Chapter 13 - PID Controllers

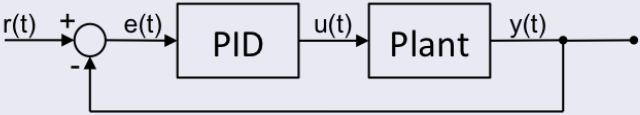
August 7, 2015

Outline

What is a PID controller?

Definition

A **P**roportional-**I**ntegral-**D**erivative (PID) controller is a control-loop feedback mechanism (controller) widely used in process industry.



Continuous-time text book equation:

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

Note: 90% (or more) of control-loops in process industry are PID.

What is a PID controller?

- **P**roportional action $u_p(t) = K_p e(t)$: it depends on the instaneous value of the error.
  - $+$ Reduces rise time
  - $+$ Reduces but **does not eliminate the steady-state error**: Only when $K \to \infty$, error $\to 0$ (unless the plant has pole(s) at $s = 0$)
- **I**ntegral action $u_i(t) = K_i \int_0^t e(\tau) d\tau$: it is proportional to the accumulated error.
  - $+$ **Eliminates the steady-state error** in some cases
  - $-$ Makes transient response slower
- **D**erivative action $u_d(t) = K_d \frac{de(t)}{dt}$: it is proportional to the rate of change of the error.
  - $+$ Increases the stability of the system, reduces overshoot, improves the transient response
  - $-$ Amplifies the noise present in the error signal

Outline

Proportional Control

The continuous-time and discrete-time implementation are identical.

For the continuous-time case we have:

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

and for the discrete-time case:

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

where e(t) or e[k] is the error signal.

Derivative Control

In continuous-time it is given by:

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

and in discrete-time by (using backward Euler):

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

with T_s the sampling time.

Integral Control

In continuous-time it is given by:

$$u_i(t) = K_i \int_0^t e(\tau)d\tau \quad \rightarrow \quad \dot{u}_i(t) = K_i e(t) \quad \rightarrow \quad \frac{U_i(s)}{E(s)} = \frac{K_i}{s}$$

and in discrete-time by (using backward Euler)

$$u_i[k] = u_i[k-1] + K_i T_s e[k] \quad \rightarrow \quad \frac{U_i(z)}{E(z)} = K_i \frac{z T_s}{z-1}$$

with T_s the sampling time.

Digital formulation (conventional version)

Digital PID controller (conventional version)

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

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

In the $\mathcal{Z}$-domain:

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

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

Digital PI controller

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

Digital PD controller

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

Alternative Digital PID controller

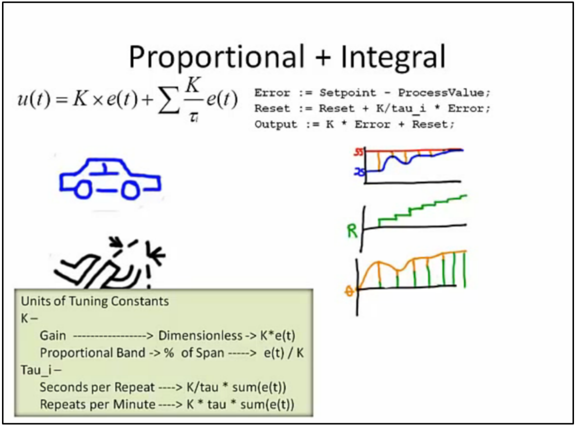
If we discretize the continuous-time (analog) PID controller using the bilinear transformation,

$$\frac{U(z)}{E(z)} = K_p + \frac{K_i}{s} + K_d s \bigg|_{s=\frac{2}{T_s}\left(\frac{z-1}{z+1}\right)}$$

we obtain an alternative form for a digital PID controller

$$\begin{aligned}\frac{U(z)}{E(z)} &= K_p + \frac{K_i T_s(z+1)}{2(z-1)} + \frac{2K_d(z-1)}{T_s(z+1)} \\ &= \frac{\alpha_2 z^2 + \alpha_1 z + \alpha_0}{(z-1)(z+1)}\end{aligned}$$

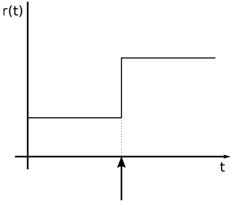
where $\alpha_2, \alpha_1, \alpha_0$ are design parameters.



https://www.youtube.com/watch?v=JEpWlTl95Tw

Alterantive Derivative Action (Continous-time)

Imagine a step change in reference signal $r(t)$. 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 (Continous-time)

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

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.

