Presentation Overview

- Introduction
- PID parameterisation and structure
- Effects of PID terms
  - Proportional, Integral and Derivative terms
- Tuning PID Controllers
- Equivalence of PID and Lead-Lag controllers
- Implementation aspects of PID
Introduction to PID Control

- Proportional-Integral-Derivative control
- Predominant controller used in industry
- Reason: it is adequate for most applications
- Initially: analogue implementation
- Now: mostly digital
Motivation and Limitations

Motivation

- Simple to get working
- Can be tuned to meet time-domain specifications
- Readily available in PLC/DCS systems
- Digital and analogue implementation easy

Limitations

- For single-input single-output systems only
- Difficult to tune to meet precise specifications
- Subtle differences in implementation causes problems
Generic Equations and Tracking Errors

- Closed-loop TF from reference to error

\[ e(s) = \frac{1}{1 + G(s)K(s)} r(s) \]

- Type 0 servo, unit-step steady-state error

\[ e_{ss} = \lim_{s \to 0} s \left( \frac{1}{1 + G(s)K(s)} \right) \frac{1}{s} = \frac{1}{1 + k_p} \]

- Type 1 servo, unit-ramp steady-state error

\[ e_{ss} = \lim_{s \to 0} \frac{1}{s^2} \left( \frac{1}{1 + G(s)K(s)} \right) = \frac{1}{k_y} \]
Idealised PID-Controller Configuration

Set-Point $r(s)$ + $e(s)$ $\rightarrow$ $u(s)$ $\rightarrow$ $G_p(s)$ $\rightarrow$ $y(s)$

Controller $K(s)$

Plant

Output
Generic PID Control Equations

- **Time domain:**
  \[ u_c(t) = k_1 e(t) + k_2 \int_0^t e(t) dt + k_3 \frac{de}{dt} \]

- **Laplace domain:**
  \[ U_c(s) = k_1 E(s) + k_2 \frac{E(s)}{s} + k_3 sE(s) \]

- However there are many different structures ...
"Ideal" PID Parameterisation

\[ u_c(t) = K_c \left[ e(t) + \frac{1}{T_i} \int_0^t e dt + T_d \frac{de}{dt} \right] \]

- \( K_c \): proportional coefficient
- \( T_i \): integral time constant
- \( T_d \): derivative time constant

Common in PLC and DCS implementations
“Parallel” PID Parameterisation

\[ u_c(t) = K_p e(t) + K_i \int_0^t e dt + K_d \frac{de}{dt} \]

- \( K_p \): proportional gain \( = K_c \)
- \( K_i \): integral gain \( = K_c / T_i \)
- \( K_d \): derivative gain \( = K_c T_d \)

Used in SW tools (e.g. Matlab) and some industrial systems
“Series” PID Parameterisation

\[ u_c(t) = K_c \left[ 1 + \frac{1}{T_i} \int_0^t (.) dt \right]\left[ e(t) + T_d \frac{de}{dt} \right] \]

- \( K_c \): proportional coefficient
- \( T_i \): integral time constant
- \( T_d \): derivative time constant

- Is found in some industrial systems
- Historically popular since can be implemented with just one op-amp
PID Parameterisation

- There are also differences in units for gain terms

- Proportional gain:
  - as a pure gain or proportional band ( = 100%/gain)

- Integral gain:
  - as reset (i.e. $K_i$, units of repeats per second or minute)
  - or integral time (i.e $T_i$, units of seconds or minutes)

- Derivative gain:
  - as derivative time (i.e $T_d$, units of seconds or minutes)
The proportional factor $K_p$ generates an output proportional to the error, it requires a non-zero error to produce the command variable.

Increasing the amplification $K_p$ decreases the error, but may lead to instability.

The integral time constant $T_i$ produces a non-zero control variable even when the error is zero, but makes the system unstable (or slower).

The derivative time $T_d$ speeds up response by reacting to an error change with a control variable proportional to the steepness of change.

Acknowledgement: Prof. Dr. H. Kirrmann, EPFL / ABB Research Center, Baden, Switzerland
Traditionally the PID terms act on the error signal, as in previous equations.

