Section 3Controller Design

December 25, 2025

1 Controller Design

This section presents the control law design for each of the seven SMC variants evaluated in this study. All controllers share a common sliding surface definition but differ in how they drive the system to and maintain it on this surface.

1.1 Sliding Surface (Common to All SMC Variants)

Definition:

The sliding surface $\sigma : \mathbb{R}^6 \to \mathbb{R}$ combines pendulum angle errors and their derivatives:

where: - $\lambda 1, \lambda 2 > 0$ - position error weights - $k1, k2 > 0$ - velocity error weights

Physical Interpretation:

The sliding surface represents a weighted combination of pendulum state errors. When $\sigma = 0$, the system evolves along a manifold in state space where angles and angular velocities satisfy the constraint $\lambda i \theta i + k i \dot{\theta} i = 0$ for $i = 1, 2$. This constraint enforces exponential convergence of each angle to zero with time constant $\tau i = ki/\lambda i$.

Design Philosophy:

- Reaching Phase: Drive system toward sliding surface ($\sigma \to 0$) - Sliding Phase: Maintain system on surface ($\sigma = 0$), ensuring exponential convergence to equilibrium - Steady-State: System remains at equilibrium ($\theta 1 = \theta 2 = 0$)

—

1.1.1 Controller Architecture Overview

All seven SMC variants in this study share a common architecture pattern but differ in specific implementation of the control law and how they handle uncertainties.

Figure 3.1: Common SMC architecture for DIP stabilization

Controller Family Tree:

Architectural Differences:

[TABLE - See Markdown version for details]

This architectural overview provides context for understanding design tradeoffs: simplicity (Classical) vs performance (STA) vs adaptability (Adaptive/Hybrid).

—

1.2 Classical Sliding Mode Control

Control Law:

where: - $u$eq - equivalent control (model-based feedforward) - $K > 0$ - switching gain (drives system to sliding surface) - $\epsilon > 0$ - boundary layer width (chattering reduction) - $kd \geq 0$ - derivative gain (damping) - sat(·) - saturation function (continuous approximation of sign function)

Equivalent Control:

The equivalent control compensates for known dynamics:

where: - $L = [0, k1, k2]$ - sliding surface gradient vector - M, C, G - inertia, Coriolis, gravity matrices from Section 2 - $B = [1, 0, 0]^T$ - control input matrix

Saturation Function (Boundary Layer):

Two options implemented:

- Hyperbolic Tangent (Default): Smooth transition, maintains control authority near $\sigma = 0$

- Linear Saturation: Piecewise linear, sharper switching

Design Parameters:

[TABLE - See Markdown version for details]

Advantages: - Simple implementation (6 gains) - Fastest computation (18.5 mus, Section 7.1) - Well-understood theory - Good energy efficiency (12.4 J, Section 7.4)

Disadvantages: - Moderate chattering (index 8.2, Section 7.3) - Larger overshoot (5.8percent, Section 7.2) - Boundary layer introduces steady-state error

Implementation Notes:

Discretization (dt = 0.01s, 100 Hz control loop):

The continuous-time control law must be discretized for digital implementation:

- Sliding Surface: Direct substitution (no discretization error)

- Equivalent Control: Use backward differentiation for stability

- Saturation Function: tanh is inherently continuous, no discretization needed

Numerical Stability:

- Matrix Inversion: M(q) is always invertible (positive definite) but can become ill-conditioned for large theta. Use LU decomposition (scipy.linalg.solve) instead of explicit inv(M) - Overflow Prevention: Clip intermediate calculations: u eq limited to ±100N before adding switching term - Derivative Estimation: Use filtered backward difference for theta (Butterworth 2nd-order, 20 Hz cutoff) to reduce noise amplification

Computational Breakdown (18.5 mus total):

[TABLE - See Markdown version for details]

Common Pitfalls:

- Chattering from small epsilon: Setting epsilon¡0.01 causes high-frequency switching (¿50 Hz). Stay above epsilon≥0.02 for dt=0.01s. - Instability from large k d: Derivative gain k d¿5.0 can cause oscillations due to noise amplification in theta estimates. - Steady-state error from large epsilon: Boundary layer epsilon¿0.1 introduces 5percent steady-state error in theta. Tune epsilon to balance chattering vs accuracy. - Matrix inversion failure: For —theta—¿/2, M(q) becomes poorly conditioned. Always check condition number: cond(M) ¡ 1000.

