1. PID Controllers

Pieter P

This first chapter gives a brief recap of PID control theory. It describes the controller architecture and derives the formulas that will be implemented in C++ in the next chapter.

Prerequisites: basic control theory, Laplace and Z-transforms.

Continuous-time

In this first section, we'll assume that all signals are functions of a continuous time variable t. Later, we'll discretize the continuous-time controllers into a discrete-time approximation that can easily be manipulated by computers and microcontrollers.

Closed-loop controllers

Figure 1 shows the block diagram of a general closed-loop or feedback control system. The output $y(t)$ of the *plant* (the system being controlled) is subtracted from the reference $r(t)$, and this error $e(t)$ is fed to the *controller*, which produces the control signal $u(t)$ that is sent to the input of the plant, in an attempt to drive the error to zero.

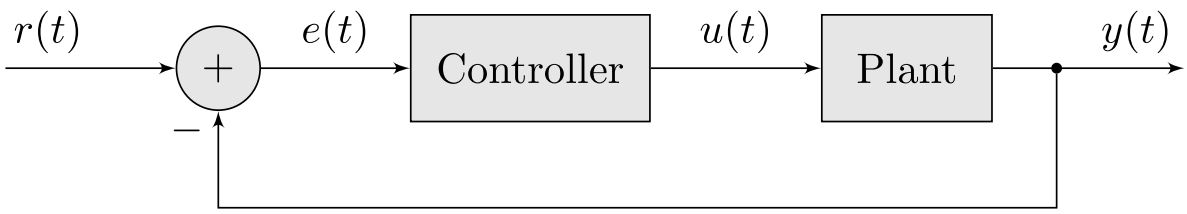

Figure 1: Block diagram of a closed-loop controller

Image source code

The PID controller

In a PID controller, the control signal is calculated as the sum of three components: a proportional component, an integral component, and a derivative component. The proportional component simply multiplies the error by a constant K_p, the integral component multiplies the time integral of the error by a constant K_i, and the derivative component multiplies the time derivative of the error by a constant K_d. Mathematically, the control law is given by

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}}{\mathrm{d}t} e(t).$$

The constants K_p, K_i and K_d are referred to as the proportional gain, the integral gain, and the derivative gain respectively. The block diagram of this type of controller is shown in Figure 2.

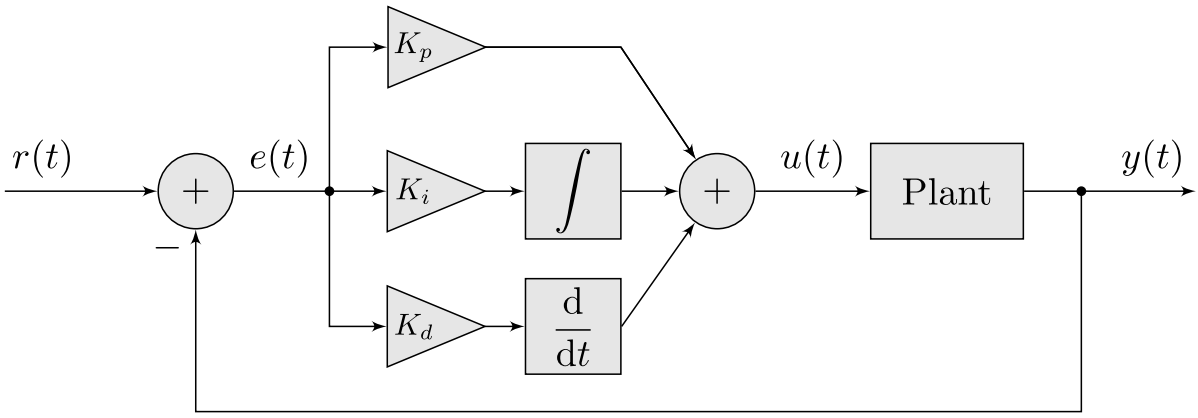

Figure 2: Block diagram of a PID controller

Image source code

You can find intuitive explanations of the purpose of each of the three components all over the internet, but in short: the proportional component makes the controller act on the instantaneous error, the integral component accumulates past errors in order to minimize the steady-state and tracking error, and the derivative component penalizes the velocity at which the output changes, which can help to reduce overshoot.

