PID Tuning Lab

Name:

Date:

By the end of this lab, you will be able to:

- Understand how each part of PID control (P, I, and D) affects response to change
- Tune a PID controller step-by-step using the P → PI → PID method
- Read performance data and make tuning decisions
- Compare different tuning methods and understand when to use each one

📖 KEY VOCABULARY

Gain: A multiplier value (like K_p, K_i, or K_d) that determines how strongly the controller responds to an input. Higher gain = stronger response.

Term: One component of the PID equation. The P-term, I-term, and D-term are calculated separately, then added together to determine the total control output.

STAGE 1: Proportional Control Only

Proportional Control (P-Control): *A control method that makes corrections based on how big the current error is. The bigger the error, the stronger the correction:*

Gain: *A multiplier value (like K_p, K_i, or K_d) that determines how strongly the controller responds to an input. Higher gain = stronger response.*

Term: *One component of the PID equation. The P-term, I-term, and D-term are calculated separately, then added together to determine the total control output.*

$$correction = K_p \times error$$

Think of proportional gain (K_p) like the sensitivity setting on a steering wheel. A high K_p makes the robot react strongly to errors, while a low K_p makes it react gently. But P-control alone has a problem on curves.

> 💡 **SIMULATOR TIP:** Hover your mouse over different parts of the PID equation at the top of the simulator. Watch how the equation highlights show you which term (P, I, or D) is contributing to the robot's current steering command.

Steady-State Error (Drift Offset): *When the robot consistently pulls to one side even while trying to correct itself. With P-only control, the robot needs some error to keep steering through curves, so it can't eliminate this drift.*

Instructions

1. Open the PID simulator and dismiss the welcome screen
2. Set gains to: K_p = 0.5, K_i = 0.0, Kd = 0.0
3. Run your first lap and write down the results in the table below
4. **Find the highest K_p that gives good performance without too much wobbling**

5. Run 5 strategic laps with different K_p values. Suggested starting points: 0.5, 1.0, 2.0, 3.0, 4.0, then fine-tune around your best value
6. Keep K_i = 0.0 and Kd = 0.0 for all Stage 1 laps

Data Collection: Proportional-Only Laps (target score: 40-60)

Lap	Kp	Ki	Kd	Score	Avg Error	Drift Offset	Roughness	Notes/Observations
1	0.5	0.0	0.0					
2	1.0	0.0	0.0					
3	2.0	0.0	0.0					
4	3.0	0.0	0.0					
5	4.0	0.0	0.0					
6		0.0	0.0					
7		0.0	0.0					
8		0.0	0.0					
9		0.0	0.0					
10		0.0	0.0					
11		0.0	0.0					
12		0.0	0.0					
13		0.0	0.0					
14		0.0	0.0					
15		0.0	0.0					

💡 **SIMULATOR TIP:** Click on a lap number in your lap history to view that lap's path on the track. Use the 'SORT BY' dropdown to organize your laps by Score, Avg Error, or other metrics to quickly find your best runs. To compare multiple laps at once, hold SHIFT and click multiple lap numbers - their paths will appear in different colors!

Stage 1 Analysis Questions

Q1: What was your best K_p value? What score did it get?

Best K_p: ___________ Score: ___________

Q2: What happened to your robot's behavior as you increased K_p? Did the drift offset get better, worse, or stay the same?

Q3: Did you see any wobbling (back-and-forth motion)? If yes, around what K_p value did it start getting bad?

Q4: [PREDICTION] - If you could magically eliminate the drift offset without changing K_p, what would happen to your score?

- ☐ Score would improve a lot (>15 points)
- ☐ Score would improve some (5-15 points)
- ☐ Score would stay about the same
- ☐ Score would get worse

> 💡 **KEY INSIGHT:** Notice that P-only control CANNOT eliminate drift offset on curves. The robot needs some error to maintain steering through turns, creating steady-state error. This limitation is WHY we need I-control in Stage 2!

STAGE 2: Adding Integral Control

Integral Control (I-Control): *A control method that looks at all the past errors and adds them up. It makes corrections based on the total amount of error over time:*

$$correction = K_i \times \Sigma(Error \times \Delta Time)$$

Integral control is like having a memory of all past mistakes. If the robot keeps drifting left, integral control remembers this pattern and applies a constant correction to the right to balance it out.

Integral Windup: *A problem that happens when the integral term gets too big, causing the system to overshoot and become unstable. This occurs when K_i is set too high.*

Instructions

7. Lock in your best K_p from Stage 1: K_p = ___________
8. Keep this K_p value the same for all Stage 2 laps
9. **Find the K_i that reduces drift offset and gives the best overall score**
10. Try these K_i values: 0.1, 0.2, 0.4, 0.6, 0.8, plus 3 more you choose. Start small!
11. Keep K_d = 0.0 for all Stage 2 laps

> ⚠ **IMPORTANT - START WITH SMALL I VALUES:** I-control is powerful! Start with small values (0.1-0.5). If you notice the robot becoming unstable or your score decreasing despite reduced drift, you may be experiencing 'integral windup' - try reducing K_i. Most successful students find optimal values between 0.2 and 0.8.