Figure 3.2: **Classical SMC** block diagram

Signal Flow: - Measure state x = [x, theta, theta, , theta, theta] - Compute sliding surface σ = lambdatheta + lambdatheta + ktheta + ktheta - Compute equivalent control u eq (model-based feedforward) - Compute switching term: -K·sat(sigma/epsilon) - Compute derivative damping: -k d·sigma - Sum all terms: u = u eq - K·sat(sigma/epsilon) - k d·sigma - Apply saturation: u sat = clip(u, -20N, +20N)

—

1.3 Super-Twisting Algorithm (STA-SMC)

Control Law:

STA employs a continuous 2nd-order sliding mode algorithm:

where: - $K1, K2 > 0$ - STA algorithm gains (satisfy Lyapunov conditions) - z - integral state (provides continuous control action) - sign(σ) - smoothed via saturation function: sign(σ) $\approx$ tanh(σ/ϵ)

Key Features:

- Continuous Control: Unlike classical SMC, uSTA is continuous (no discontinuity at $\sigma = 0$) - Finite-Time Convergence: Guaranteed convergence to $\sigma = 0$ in finite time (not just asymptotic) - Chattering Reduction: Continuous action inherently eliminates chattering

Gain Selection (Lyapunov-Based):

For stability, gains must satisfy:

where $\bar{d}$ is the upper bound on disturbances.

Convergence Time Estimate:

Upper bound on reaching time:

Design Parameters:

[TABLE - See Markdown version for details]

Advantages: - Best overall performance (1.82s settling, 2.3percent overshoot) - Lowest chattering (index 2.1, 74percent reduction vs Classical) - Most energy-efficient (11.8 J) - Finite-time convergence guarantee

Disadvantages: - +31percent compute overhead vs Classical (24.2 mus) - More complex gain tuning (Lyapunov conditions) - Less intuitive than classical SMC

Figure 3.3: Super-Twisting Algorithm (STA) block diagram

Signal Flow: - Measure state x = [x, theta, theta, , theta, theta] - Compute sliding surface σ = lambdatheta + lambdatheta + ktheta + ktheta - Compute equivalent control u eq (model-based feedforward) - Compute proportional term: -K—sigma—$^(1/2) sign(sigma) - Computeintegralstate$: $\dot{z} = -K sign(sigma) - SumSTAterms : uSTA = -K|sigma|^(1/2) sign(sigma) + z - Totalcontrol$: $u = ueq + uSTA - Applysaturation : usat = clip(u, -20N, +20N)$

Implementation Notes:

Discretization (dt = 0.01s):

- Fractional Power Term: —sigma—$^(1/2) cancausenumericalissuesforsmallsigma. Usesafetythreshold$:

- Integral State Update: Use backward Euler for stability:

- Sign Function Smoothing: Replace discontinuous sign with smooth saturation:

Numerical Stability:

- Integral Windup: Clip z to prevent unbounded growth: z [-100, +100] - Division by Zero: Check —sigma— ¿ epsilon min before computing fractional power - Overflow Protection: Clip u STA before adding to u eq: u STA [-50N, +50N]

Common Pitfalls:

- Instability from violating Lyapunov conditions: Ensure K² $\geq$ 2Kd where d is disturbance bound (1.0 for DIP) - Integral windup: Without anti-windup (z clamping), integral state can grow unbounded during saturation - Chattering from small epsilon: If epsilon¡0.005, sign function becomes too sharp -¿ high-frequency switching - Slow convergence from small K: If K¡8.0, reaching time increases beyond acceptable limits (¿5s)

—

1.4 Adaptive Sliding Mode Control

Control Law:

where: - $K(t)$ - time-varying adaptive gain - $\gamma > 0$ - adaptation rate (increase when $|\sigma|$ large) - $\beta > 0$ - leak rate (decay toward Kinit when $|\sigma|$ small) - $\delta > 0$ - dead-zone threshold - Kinit - nominal gain value

Adaptation Mechanism:

- Outside Dead-Zone ($|\sigma| > \delta$): Gain increases proportionally to sliding surface magnitude, providing more control authority when far from surface - Inside Dead-Zone ($|\sigma| \leq \delta$): Gain decays toward nominal value, preventing unbounded growth

Bounded Gain Constraint:

Prevents gain saturation or underflow.

