Tuning?

This section justifies the choice of PSO over alternative optimization algorithms, providing comparative context for the methodology.

Table 5.2: Optimization Algorithm Comparison for Controller Gain Tuning

[TABLE - See Markdown version for details]

Why PSO is Optimal for This Problem:

- Multi-Modal Fitness Landscape: - SMC cost function (Eq. 5.2) exhibits multiple local minima - Different gain combinations can achieve similar performance - PSO swarm explores broadly -¿ discovers multiple promising regions - Advantage over SA, Nelder-Mead: Better global search capability

- Moderate Dimensionality (6-8 Parameters): - **Classical SMC**: 6 parameters, STA: 6, Adaptive: 8 - PSO's sweet spot: 4-15 parameters - Advantage over Bayesian Opt: Not too low-dimensional (would waste Gaussian Process overhead) - Advantage over CMA-ES: Not too high-dimensional (CMA-ES better for ¿20 params)

- Fast Fitness Evaluation (0.5s per simulation): - DIP simulation: 10s duration, dt=0.01s -¿ 1000 time steps, 0.5s compute - PSO's 8,000 evaluations feasible: 40 particles x 200 iterations x 0.5s = 1.1 hours - Advantage over Bayesian Opt: Fitness not expensive enough to justify surrogate modeling - Advantage over Grid Search: 10 evaluations (6 params x 10 values) = 138 hours (infeasible)

- No Gradient Information Available: - SMC cost not differentiable w.r.t. gains (chattering introduces discontinuities) - Finite differences unreliable due to noise and stochastic dynamics - Rules out: Gradient descent, L-BFGS, conjugate gradient - Requires: Gradient-free algorithms (PSO, GA, SA, CMA-ES)

- Robust Convergence with Standard Hyperparameters: - PSO works well with w=0.7, c=c=2.0 (no custom tuning needed) - Advantage over GA: Fewer hyperparameters (3 vs 5+), less tuning effort - Advantage over Bayesian Opt: No kernel selection, acquisition function tuning

- Implementation Simplicity: - PySwarms library: validated, vectorized, GPU-accelerated PSO - 50 lines of code for complete integration - Advantage over Bayesian Opt: GPyOpt/Optuna complex, high learning curve - Advantage over CMA-ES: pycma less mature, fewer features

- Parallelizable: - Batch simulation: evaluate all 40 particles simultaneously - NumPy vectorization: 15x speedup (Section 5.4) - Advantage over SA: Inherently serial (no parallelism)

When Alternative Algorithms Preferred:

[TABLE - See Markdown version for details]

Not Recommended for SMC Gain Tuning:

- Grid Search: 10 evaluations for 6 params (infeasible time/compute) - Gradient Descent: Cost not smooth (chattering discontinuities) - Simulated Annealing: Slower convergence than PSO (3-5x more iterations) - Random Search: Poor exploration efficiency vs PSO swarm intelligence

Conclusion:

PSO is the optimal choice for SMC gain tuning given: - Multi-modal landscape (require global search) - 6-8 parameters (PSO sweet spot) - Fast fitness evaluation (0.5s, feasible for 8,000 evals) -

No gradient information (require gradient-free method) - Standard hyperparameters work well (no custom tuning needed) - Simple implementation (PySwarms library)

For different problem characteristics (e.g., expensive fitness ¿10s, high-dimensional ¿15 params), alternative algorithms may be more appropriate (see Table 5.2).

—