SMARTDAMP: Simple, Model-based Automatic Tuning with Robust Damping for PID Position Control of Electric Motors

Ben Caunt

Abstract

In robotic systems Proportional Integral Derivative (PID) control [1] is an algorithm used to move systems powered by electric motors to a desired position. The process for implementing this algorithm involves tuning the three gains: the Proportional, Integral, and Derivative terms. Correct tuning of these gains is necessary for the system to reach the desired position with the desired response characteristics [2]. The Ziegler-Nichols (ZN) procedure is one of the most common methods for tuning PID Controllers. ZN is a heuristic approach which requires the engineer to increase the proportional gain until the system reaches a steady oscillation. While easy to accomplish, ZN tuning can damage the system during this oscillation if hardstops or current limits are exceeded. In addition, ZN often produces aggressive responses [3] that may not be effective in day-to-day operations but may also degrade the lifespan of this system. This paper introduces the derivation for the SMARTDAMP ("Simple, Model-based Automatic Tuning with Robust Damping") algorithm, which provides a method to derive optimal PID coefficients safely. We used an android-based, four-motor holonomic mobile robot to compare the ZN procedure vs. SMARTDAMP. The result of this process was that we collected sufficient evidence to conclude that the proposed SMARTDAMP algorithm outperforms ZN for motor-based systems in terms of disturbance rejection and trajectory tracking error.

1. Introduction

Electric Motors are used in systems all across the world. While technological advances are made in many connecting engineering disciplines, the main idea for controlling motor positions has stayed roughly the same for decades. The procedure begins with a sensor, such as an encoder, to measure the system's position, then using a microcontroller implementation of a PID Controller to compute the motor controller's output. This algorithm is then updated hundreds if not thousands of times per second to ensure the system arrives at the desired output state. The Da Vinci Surgical System [4] and other life-saving medical equipment use PID Control to translate the surgeon's inputs into precise robotic motion. In computer-numerical-control (CNC), the same PID Control approach remains dominant in controlling systems such as mills and lathes to manufacture parts within a tolerance of less than $\pm\ 0.0001$ inches [5]. The core principles behind PID control are now nearly 100 years old [6], with the critical difference being that most PID controllers today are implemented in software rather than with analog circuits. The most critical factor in the success of these systems is how well their PID coefficients are tuned.

2. Background

In robotics, we often want to move our system not only to a desired position but also to regulate our velocity and acceleration to improve consistency and reduce long-term wear and tear on the system. A common approach is using a trapezoidal motion profile [7] to generate the trajectory reference between the initial and desired positions while respecting the previously mentioned constraints.

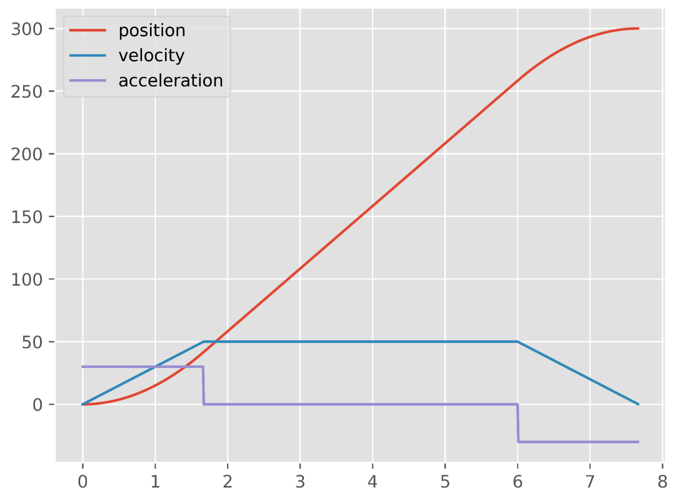


Figure 1: Trapezoidal Motion Profile Trajectory traveling 300 inches with a max velocity of 50 in/s and max acceleration of 30 in/s^2

In specific applications, it is possible to use motion profiles without the use of feedback control. Feedforward control instead uses a model to approximate the optimal control input. While feedforward is often less robust to external and unmodeled disturbances such as mechanical failures, it can be immune to sensor noise or outright sensor failures [8].

