Introduction to PID Control

Cody L. Wellman

***Abstract*—In this paper, we explore examples of PID control loops and the theory of how they operate. We also explore the tuning of these loops and how we can optimize them to perform the best.**

***Index Terms*—Proportional-integral-derivative control, PID theory, PID tuning, control systems**

I. Introduction

A proportional-integral-derivative controller, commonly shortened to PID controller, is a control loop mechanism employing feedback that is commonly used in industrial control systems and a variety of other applications that require continuous control. A PID controller continuously calculates an error value $e(t)$ which is the difference between the desired setpoint $r(t)$ and the currently measured process variable $y(t)$ and it applies a correction based off of the proportional, integral, and derivative terms, hence the naming.

Essentially, PID control automatically applies accurate and responsive correction to a control system. An everyday example could be a thermostat which controls the temperature of a room. Let's say you set the thermostat to a specific temperature and some external device is introduced into the room and cools down the room to a temperature other than what was set on the thermostat, then the PID controller's algorithm is going to come in and bring the current temperature of the room back to what your desired temperature is with minimal delay and overshoot by adjusting the temperature in a controlled manner.

II. Control Loop Example

Now, let's look at a more detailed example of how a PID control loop is implemented. So, let's imagine a robotic arm [2] which can be moved and positioned how we want by a control loop. An electric motor can raise or lower the arm depending on forward or reverse power that is applied. Power applied is unable to be a simple function of position due to the inertial mass of the arm, the force of gravity, and any other external forces.

- The currently sensed position is the process variable $y(t)$.
- The desired position is called the setpoint $r(t)$.
- The difference between the $y(t)$ and $r(t)$ is the error value $e(t)$, which tells us if the arm is raised too high or if it is lowered too much and by how much.
- The input to the PID controller $u(t)$, which in this case is the current of the motor, is also the output of the PID controller $u(t)$.

By measursing the currently sensed position, the process variable $y(t)$, and subtracting it from our desired setpoint $r(t)$, we get our error value $e(t)$, and with that our PID controller can calculate how much current will be supplied to the motor $u(t)$.

A. Proportional Control

The first method of control we can use is proportional control, this is where the motor current is going to be set in proportion to the existing error. This method of control can fail if the arm has to lift weights in different sizes, if this is the case then a greater weight needs a greater force applied for the respective error. Inversely, if a smaller force is being applied then a smaller respective error will be present. This is where integral and derivative control will come in.

B. Integral Control

Integral control increases action not only in relation to error but also the length of time for which the action has taken place. When an applied force is not enough to bring the error to zero, the force will then be increased as time passes. When purely integral control is being used, the controller may bring the error to zero, but it would start with a slow reaction time which would have small action and increase as time goes on, and near the end the action will increase as long as the error is positive, even if the error is nearing zero.

If too much integral is being applied when the error is small and decreasing, it will usually lead to the controller overshooting. After the controller has been overshooting, applying a large correction in the opposite direction and reapeadity overshooting the desired position again, would cause the output to oscillate around the setpoint. If the amplitude of the oscillations increase with time then the system would be considered unstable. If the oscillations decrease with time then the system would be considered stable.

C. Derivative Control

A derivative controller does not consider the magnitude of an error, which means a purely derivative controller is unable to bring a system to its setpoint since it can't bring an error to zero, but a derivative controller considers the rate of change of an error, which it tries to bring that to zero. It attempts at flattening the trajectory of error into a flat horizontal line, which dampens the force applied and thus will reduce overshooting.

III. PID Controller Theory

The PID control scheme is named after its three terms used in correction; the sum of those terms and calculations constitute the controller's final output $u(t)$. See (1) for the final equation of the PID controller. It is important to remember that

a PID controller can be used without having all terms fulfilled, in place of those terms a 0 is substituted during configuration of the constant gains.

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de}{dt} \tag{1}$$

A. Proportional Term

The proportional term produces the controller output $u(t)$ which is proportional to the current error value $e(t)$.The proportional control response is adjusted by multiplying the error value $e(t)$ by the proportional gain constant K_p, see (2) below for the equation that proportional control performs.

$$u(t) = K_p e(t) \tag{2}$$

When the proportional gain constant is too large in value it can cause a large change in the controller output response for the given error which can cause a system to become unstable. Inversely, when the proportional gain constant is too small in value it can cause a small controller output response and the controller will be less responsive and less sensitive. Thus tuning of the proportional gain constant is important in order to have a responsive and sensitive controller.

1) Steady-State Error

The steady-state error is the difference between the desired final output and the actual one being measured after the system reaches equilibrium. An error that is not a zero is required to drive it, so a proportional controller generally operates with some steady-state error. A common way of eliminating steady-state error is to integrate the error and add it to the controller's input but a better approach to eliminate it would be to use a feedforward or constrain when the integral control acts.