Design Parameters:

[TABLE - See Markdown version for details]

Advantages: - Adapts to model uncertainty online - Predicted best robustness to parameter errors (15percent tolerance, Section 8.1) - Bounded gains prevent instability

Disadvantages: - Slowest settling (2.35s, Section 7.2) - Highest chattering (index 9.7, Section 7.3) - Highest energy (13.6 J, +15percent vs STA) - Most complex computation (31.6 mus)

—

1.5 Hybrid Adaptive STA-SMC

Control Law:

Hybrid controller switches between STA mode and Adaptive mode based on sliding surface magnitude:

where: - uSTA - STA control law (Section 3.3) - uAdaptive - Adaptive control law (Section 3.4) - σswitch - mode switching threshold

Switching Logic:

- Reaching Phase ($|\sigma|$ large): Use STA for fast, chattering-free convergence - Sliding Phase ($|\sigma|$ small): Use Adaptive for robustness to model uncertainty - Hysteresis: Implement hysteresis band to prevent chattering between modes

Mode Transition:

where Δ is hysteresis margin.

Design Parameters:

[TABLE - See Markdown version for details]

Advantages: - Balanced performance (1.95s settling, 3.5percent overshoot) - Best predicted robustness (16percent model uncertainty tolerance) - Good disturbance rejection (89percent attenuation) - Combines STA speed with Adaptive robustness

Disadvantages: - Complex switching logic requires validation - Moderate compute overhead (26.8 mus) - Requires tuning both STA and Adaptive gains

Figure 3.4: Hybrid Adaptive **STA-SMC** with mode switching

Signal Flow: - Measure state x = [x, theta, theta, , theta, theta] - Compute sliding surface σ = lambdatheta + lambdatheta + ktheta + ktheta - Compute equivalent control u eq (model-based feedforward) - Evaluate mode selector: - If —sigma— ¿ σ switch + Delta -¿ Mode = STA - If —sigma— ¡ σ switch - Delta -¿ Mode = Adaptive - Otherwise -¿ Keep previous mode (hysteresis) - Compute control based on mode: - STA mode: u sw = -K—sigma—$^{(}1/2) sign(sigma) + z - Adaptivemode : usw = -K(t) sat(sigma/epsilon) - kdsigma - Totalcontrol : u = ueq + usw - Applysaturation : usat = clip(u, -20N, +20N)$

Implementation Notes:

Mode Switching Logic (Critical for Safety):

- Hysteresis Implementation:

- State Continuity: When switching modes, ensure control continuity: - Transfer integral state z from STA to Adaptive K(t) - Use smooth transition: u[k] = alpha·u STA + (1-alpha)·u Adaptive where α [0,1] based on hysteresis position

- Mode Initialization: - Start in STA mode (typical for large initial errors) - Initialize z=0, K(t)=K init - Track mode transitions for debugging

Numerical Stability:

- Bumpless Transfer: During mode switch, match initial conditions: - STA-¿Adaptive: Set K(t) = current equivalent switching gain - Adaptive-¿STA: Set z = accumulated adaptive correction - Anti-Windup: Reset integral states (z or K) if control saturates for ¿100ms - Mode Chattering Prevention: Enforce minimum dwell time (50ms) in each mode

Common Pitfalls:

- Mode chattering: If Delta too small (¡0.005), controller oscillates between modes -¿ instability - Discontinuous control: Without bumpless transfer, u jumps at mode switches -¿ excites high-frequency dynamics - Incorrect state initialization: Forgetting to transfer integral states causes transient spikes (¿20percent overshoot) - Hysteresis too large: If Delta ¿ σ switch/2, mode never switches -¿ defeats hybrid design purpose

—

1.6 Swing-Up SMC

Two-Phase Control:

Swing-up SMC operates in two distinct modes:

Phase 1: Swing-Up (Energy-Based Control)

When total system energy $E < E$threshold:

where: - $k$swing > 0 - swing-up gain - Energy pumping: Adds energy when $\cos(\theta 1)\dot{\theta}1 > 0$ (constructive phase)

Phase 2: Stabilization (SMC)

When $E \geq E$threshold and $|\theta 1|, |\theta 2| < \theta$switch:

Uses any SMC variant (typically Classical or STA) for stabilization.

