Chapter 11 - PID Controllers

July 12, 2015

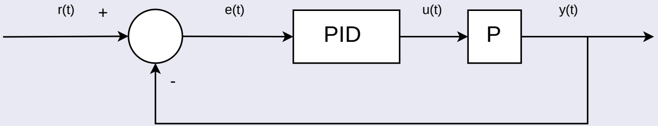
Outline

What is a PID controller?

Definition

A **P**roportional **I**ntegral **D**eriviative controller is a loop feedback controller with continuous time equation:

$$u(t) = \underbrace{K_p e(t)}_{\text{Proportional Action}} + \underbrace{K_i \int_0^t e(\tau) d\tau}_{\text{Integral Action}} + \underbrace{K_d \frac{de(t)}{dt}}_{\text{Derivative Action}}$$



More than 90% of all closed loop controllers are PID

What is a PID controller?

- Proportional action $K_p e(t)$: All advantages of a high loop gain
  - + Reduces rise time
  - + Improves robustness: $K \rightarrow \infty \Rightarrow S_P^T \rightarrow 0$
  - – Reduces but **does not eliminate steady-state error**: Only when $K \rightarrow \infty$, error $\rightarrow 0$ (unless plant has pole(s) at $s = 0$)
  - – **Mind stability**: limit on gain

What is a PID controller?

- Proportional action $K_p e(t)$: All advantages of a high loop gain
  - \+ Reduces rise time
  - \+ Improves robustness: $K \to \infty \Rightarrow S_P^T \to 0$
  - – Reduces but **does not eliminate steady-state error**: Only when $K \to \infty$, error $\to 0$ (unless plant has pole(s) at $s = 0$)
  - – **Mind stability**: limit on gain
- Integral action $K_i \int_0^t e(\tau) d\tau$: Reacts on constant errors
  - \+ Augments the type of $P(s)C(s)$ by one, thus **can eliminate steady state error** in some cases
  - – Makes transient response slower

What is a PID controller?

- Proportional action $K_p e(t)$: All advantages of a high loop gain
  - $+$ Reduces rise time
  - $+$ Improves robustness: $K \to \infty \Rightarrow S_P^T \to 0$
  - $-$ Reduces but **does not eliminate steady-state error**: Only when $K \to \infty$, error $\to 0$ (unless plant has pole(s) at $s = 0$)
  - $-$ **Mind stability**: limit on gain
- Integral action $K_i \int_0^t e(\tau)d\tau$: Reacts on constant errors
  - $+$ Augments the type of $P(s)C(s)$ by one, thus **can eliminate steady state error** in some cases
  - $-$ Makes transient response slower
- Derivative action $K_d \frac{de(t)}{dt}$: Reacts on quick variations
  - $+$ Damping effect: reduces overshoot, improves transient response
  - $-$ Sensitive for noise, amplifies it if present

Outline

Proportional Control

The continuous-time and discrete implementation are identical

Continuous:

$$u_p(t) = K_p e(t) \quad \leftrightarrow \quad \frac{U_p(s)}{E(s)} = K_p$$

Discrete:

$$u_p[k] = K_p e[k] \quad \leftrightarrow \quad \frac{U_p(z)}{E(z)} = K_p$$

Derrivative Control

Discretization can be done with **backward Euler**, applying $\dot{y} \approx \frac{y[k]-y[k-1]}{T}$ or $s = \frac{z-1}{Tz}$

Continuous:

$$u_d(t) = K_d \frac{de(t)}{dt} \quad \leftrightarrow \quad \frac{U_d(s)}{E(s)} = K_d s$$

Discrete:

$$u_d[k] = K_d \frac{e[k] - e[k-1]]}{T} \quad \leftrightarrow \quad \frac{U_d(z)}{E(z)} = K_d \frac{z-1}{Tz}$$

with T the sampling time.

Other discretization methods can be used, but forward Euler can introduce instability.

