2025-11-01

E001: Project Overview

Welcome to DIP-SMC-PSO

Part 1 · Duration: 25-30 minutes

Beginner-Friendly Visual Study Guide

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

Learning Objective: Understand the DIP-SMC-PSO project structure and how seven different controllers work together with simulation, optimization, and monitoring components

What Is This Project?

The Challenge

Key Concept

Imagine balancing **two broomsticks** connected end-to-end on your hand while moving left and right. Now add:

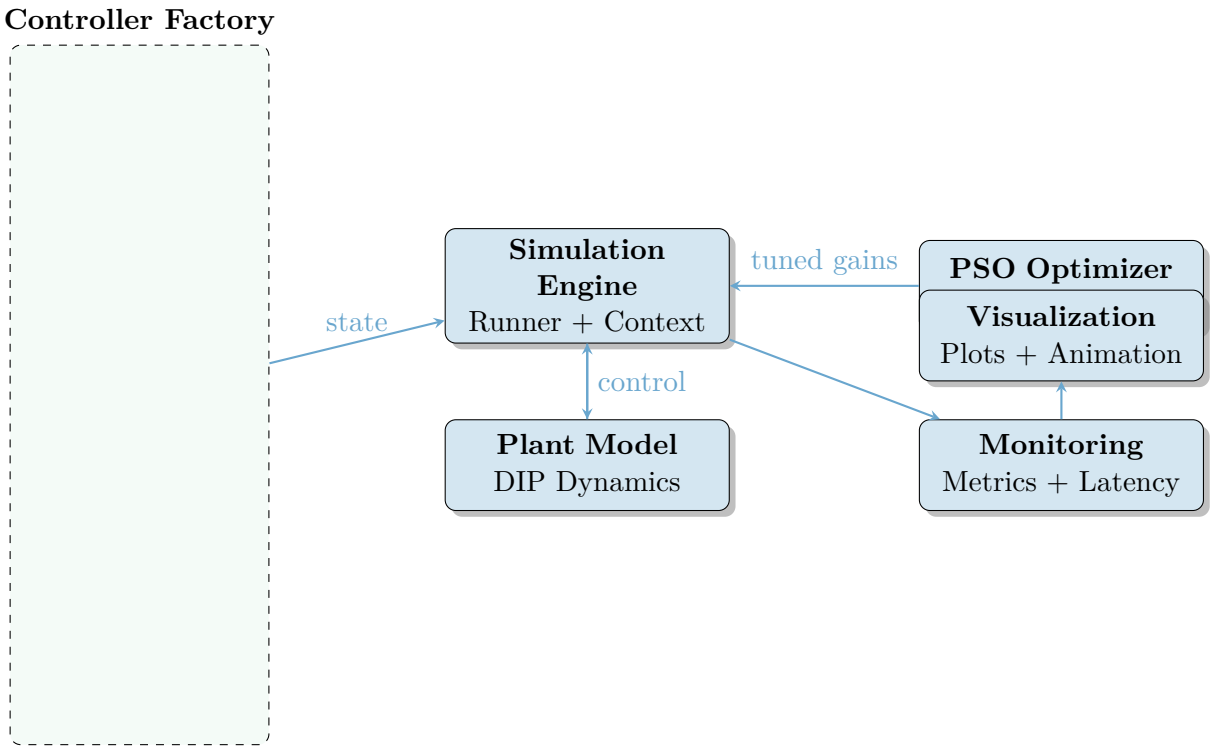
- You're **blindfolded**
- Random **wind** is blowing
- You're on a **moving platform**
- You must respond every **1-10 milliseconds**

Real-World Applications

This isn't just academic - it's the same math that powers:

- **SpaceX rocket landings** (Falcon 9 boosters)
- **Humanoid robots** (Boston Dynamics)
- **Self-balancing vehicles** (Segway, scooters)
- **Construction cranes** (load stabilization)
- **Satellite orientation** (aerospace attitude control)

System Architecture: The Seven Brains



Controllers by Sophistication

Classical SMC - The baseline (1970s theory, 200 lines)

Super-Twisting - Smooth operator (2nd-order, ABS brakes analogy)