Energy Calculation:

Mode Transition Logic:

Design Parameters:

[TABLE - See Markdown version for details]

Advantages: - Global controller (works from any initial condition) - Can bring pendulum from downward to upward position - Combines energy-based and model-based control

Disadvantages: - Complex mode logic requires careful tuning - Swing-up phase performance not guaranteed (heuristic energy pumping) - Not applicable to small perturbation stabilization (this study's focus)

—

1.7 Model Predictive Control (MPC)

Optimization Problem:

At each time step, solve finite-horizon optimal control problem:

where: - N - prediction horizon (number of future time steps) - Q, R, Qf - state, input, terminal cost weight matrices - $f(\cdot,\cdot)$ - discretized nonlinear dynamics (Section 2) - $u$max - actuator limit

Linearization (For Computational Efficiency):

Approximate nonlinear dynamics around current trajectory:

where $A(k), B(k)$ are Jacobians computed via finite differences.

Implementation:

Uses 'cvxpy' library to solve quadratic program (QP) at each time step.

Design Parameters:

[TABLE - See Markdown version for details]

Advantages: - Explicit handling of constraints (actuator limits, state bounds) - Optimal control over finite horizon - Can incorporate future reference trajectories

Disadvantages: - Computationally expensive (requires external optimizer) - Not self-contained (depends on 'cvxpy') - Real-time feasibility questionable for 10 kHz control - Excluded from main comparative analysis (dependency issue)

—

1.8 Summary and Comparison

Table 3.1: Controller Characteristics Comparison

[TABLE - See Markdown version for details]

Convergence Guarantees:

[TABLE - See Markdown version for details]

Design Complexity:

- Simplest: **Classical SMC** (6 scalar gains) - Moderate: STA SMC (2 gains + Lyapunov conditions), **Adaptive SMC** (5 gains + adaptation law) - Complex: Hybrid STA (8 gains + switching logic) - Most Complex: Swing-Up SMC (energy calculation + mode transitions), MPC (weight matrices + optimization)

Computational Complexity Analysis:

Table 3.2: Detailed Computational Breakdown

[TABLE - See Markdown version for details]

Common Operations (All Controllers): - M, C, G Evaluation: 8.2 mus, 120 FLOPs (inertia matrix, Coriolis, gravity) - Matrix Inversion: 4.1 mus, 60 FLOPs (3x3 LU decomposition for $M^{-1}) - Overhead : 1.3 - 1.5 mus (functioncalls, memoryaccess, statecopying)$

Controller-Specific Costs:

- **Classical SMC** (4.9 mus control law): - Sliding surface sigma: 0.9 mus (10 FLOPs: 4 multiplies + 3 adds) - Equivalent control u eq: 2.8 mus (40 FLOPs: matrix-vector products) - Switching term: 1.2 mus (5 FLOPs: saturation + multiply) - Bottleneck: u eq calculation (58percent of control law time)

- STA SMC (10.6 mus control law): - Sliding surface sigma: 0.9 mus (same as Classical) - Equivalent control u eq: 2.8 mus (same as Classical) - Fractional power —sigma—$^{1/2}: 3.2 mus (sqrtoperation\ 50 cycles) - Integralstateupdate\dot{z} : 2.1 mus (signfunction + integration) - Signsmoothing(tanh) : 1.6 mus (\ 40cyclesfortanhap$ $Bottleneck : Fractionalpowerterm(30 percentof controllawtime)$

- **Adaptive SMC** (17.8 mus control law): - Sliding surface sigma: 0.9 mus - Equivalent control u eq: 2.8 mus - Switching term: 1.2 mus (same as Classical) - Gain adaptation update: 8.4 mus (dead-zone check, conditional update, bounds checking) - State history management: 4.5 mus (circular buffer for derivative estimation) - Bottleneck: Gain adaptation (47percent of control law time)

- Hybrid STA (13.2 mus control law): - Sliding surface sigma: 0.9 mus - Equivalent control u eq: 2.8 mus - Mode selector logic: 2.1 mus (hysteresis check, mode transitions) - Dual control law computation: 6.2 mus (compute both STA and Adaptive in parallel) - Bumpless transfer: 1.2 mus (state continuity during mode switch) - Bottleneck: Dual control law (47percent of control law time)