Frequency domain

In the frequency or s-domain, the PID control law can be written as

$$U(s) = \left(K_p + K_i \frac{1}{s} + K_d \, s\right) E(s),$$

where $U(s)$ and $E(s)$ are the Laplace transforms of the respective time-domain signals $u(t)$ and $e(t)$. This formulation is represented by Figure 3.

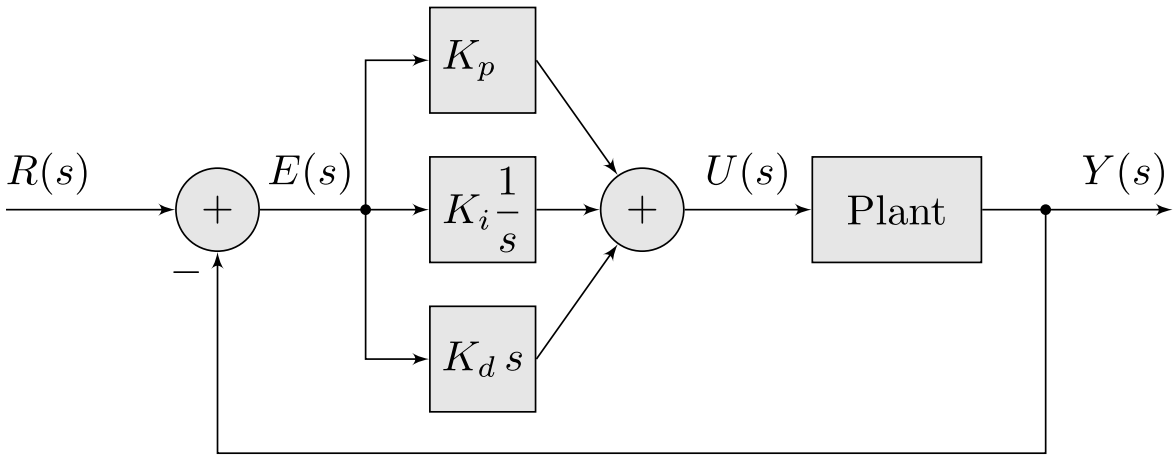

Figure 3: Block diagram of a PID controller using Laplace notation

Image source code

Derivative filtering

The derivative of the error can be rather noisy, so practical PID controllers often include a low-pass filter. Let $e_f(t)$ be the low-pass filtered error, then the control law can be modified into

$$u(t) = K_p \, e(t) + K_i \int_0^t e(\tau) \mathrm{d}\tau + K_d \, \frac{\mathrm{d}}{\mathrm{d}t} e_f(t).$$

Or, in the frequency domain,

$$U(s) = \left(K_p + K_i \frac{1}{s} + K_d \, s H(s)\right) E(s).$$

Here, $H(s)$ is the transfer function of the low-pass filter for the derivative component.

For the sake of simplicity, we'll use a single-pole low-pass filter to filter the error before taking the derivative. The transfer function of this filter is

$$H(s) = \frac{1}{1 + s \, T_f},$$

where T_f is the filter's time constant, a parameter we can tune later.

Discrete-time

Since computers and microcontrollers cannot deal with continuous time, the control law has to be discretized. We'll use T_s to note the time step or sampling interval.

Discrete-time signals

Given the continuous-time error signal $e : \mathbb{R} \to \mathbb{R} : t \mapsto e(t)$, define the discrete-time error signal $e[k]$ as $e(t)$ sampled at $t = k\,T_s$ (with sampling interval T_s),

$$e[\cdot] : \mathbb{Z} \to \mathbb{R} : k \mapsto e[k] \triangleq e(kT_s).$$

We will use the same letters for continuous-time and discrete-time transfer functions and signals in the s- and z-domain, it should be clear from the context and the variables used (s or z) whether it's a continuous-time or discrete-time signal. For example, $H(s)$ is a continuous-time transfer function, and $H(z)$ is a discrete-time transfer function, defined by different rational functions.

Forward Euler

The first discretization method we'll have a look at is the forward Euler method, it is one of simplest methods available to approximate a continuous-time ordinary differential equation by a discrete-time difference equation or recurrence relation.

Integral

When the time step T_s is sufficiently small, the integral term of the PID control law at time $t = kT_s$ can be approximated by a Riemann sum:

$$e_i(t) \triangleq \int_0^t e(\tau)\mathrm{d}\tau \approx \sum_{n=0}^{k-1} e[n]\,T_s \triangleq e_i[k]$$

Note that this is an approximation, $e_i(kT_s) \approx e_i[k]$, they are not exactly equal.