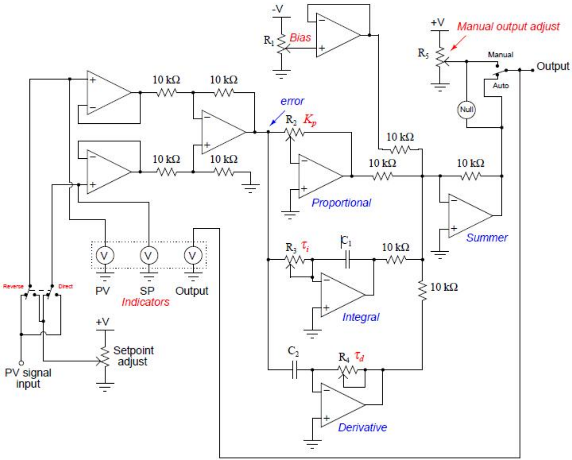
In the time-domain:

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

Outline

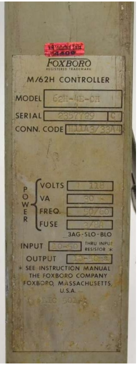
Analog Implementation

The key building block is the operational amplifier (op-amp).



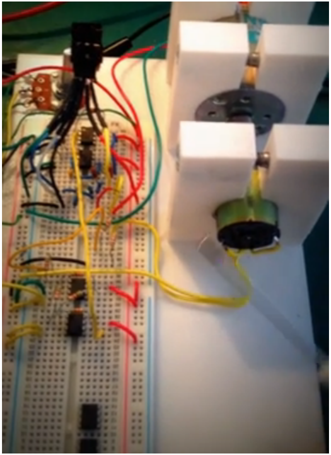
- PV - Process Variable $y(t)$
- SP - Set Point $r(t)$
- Output - Control action $u(t)$

Analog Implementation



Analog PID controller: FOXBORO 62H-4E-OH M/62H

Analog PI Motor Speed Control



https://youtu.be/6W3PLiVIcmE

Digital Implementation

The difference equations are typically implemented in a microcontroller in an FPGA (field-programmable gate array device):

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

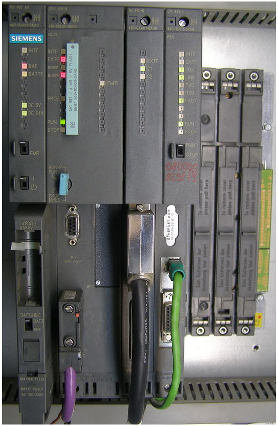
$$\text{with } u_i[k] = u_i[k-1] + K_i T_s e[k]$$

Pseudocode

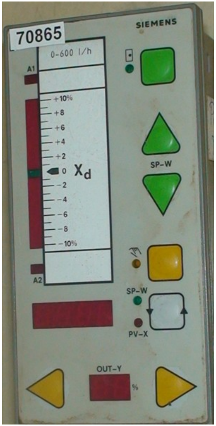
```
previous_error = 0
integral = 0
Start:
error = setpoint - measured_value
proportional = KP * error
integral = integral + Ki*sampling_time*error
derivative = Kd*(error-previous_error)/sampling_time
output = proportional + integral + derivative
previous_error = error
wait (sampling_time)
goto Start
```

Digital Implementation

PLC with a digital PID module:



Digital PID:



PLC

A Programmable **L**ogic **C**ontroller (PLC) is a digital computer used for automation in process industry.

What is a PLC? Basics of PLCs

https://youtu.be/iWgHqqunsyE

Outline

Manual Tuning

The effects of each of the controller parameters K_p, K_i and K_d on the closed-loop system are summarized in the table below.