In process applications it is common to have derivative acting on the output rather than error:
- Called “PI-D” or “Derivative on PV”

In some cases the proportional control can act on the loop output as well:
- Called “I-PD” or “SP on I-only”

**Note:** if the set-point is constant then all three forms are equivalent, they only differ when the set-point changes ….
PID Control Action

Response of a PID acting on the error signal

The big spike in controller output is due to the set-point step being fed directly through derivative.
Derivative often used in feedback path only
• i.e PI-D control

\[ u_c(s) = K_c(e(s) - T_d s y(s)) \]

Eliminating \( e(s) \), using \( e(s) = r(s) - y(s) \), gives .....
Derivative - on Loop Output

Derivative often used in feedback path only

- i.e PI-D control

\[ u_c(s) = K_c r(s) - K_c \left( y(s) + T_d s y(s) \right) \]

![Block diagram of a control system with derivative action](image)

\[ CLTF = \frac{G K_c}{1 + G K_c (1 + T_d s)} \]
PI-D Control Action

- Response of a PI-D (i.e. Derivative on PV)

The spike in controller output is smaller - derivative “kick” has been eliminated.
I-PD Control Action

- Response of an I-PD (i.e. Proportional & Derivative on PV)

I-PID Control should **not** be used on integrating processes (e.g. level control)

No spike in controller output - both proportional and derivative “kick” eliminated
These many different forms of PID do cause problems:

- each has different behaviour
- each require different tuning rules
- you must know which form is used before doing any tuning, design and/or simulation

However, it can provide an additional degree of freedom when designing a control system
Effects of P, I & D Terms

- **Proportional Action**
  - First-order plant
  - Second-order plant

- **Integral Action**
  - First-order plant

- **Derivative Action**
  - First-order plant
  - Second-order plant
  - Rate feedback
First-order plant:

\[ G(s) = \frac{1}{s + a} \]

- Pole at \( s = -a \), steady-state gain \( 1/a \)

Closed-loop TF:

\[
\frac{y(s)}{r(s)} = \frac{G(s)K(s)}{1 + G(s)K(s)} = \frac{K_c}{s + a + K_c}
\]

- New pole \( s = -(a + K_c) \)

Steady-state error with step input:

However, proportional gain **cannot** just be increased arbitrarily to remove steady-state error. Problems arise with actuator saturation and instability with real, higher-order plant
Proportional - 1st Order Plant

Response to unit step set-point change:
Proportional - 2nd Order Plant

- Second-order plant: 
  \[ G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \]

- Closed-loop TF: 
  \[ \frac{y(s)}{r(s)} = \frac{\omega_n^2 K_c}{s^2 + 2\zeta\omega_n s + \omega_n^2 (K_c + 1)} \]

- As \( K_c \) increases:
  - CL natural frequency increases
  - CL damping ratio decreases \textbf{overshoot increases}
  - steady state error decreases
Proportional - 2nd Order Plant
Steady-State Error

\[ e_{ss} = \lim_{s \to 0} s \frac{1}{s} \frac{1}{1 + G(s)K(s)} = \lim_{s \to 0} \frac{1}{s} \frac{1}{1 + \frac{\omega_n^2 K_c}{s^2 + 2\zeta \omega_n s + \omega_n^2}} = \frac{1}{1 + K_c} \]

- If \( K_c \) larger, error smaller
Proportional - 2nd Order Plant

Response to unit step set-point ($\zeta = 0.4, \omega_n = 3$)
Integral Action

- Objective is to remove steady-state error

- PI controller in time domain: 
  \[ u_c(t) = K_c \left( e + \frac{1}{T_i} \int_0^t e dt \right) \]

- Laplace domain: 
  \[ K(s) = K_c \left( 1 + \frac{1}{T_i s} \right) \]

- With a constant error: 
  \[ u_c(t) = K_c \left( e + \frac{1}{T_i} \int_0^t e dt \right) = K_c \left( e + \frac{e}{T_i} \int_0^t dt \right) = K_c \left( e + \frac{et}{T_i} \right) \]

That is, if a constant error exists the controller output will keep increasing, until the error is zero ……
Integral Action - Constant Error

The diagram illustrates the behavior of an integral control system. Over time, the integral output accumulates with each input error, as shown by the slopes of the lines. The proportional output is also depicted, showing how it changes with time.
**Integral Action - 1st Order Plant**