- Swing-Up SMC (8.5 mus control law): - Energy calculation: 3.8 mus (kinetic + potential energy terms) - Mode selector: 0.8 mus (energy threshold check) - Swing-up term: 1.4 mus (k swing cos(theta) theta) - SMC stabilizer: 2.5 mus (simplified **Classical SMC**) - Bottleneck: Energy calculation (45percent of control law time)

Real-Time Feasibility (100 Hz Control Loop):

[TABLE - See Markdown version for details]

Notes: - All SMC variants have ¿99.6percent timing margin -¿ safe for 100 Hz deployment - MPC requires optimization solver (10-50 iterations) -¿ not real-time feasible without warm-start - Worst-case timing (**Adaptive SMC**): 31.6 mus ¡¡ 10 ms deadline (0.32percent utilization)

Scalability to Faster Control Loops:

[TABLE - See Markdown version for details]

Observations: - SMC variants scale to 5 kHz (200 mus budget) with ¿84percent margin (Classical) or ¿84percent margin (Adaptive) - **Classical SMC** fastest -¿ best for high-frequency applications (robotics: 1-10 kHz) - MPC limited to ¡100 Hz without hardware acceleration (GPU, FPGA)

1.9 Parameter Tuning Guidelines

This section provides step-by-step tuning procedures for each controller, based on system characteristics and performance requirements.

General Tuning Principles:

- Start Conservative: Begin with small gains, increase gradually until performance meets requirements - One Parameter at a Time: Change single parameter, observe response, iterate - Measure Performance: Track settling time, overshoot, chattering index after each change - Document Baseline: Record initial parameters and performance for comparison

System Characterization (Required Before Tuning):

Before tuning any controller, characterize the DIP system: - Mass ratios: m/m, m/m (affects inertia coupling) - Length ratios: L/L cart, L/L (affects angular dynamics) - Natural frequencies: omega = (g/L), omega = (g/L) (sets response timescales) - Disturbance levels: Measure typical external force magnitudes d (wind, friction) - Actuator limits: u max (typically $\pm$20N for DIP)

—

3.9.1 **Classical SMC** Tuning Procedure

Step 1: Design Sliding Surface (lambda, lambda, k, k)

- Choose convergence rates based on natural frequencies: Rule: 2x natural frequency provides good damping without excessive speed

- Choose sliding gains for critically damped surface: Rule: k i = lambda i/2 gives critically damped sliding variable dynamics

Step 2: Tune Switching Gain K

- Estimate disturbance bound: d = max observed —disturbance— (typically 0.5-1.5 for DIP) - Set initial K = 1.5·d (50percent margin) - Simulate and observe: - If oscillations persist -¿ increase K by 20percent - If chattering excessive -¿ decrease K by 10percent, increase epsilon - Final K typically 1.2-2.0x disturbance bound

Step 3: Tune Boundary Layer epsilon

- Start with epsilon = 0.05 (large boundary layer, low chattering) - Gradually decrease epsilon while monitoring chattering index: - If chattering index ¿ 15 -¿ stop, increase epsilon - Final epsilon typically 0.02-0.05 for DIP (balance accuracy vs chattering)

Step 4: Tune Derivative Gain k d

- Start with k d = 0 (no damping) - Increase k d in steps of 0.5 until overshoot ¡ 5percent - Typical range: k d [1.0, 3.0] - Warning: k d ¿ 5.0 amplifies sensor noise -¿ instability

Expected Performance (after tuning): - Settling time: 2.0-2.5s - Overshoot: 5-8percent - Chattering index: 7-10 - Computation: 18.5 mus

—

3.9.2 **STA-SMC** Tuning Procedure

Step 1: Estimate Disturbance Bound d

Same as **Classical SMC** (typically 0.5-1.5 for DIP)

Step 2: Apply Lyapunov Conditions

- Choose K to dominate disturbances: For d=1.0, epsilon=0.01 -¿ K ¿ 200 Practical choice: K = 250 (25percent margin)

- Choose K to satisfy stability: For K=250, d=1.0 -¿ K ¿ (500) = 22.4 Practical choice: K = 30 (34percent margin)

Step 3: Tune for Performance

- Start with Lyapunov-based values (K=30, K=250) - If convergence too slow -¿ increase K by 20percent - If chattering observed -¿ decrease K by 10percent, increase epsilon - Final gains typically: K [12, 20], K [8, 15] (after PSO optimization)