	Closed-loop response			
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

Important note

Changing one parameter can influence the effect of the other two. Therefor, use this table only as a reference.

Manual Tuning

One possible way is as follows (the controller is connected to the plant):

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 disturbances and/or dampen the response.

Heuristic Methods

When the **mathematical model of a plant is unknown** or it is too complicated to obtain, an analytical approach to design a PID controller is not possible. Therefore in these cases we have to use an experimental approach to tune the PID controller. Among the existing experimental approaches we can mention:

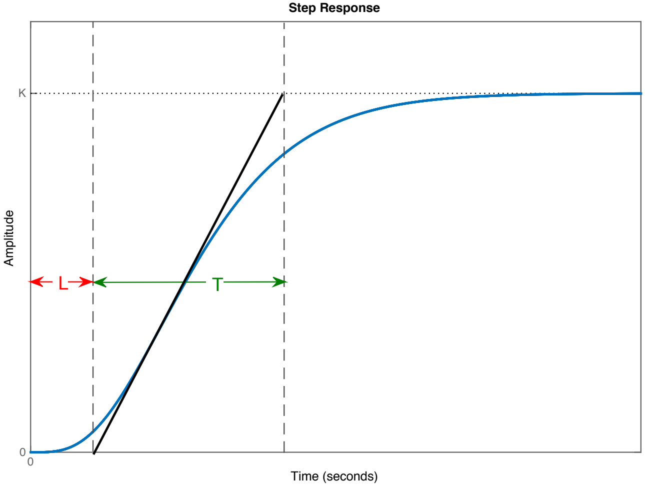
- Ziegler-Nichols tuning rule based on step response (First method)
- Ziegler-Nichols tuning rule based on critical gain and critical period (Second method)

Ziegler-Nichols tuning rule: First Method

In this method, the response of the plant to a unit-step input is obtained experimentally. If the plant involves neither integrators nor dominant complex-conjugate poles, then such a unit-step response curve may look S-shaped as shown in the figure on the next slide. This S-shaped curve may be characterized by two constants: delay time L and time constant T. The delay time and the time constant are determined by drawing a tangent line at the inflection point of the S-shaped curve and determining the intersections of the tangent line with the time axis and a horizontal line crossing the y-axis at point (0,K).

If the response to a step input does not exhibit an S-shaped curve, the first method does not apply.

Ziegler-Nichols tuning rule: First Method



Ziegler-Nichols tuning rule: First Method

Ziegler and Nichols suggested to set the values of K_p, K_i, and K_d according to the formulas shown in the following table:

Control Type	K_P	K_I	K_D
P	$\frac{T}{L}$	0	0
PI	$0.9\frac{T}{L}$	$K_p\frac{0.3}{L}$	0
PID	$1.2\frac{T}{L}$	$\frac{K_p}{2L}$	$0.5LK_P$

Ziegler-Nichols tuning rule: First Method

The PID controller tuned by the first method of Ziegler-Nichols rules gives:

$$\begin{aligned}\frac{U(s)}{E(s)} &= K_P + \frac{K_I}{s} + K_D s \\ &= 1.2\frac{T}{L}\left(1 + \frac{1}{2Ls} + 0.5Ls\right) \\ &= 0.6T\frac{\left(s+\frac{1}{L}\right)^2}{s}\end{aligned}$$

with a pole at the origin and a double zero at $s = -\frac{1}{L}$.

Ziegler-Nichols tuning rule: Second Method

This method is based on the value of K_p that results in marginal stability when only proportional control action is used.

1. Set the integral and derivative gains to zero ($K_p = K_d = 0$);
2. Increase the proportional gain K_p until the output of the control loop starts oscillating with a 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 of the closed-loop system;
4. Use K_u and T_u to determine the gains of the PID controller according to the tuning rule table shown in the next slide.

Ziegler-Nichols tuning rule: Second Method

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	$0.2K_u$	$2K_p/T_u$	$K_pT_u/3$

If the output does not exhibit sustained oscillations for whatever value K_P may take, then the second method does not apply.