- **Plant:** \( G(s) = \frac{1}{s + a} \)

- **Controller:** \( K(s) = \frac{1}{T_i s} \)

- **Closed-loop TF:**
  \[
  \frac{y(s)}{r(s)} = \frac{1}{T_i s^2 + a T_i s + 1} = \frac{\omega_{ncl}^2}{s^2 + 2 \zeta_{cl} \omega_{ncl} s + \omega_{ncl}^2}
  \]

- **CL natural frequency and damping ratio now functions of** \( T_i \)

- **Steady state error now removed:**
  \[
  e_{ss} = \lim_{s \to 0} \frac{1}{1 + \frac{1}{s + a} \frac{1}{T_i s}} = \lim_{s \to 0} \frac{s}{s + \frac{1}{s + a} \frac{1}{T_i}} = 0 + \frac{1}{a T_i} = 0
  \]

**Integral Action - 1st Order Plant**

Response to unit step set-point change:
Objective: stabilise system; slow down transients

PD controller: \[ u_c(t) = K_c \left( e(t) + T_d \frac{de}{dt} \right) \]

In Laplace domain: \[ K(s) = K_c \left( 1 + T_d s \right) \]

With constant error, derivative action = 0
- no contribution to steady-state behaviour

With transients on error, derivative action = large
- risk actuator saturation from measurement noise, step changes in set-point
Derivative - 1st Order Plant

Plant: \( G(s) = \frac{1}{s + a} \)

Controller: \( K_{ff}(s) = K_c \)
\( K(s) = K_c \left(1 + T_d s\right) \)

Closed-loop TF: \( \frac{y(s)}{r(s)} = \frac{K_c}{(1 + K_c T_d)s + a + K_c} \)

Closed-loop pole: \( s = -\frac{(a + K_c)}{(1 + K_c T_d)} \)

- as \( T_d \) is increased, CL response gets slower

Steady state error not effected by \( T_d \):
\[
e_{ss} = \lim_{s \to 0} \frac{1}{1 + \frac{1}{s + a} K_c (1 + T_d s)} = \frac{1}{1 + \frac{K_c}{a}}
\]
Derivative - 1st Order Plant

Response to unit step set-point change:

![Step Response Graph]

- **Time (sec.)**
- **Amplitude**

- $T_d = 1$
- $T_d = 2$
- $T_d = 5$
Derivative - 2nd Order Plant

- Generalised Plant:
  \[ G(s) = \frac{\omega_{np}^2}{s^2 + 2\zeta_p \omega_{np} s + \omega_{np}^2} \]

- Closed-loop TF:
  \[ \frac{y(s)}{r(s)} = \frac{K_c \omega_{np}^2}{s^2 \left(2\zeta_p \omega_{np} + K_c \omega_{np}^2 T_d\right) s + (1 + K_c) \omega_{np}^2} \]

- CL natural frequency:
  \[ \omega_{ncl} = \omega_{np} \sqrt{1 + K_c} \]

- CL damping ratio:
  \[ \zeta_{cl} = \frac{\zeta_p}{\sqrt{1 + K_c}} + \frac{K_c \omega_{np}}{2\sqrt{1 + K_c}} T_d \]

- Steady-state error is not effected by \( T_d \)
Derivative - 2nd Order Plant

Steady-state error:

\[
e_{ss} = \lim_{s \to 0} \frac{1 + \frac{\omega_{np}^2}{s^2 + 2\zeta_p \omega_{np} s + \omega_{np}^2} K_c T_d s}{1 + \frac{\omega_{np}^2}{s^2 + 2\zeta_p \omega_{np} s + \omega_{np}^2} K_c (1 + T_d s)} = \frac{1}{1 + K_c}
\]
Derivative - 2nd Order Plant

Response to unit step set-point change:
Derivative - as Rate Feedback

- Implementation using direct measure of derivative
- Similar to P-D control
  - with $K_r = K_c T_d$

\[
CLTF = \frac{G K_c}{1 + G (K_c + K_r s)}
\]
Tuning PID Controllers

- Fundamental Trade-offs
- Applicability of PID
- Process Reaction Tuning
- Sustained Oscillation Tuning
- IMC PID Tuning
- Advanced Tuning Methods
**Fundamental Trade-offs**