Step 4: Adjust Sign Function Smoothing epsilon

- Start with epsilon = 0.01 (tight smoothing) - If chattering index ¿ 5 -¿ increase epsilon to 0.02 - STA should achieve chattering index ¡ 3 with epsilon=0.01

Expected Performance (after tuning): - Settling time: 1.8-2.0s - Overshoot: 2-4percent - Chattering index: 1-3 - Computation: 24.2 mus

—

3.9.3 **Adaptive SMC** Tuning Procedure

Step 1: Set Initial Gain K init

Choose K init = 1.2·d (similar to **Classical SMC** switching gain)

Step 2: Tune Adaptation Rate gamma

- Start with gamma = 5.0 (moderate adaptation) - Simulate with large disturbance (e.g., 50percent parameter error) - If tracking error persists -¿ increase gamma by 50percent - If gain K(t) oscillates -¿ decrease gamma by 25percent - Final gamma typically 3.0-7.0

Step 3: Tune Leak Rate beta

- Start with $\beta = 0.1$ (slow decay) - If K(t) grows unbounded -¿ increase β to 0.2 - If K(t) doesn't adapt fast enough -¿ decrease β to 0.05 - Final β typically 0.05-0.15

Step 4: Set Dead-Zone delta

- Choose delta = 2epsilon (twice boundary layer width) - Ensures adaptation stops when on sliding surface - Typical delta = 0.01-0.02

Step 5: Set Gain Bounds

- Lower bound: K min = 0.5·K init (prevent gain collapse) - Upper bound: K max = 5·K init (prevent excessive control effort) - Typical: K min=5.0, K max=50.0

Expected Performance (after tuning): - Settling time: 2.3-2.5s - Overshoot: 4-6percent - Chattering index: 9-11 - Robustness: 15percent model uncertainty tolerance

—

3.9.4 Hybrid Adaptive **STA-SMC** Tuning Procedure

Step 1: Tune STA and Adaptive Controllers Independently

Follow Sections 3.9.2 and 3.9.3 to obtain nominal gains for both modes.

Step 2: Set Switching Threshold σ switch

- Analyze typical sliding variable range during transient response - Choose σ switch at 50-70percent of peak —sigma— during reaching phase - Typical: σ switch = 0.05 (5percent of initial error)

Step 3: Set Hysteresis Margin Delta

- Start with Delta = σ switch/5 (20percent hysteresis band) - If mode chattering observed -¿ increase Delta by 50percent - If mode switches too infrequently -¿ decrease Delta by 25percent - Final Delta typically 0.01-0.02 (10-20percent of σ switch)

Step 4: Verify Bumpless Transfer

- Simulate mode transitions and check control discontinuity: - If Deltau ¿ 0.2·u max -¿ adjust state initialization logic - Target: Deltau ¡ 0.1·u max (bumpless transfer)

Step 5: Test Robustness Across Modes

- Simulate with: - Large initial errors (test STA mode) - Model uncertainty (test Adaptive mode) - Mode transitions (test hysteresis) - Verify no chattering at mode boundaries

Expected Performance (after tuning): - Settling time: 1.9-2.1s - Overshoot: 3-5percent - Chattering index: 4-6 - Robustness: 16percent model uncertainty tolerance

—

3.9.5 Common Tuning Pitfalls

[TABLE - See Markdown version for details]

—

3.9.6 PSO-Based Automated Tuning (Recommended)

Manual tuning can be labor-intensive. PSO optimization (Section 5) automates the process:

Advantages: - Explores parameter space systematically (swarm-based search) - Optimizes multi-objective cost (settling time + overshoot + chattering) - Finds near-optimal gains in 50-100 iterations (10 minutes)

Procedure: - Define parameter bounds (e.g., K [5, 30], epsilon [0.01, 0.1]) - Choose cost function: J = w·t settle + w·overshoot + w·chattering - Run PSO with 20 particles, 50 iterations - Verify performance on validation scenarios (different initial conditions)

Typical Results: - **Classical SMC**: K=15.0, epsilon=0.02, k d=2.0 -¿ 18percent better than manual tuning - STA SMC: K=12.0, K=8.0, epsilon=0.01 -¿ 22percent better performance - Hybrid STA: σ switch=0.05, Delta=0.01 -¿ optimal mode switching

See Section 5 for complete PSO methodology.

—