PID Overview

Jeremy Evans

---

1 Introduction

This document introduces the theory of PID control and its programmatic implementation. We will define what a PID controller is, what its purpose is, loop tuning, and its implementation in software. It is meant to serve as an aid for designing the programming logic of a PID controller deployable to an MSP430 MCU.

A working understanding of the Laplace and Z transforms would be helpful in reading this document (if only to verify the correctness of the implementation). Nevertheless, we will document the derivation in full (though we may omit the calculus and algebra in the final progress report).

2 Overview

A PID controller (proportional-integral-derivative controller) is a feedback control loop mechanism that receives a desired setpoint $r(t)$ as input and outputs a *process variable* (PV), denoted as $y(t)$, which is then fed back into the system as input. It continuously calculates an error value $e(t)$ as the difference between the desired setpoint and the process variable and applies a correction based on proportional (P), integral (I), and derivative (D) terms, which are summed together to make up the control variable, $u(t)$ affecting the value of the process variable, hence the name. In applying this correction over time, it attempts to stabilize the output, i.e., eliminate the oscillation of the error value (or achieve marginal stability, or bounded oscillation, though specifications vary between applications).

Corrections are achieved by multiplying the P, I, and D terms by constants K_p, K_i, and K_d respectively, each chosen through a process known as *loop tuning*. Improperly chosen constants would result in the error value diverging (with or without oscillation), whereas properly chosen constants would have the desired effect of the error value converging to zero (or oscillating within an acceptable bounds).

2.1 Mathematical Definition

PID control is mathematically defined as:

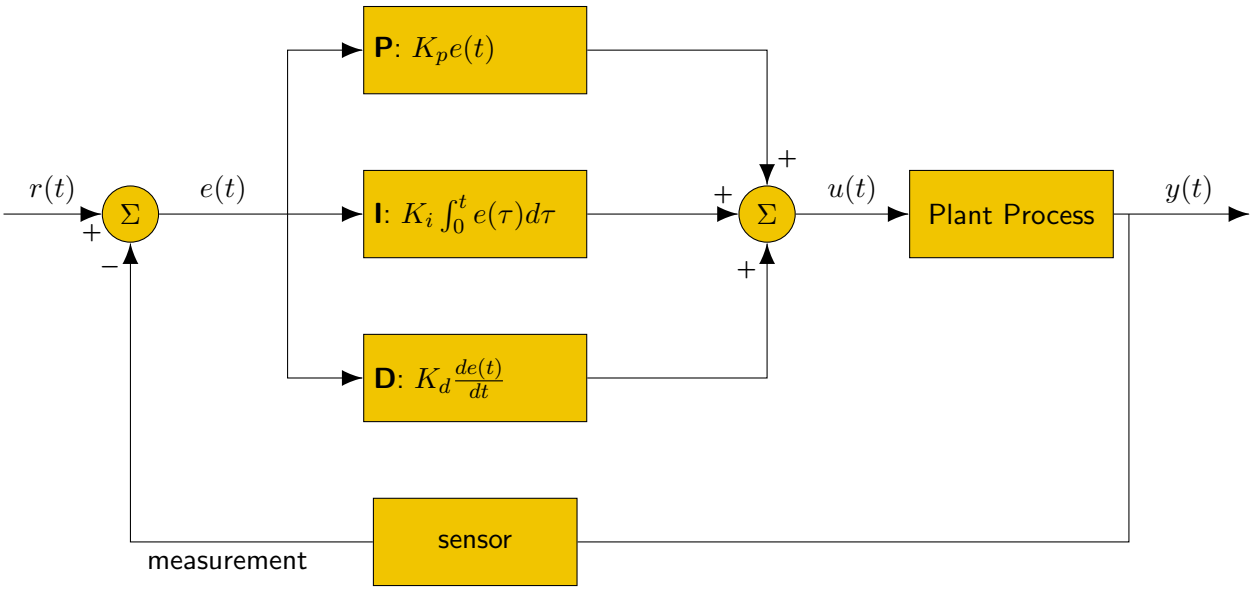
$$\boxed{u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}} \tag{1}$$

where

- $u(t)$ is the *control variable*, the parameter that is controlled,
- $K_p \in \mathbb{R} \geq 0$ is the *proportional gain*,
- $K_i \in \mathbb{R} \geq 0$ is the *integral gain*,
- $K_d \in \mathbb{R} \geq 0$ is the *derivative gain*,
- $e(t) = r(t) - y(t)$ is the error ($r(t) = SP$ is the *setpoint*, $y(t) = PV$ is the *process variable*),
- t is the *instantaneous time* (i.e., the current time),
- $\tau \in \mathbb{R}$ are measurements of time in the range $[0, t]$.

