

PID vs. Neural Network Control: MATLAB Lab

Lab manual for students, with notes for instructors

Akin Tatoglu, Ph.D. · October 2026

Overview

In this lab you tune PID controllers by hand, train a neural network controller by trial and error, and stress-test both on simulated robots in MATLAB. Plan on about 3 hours plus the write-up. Parts 1–4 are required; Parts 5 and 6 are optional.

By the end you can: tune a PID and explain each gain; train a small network and read its training curve; compare both controllers on accuracy, speed, motor effort and robustness; and explain why a controller that works on the normal robot can fail on a changed one.

You need: MATLAB R2019b or newer, no toolboxes, and `arai_pid_vs_nn.zip` from the ARAI Lab tools page. Unzip it, open the folder in MATLAB, and run one section at a time with Ctrl+Enter (Cmd+Enter on a Mac). The *Lecture notes* PDF explains the math, and the online playground shows the same robots live.

The building blocks

Every script uses the same five calls. Read this once and the rest of the lab is easy to follow.

```
P      = arai_params('arm');           % pick a robot: 'arm', 'segway', 'twolink', 'drone'
cond   = default_conditions();         % the normal robot (no stress tests)
tr     = simulate_robot(P, make_pid([30 20 4]), cond); % run one episode with a PID
m      = compute_metrics(P, tr);       % overshoot, settling time, errors, jitter
plot_compare(P, {tr}, {'My PID'});     % plot the response and the motor command
```

To use the pretrained network instead of a PID, replace `make_pid([30 20 4])` with `make_nn(pretrained_weights('arm'))`. To switch on a stress test, change a field of `cond`:

```
cond.payload = 2.5; % heavier load (use 1.6 for the drone)
cond.friction = true; % sticky joint, rough ground, stiff joints or strong drag
cond.push     = true; % a push or wind gust during the run
cond.noise    = true; % noisy sensors
```

Pre-lab questions

Answer before the session.

1. The arm holds a 1 kg load 0.5 m from the joint. What steady torque holds it at 90° from hanging down? At 100° ? ($g = 9.81 \text{ m/s}^2$.)
2. With $K_i = 0$ and $K_p = 30 \text{ N}\cdot\text{m/rad}$, estimate the steady error near 100° from your answer to question 1. Why is it not zero?
3. The PID uses $-K_d\dot{y}$ instead of $K_d\dot{e}$. What happens to the motor command at the instant the target jumps, in each case?
4. A balancing robot wants to drive forward from standing still. Which way must its wheels move first, and why?
5. The arm's network sees $\sin \theta$ as an input. Why might that help it hold the arm against gravity?

Part 1: Tune a PID on the arm joint

The arm starts hanging down, moves to 60° , then to 100° at $t = 2.5 \text{ s}$. The motor is limited to $\pm 12 \text{ N}\cdot\text{m}$.

Run `lab1_pid_arm.m`. Its core is a loop over gain sets:

```
G = [30 20 4; 30 0 4; 30 20 0; 80 20 4]; % rows: [Kp Ki Kd]
for k = 1:size(G,1)
    tr{k} = simulate_robot(P, make_pid(G(k,:)), cond);
    M(k) = compute_metrics(P, tr{k});
end
print_metrics(P, names, M);
```

Then change `myGains` until the arm settles in under 0.5 s with less than 5% overshoot.

Kp	Ki	Kd	Overshoot (%)	Settling (s)	Final error (°)	RMS (°)	Jitter (%/s)
30	20	4					
30	0	4					
30	20	0					
80	20	4					

My gains:

Questions. (1) Which term removes the steady error, and how big is the error without it? (2) Describe from the plot what changes when $K_d = 0$. (3) What did $K_p = 80$ improve, and what did it cost?

Part 2: Train a neural network

The network has 71 weights and starts random. Each training round scores 12 pairs of slightly changed copies on practice runs with random targets, always on the normal arm, and moves the weights toward the better copies.

```
[pMine, loss] = train_es('arm', 300, 1, struct('pop', 12, 'eps', 2));
semilogy(loss); xlabel('Training round'); ylabel('Training error');
trMine = simulate_robot(P, make_nn(pMine), cond);
trPre = simulate_robot(P, make_nn(pretrained_weights('arm')), cond);
```

Run lab2_train_nn.m with 50 rounds, then with 300. Your network is saved to my_arm_network.mat.