B. Integral Term

The integral term's contribution to a PID controller is the sum of the accumulated error value $e(t)$ over a specific amount of time which is known as the delta-time dt and it provides the accumulated offset that should have been corrected on previous iterations. The accumulated error value $e(t)$ is then multiplied by the integral gain constant K_i and it then gets added to the controller output $u(t)$. See (3) below for the equation that integral control performs.

$$u(t) = K_i \int e(t)dt \tag{3}$$

The integral term is responsible for accelerating the movement of the process towards the given setpoint and should eliminate the residual steady-state error that occurs with just a proportional controller. However, since the integral term responds to the accumulated errors from previous iterations, it can cause the currently measured process value to overshoot the desired setpoint value. Overshooting is when the measured value exceeds or goes higher in value than what the desired value is.

C. Derivative Term

The derivative term is responsible for the calculation of the derivative of the error value and it does this by determining the slope of the error over time and multiplying that by the derivative gain constant K_d and it then gets added to the controller output $u(t)$. See (4) below for the equation that derivative control performs.

$$u(t) = K_d \frac{de}{dt} \tag{4}$$

The derivative term provides a dampening effect that helps to limit overshoot and improves stability as well as improving settling time. It predicts future error and behavior of the system and compenstatines the output ahead of time.

IV. Control Loop Tuning

Tuning a PID control loop is the process of adjusting all of its control parameters and proportional, integral, and derivative gain constants to the optimum calculated values for the desired controller response. Stability, which is when there is no unbound oscillation around the setpoint, is a basic requirement, beyond that there is different behavior that different systems may have with different requirements. There are various methods of control loop tuning, this section describes some traditional, manual, and software methods for loop tuning.

A. Stability

When the gain constants of a PID control loop are not adjusted properly then a controller's process input may be considered unstable, its output will diverge from the setpoint and it may or may not have oscillation. Instability is caused when a gain is in excess, particularly along with being in the presence of significant lag.

Generally, a stabilization of the controller's response is required and the process input should not oscillate for any condition, except sometimes bounded oscillation may occur and can be acceptable or desired in certain applications.

B. Optimal Behavior

The optimal behavior of a change in the process variable or a change in the setpoint will vary depending on the application and the desired response.

There are two basic requirements when it comes to the optimal behavior of a controller, they are regulation, which is the action of staying at a desired setpoint, and tracking, which is implementing the change of a setpoint. These two terms refer to how well the controlled variable tracks the desired setpoint. Specific criteria for tracking includes the rise time, which is the time it takes for a signal to go from a low value to a high value, and settling time, which is the time it takes for a value to settle or get to the desired setpoint. Some processes

must not allow an overshoot, which is when the currently measured process variable goes above the desired setpoint.

C. Tuning Methods

There are several methods for tuning a PID control loop. The most effective methods are generally going to involve the development of some form of model and choosing the P, I, and D based on the dynamic model parameters.

Manual tuning methods may be time-consuming and generally are not recommended especially for systems with long control loop times. The Ziegler-Nichols method is going to be quicker than doing it manually but may take some trial-and-error also. Software tuning of a PID control loop is most likely going to be the most accurate way of tuning but is going to heavily depend on specific applications and software.

1) Manual Tuning

The first tuning method is done manually and it starts with setting K_i and the K_d gain to zero. Then you start increasing the K_p gain, it is generally suggested to start off with a K_p of 0.01 and increase by multiplying that by 10 until the output of the loop starts to oscillate and set K_p to half of that value. Then continue onto K_i where you will be increasing the value until any offset is corrected but not too high of a value or else it could cause instability. Finally, start increasing K_d if it is desired. Too much K_d could cause excessive response and overshooting.

2) Ziegler-Nichols Method

Another tuning method is known as the Ziegler-Nichols method, which was introduced by John G. Ziegler and Nathaniel B. Nichols in the 1940s. As mentioned before, you start off with setting K_i and the K_d gain constant to zero. The proportional gain constant is increased until it reaches K_u, which is referred to as the ultimate gain and is when the controller output starts to oscillate constantly. K_u and the oscillation period T_u are then used to set the rest of the gain constants. Please refer to the table below.

TABLE I
ZIEGLER-NICHOLS METHOD [1]

Control Type	K_p	K_i	K_d
P	$0.50K_u$	—	—
PI	$0.45K_u$	$0.54K_u/T_u$	—
PD	$0.8K_u$	—	$0.10K_uT_u$
PID	$0.60K_u$	$1.2K_u/T_u$	$3K_uT_u/40$

3) Software Tuning

Most modern industrial embedded PID controllers no longer use manual calculation methods shown above to tune control loops. PID control loop tuning and optimization is primarily done using software packages to ensure consistent and accurate values. These software packages gather data, develop models, and suggest optimal tuning values.

REFERENCES

[1] Ziegler, J.G & Nichols, N. B. (1942). *Optimum Settings for Automatic Controllers*.

[2] Hacettepe University Department of Electrical and Electronics Engineering. (2009) *Position Control System.*