2.2 PID Block Diagram

The PID algorithm can be represented by the following block diagram:



Represented in a high-level model of a heat treatment furnace, $r(t)$, the setpoint, would be the desired furnace temperature; $u(t)$, the control variable, would be the gas flow rate; and $y(t)$, the process variable, would be the measured furnace temperature.

2.3 PID Terms

2.3.1 Proportional

Given by $P(t) := K_p e(t)$, the proportional term produces an output that is proportional to the current error value $e(t)$; the greater the error value, the greater the control output. This output is multiplied by a gain factor $K_p \in \mathbb{R} \geq 0$ that determines how responsive the controller should be with a given error-value. Large values of K_p result in a large change in the output for a given change in the error. However, exceptionally large values may result in the output *overshooting* the setpoint (exceeding the value of the setpoint) and, in the worst case, destabilizing the system whereupon the error rate diverges.

One caveat to the proportional term is that a non-zero error is required for it to have any effect; if the error is zero, it will not produce a response. Because of this, it generally operates with a *steady-state error,* which is the difference between the desired final output and the actual one.

2.3.2 Integral

Given by $I(t) := K_i \int_0^t e(\tau) d\tau$, the integral term's output is proportional to both the magnitude and duration of the error; its integral is the sum of the instantaneous error over time and represents the accumulated offset. Because it sums the error over time, the integral response will continually increase over time so long as the error is non-zero, effectively driving the steady-state error inherent in the proportional term to zero.

If the setpoint suddenly changes to a drastic degree or if the measured process variable jumps significantly, *integral windup* can occur wherein the plant process of a PID controller can saturate; that is, reach its physical limit (e.g., drive a valve to be fully open or closed). In this case, no physical change in the system can affect the integral term; it will continue to grow without bounds.o

Several solutions to this problem are:

- Increase the setpoint incrementally (in a suitable ramp)
- Initialize the controller integral to a desired value (e.g., before the problem occurs)
- Disable the integral function until the process variable has entered the controllable region

2.3.3 Derivative

Given by $D(t) := K_d \frac{de(t)}{dt}$, the output of the derivative term is proportional to the rate of change of the error value and is intended to decrease the rate at which the error increases. Large values of K_d correlate to large responses to changes in the error. However, it is sensitive to noise in the process variable (moreso for large values of K_d) and is often multiplied by a low-pass filter to mitigate the high frequency gain and noise.

3 Implementation

To implement the PID algorithm in software, we need to transform the equation into a form that can be represented in code. One way to do this is to derive the transfer function of the PID algorithm in the *s-domain* (where output is viewed with respect to frequency rather than time), and discretize the transfer function using what is called the *bilinear transform* or *Tustin's method*. By discretizing the transfer function, we can arrive at a linear difference equation (also known as a recurrence relation), which can easily be implemented in code.

3.1 PID Transfer Function

The Laplace transform of a function $f(t)$, defined for all real numbers $t \geq 0$, is the function $F(s)$, which is a unilateral transform defined by

$$\boxed{F(s) := \mathcal{L}\{f(t)\} = \int_0^\infty f(t)e^{-st}dt} \tag{2}$$

where $s = \sigma + i\omega$ (with real numbers σ and ω) is a complex number denoting frequency.

Divide **eq. 1** into three functions $P(t)$, $I(t)$, and $D(t)$, where

$$\begin{aligned} P(t) &:= K_p e(t) \\ I(t) &:= K_i \int_0^t e(\tau)d\tau \\ D(t) &:= K_d \frac{de(t)}{dt} \end{aligned}$$

Below, we apply the Laplace transform to each in turn.

3.1.1 Laplace: P(t)

$$\begin{aligned} P(s) &:= \mathcal{L}\{P(t)\} \\ &= \int_0^\infty K_p e(t)e^{-st}dt \\ &= K_p \int_0^\infty e(t)e^{-st}dt \\ &= K_p \mathcal{L}\{e(t)\} \\ &= K_p E(s) \end{aligned}$$

3.1.2 Laplace: I(t)

$$
\begin{aligned}
I(s) &:= \mathcal{L}\{I(t)\} \\
&= \int_0^\infty \left(K_i \int_0^t e(\tau) d\tau \right) e^{-st} dt \\
&= K_i \int_0^\infty e^{-st} \int_0^t e(\tau) d\tau dt \\
&= K_i \left(\int_0^t e(\tau) d\tau \left(\frac{e^{-st}}{-s} \right) \right) \Bigg|_0^\infty - \int_0^\infty e(t) \left(\frac{e^{-st}}{-s} \right) dt; \text{(integration by parts)} \\
&= K_i \left(0 - 0 + \frac{1}{s} \int_0^\infty e(t) e^{-st} \right) \\
&= K_i \left(\frac{\mathcal{L}\{e(t)\}}{s} \right) \\
&= \frac{K_i E(s)}{s}
\end{aligned}
$$

3.1.3 Laplace: D(t)