Integral Control

Discretization can be done with **backward Euler**, applying $\dot{y} \approx \frac{y[k]-y[k-1]}{T}$ or $s = \frac{z-1}{Tz}$

The continuous equation is:

$$u_i(t) = K_i \int_0^t e(\tau) d\tau \quad \leftrightarrow \quad \frac{U_i(s)}{E(s)} = \frac{K_i}{p}$$

Differentiating this gives:

$$\dot{u}_i = K_i e(t)$$

Integral Control

Discretization can be done with **backward Euler**, applying $\dot{y} \approx \frac{y[k]-y[k-1]}{T}$ or $s = \frac{z-1}{Tz}$
The continuous equation is:

$$u_i(t) = K_i \int_0^t e(\tau)d\tau \quad \leftrightarrow \quad \frac{U_i(s)}{E(s)} = \frac{K_i}{p}$$

Differentiating this gives:

$$\dot{u}_i = K_i e(t)$$

Then applying backward Euler:

$$u_i[k] = u[k-1] + K_i T e[k] \quad \leftrightarrow \quad \frac{U_i(z)}{E(z)} = \frac{K_i T}{1 - z^{-1}}$$

with T the sampling time.
Other discretization methods can be used, here is no significant difference.

Digital formulation combined

Digital PID controller

$$u[k] = K_p e[k] + \frac{K_d}{T}(e[k] - e[k-1]) + u_i[k]$$

$$\text{with } u_i[k] = u_i[k-1] + K_i T e[k]$$

In z-domain:

$$\frac{U(z)}{E(z)} = K_p + \frac{K_d}{T}\frac{z-1}{z} + K_i T \frac{z}{z-1}$$

where $\frac{K_d}{T}$ and $K_i T$ are the new derivative and integral gains.

Also controllers where the integral or derivative function is omitted exist. These are called PI or PD controllers respectively.

Alternative Digital PID controller

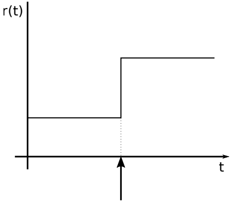
We can also discretize using the **bilinear transformation**:

$$\begin{aligned}\frac{U(z)}{E(z)} &= K_p + \frac{K_i}{s} + K_d s\bigg|_{s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)} \\ &= K_p + \frac{K_i T(z+1)}{2(z-1)} + \frac{2K_d(z-1)}{T(z+1)} \\ &= \frac{\alpha_2 z^2 + \alpha_1 z + \alpha_0}{(z-1)(z+1)}\end{aligned}$$

where $\alpha_2, \alpha_1, \alpha_0$ are design parameters.

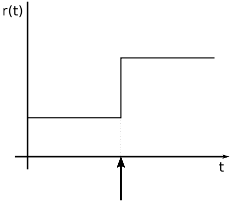
Alterantive Derivative Action

Imagine a jumping set point or rapidly changing signal. This results in a theoretically infinite, practically very large response of the derivative term.



Alterantive Derivative Action

Imagine a jumping set point or rapidly changing signal. This results in a theoretically infinite, practically very large response of the derivative term.



$\Rightarrow$ Add a low-pass filter to the derivative term:

$$\frac{U_d(s)}{E(s)} = \frac{K_d s}{1 + s\tau}$$

With $s = j\omega$, breakpoint at $\omega = 1/\tau$. This prevents amplification of high frequencies.

Alterantive Derivative Action

$$\frac{U_d(s)}{E(s)} = \frac{K_d s}{1 + s\tau}$$

Further $e(t)$ is replaced by $c \cdot r(t) - y(t)$ with c the set point weighting, which is often set to zero to further reduce immediate influence of a sudden set point jump.
In time domain:

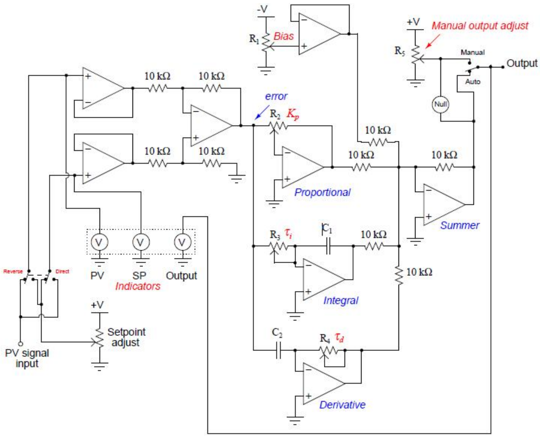
$$u_d(t) = -\tau \frac{du_d}{dt} + K_d(c \cdot r(t) - y(t))$$

This can also be discretized, but the bilinear method then introduces *ringing*, i.e. large oscillations in transient response.