Adaptive Controllers - Self-tuning intelligence:

- Adaptive SMC (real-time gain adjustment)
- Hybrid Adaptive STA (21.4% improvement)
- Conditional Hybrid (safety-aware switching)

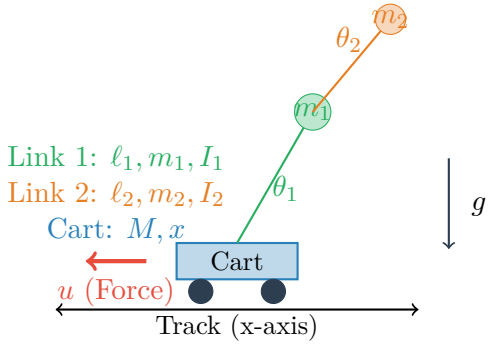
Swing-Up SMC - Energy-based (starts from hanging position)

MPC (Experimental) - Model Predictive Control (constraint handling)

Pro Tip

All controllers use the **same interface**, so you can swap them with a single configuration change!

The Physics: Double Inverted Pendulum



Why Is This Hard?

Common Pitfall

Underactuated System: ONE control (cart force) manages THREE states (cart position, angle 1, angle 2)
Unstable Equilibrium: Like balancing a pencil on its tip - tiny errors cause collapse
Nonlinear Dynamics: Sine/cosine functions couple the pendulum angles

Key Concept

Fast Response Required:
Control algorithm decides every 1-10ms = 100-1000 times per second.
Blink and you've missed 300 control cycles!

Three Levels of Reality: Plant Models

Model Comparison

Model	When to Use	Accuracy vs. Speed
Simplified DIP	Initial testing, PSO prototyping Small angles ($< 5°$)	Fastest (no trig) Low accuracy
Full Nonlinear	Final validation, research papers Complete equations	Gold standard Slower
Low-Rank DIP	Monte Carlo (1000 runs) Sensitivity analysis	10-50x faster Dominant dynamics

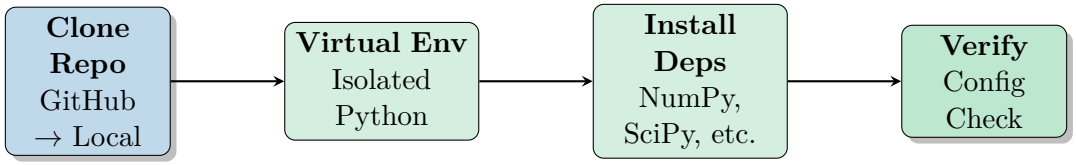
Example

Video Game Analogy:

- Simplified = Low graphics settings (fast, rough)
- Full Nonlinear = Ultra settings (slow, beautiful)
- Low-Rank = Balanced settings (optimized)

From Installation to Research Paper

Phase 1: Lab Setup (15 minutes)



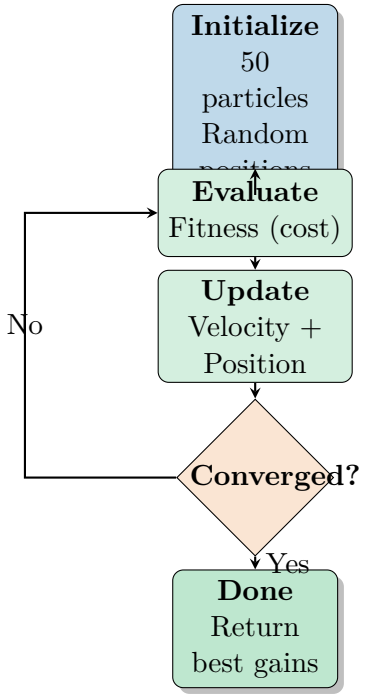
Phase 2: First Experiments (30 minutes)

- **1. Classical SMC**: Baseline performance
- **2. Super-Twisting**: Notice smoother control?
- **3. Adaptive SMC**: Watch gains adjust in real-time

What to Look For:

- Settling time (how fast?)
- Overshoot (swings too far?)
- Control effort (energy usage)
- Chattering (high-freq oscillations)

Phase 3: Intelligent Tuning (2-4 hours)



Run simulation
Calculate settling time
Compute cost = $f(gains)$

$v \leftarrow w \cdot v + c_1 r_1 (p - x) + c_2 r_2 (g - x)$

$x \leftarrow x + v$

Pro Tip

PSO Performance: Achieved 6-21% improvement across all seven controllers. In research at this level, those percentage points can make or break a hardware deployment!