- **Set-point tracking:**
  - Good tracking performance $\rightarrow$ $K(s)G(s)$ large

- **Stability:**
  - Keep $K(s)G(s)$ away from -1 $\rightarrow$ limit on $K(s)$

- **Disturbance rejection:**
  - Reduce effect of disturbances $\rightarrow$ $K(s)$ large

- **Noise immunity:**
  - Noise should not excite $u(s)$ $\rightarrow$ limit on $K(s)$
Applicability of PID

- PI controllers adequate for most applications
- Derivative action is rarely utilised
  - often misunderstood due to the many different forms and the different ways in which they work
- For dominant second (and higher) order plants
  PID control may be more appropriate
  - additional D term can provide damping to reduce overshoot
  - D can be beneficial for slow thermal processes
The following generalisation is from Shinsky ("Process Control Systems", McGraw Hill, 1979)

There will always be exceptions, so need tuning rules ...

## Applicability of PID

<table>
<thead>
<tr>
<th>Process</th>
<th>Proportional Band (%)</th>
<th>Integral</th>
<th>Derivative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow</td>
<td>100-500</td>
<td>essential</td>
<td>no</td>
</tr>
<tr>
<td>Pressure, liquid</td>
<td>50-200</td>
<td>essential</td>
<td>no</td>
</tr>
<tr>
<td>Pressure, gas</td>
<td>0-5</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Level</td>
<td>5-80</td>
<td>seldom</td>
<td>no</td>
</tr>
<tr>
<td>Vapour (T and p)</td>
<td>10-100</td>
<td>yes</td>
<td>essential</td>
</tr>
<tr>
<td>Chemical Composition</td>
<td>100-1000</td>
<td>essential</td>
<td>if possible</td>
</tr>
</tbody>
</table>
Process Reaction Tuning

- Plant in open loop with and input $u_{nom}$

- Apply a step to input ($u_{step}$) and record the response

\[
\Delta y = \frac{y_{step} - y_{nom}}{\Delta T}
\]

- Determine $L$ and $R$ from the response
Process Reaction Tuning

- Normalise $R$:

\[ R_N = \frac{R}{u_{step} - u_{nom}} \]

- Use $L$ and $R_N$ in this table to give PID settings

<table>
<thead>
<tr>
<th>Controller</th>
<th>$K_p$</th>
<th>$T_i$</th>
<th>$T_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>$1.0/(R_N L)$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>P+I</td>
<td>$0.9/(R_N L)$</td>
<td>$3L$</td>
<td>-</td>
</tr>
<tr>
<td>P+I+D</td>
<td>$1.2/(R_N L)$</td>
<td>$2L$</td>
<td>$1/(2L)$</td>
</tr>
</tbody>
</table>
Sustained Oscillation Tuning

- Carried out with plant in closed-loop
  - useful for open-loop unstable plant

- Using a proportional-only controller, increase the gain until sustained oscillations are attained
  - need to be careful that oscillations do not upset normal running

- Record the gain ($K_c$) and the period of the observed oscillations ($T_c$)
## Sustained Oscillation Tuning

- Use $K_c$ and $T_c$ in this table to give PID settings

<table>
<thead>
<tr>
<th>Controller</th>
<th>$K_p$</th>
<th>$T_i$</th>
<th>$T_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>$0.5K_c$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>P+I</td>
<td>$0.45K_c$</td>
<td>$0.8T_c$</td>
<td>-</td>
</tr>
<tr>
<td>P+I+D</td>
<td>$0.6K_c$</td>
<td>$0.5T_c$</td>
<td>$0.125T_c$</td>
</tr>
</tbody>
</table>

- These are just initial values to get the controller to work
  - will need “fine tuning”
IMC PID Tuning

- “IMC” = Internal Model Control

- Requires plant model
  - model structure and order (e.g. first order with dead-time)
  - parameters such as gains, dead-time and time constants
  - usually obtained from plant tests

- … and *desired* closed loop time constant ($\beta$)

- IMC formulae give required PID gains for specified plant model and $\beta$
If the plant model is:
- first order    - controller = PI
- second order   - controller = PID
- first order + dead-time - controller = PID

For higher order plant models, resulting PID is only an approximation of true IMC controller.