This signal $e_i[k]$ can also be defined by the following recurrence relation

$$\begin{cases} e_i[k] = e_i[k-1] + e[k-1]\,T_s \\ e_i[0] = 0. \end{cases}$$

In the z-domain, the forward Euler discretization we carried out in the previous paragraph can be expressed as

$$\begin{aligned} & E_i(z) = z^{-1}\,E_i(z) + T_s\,z^{-1}\,E(z) \\ \Leftrightarrow \quad & E_i(z) = \frac{T_s}{z-1}\,E(z). \end{aligned}$$

Recall that in the s-domain, the relation between $E_i(s)$ and $E(s)$ was given by $E_i(s) = \frac{1}{s}E(s)$, so in general, we could define forward Euler discretization as the mapping from the s-domain to the z-domain where $s \mapsto \frac{z-1}{T_s}$.

Backward Euler

The backward Euler method is very similar to forward Euler, but it has a different time delay:
When applied to the derivative $y(t) = \frac{\mathrm{d}}{\mathrm{d}t}x(t)$, the forward Euler method results in the discrete-time recurrence relation $y[k] = \frac{x[k+1]-x[k]}{T_s}$, which is non-causal (the output $y[k]$ depends on the future input $x[k+1]$). The following section introduces the backward Euler method, which will discretize this derivative as the causal recurrence $y[k] = \frac{x[k]-x[k-1]}{T_s}$.

Derivative

We can approximate the derivative term in the control law using finite differences:

$$e_d(t) \triangleq \frac{\mathrm{d}}{\mathrm{d}t}e_f(t) \approx \frac{e_f(t) - e_f(t-T_s)}{T_s} \triangleq e_d[k]$$

In the z-domain, this is equivalent to

$$E_d(z) = \frac{1 - z^{-1}}{T_s} E_f(z)$$

$$\Leftrightarrow \quad E_d(z) = \frac{z - 1}{z\,T_s} E_f(z).$$

In the s-domain, we have $E_d(s) = s\,E_f(s)$, so backward Euler discretization is the mapping $s \mapsto \frac{z-1}{z\,T_s}$.

Low-pass filter

Applying this mapping to the transfer function of the low-pass filter for the derivative results in the following,

$$\begin{aligned} E_f(s) &= \frac{1}{1 + s\,T_f} E(s) \\ E_f(z) &= \frac{1}{1 + \frac{z-1}{z\,T_s} T_f} E(z) \\ &= \frac{z\,T_s}{z\,(T_s + T_f) - T_f} E(z) \\ &= \frac{z\,\beta}{z - (1 - \beta)} E(z), \end{aligned}$$

where $\beta \triangleq \frac{T_s}{T_s + T_f}$. You might recognize this expression as the transfer function of the exponential moving average filter, usually defined by the recurrence relation $e_f[k] = \beta\,e[k] + (1 - \beta)\,e_f[k-1]$.

In practice, one often treats the derivative term as a whole, discretizing the derivative and the low-pass filter in one go by combining their transfer functions and then applying forward Euler:

$$\begin{aligned} E_d(s) &= sH(s)E(s) \\ &= \frac{s}{1 + s\,T_f} E(s) \\ &= \frac{1}{\frac{1}{s} + T_f} E(s) \\ \\ E_d(z) &= \frac{1}{\frac{T_s}{z-1} + T_f} E(z) \\ &= \frac{z - 1}{T_s - T_f + z\,T_f} E(z) \end{aligned}$$

In the time domain, this becomes

$$\begin{aligned} (T_s - T_f)\,e_d[k-1] + T_f\,e_d[k] &= e[k] - e[k-1] \\ e_d[k] &= \alpha\,\frac{e[k] - e[k-1]}{T_s} + (1 - \alpha)\,e_d[k-1], \end{aligned}$$

where $\alpha \triangleq \frac{T_s}{T_f}$. This can be written as

$$e_d[k] = \frac{e_f[k] - e_f[k-1]}{T_s}$$

$$e_f[k] \triangleq \alpha\,e[k] + (1 - \alpha)\,e_f[k-1].$$

The first equation is the finite differences approximation of a derivative, and the second is again an exponential moving average filter, but with a different weight factor compared to the result we got earlier using backward Euler.

