

## Chapter 13 - PID Controllers

August 6, 2015

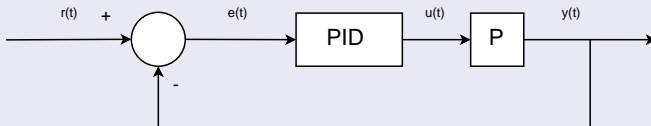
# Outline

- 1 Introduction
- 2 Analog and Digital formulations
- 3 Implementation Examples
- 4 PID Tuning

# What is a PID controller?

## Definition

A **P**roportional **I**ntegral **D**erivative 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:** More than 90% of all closed loop controllers are PID.

# What is a PID controller?

- **Proportional action**  $u_p(t) = K_p e(t)$ : Action depends on the instantaneous value of the control error.
  - + Reduces rise time
  - Reduces but **does not eliminate steady-state error**: Only when  $K \rightarrow \infty$ , error  $\rightarrow 0$  (unless plant has pole(s) at  $s = 0$ )
- **Integral action**  $u_i(t) = K_i \int_0^t e(\tau) d\tau$ : Gives a controller output that is proportional to the accumulated error. Reacts on constant errors
  - + **Can eliminate steady state error** in some cases
  - Makes transient response slower
- **Derivative action**  $u_d(t) = K_d \frac{de(t)}{dt}$ : Acts on the rate of change of the control error.
  - + Damping effect: reduces overshoot, improves transient response
  - Sensitive for noise, amplifies it if present

# Outline

- 1 Introduction
- 2 Analog and Digital formulations
- 3 Implementation Examples
- 4 PID Tuning

# Proportional Control

The continuous-time and discrete implementation are identical  
Continuous:

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

Discrete:

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

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

# Derivative Control

Continuous:

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

Discrete (using **backward Euler**):

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

with  $T_s$  the sampling time.

# Integral Control

The continuous equation is:

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

Differentiating this gives :

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

Then applying backward Euler:

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

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 z-domain:

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

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

## Digital PI controller

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

## Digital PD controller

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

## Alternative Digital PID controller

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

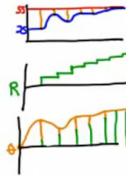
$$\begin{aligned}\frac{U(z)}{E(z)} &= K_p + \frac{K_i}{s} + K_d s \bigg|_{s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)} \\ &= K_p + \frac{K_i T(z+1)}{2(z-1)} + \frac{2K_d(z-1)}{T(z+1)} \\ &= \frac{\alpha_2 z^2 + \alpha_1 z + \alpha_0}{(z-1)(z+1)}\end{aligned}$$

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

## Proportional + Integral

$$u(t) = K \times e(t) + \sum \frac{K}{\tau_i} e(t)$$

Error := Setpoint - ProcessValue;  
 Reset := Reset + K/tau\_i \* Error;  
 Output := K \* Error + Reset;



Units of Tuning Constants

K -

Gain -----> Dimensionless ->  $K \cdot e(t)$

Proportional Band -> % of Span ----->  $e(t) / K$

Tau\_i -

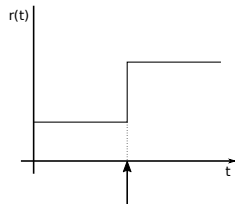
Seconds per Repeat ----->  $K / \tau_i \cdot \sum(e(t))$

Repeats per Minute ----->  $K \cdot \tau_i \cdot \sum(e(t))$

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

# Alternative Derivative Action(Continuous time)

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



⇒ 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 time domain:

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

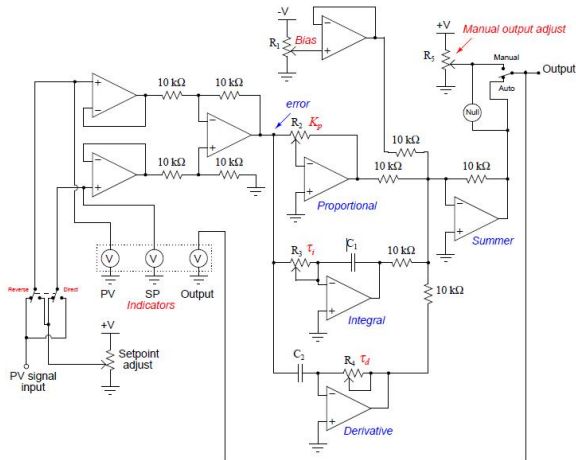
This can also be discretized, but the bilinear method then introduces *ringing*, i.e. large oscillations in transient response.

# Outline

- 1 Introduction
- 2 Analog and Digital formulations
- 3 Implementation Examples**
- 4 PID Tuning

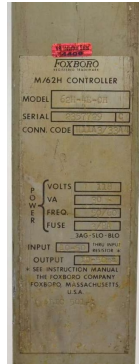
# Analog Implementation

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



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

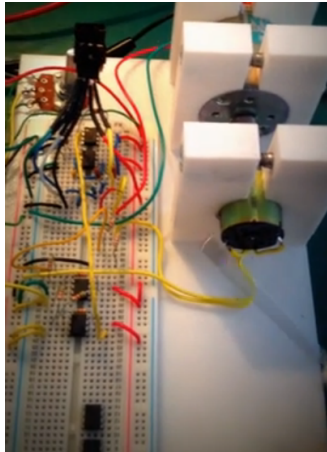
# Analog Implementation



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



# Analog PI Motor Speed Control



<https://youtu.be/6W3PLiVlcmE>

# Digital Implementation

The difference equations are typically implemented in a micro controller or FPGA (field-programmable gate array):