Controller	Rounds	Train time (s)	Overshoot (%)	Settling (s)	Final error (°)	Jitter (%/s)
Default PID	—	—				
Pretrained network	1,200	—				
My network	50					
My network	300					

Questions. (1) Describe the training curve. Why does it jump between rounds? (2) How does the 50-round network fail? (3) Which controller switches its motor command more, and why might training produce that? (4) In episode_cost.m the effort penalty is $0.002 \cdot u^2$. Predict what raising it to 0.05 would do.

Part 3: Stress tests

None of these conditions appeared in training. Run lab3_stress_tests.m with robotKey = 'arm'; it loops over the tests:

```
for k = 1:5
    cond = default_conditions();
    if k == 2, cond.payload = 2.5; end
    if k == 3, cond.friction = true; end
    if k == 4, cond.push = true; end
    if k == 5, cond.noise = true; end
    trP = simulate_robot(P, make_pid(P.pid), cond, 3);
    trN = simulate_robot(P, make_nn(pretrained_weights('arm')), cond, 3);
    print_metrics(P, {'PID', 'Network'}, [compute_metrics(P, trP), compute_metrics(P, trN)]);
end
```

Test	Final err PID (°)	Final err NN (°)	RMS PID (°)	RMS NN (°)	Jitter PID	Jitter NN
Normal						
Payload ×2.5						
Friction						
Push						
Noisy sensor						

Questions. (1) Which test hurts the network most, and why, given what it can and cannot see? (2) Which controller recovers better from the push? (3) Why does motor jitter matter on real hardware? (4) Suggest a training change that would help with the heavy payload, and its cost.

Part 4: Self-balancing robot

The robot starts tilted 3° , drives to 1 m, and returns to 0 at $t = 4$ s. It falls past 60° . A fixed outer loop turns position error into a target lean; you tune the tilt PID. Run `lab4_segway.m`.

```
P = arai_params('segway');
tr = simulate_robot(P, make_pid([50 10 0]), default_conditions());
if tr.fell, fprintf('Fell over at %.2f s\n', tr.fellAt); end
plot(tr.t, tr.x(:,2)*180/pi); ylabel('Tilt (deg)');
```

Kp	Ki	Kd	Fell over? (time)	Settling (s)	Final error (m)	Jitter (%/s)
50	10	18				
50	10	0				
20	10	18				
Network	—	—				

Questions. (1) Why does it fall when $K_d = 0$? (2) Which way does it lean in the first half second, and why? (3) The network moves sooner but stops short of the target. Explain using the integral term. (4) Run Part 3 with `robotKey = 'segway'`: which part of the PID design fails on rough ground?

Part 5 (optional): Two-link manipulator

Run `lab5_manipulator.m`. It compares PID, PID with $K_i = 0$ and the network, and plots the gripper paths. Compare motor jitter, explain why removing K_i leaves the gripper short of point B, and say which path you would want near people.

Part 6 (optional): Quadrotor drone

Run `lab6_drone.m`. The drone hovers at 1 m, then climbs to 2 m and flies 1 m right. Its altitude PID gets no hover thrust for free.

```
P = arai_params('drone');
tr = simulate_robot(P, make_pid([30 0 8]), default_conditions()); % Ki = 0
fprintf('Final altitude: %.2f m (target 2.00 m)\n', tr.y(end));
```

Explain why the drone hovers low with $K_i = 0$, using the force balance $T = mg$ at hover.

Deliverables

- Pre-lab answers, handed in at the start of the session.
- Tables for Parts 1–4, filled in, with the plots each part asks for.
- Answers to every question block.
- A one-page discussion: *a company asks you to choose the controller for a warehouse arm carrying boxes from 0.5 to 8 kg near people, 16 hours a day. Recommend PID, a network or a combination, support it with at least three numbers from your tables, and say how you would test it.*
- An appendix with any MATLAB code you changed.

Grading rubric

Component	Points	Full credit means
Pre-lab	10	Correct torque estimates with work; clear reasoning on derivative kick and balance
Part 1	15	Complete table; tuned gains meet the 0.5 s and 5% targets
Part 2	20	Complete table; training curve described; effort prediction reasoned
Part 3	20	Failures explained by what the network can and cannot see
Part 4	15	Fall and lean-before-driving explained physically
Discussion	15	A clear recommendation backed by the student's own numbers and a test plan
Report quality	5	Labeled plots with units, readable tables, concise writing
Total	100	Parts 5 and 6 can add up to 5 bonus points each

Reference values and the pre-lab answer key are in the separate instructor key.

© 2026 Akin Tatoglu, ARAI Lab, University of Hartford. Free for teaching and learning with attribution.