Data Collection: PI Control Laps (target score: 50-65)

Lap	Kp	Ki	Kd	Score	Avg Error	Drift Offset	Roughness	Notes/Observations
16		0.1	0.0					
17		0.2	0.0					
18		0.4	0.0					
19		0.6	0.0					
20		0.8	0.0					
21			0.0					
22			0.0					
23			0.0					
24			0.0					
25			0.0					
26			0.0					
27			0.0					
28			0.0					
29			0.0					
30			0.0					

💡 **SIMULATOR TIP:** The error history graph at the bottom shows how far off-center the robot was over time. When viewing a saved lap, this graph shows that lap's error pattern. When comparing multiple laps (using SHIFT+CLICK), each lap's error appears in a different color so you can see which gain settings kept the robot closer to the line!

Stage 2 Analysis Questions

Q1: What was your best K_i value? What score did it get?

Best K_i: ___________ Best Score: ___________

Q2: Compare your best PI lap to your best P-only lap. How much did your score improve?

Stage 1 best: ___________
Stage 2 best: ___________
Improvement: ___________ points

Q3: What happened to your drift offset when you added integral control? Use specific numbers from your data.

Q4: [REFLECTION ON YOUR PREDICTION] Look back at Stage 1, Q4. You predicted what would happen if drift offset was eliminated. Were you right? Why or why not?

💡 **KEY INSIGHT - THE I-GAIN PARADOX:** You may notice something surprising: Higher I gains reduce drift offset, but can actually LOWER your overall score! This happens because too much I-control causes 'integral windup' - the accumulated error becomes so large that the system overshoots and becomes unstable. This is a real engineering tradeoff: drift correction vs. system stability.

🎯 **SUCCESS METRIC:** Most students see a 50-70% reduction in drift offset when adding I-control. This demonstrates the PRIMARY PURPOSE of integral control - eliminating steady-state error that P-control alone cannot fix!

STAGE 3: Adding Derivative Control

Derivative Control (D-Control): *A control method that looks at how fast the error is changing and tries to predict future errors. It provides corrections based on the rate of change:*

$$correction = K_d \times (\Delta Error / \Delta Time)$$

Think of derivative control like a fortune teller that can see where the error is heading. If the error is changing quickly, derivative control applies a counter-force to slow down that change and prevent overshooting.

Damping: *The process of reducing oscillations (back-and-forth wobbling) in a system. Derivative control acts like shock absorbers on a car, smoothing out rapid changes.*

Instructions

12. Lock in your best gains from Stage 2: K_p = __________ K_i = __________
13. Keep K_p and K_i the same for all Stage 3 laps
14. **Find the Kd that reduces wobbling and gives the best overall score**
15. Try these Kd values: 0.01, 0.02, 0.05, 0.1, 0.2, plus 3 more you choose. Start small!

WARNING: Derivative control is very sensitive! Too much Kd can make the system unstable and noisy!

Data Collection: Full PID Control Laps (target score: 80-130)

Lap	Kp	Ki	Kd	Score	Avg Error	Drift Offset	Roughness	Notes/Observations
31			0.01					
32			0.02					
33			0.05					
34			0.1					
35			0.2					
36								
37								
38								
39								
40								
41								
42								
43								
44								

Stage 3 Analysis Questions

Q1: What was your best full PID combination? What score did it get?

K_p = ______ K_i = ______ K_d = ______ Score: ___________

Q2: Compare your smoothness scores from Stage 2 (PI) versus Stage 3 (PID). How did adding derivative control affect smoothness?

Q3: What happened when you tried higher KDvalues? Did you reach a point where performance got worse?

STAGE 4: Ziegler-Nichols Tuning Method

Now you'll learn a mathematical method for tuning PID controllers that engineers have used since 1942. Instead of guessing values, this method uses a specific procedure to calculate the gains.

Ziegler-Nichols (Z-N) Method: *A systematic way to tune PID controllers by first finding the point where the system just starts to oscillate, then using mathematical formulas to calculate the optimal gains.*

Ultimate Gain (Ku): *The lowest value of K_p (with K_i=0, K_e=0) that causes the system to oscillate continuously at a steady rate. This is the 'edge of stability.'*

Ultimate Period (Pu): *The time it takes for one complete oscillation cycle when the system is operating at the ultimate gain. Measured from peak to peak.*

> 📖 **HISTORICAL NOTE:** Z-N was developed in 1942 for slow industrial processes (temperature control, chemical reactors) with response times of minutes to hours. Our robot responds in milliseconds, a very different system! We'll test how well Z-N works for fast systems.