One such Feedforward model is the simple motor feedforward model [9]:

$$u = K_v\dot{x} + K_a\ddot{x} \qquad (1.1)$$

Where u is the power sent to the motor, $\dot{x}$ is the desired velocity of the system, $\ddot{x}$ is the desired acceleration of the system, and K_v *and* K_a are tuned constants. Manually tuning this model to acceptable precision is relatively straightforward. During operation of the robot, the reference velocity and accelerations are substituted into $\dot{x}$ and $\ddot{x}$ the feedforward equation 1.1. Increasing K_a will eliminate phase-lag.

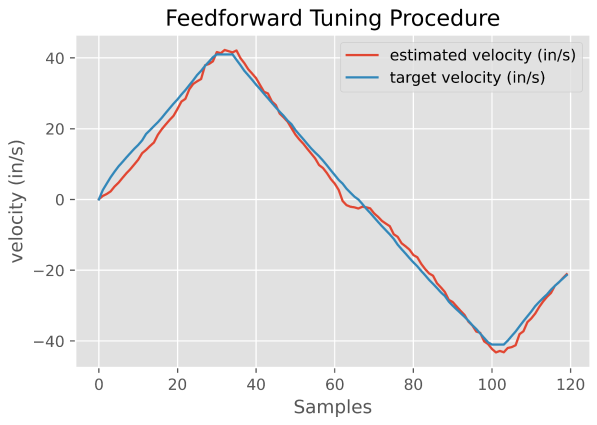


Figure 2: Feedforward Tuning Procedure ran our Mecanum Drive Robot, kV = 0.0132 and kA = 0.003.

Note: Oftentimes systems will implement a friction feedforward term (1.1.1). This works well and you should use it. For the purposes of this work we ignore this term due to transfer functions requiring a linear differential equation.

$$u = K_v \dot{x} + K_a \ddot{x} + kStatic \cdot sign(\dot{x}) \qquad (1.1.1)$$

Why Feedback?

As we can see, the feedforward accurately tracks the velocity reference from the motion profile to see how close it gets to the end position.

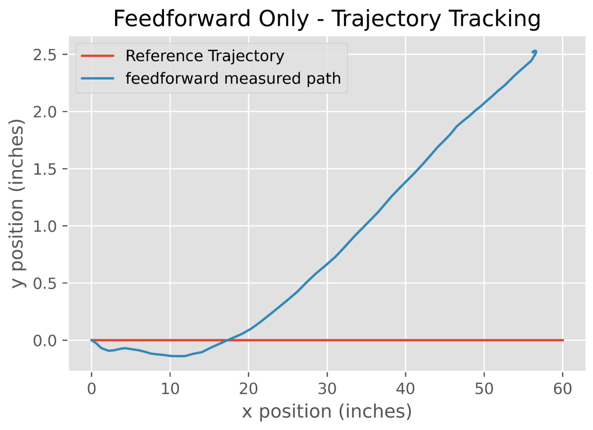


Figure 3: Position of Feedforward Only Trajectory Tracking Performance

Since the feedforward-only approach cannot correct disturbances such as friction or debris in the testing area, it cannot track the reference trajectory accurately.

The solution to this problem is to add a feedback controller like PID into the loop. A PID controller forces the error to be much smaller as it actively compensates for the disturbances by changing the motor output based on the system's estimated state. The problem now is with the optimal tuning of the PID Controller.

Ziegler-Nichols PID Tuning

While it is typical for manual or heuristic tuning methods such as the Ziegler-Nichols procedure to be used, the Ziegler-Nichols procedure is often ineffective at producing optimal gains and conducting the tuning procedure requires increasing the proportional gain until the system is marginally stable in a continual oscillation. Doing this poses the risk of making the system go unstable, which for some systems can cause damage and be expensive to repair.

