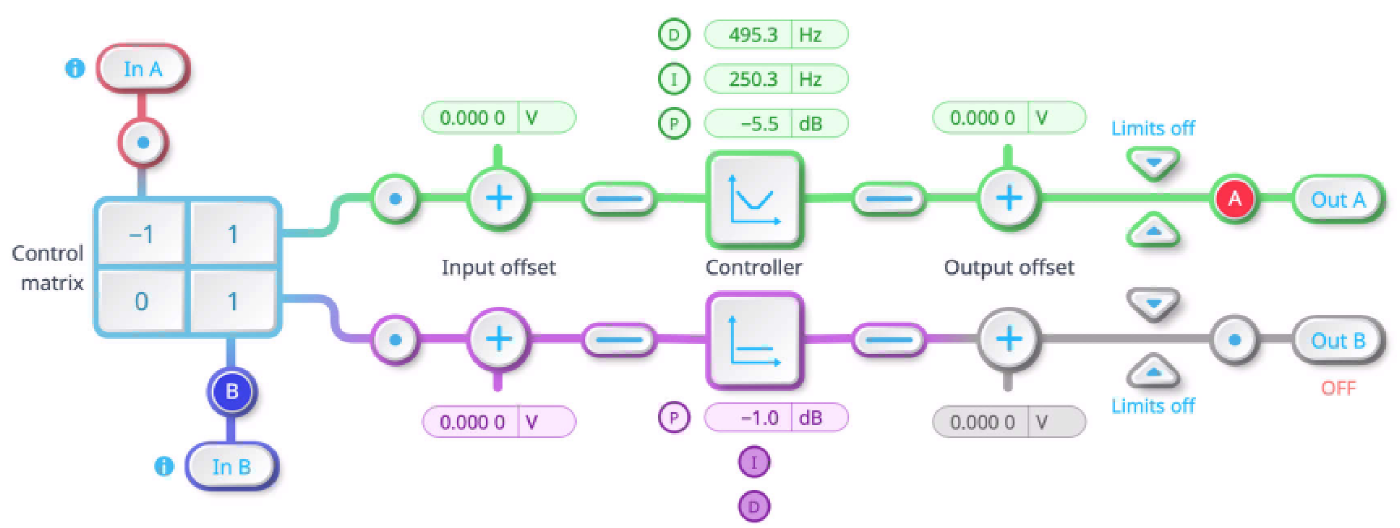
PID Controller tuning with the Ziegler-Nichols method

A quick start guide to tuning your Moku PID Controller

Ziegler-Nichols tuning rules can serve as a useful starting point for tuning, especially when plant dynamics are unknown. In this guide, you will learn how to implement the step response method and self-oscillation method using real-time, interactive Bode plots to design the control system's frequency response. By watching the time-domain response on the Oscilloscope embedded in the Moku PID Controller, you can gain a better understanding of how changes in **P**, **I**, and **D** parameters affect your system under test, leading to easier, more effective tuning.

PID gain profiles



P (Proportional): Reacts to present error.
Too high? You'll see overshoot & oscillations. Too low? You'll get a slow response.

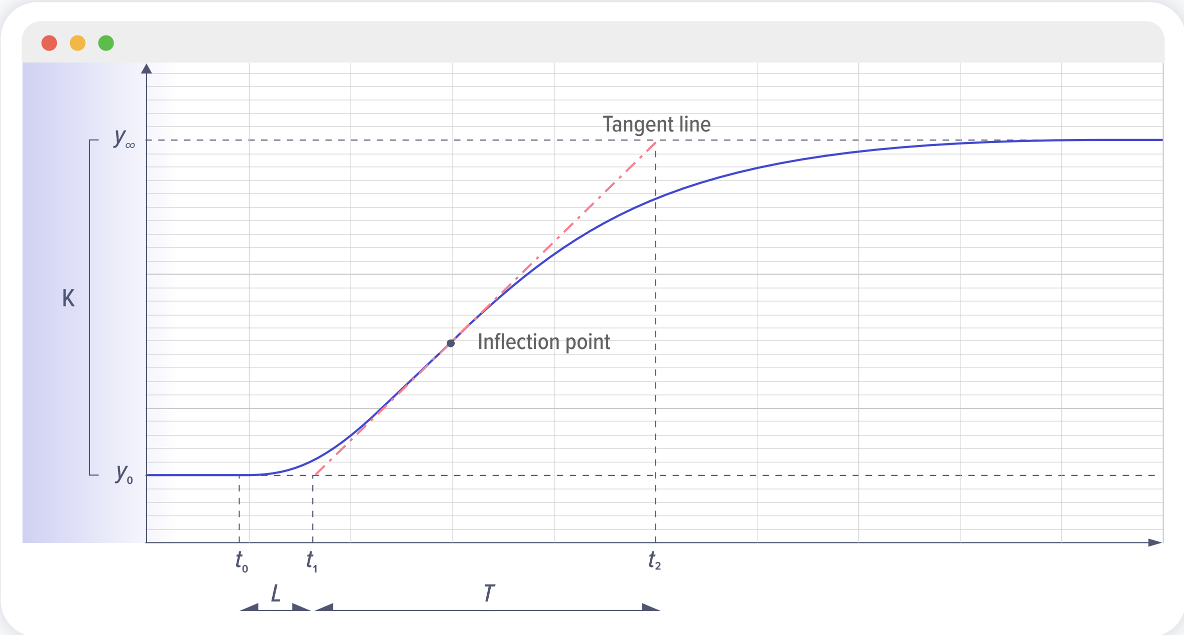
I (Integral): Reacts to past errors.
Too high? You'll see overshoot & instability. Too low? You'll get a slow correction of steady-state error.