Step 1: Find Your Ultimate Gain (Ku)

GOAL: Find the LOWEST P value where the robot wobbles back and forth steadily.

1. Set Ki = 0.0 and Kd = 0.0 (this is P-only control).
2. Start with P = 2.0. Click "Start Single Lap" to begin.
3. Watch the robot's path and the error graph at the bottom. Does it wobble back and forth in a regular pattern, like a wave? If not, stop the lap, raise P a bit (try +0.5 each time), and start again. Keep going until you see steady wobbles that don't grow or shrink.
4. Once you spot steady wobbles, pause the simulation right away. Look at the error graph; it's like a wavy line showing the robot's mistakes over time. Measure Pu looking for a yellow box to the left of gain. Record this P as Ku and the oscillation time as Pu.

> 👀 **WHAT TO LOOK FOR:** Use SHIFT+CLICK to compare multiple laps. The Ku lap should show a clear wave pattern. Wobbles should be regular and consistent - not random, not growing, not shrinking. Think of a sine wave!

Ultimate Gain: Ku = ________ **Ultimate Period: Pu = ________ seconds**

Common Questions & Problems

Q: "Every P value I try shows oscillations!"

A: Look for SUSTAINED wobbling, not just any wobble. At low P, the robot drifts. At Ku, it actively wobbles back and forth in a regular pattern. Choose the LOWEST P where you see this steady wobbling.

Q: "My Ku seems really low compared to my Stage 1 optimal P."

A: That's EXPECTED! Ku is the edge of stability (where oscillations start). Your optimal P from Stage 1 (probably 3-4) was higher because some oscillation was acceptable for better performance. Z-N is conservative - it starts from the edge and backs off for safety.

Q: "Should I pick the P value that gives the best score?"

A: NO! Ku is NOT about score. It's about finding the oscillation point. Pick the LOWEST P with steady wobbling, regardless of score. The Z-N formulas will handle the rest.

Step 2: Calculate Z-N Gains

Use these formulas to calculate your Ziegler-Nichols gains:

Controller Type	K_p	K_i	K_d
P-only	0.5 × Ku	0	0
PI	0.45 × Ku	$(1.2 \times K_p)$ / Pu	0
Full PID	0.6 × Ku	$(2.0 \times K_p)$ / Pu	$(K_p \times$ Pu) / 8

Calculate your Z-N gains:

Z-N P-only:

K_p = 0.5 × __________ = __________

Z-N PI:

K_p = 0.45 × __________ = __________

K_i = (1.2 × __________) ÷ __________ = __________

Z-N PID:

K_p = 0.6 × __________ = __________

K_i = (2.0 × __________) ÷ __________ = __________

K_d = (__________ × __________) ÷ 8 = __________

Step 3: Predict Before Testing

[PREDICTION] Based on what you learned in Stages 1-3, do you think the Z-N method will give better or worse results than your manual tuning? Why?

Step 4: Test Your Z-N Gains

Test each of the three Z-N configurations and record the results:

Z-N Method	Kp	Ki	Kd	Score	Avg Error	Drift Offset	Roughness	Notes/Observations
Z-N P-only		0.0	0.0					
Z-N PI			0.0					
Z-N Full PID								

Comparison: Z-N vs. Manual Tuning

Compare your Z-N results to your best manual results from Stages 1-3:

Configuration	Manual Score	Z-N Score	Which Was Better?
P-only			
PI			
Full PID			

Z-N Analysis Questions

Q1: Which method gave better results overall - your manual tuning or the Z-N method? Why do you think this happened?

Q2: The Z-N method was created in 1942 for slow industrial processes (like temperature control). Our robot simulator is much faster. How might this explain any differences between Z-N and your manual results?

Q3: If you were working as an engineer tuning a real industrial system, would you: a) Use only Z-N, b) Use only manual tuning, or c) Use Z-N as a starting point, then manually fine-tune? Explain your choice.

Engineering Reflection

Q1: Which control term (P, I, or D) had the biggest impact on your system's performance? Explain why.

Q2: If you had to explain PID control to a friend using everyday comparisons, what would you say?

Proportional (P):

Integral (I):

Derivative (D):

Q3: Based on your experiments, what advice would you give to someone just starting to tune a PID controller?

Q4: Would you recommend your teacher uses this activity again? Explain why or why not.

PID control is everywhere in modern life - from the cruise control in cars to the temperature control in buildings to autopilot systems in aircraft. Through this lab, you've experienced the same systematic approach that engineers use to tune these controllers.

Good control system design requires both understanding the theory and testing in the real world. The 'best' gains depend on specific system and what it needs to do.