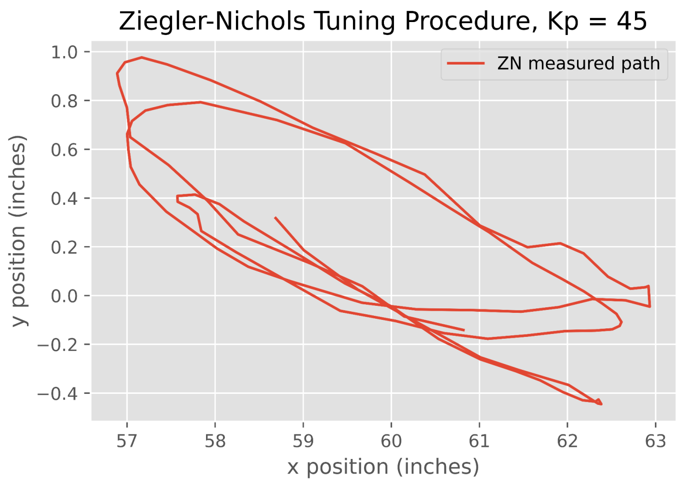


Figure 4: Kp at Critical Stability value where the robot reaches a marginally stable oscillation.

System Modeling

The method with the best performance guarantees, is model-based PID tuning [10]. A popular modeling technique in control theory is that of transfer function modeling. Instead of time domain techniques such as differential equations, transfer functions are in terms of the complex variable s that represents the set of frequencies a given system can respond to.

To model our system, we can start with our general equation of the system's input in equation 1.1 and then re-arange in terms of acceleration to obtain a differential equation of our system. Using this two-parameter feedforard is useful as it allows us to approximately model our complex motor based system through a purely experimental process by tuning the feedforward model as in Figure 2.

Transfer Function Model

Given the feedforward from equation 1.1: $u = K_v \dot{x} + K_a \ddot{x}$, one can re-arrange the equation in terms of acceleration:

$$\ddot{x} = \frac{1}{K_a}u - \frac{K_v}{K_a}\dot{x} \qquad (1.2)$$

Now we can take the Laplace transform of this differential equation. Simply put, each state variable becomes the complex variable s to the power of the order of the derivative of the state variable.

$$s^2 = \frac{1}{K_a} - \frac{K_v}{K_a}s \qquad (1.3)$$

The input cancels ($s^0 = 1$), and the acceleration and velocity states became their associated derivatives.

Finally, we need to re-arrange into a rational expression of input over output:

$$G(s) = \frac{1/k_a}{s^2 + \frac{K_v}{K_a}s} \quad (1.4)$$

Position vs. Input Transfer Function

Transfer functions are great for system modeling as they give crucial information about a system's motion characteristics and stability.

To find these characteristics, we need to find the characteristic roots or *poles* of the system. The value of poles tells us if our system is stable or not.

Pole Locations and Stability impact

Pole Location	Is Stable?	Will Oscillate?
Real, Negative	Yes	No
Complex, Negative	Yes	Yes
Real, 0	Marginally Stable	No
Complex, 0	Marginally Stable	Yes
Real, Positive	No	No
Complex, Positive	No	Yes

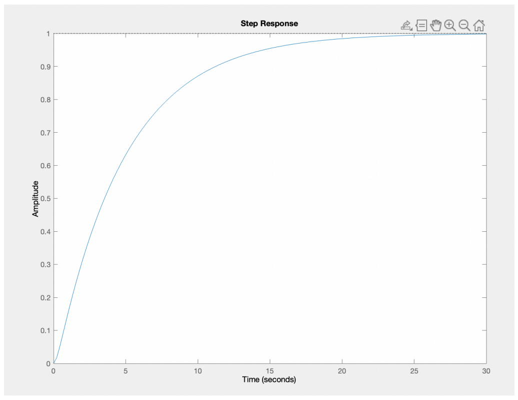
An important concept to grasp is that all poles of a system need to be negative for the system to be stable. This is because solving a linear dynamic system takes the form:

$$f(t) = \sum_{i=1}^{n} e^{\lambda_i t} \qquad (1.5)$$

Where λ_i is a given pole in our system, we can visualize if our system is stable with equation 1.5. If all of our poles are negative, then f(t) will decay to 0, but if just one pole is non-negative, then convergence will not occur, and thus the system is unstable.

