FIELD STUDY — CONTROL SYSTEMS

PID CONTROLLER TUNING FOR DC MOTOR SPEED CONTROL USING PSO AND ACO

Two swarm-intelligence algorithms — particle swarm and ant colony — pitted against classical Routh–Hurwitz tuning to control the speed of a separately excited DC motor.

PID TUNING | PARTICLE SWARM OPTIMIZATION | ANT COLONY OPTIMIZATION | MATLAB & SIMULINK

■ PLANT: DC MOTOR

■ TOOL: MATLAB/SIMULINK

■ DISTURBANCE: T=10S

01 — THE PROBLEM

A FIXED-GAIN CONTROLLER ISN'T GOOD ENOUGH

The DC motor's speed loop was first closed with a PID controller whose gains were chosen using the conventional **Routh–Hurwitz** criterion — picking values that merely satisfy a stability inequality, not ones tuned to minimize error, overshoot, or settling time.

Kp = 2	Ki = 15	Kd = 0.01	0.2680 p.u.	0.5690 s
PROPORTIONAL	INTEGRAL	DERIVATIVE	OVERSHOOT	SETTLING

** Gains satisfy a stability bound, not a performance target.

** Large overshoot when a load-torque disturbance hits at t = 10s.

** Slow to settle back to the reference speed afterward.

So the tuning problem was reframed as an optimization task — minimize overshoot and settling time directly — and handed to two

02 — METHOD

BORROWING FROM BIRDS AND ANTS

Each algorithm searches the same three-dimensional space of gains — Kp, Ki, and Kd — but explores it the way its namesake swarm does.

JOB: PSO STATUS: RUN

PARTICLE SWARM

Modeled on bird flocking and fish schooling. Each candidate gain-set is a "particle" that remembers its own best result (pbest) and is pulled toward the swarm's best (gbest) at every step.

Population size **10**
Dimensions **3** (Kp, Ki, Kd)
Max iterations **300**
Inertia weight **0.9 → 0.4**
Acceleration C1 **0.7**, C2 **0.5**

JOB: ACO STATUS: RUN

ANT COLONY

Modeled on ant foraging. Each "ant" walks a candidate solution and deposits pheromone in proportion to how good it is — over time the colony's trail converges on the shortest path to the optimum.

Ant population **20**
Evaporation constant **0.4**
Initial pheromone **0.0**
Visibility factor **1.0**
Visibility enhancement **1.0**

03 — RESULTS

THREE TUNING METHODS, ONE PLANT

Each method was run against the same DC motor model in Simulink, with an identical load-torque disturbance at t = 10s.

CONTROLLER GAINS AND STEP RESPONSE, BY TUNING METHOD

METHOD	KP	KI	KD	OVERSHOOT	SETTLING TIME
Conventional (Routh–Hurwitz)	2.00	15.00	0.01	0.2680 p.u.	0.5690 s
PSO-tuned	15.06	31.58	0.68	**0.000 p.u.**	**0.0460 s**
ACO-tuned	4.85	9.30	0.53	**0.000 p.u.**	1.4545 s

Both swarm methods eliminate overshoot entirely; the gap between them shows up in how quickly the speed settles back to reference.

04 — COMPARISON

PSO REACHES THE OPTIMUM FASTER

TABLE 6.1 — PSO VS. ACO, COMPUTATIONAL CHARACTERISTICS

METRIC	PSO	ACO
Iterations to reach objective	**57**	120
Objective function value at 250 iterations	**0.070361**	0.708562
Overshoot	**0.000 p.u.**	**0.000 p.u.**
Settling time	**0.0460 s**	1.4545 s

Both algorithms converge to a near-zero overshoot, but PSO gets there in roughly half the iterations and settles about thirty times faster than the ant colony method — at comparable computational cost.

05 — CONCLUSION

VERDICT: PARTICLE SWARM OPTIMIZATION

Both PSO and ACO reach near-optimal gains with far less trial-and-error than the conventional method, and both hold up cleanly through the load-torque disturbance. But for this plant, particle swarm optimization converges faster, needs fewer iterations, and yields the tighter dynamic response — the better tool for the job.

PID TUNING | PARTICLE SWARM OPTIMIZATION | ANT COLONY OPTIMIZATION | MATLAB & SIMULINK

* END OF REPORT — BEST-PERFORMING METHOD*

PARTICLE SWARM OPTIMIZATION

0.0460 s settling · 0.000 p.u. overshoot