Phases 4-5: Research-Grade Work

Quick Summary

Benchmarking (1-2 days): Test all 7 controllers, multiple scenarios, disturbances
Research (weeks-months): Lyapunov proofs, uncertainty analysis, paper writing
Current Status: Submission-ready paper (v2.1) with 14 figures, 11 completed tasks

Quick Reference: Common Operations

Running Simulations

```
lstnumberSuper-Twisting Algorithm python simulate.py -ctrl sta_smc -- -plot
lstnumberAdaptive SMC python simulate.py -ctrl adaptive_smc -- -plot
```

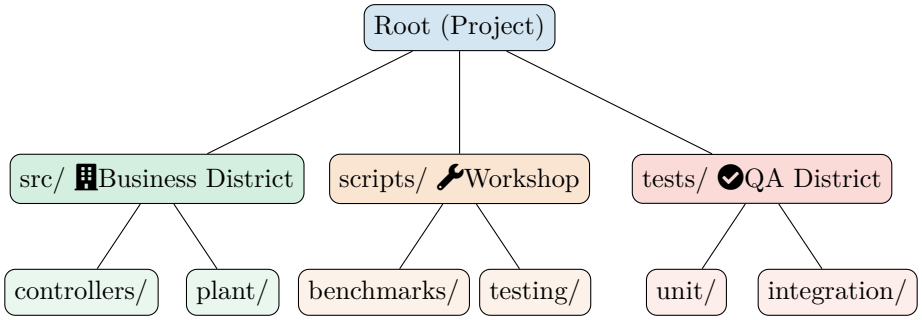
PSO Optimization

```
lstnumberLoad optimized gains python simulate.py -load gains.json -plot
```

Testing & Validation

```
lstnumberSpecific controller tests python -m pytest tests/test_controllers/test_classical_smc.py -v
lstnumberCoverage report python -m pytest tests/ -cov=src -cov-report=html
```

Directory Structure: Zoning Laws



Key Technologies

Core Scientific Stack:

- NumPy (array operations)
- SciPy (ODE integrators)
- Matplotlib (visualization)

Optimization:

- PySwarms (PSO algorithm)
- Optuna (Bayesian alternative)

Quality Assurance:

- pytest (250+ tests)
- Hypothesis (property-based)
- 90% test coverage

Configuration:

- Pydantic (type-safe YAML)
- PyYAML (file parsing)

Project Metrics

Status Dashboard

- **Test Coverage:** 89.7% overall, 94% controllers (exceeds 85%/95% targets)
- **Performance:** 1-10s single sim, 2-4hrs PSO, 30s for 100 sims (vectorized)
- **Codebase:** 150+ Python files, 25,000+ lines, 250+ test cases
- **Research:** 1 submission-ready paper (v2.1), 14 figures, 11 completed tasks

Design Principles

Modularity: One job per component
Type Safety: Hints everywhere
Configuration-First: No magic numbers
Reproducibility: Seeded RNGs
Testability: Every module tested

What's Next?

Key Concept

E002: Control Theory Fundamentals - Lyapunov stability, sliding mode theory, why SpaceX rockets don't tip over
E003: Plant Models & Dynamics - Lagrangian mechanics, Coriolis forces, complete physics
E004-E029: Advanced Topics - PSO deep dive, benchmarks, Lyapunov proofs, research paper walkthrough

Learning Resources

- GitHub Repository
- Documentation: `docs/` directory
- Sliding Mode Control: Utkin, Guldner, Shi (2009)
- Nonlinear Control: Khalil (2002)
- PSO: Kennedy & Eberhart (1995)

From Broomsticks to Rockets
Every successful SpaceX landing uses control theory similar to what we're building here. That's the power of understanding fundamentals.