The poles of a transfer function system are found by finding when the denominator of our transfer function goes to zero. For equation 1.4, this is when $s^2 + \frac{K_v}{K_a}s = 0$. We can use the quadratic formula to solve for the pole locations. Using this, we know we have a pole at 0 and at $-\frac{K_v}{K_a}$. The presence of a zeroed pole tells us that no matter the motor's constants, in order to make this system stable, we must use feedback control.

When designing our feedback controller, we aim to ensure that all of our poles lie in the left plane of the imaginary axis. Of the stable pole locations, there are three main types of responses: Under-damped, Over-damped, and Critically-damped responses.



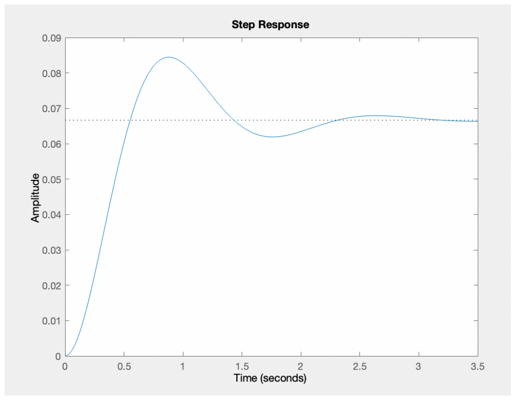


Figure 4: Over-Damped Response

Figure 5: Under-Damped Response

w

Figure 6: Critically-Damped Response

Overdamped responses occur when a system's poles are both real and negative. Underdamped responses occur when the system's poles are complex and negative. A critically damped response occurs whenever both poles are equal. The transfer function of a PID controller is $C(s) = K_p + \frac{K_i}{s} + K_d s$ [12]. For our purposes, our system already has a pole at the origin, so we opt to use the PD controller only $C(s) = K_p + K_d s$ (equation 1.6). Our goal now is to tune these two coefficients to place our system's closed-loop poles at the same location. First, we need to solve the closed loop poles. The general equation for a closed-loop feedback controller is $\frac{G(s)C(s)}{1+G(s)C(s)}$. G(s) is our open-loop system, and C(s) is our feedback controller. Substituting in equation 1.4 and equation 1.6, we can find our closed loop transfer function is

$$CL(s) = \frac{K_p + K_d s}{K_p + K_d s + K_v s + K_a s^2} \quad (1.7)$$

Our characteristic polynomial, is then, therefore, $K_p + K_d s + K_v s + K_a s^2$.

We can then plug the characteristic polynomial into the quadratic formula to find each pole:

$$\lambda_1 = \frac{K_d + K_v - \sqrt{K_d^2 + 2K_d K_v + K_v^2 - 4K_a K_p}}{2K_a}$$

$$\lambda_2 = \frac{K_d + K_v + \sqrt{K_d^2 + 2K_d K_v + K_v^2 - 4K_a K_p}}{2K_a}$$

We know that for a system to be critically damped, $\lambda_1 = \lambda_2$

$$\frac{K_d + K_v - \sqrt{K_d^2 + 2K_d K_v + K_v^2 - 4K_a K_p}}{2K_a} - \frac{K_d + K_v + \sqrt{K_d^2 + 2K_d K_v + K_v^2 - 4K_a K_p}}{2K_a} = 0$$

(1.8)

Which when one solves for K_d yields:

$$K_d = 2\sqrt{K_a K_p} - K_v \quad (1.9)$$

$$\text{When } K_p \geq \frac{K_v^{\,2}}{4K_a}$$

Thus, given our two motor constants, Kv and Ka as well as with any arbitrary choice of Kp which exceeds the bounds, we can use equation 1.9 to guarantee our closed loop system will have a critically damped response. This takes away the pressure of tuning the PID for stability and instead allows the engineercto tune for performance.