Other discretization methods

An alternative method is the bilinear transform (also known as the trapezoidal rule or Tustin's rule), it is of a higher order than forward and backward Euler, and has some nice properties such as the fact that stable poles in one domain map to stable poles in the

other. Other techniques include pole-zero matching, matched step response, frequency response approximations, but these are outside of the scope of this article as they are not usually applied to PID controllers.

Overview

The following table gives an overview of all signals that make up the PID control law, as well as their discretizations. The third column is the most important one, because the discrete-time recurrence relations can easily be implemented in software.

Continuous-time	**s-domain**	**Discrete-time**	**z-domain**
$e(t)$	$E(s)$	$e[k]$	$E(z)$
$e_i(t) = \int_0^t e(\tau)\mathrm{d}\tau$	$E_i(s) = \frac{1}{s} E(s)$	$e_i[k] = e_i[k-1] + T_s\, e[k-1]$	$E_i(z) = \frac{T_s}{z-1} E(z)$
$e_d(t) = \frac{\mathrm{d}}{\mathrm{d}t} e_f(t)$	$E_d(s) = sE_f(s)$	$e_d[k] = \frac{e_f[k]-e_f[k-1]}{T_s}$	$E_d(z) = \frac{z-1}{z\,T_s} E_f(z$
$e_f(t) = e(t) - T_f \frac{\mathrm{d}}{\mathrm{d}t} e_f(t)$	$E_f(s) = \frac{1}{1+s\,T_f} E(s)$	$e_f[k] = \alpha\, e[k] + (1-\alpha)\, e_f[k-1]$	$E_f(z) = \frac{\alpha\, z}{z-(1-\alpha)} E$

Derivative on measurement

One disadvantage of the PID topology discussed above is that the derivative component will become very large if the reference $r(t)$ suddenly changes. This effect is known as “derivative kick”.
The solution is really simple: instead of the derivative of the error, the derivative of the measurement is used. The former is known as “derivative on error”, the latter as “derivative on measurement”. Both topologies are equivalent if the reference is constant, because if $\frac{\mathrm{d}}{\mathrm{d}t} r(t) = 0$, then $\frac{\mathrm{d}}{\mathrm{d}t} e(t) = -\frac{\mathrm{d}}{\mathrm{d}t} y(t)$.

Figure 4 shows a block diagram of this new derivative on measurement topology, including the low-pass filter on the derivative.

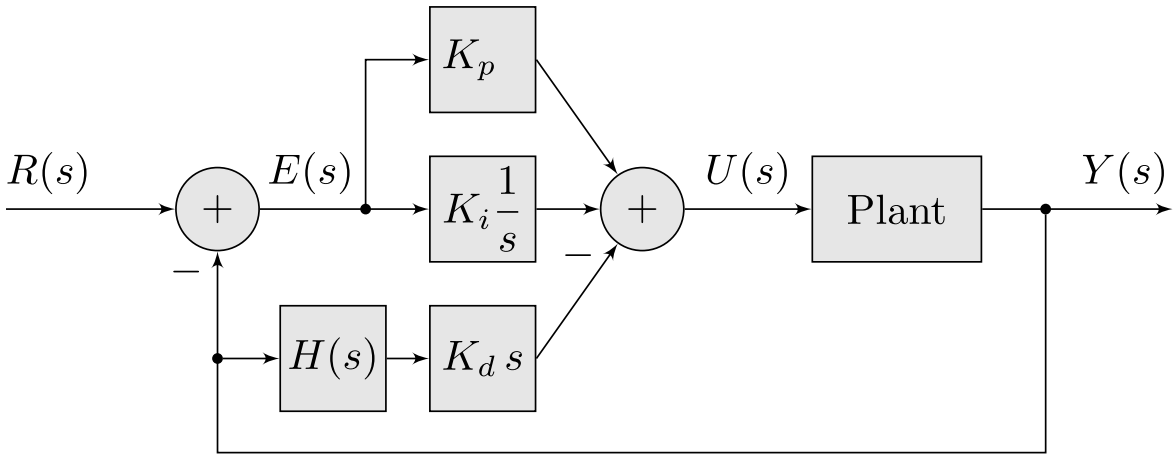

Figure 4: Block diagram of a PID controller with “derivative on measurement”

Image source code