D (Derivative): Reacts to future trends.
Too high? Your system will face noise sensitivity. Too low? You'll get a sluggish response.

Ziegler-Nichols tuning

Unit step response method

1. Apply a unit step input to the open-loop system and observe the response using the Moku PID Controller.
2. Record the process reaction curve, or output response over time using the embedded Oscilloscope.
3. Identify the **delay time**, **L**, and the **time constant**, **T**, from the curve.



Tip: Find *L* by drawing a straight line tangent to the inflection point of your process reaction curve. *L* is the time between t_0, the time at which your process reaction begins, and t_1, the point at which the tangent line crosses the steady-state amplitude. Find *T* by determining the difference between t_1, where the tangent line through the inflection point intercepts y_0, and t_2, where the tangent line intercepts the steady-state gain, y_∞. *K*, the system gain, is found by recording the steady state response and input response $\left(\frac{y_\infty - y_0}{u_\infty - u_0}\right)$, where $u_\infty - u_0$ is the amplitude of the step function.

4. Use the table below to set PID values.

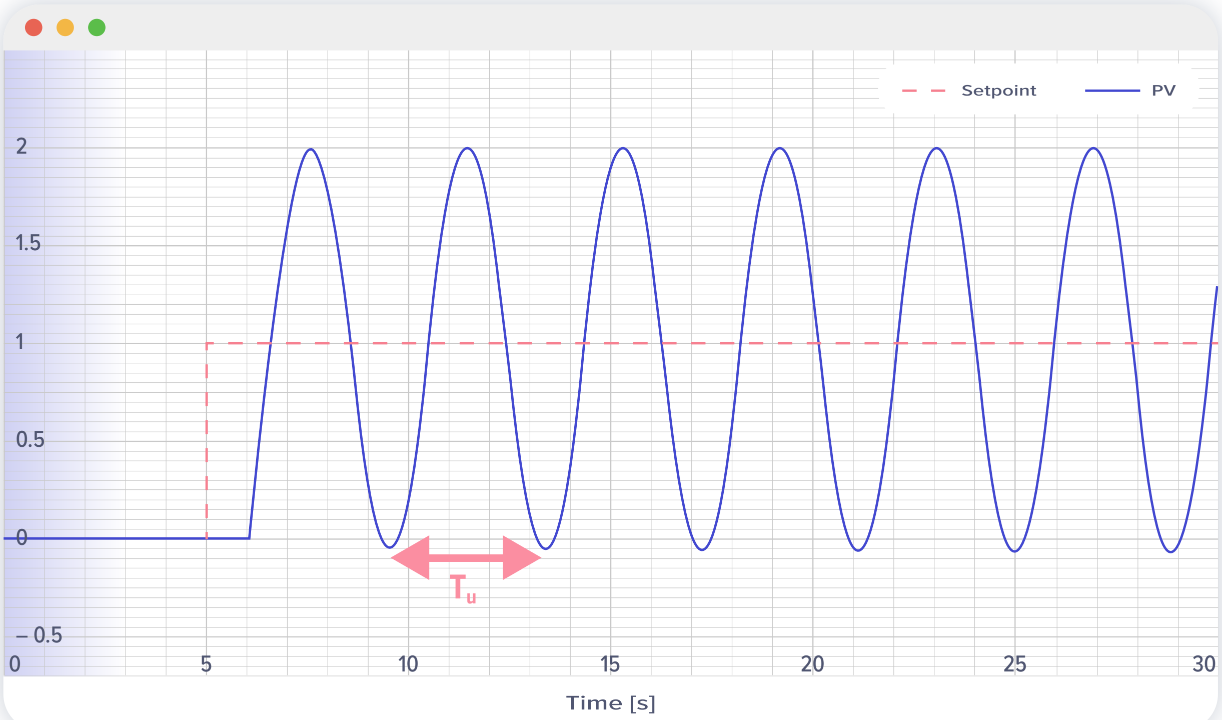
Type	K_P	T_I	T_D
P	T/KL	–	–
PI	0.9 * (T/KL)	3*L	–
PID	1.2 * (T/KL)	2*L	L/2

[1], [2]

Self-oscillation method

Note: This method can force your plant into oscillation, which could be dangerous. Use this method with caution.

1. Disable **I** and **D** and increase **P** until sustained oscillations occur.



2. Record this value, the **Ultimate Gain (K_u)**, or point of instability, and **Ultimate Period (T_u)**, or oscillation period.
3. Use the table below to set PID values.

Type	K_P	T_I	T_D
P	0.50 * K_u	–	–
PI	0.45 * K_u	T_u/1.2	–
PID	0.60 * K_u	T_u/2	T_u/8

[1]

Tuning adjustments

To further tune your controller, use this table to help guide your manual tuning adjustments.
Increasing each gain profile will lead to a different effect on various system responses.

Parameter increase	Rise time	Overshoot	Settling time	Steady-state error
K_P	Decreases	Increases	Slight change	Decreases
K_I	Decreases	Increases	Increases	Eliminates
K_D	Slight change	Decreases	Decreases	Slight change

[3]

References

[1] Y. Mo, "EE3011 Lecture 9," Yilin Mo's Website. [Online]. Available: https://yilinmo.github.io/EE3011/Lec9.html.

[2] "Ziegler-Nichols method for tuning PID controllers," Picuino. [Online]. Available: https://www.picuino.com/en/control-ziegler-nichols.html.

[3] F. A. Salim and N. Y. Park, "PID Controller," Cornell University ECE4760 Final Projects, 2012. [Online]. Available: https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2012/fas57_nyp7/Site/pidcontroller.html.

Ready to test?

Explore configurations →

Have questions?

Speak with an engineer →