The SMARTDAMP Procedure

Now that all equations have been defined, we can outline the procedure one must follow to utilize the SMARTDAMP algorithm. First, they use the feedforward model to follow a ramping trajectory, such as the motion profile in Figure 1. They then will fit the Kv, and Ka coefficients by hand or through an automatic tuning process. This process of tuning the Kv and Ka constants is much more intuitive and has less risk of instability compared to immediately jumping into PID tuning. Finally, the operator can use equation 1.9 to tune the derivative term given any choice of Kp, and they can be confident that the system will remain stable no matter the value they choose. Kp can be adjusted manually until the response is as fast as desired, without worrying about the potential for overshoot. This approach is critical because it combines the precision and guarantees of model-based gain tuning with the low barrier of entry associated with heuristic tuning procedures such as ZN.

Results

Using a holonomic mobile robot, we tested the trajectory tracking and disturbance rejection qualities of our proposed SMARTDAMP approach compared to the Ziegler-Nichols “no-overshoot” and “PD” settings. We had the robot follow a straight-line, motion-profiled trajectory of 60 inches in length. For each tuning configuration, we tested one run undisturbed and a second with a 10-inch disturbance to see how the controller responded.

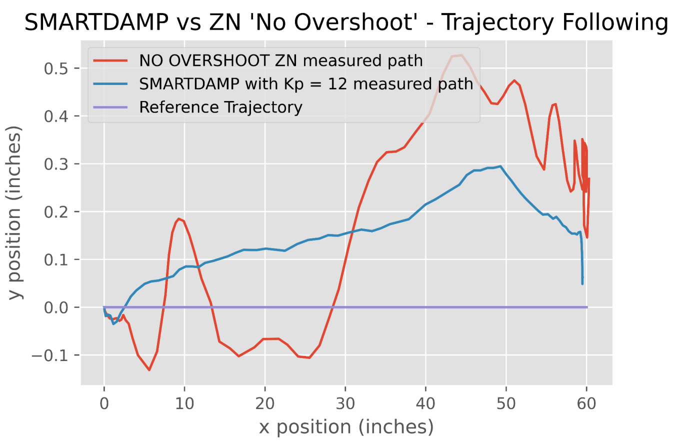


Figure 7: Trajectory tracking of Ziegler-Nichols “No-Overshoot” vs. SMARTDAMP

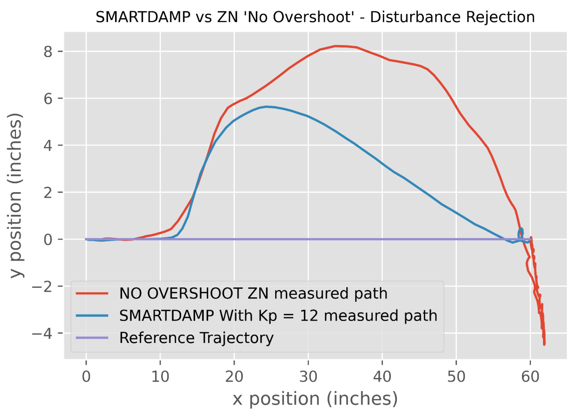


Figure 8: Disturbance Rejection of Ziegler-Nichols “No-Overshoot” vs. SMARTDAMP.

From these results, we can observe that for ideal conditions, the settling time of SMARTDAMP will be much faster than that of the Ziegler-Nichols tuning procedure. Just as critical, whenever external disturbances such as collisions with other robots occur, we can be confident that SMARTDAMP will quickly and smoothly correct the system and reject any disturbances.