Keep in mind that we are working with heuristic tuning rules, and therefore some additional fine tuning might be necessary.

Ziegler-Nichols tuning rule: Sedond Method (example)

Example

Consider a plant with a given model:

$$P(s) = \frac{1}{(s+1)^3}$$

- We compute the critical gain K_u. This is the value of K_p for which $\angle(K_p P(s)) = -180^\circ$. On the Nyquist plot this is the value of K_p for which $K_p P(s)$ passes through $(-1, 0)$.

$$K_u P(j\omega_u) = -1$$

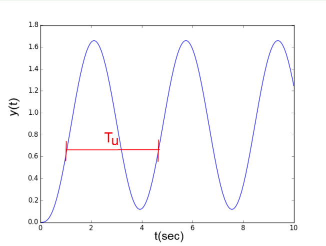
$$\Leftrightarrow K_u = -(j\omega_u + 1)^3$$

$$= (3\omega_u^2 - 1) + j(\omega_u^3 - 3\omega_u)$$

$$\omega_u^3 - 3\omega_u = 0 \Rightarrow \omega_u = \sqrt{3}$$

$$K_u = 8,\ T_u = \frac{2\pi}{\omega} = 3.628$$

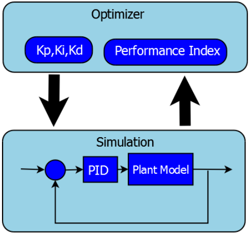
$$K_p = 4.8,\ K_I = 2.6448 K_p,\ K_d = 2.16$$



Numerical Optimization Methods

The tuning of a PID controller is posed as a constrained optimization problem.

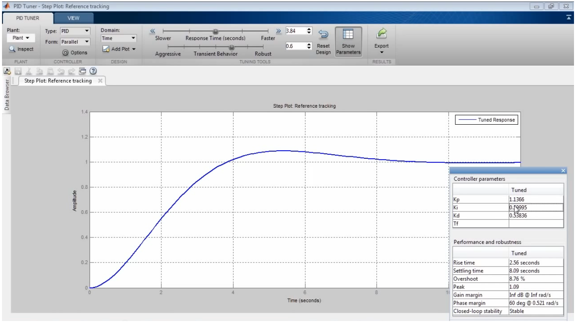
- For a given set of parameters K_p, K_i and K_d run a simulation of the closed-loop system, and compute some performance parameters (e.g. settling time, rise time, etc.) and a performance index.
- Optimize the performance index over the three PID gains.



Some Software Tools

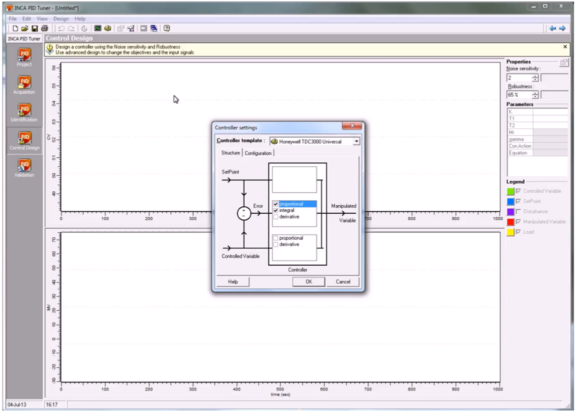
Software Tool	Brief Description
pidtool / pidTuner	It is a Matlab tool to interactively design a SISO PID controller in the feed-forward path of single-loop, unity-feedback control configuration
Pidpy	It is a modular PID control library for python that supports PID auto tuning. https://pypi.python.org/pypi/pypid/
INCA PID Tuner	It is a commercial tuning tool developed by IPCOS. It has a vast library of PID structures for DCS and PLC Systems including Siemens, ABB, Honeywell, Emerson, etc. http://www.ipcos.com/advancedprocesscontrol/advanced-process-control/pid-tuning-software/inca-pid-tuning/

pidtool /pidTuner - Demo



https://www.youtube.com/watch?v=2tKe0caUv1I

INCA PID Tuner - Demo



https://www.youtube.com/watch?v=XH2bkq1URSg