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

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

Steps to be implemented:

```
previous_error = 0
```

```
integral = 0
```

```
Start:
```

```
    error = setpoint - measured_value
```

```
    proportional = K_P * error
```

```
    integral = integral + K_i*sampling_time*error
```

```
    derivative = K_d*(error-previous_error)/sampling_time
```

```
    output = proportional + integral + derivative
```

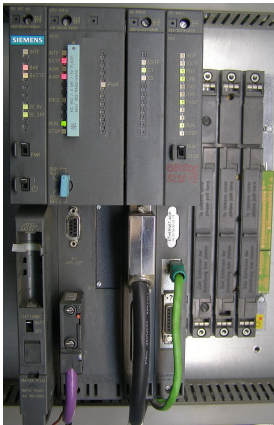
```
    previous_error = error
```

```
    wait(sampling_time)
```

```
    goto Start
```

# Digital Implementation Example

PLC with a digital PID module:



Digital PID's:



# PLC

**P**rogrammable **L**ogic **C**ontroller is a digital computer used for automation in the process industry.



# What is a PLC? Basics of PLCs



<https://youtu.be/iWgHqqunsyE>

# Outline

- 1 Introduction
- 2 Analog and Digital formulations
- 3 Implementation Examples
- 4 PID Tuning

# Manual Tuning

Effects of adjusting the parameters  $K_p$ ,  $K_i$ ,  $K_d$ :

PID gains	Rise Time	Overshoot	Settling 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

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 is 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 tuning rule

This method relies on empirically determining two parameters of the system:

- 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
- 4 Adjust the controller parameters according the table on the next slide.

## Heuristic Methods: Ziegler-Nichols tuning 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	$0.8K_u$	-	$K_p T_u/8$
PID	$0.6K_u$	$2K_p/T_u$	$K_p T_u/8$
Pessen Integral Rule	$0.7K_u$	$2.5K_p/T_u$	$3K_p T_u/20$
Some overshoot	$0.33K_u$	$2K_p/T_u$	$K_p T_u/3$
No overshoot	$0.2K_u$	$2K_p/T_u$	$K_p T_u/3$

# Heuristic Methods: Ziegler-Nichols tuning rule(example)

## Example

Consider a plant with a given model:

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

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

$$K_c P(j\omega_c) = -1$$

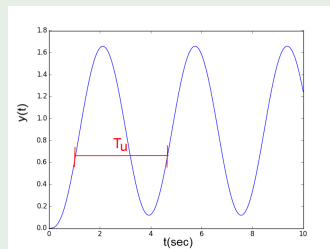
$$\Leftrightarrow K_c = -(j\omega_c + 1)^3$$

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

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

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

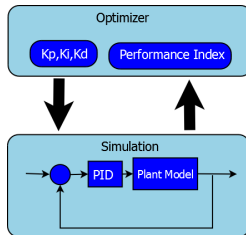
$$K_p = 4.8, K_I = 0.551K_p, K_d = 0.45K_p$$



# Numerical Optimization Methods

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

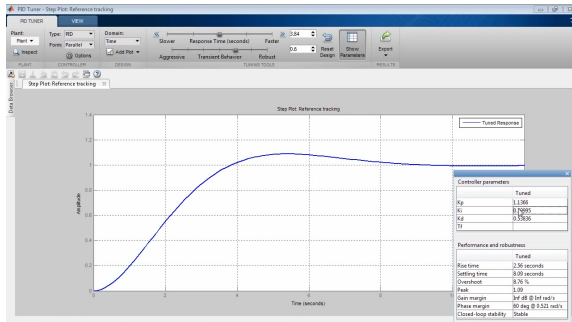
- For a given set of parameters  $K_p$ ,  $K_i$  and  $K_d$  run a simulation of the closed-loop system, and compute some performance parameters (e.g. setting time, rise time, etc.) and a performance index.
- Optimize the performance index over the three PID gains.



## Some Software Tools

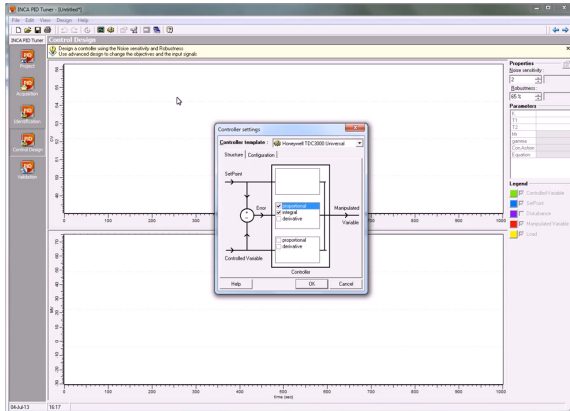
Software Tool	Brief Description
pidtool / pidTuner	It is a Matlab tool to interactively design a SISO PID controller in the feed-forward path of single-loop, unity-feedback control configuration
Pidpy	It is a modular PID control library for python that supports PID auto tuning. <a href="https://pypi.python.org/pypi/pypid/">https://pypi.python.org/pypi/pypid/</a>
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. <a href="http://www.ipcos.com/advancedprocesscontrol/advanced-process-control/pid-tuning-software/inca-pid-tuning/">http://www.ipcos.com/advancedprocesscontrol/advanced-process-control/pid-tuning-software/inca-pid-tuning/</a>

# pidtool / pidTuner - Demo



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

# INCA PID Tuner Demo



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