Conclusion

In this work, we demonstrated how we could reduce the complexity of PID tuning for the end user by having them first perform a manual system identification of the plant. Then by using the Laplace transform and the generalized feedback equation, we derive optimal, critically damped PID gains for the system in question. We reveal that one can find the critically damped derivative gain as $K_d = 2\sqrt{K_a K_p} - K_v$ (when $K_p \geq \frac{K_v^2}{4K_a}$). We were able to mathematically defend that not only is SMARTDAMP guaranteed to be stable due to forcing a controller with negative pole locations, but also guarantees extremely little/no overshoot due to calculating the gain to make the pole locations identical. This property of SMARTDAMP allows the control engineer to tune the system and purely be concerned with how fast the system's response is without needing to be burdened by concerns of closed-loop stability. Using a Holonomic mecanum drive robot, we tested SMARTDAMP against the Ziegler-Nichols tuning procedure to assess trajectory tracking performance and disturbance rejection. While the Ziegler-Nichols tests also did successfully track the reference trajectory and reject the disturbance, the responses were much slower, less stable, and less predictable than SMARTDAMP.

References

1. Kiam Heong Ang, G. Chong and Yun Li, "PID control system analysis, design, and technology," in *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559-576, July 2005, doi: 10.1109/TCST.2005.847331.

2. “Driver PID Settings.” Thorlabs, Inc. - Your Source for Fiber Optics, Laser Diodes, Optical Instrumentation and Polarization Measurement &amp; Control, https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=9013#:~:text=To%20tune%20your%20PID%20controller,to%20roughly%20half%20this%20value.

3. Trammell, Larry. “Ziegler-Nichols Tuning Rules for PID.” *Ziegler-Nichols Tuning Rules for PID*, https://www.mstarlabs.com/control/znrule.html.

4. Li, Bing, et al. “Study on the Low Velocity Stability of a Prostate Seed Implantation Robot’s Rotatory Joint.” *Electronics*, vol. 9, no. 2, 2020, p. 284. https://doi.org/10.3390/electronics9020284.

5. Fu C-B, Tian A-H, Li Y-C, Yau H-T. Active controller design for precision computerized numerical control machine tool systems. Journal of Low Frequency Noise, Vibration and Active Control. 2019;38(3-4):1149-1159. doi:10.1177/1461348418797239

6. Bennett, S. (1984). Nicholas Minorsky and the automatic steering of ships. IEEE Control Systems Magazine, 4, 10-15.

7. *Trapezoidal motion profiles in WPILIB*. FIRST Robotics Competition Documentation. (n.d.). Retrieved January 4, 2023, from https://docs.wpilib.org/en/stable/docs/software/advanced-controls/controllers/trapezoidal-profiles.html

8. Douglas, B. (2018, September 25). What is Feedforward Control? | Control Systems in practice. YouTube. Retrieved January 6, 2023, from https://www.youtube.com/watch?v=FW_ay7K4jPE

9. Veness, T. (n.d.). *Controls Engineering in FRC - Tavsys.net*. Retrieved January 6, 2023, from https://file.tavsys.net/control/controls-engineering-in-frc.pdf

10. Douglas, B. (2018, July 24). Manual and automatic PID tuning methods | understanding PID control, part 6. YouTube. Retrieved January 6, 2023, from https://www.youtube.com/watch?v=qj8vTO1eIHo

11. HoustonMathPrep. (2013, October 8). *Using Laplace transforms to solve differential equations*. YouTube. Retrieved January 6, 2023, from https://www.youtube.com/watch?v=FBl6XSTaS7k

12. Wikimedia Foundation. (2023, January 3). Pid Controller. Wikipedia. Retrieved January 6, 2023, from https://en.wikipedia.org/wiki/PID_controller

13. *Intro to control - 10.2 closed-loop transfer function*. YouTube. (2014, October 31). Retrieved January 6, 2023, from https://youtu.be/GBv5iiKA6gw