Outline

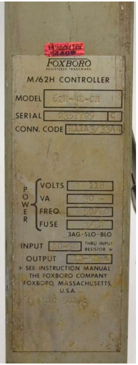
Analog Implementation

The key building block in the op-amp



- PV - Process Variable $y(t)$
- SP - Set Point $r(t)$
- Output - Control action $u(t)$

Analog Implementation



FOXBORO 62H-4E-OH M/62H

Digital Implementation

The difference equations are typically implemented in a micro controller or FPGA:

$$u[k] = K_p e[k] + \frac{K_d}{T}(e[k] - e[k-1]) + u_i[k]$$

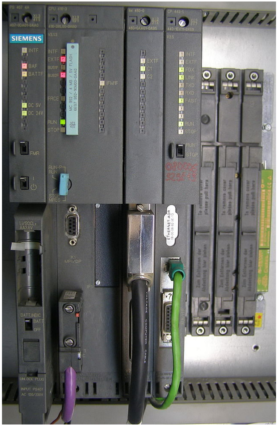
$$\text{with } u_i[k] = u_i[k-1] + K_i T e[k]$$

Steps to be implemented:

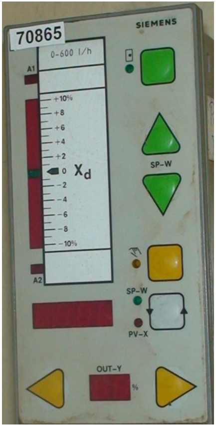
1. Wait for clock interrupt
2. Read analog input
3. Compute control signal
4. Set analog output
5. Update controller variables
6. Go to 1

Digital Implementation Example

PLC with a digital PID module:



Digital PID's:



Outline

Manual Tuning

Effects of adjusting the parameters K_p, K_i, K_d:

PID gains	Rise Time	Overshoot	Setlling time	Steady-State error
$K_p \uparrow$	Decrease	Increase	Small Change	Decrease
$K_i \uparrow$	Decrease	Increase	Increase	Eliminate
$K_d \uparrow$	Small change	Decrease	Decrease	No change

Note: Changing one parameter can influence the effect of the other two. Use this table only as an indication.

Manual Tuning

In case the controller can be tuned while connected to the plant, following routine can be used:

1. Set K_i and K_d equal to 0
2. Increase K_p until you observe that the step response is fast enough and the steady-state error in small
3. Start adding some integral action in order to get rid of the steady state error. Keep in mind that too much K_i can cause instability!
4. Add some derivative action in order to quickly react to disturbance and/or dampen the response

Heuristic Methods: Ziegler-Nichols rule

This method relies on empirically determining two parameters of the system (should again be practical possible):

1. Set the integral and derivative gains to 0
2. Increase the proportional gain K_p until the output of the control loop starts oscillating with at constant amplitude. The value of K_p at this point is referred to as ultimate gain $K_u \triangleq K_p$
3. Measure the period of the oscillations T_u at the output

Heuristic Methods: Ziegler-Nichols rule

With K_u and T_u determined like in the previous slide, a starting point for the parameters can be determined:

Control Type	K_p	K_i	K_d
P	$0.5K_u$	-	-
PI	$0.45K_u$	$1.2K_p/T_u$	-
PD	$0.8K_u$	-	$K_pT_u/8$
PID	$0.6K_u$	$2K_p/T_u$	$K_pT_u/8$
Pessen Integral Rule	$0.7K_u$	$2.5K_p/T_u$	$3K_pT_u/20$
Some overshoot	$0.33K_u$	$2K_p/T_u$	$K_pT_u/3$
No overshoot	$0.2K_u$	$2K_p/T_u$	$K_pT_u/3$

Numeriacal Optimization Methods