Tools are available that contain the formulae
- e.g. “EZYtune” (http://www.unac.com.au/ezytune)
Advanced Tuning Methods

■ Aström and Hagglund ("Relay") Auto-tuning:
  • similar to sustained oscillation method, but uses a relay with dead-zone to establish oscillations
  • tuning rules built into controller so that it can perform this test and automatically re-tune itself

■ PID tuning using system identification and closed loop pole placement
  • called "self-tuning" if carried out on-line and automatically
Loop Tuning Check List

☑ define desired performance
  • rise time, peak overshoot, stability margins, etc.

☑ identify the dominant plant dynamics

☑ look at interaction with neighbouring loops

☑ which PID configuration is being used?

☑ consider saturation

☑ examine set-point tracking and disturbance rejection of CL system
  • has the desired performance been met?
PID and Lead-Lag Controller Characteristics

- PI similar to lag-compensator
- PD similar to lead-compensator

.... shown in the following bode plots
Bode plot:

**Magnitude:**
- PI: rising for decreasing frequency
- Lag: levels out at low frequencies

**Phase:**
- PI: 90° phase lag for frequencies below breakpoint
- Lag: phase lag only between breakpoints
**PD & Lead Compensator**

**Bode plot:**

**Magnitude:**
- PD: rising for increasing frequency
- Lead: levelling out

**Phase:**
- PD: 90° phase lead for frequencies above breakpoint
- Lead: phase lead only between breakpoints
Implementation Aspects of PID

- Bumpless Transfer
- Derivative Filtering
- Integral Windup
- Digital Implementation
Bumpless Transfer

- Prevents spiky demand signals to actuators when switching between different control modes

- Occurs when switching from manual to automatic
  - Cause: controller output different from current signal
  - Solution: set reference to follow current plant output value
    - called a “tracking” controller

- Derivative action on PV also helps
Derivative Filtering

Most industrial PID controllers have some filtering on derivative action.

Prevents noisy measurements giving noisy controller outputs.

Usually applied as:

\[
\frac{T_ds}{\alpha T_ds + 1}
\]

\(\alpha = \text{derivative time multiplier}\)

- typically in range 0.06 \(\rightarrow\) 0.13
Integral Windup

- Caused by integral action and actuator saturation

- Can lead to instability
This “extra control” is a time delay - which can destabilise the plant.
Anti-Windup Mechanisms

Analogue implementation:

In practice it is difficult to set-up gain $K$. 
Anti-Windup Mechanisms

- Logic/digital implementation easier:

- Anti-windup present in *most* industrial controllers
Digital Implementation

A continuous time PID controller is given by:

\[ u_c(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de}{dt} \]

This can be cast into this discrete time form:

\[ u_c(k) = K_p e(k) + K_i \tau_s \sum_{j=0}^{k} e(j) + \frac{K_d}{\tau_s} [e(k) - e(k-1)] \]

where \( k \) is the sample number and \( \tau_s \) is the sampling period.
Digital Implementation

This may be re-written as:

\[ u_c (k) = u_c (k - 1) + K_p [e(k) - e(k - 1)] + K_i \tau_s e(k) + \frac{K_d}{\tau_s} [e(k) - 2e(k - 1) + e(k - 2)] \]

or:

\[ u_c (k) = u_c (k - 1) + \Delta u_c (k) \]

The change in controller output (\( \Delta u_c \)) is calculated and the inherent digital integrator forms \( u_c (k) \)

Called an “incremental PID” controller
Why Does It Work and is so Successful?

- Has high gain at LF for disturbance rejection
- Has only single pole so small phase lag introduced
- With D term can have phase advance offsets lags in plant (amplifies noise of course).
- Can be tuned just with plant data – no model essential.
- No advanced theory so a technician may easily try to tune.
- Easily often related to physical plant parameters. by pole placement for example.
Concluding Remarks

1. One of the most popular control techniques.
2. Unrealistic high frequency gain.
3. Simple to tune.
4. Mostly for single input single output systems.
5. Auto-tuning now used for some processes.
6. Integral wind-up a problem.
7. Derivative kick must also guard against.