Driver assistance system design A

Proportional Integral Derivative control

Carlo Novara

Politecnico di Torino
Dip. Elettronica e Telecomunicazioni

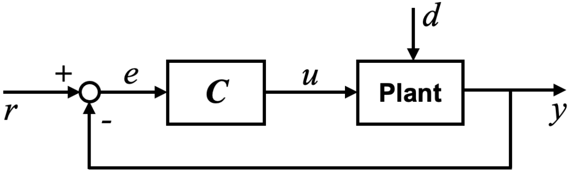
Outline

Introduction

- Proportional Integral Derivative (PID) control is probably the most popular control approach in industrial applications:
  - automotive
  - aerospace
  - electrical systems
  - power systems
  - chemical processes
  - many others ...
- The main reason for such a popularity is its simplicity: a standard PID controller is characterized by a few parameters,
- The PID parameters can be suitably tuned
  - off-line, via computer design and simulation,
  - on-line, directly on the physical plant of interest.
- PID control is in general effective for relatively simple systems. It may not be suitable for complicated/complex systems.

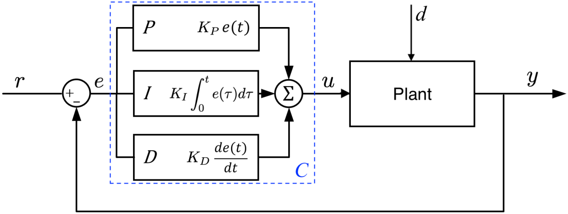
Introduction

A general control architecture



- Relevant blocks:
  - Plant: system to control
  - C: controller.
- Relevant signals:
  - u: plant command input
  - y: plant output
  - r: reference
  - $e = r - y$: tracking error
  - d: disturbances/noises.
- The **goal of control** is to ensure a "small" tracking error $e(t)$.

PID control: time-domain description



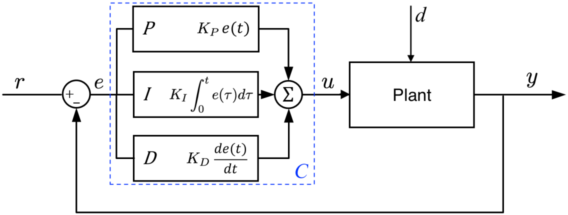
- The continuous-time (CT) PID control law is

$$u(t) = K_P\, e(t) + K_I \int_0^t e(\tau)d\tau + K_D\, \dot{e}(t).$$

  - $K_P\, e(t)$: proportional action
  - $K_I \int_0^t e(\tau)d\tau$: integral action
  - $K_D\, \dot{e}(t)$: derivative action.

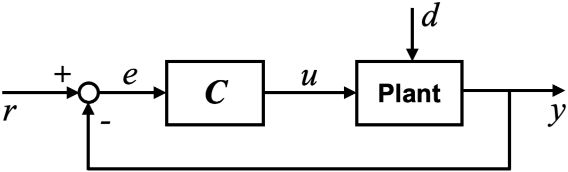
PID control: time-domain description

Interpretation



- $K_P\, e(t)$: Proportional action, accounts for the present. It provides a command finalized at reducing the current tracking error.
- $K_I \int_0^t e(\tau)d\tau$: Integral action, accounts for the past (the integral is the sum of past values). It allows precise tracking error for constant or slowly-varying references.
- $K_D\, \dot{e}(t)$: Derivative action, accounts for the future (the derivative gives the trend of the signal). It improves the dynamic performance and robustness.

PID control: Laplace-domain description



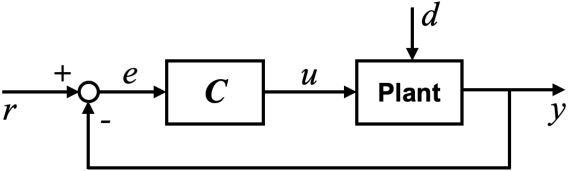
- The PID controller is

$$C(s) = K_P + \frac{K_I}{s} + K_D\, s$$

where s is the Laplace variable and

- K_P: proportional action
- $\frac{K_I}{s}$: integral action
- $K_D\, s$: derivative action.

PID control: $\mathcal{Z}$-domain description



- The discrete-time (DT) PID controller is

$$C(z) = K_P + K_I \frac{T_s}{z-1} + K_D \frac{z-1}{T_s}$$

where z is the $\mathcal{Z}$-transform variable, T_s is the discretization time and

- K_P: proportional action
- $K_I \frac{T_s}{z-1}$: integral action
- $K_D \frac{z-1}{T_s}$: derivative action.

PID control: derivative filter

- In both the CT and DT cases, the derivative term may give noise amplification, especially at high frequencies.
- A first-order filter $H(s)$ or $H(z)$ is usually introduced in the derivative term to reduce noise effects:

$$
\begin{aligned}
H(s) &\doteq \tfrac{1}{T_f s+1} && \text{CT}\\
H(z) &\doteq \tfrac{T_s}{T_f(z-1)+T_s} && \text{DT.}
\end{aligned}
$$

- In this way, PIDF controllers are obtained:

$$
\begin{aligned}
C(s) &= K_P + \tfrac{K_I}{s} + K_D\, sH(s) && \text{CT}\\
C(z) &= K_P + K_I \tfrac{T_s}{z-1} + K_D \tfrac{z-1}{T_s} H(z) && \text{DT.}
\end{aligned}
$$

Comparison between CT and DT systems

CT systems

Time: $t \in [0, \infty) \subset \mathbb{R}$

NL state equations

$$\begin{aligned} \dot{x} &= f(x, u) \\ y &= g(x, u) \end{aligned}$$

LTI state equations

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned}$$

DT systems

Time index: $k \in \mathbb{Z}$, $k = 0, 1, 2, \ldots$

NL state equations

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) \\ y(k) &= g(x(k), u(k)) \end{aligned}$$

LTI state equations

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned}$$

Comparison between CT and DT systems

CT LTI systems

Asymptotic stability:
$\mathrm{Re}(\lambda_i) < 0, \ \forall i$

Marginal stability:
$\mathrm{Re}(\lambda_i) \leq 0, \ \forall i$
$\mathrm{mult}(\lambda_i) = 1 \text{ if } \mathrm{Re}(\lambda_i) = 0$

Instability:
otherwise

DT LTI systems

Asymptotic stability:
$|\lambda_i| < 1, \ \forall i$

Marginal stability:
$|\lambda_i| \leq 1, \ \forall i$
$\mathrm{mult}(\lambda_i) = 1 \text{ if } |\lambda_i| = 1$

Instability:
otherwise

Comparison between CT and DT systems

CT systems

Laplace transform
$\mathcal{L}\{f(t)\} = \int_0^\infty f(t)e^{-st}dt$

linearity

final value theorem

derivative:
$\mathcal{L}\{\dot{f}(t)\} = s\mathcal{L}\{f(t)\} - f(0)$

$s \leftrightarrow$ derivative, $s^{-1} \leftrightarrow$ integral

$$sF(s) \leftrightarrow \dot{f}(t)$$
$$s^{-1}F(s) \leftrightarrow \int f(\tau)d\tau$$

DT systems

$\mathcal{Z}$-transform
$\mathcal{Z}\{f(k)\} = \sum_{k=0}^\infty f(k)z^{-k}$

same property

similar result

time shift:
$\mathcal{Z}\{f(k-1)\} = z^{-1}\mathcal{Z}\{f(k)\}$

$z \leftrightarrow$ time shift:

$$zf(k) = f(k+1)$$
$$z^{-1}f(k) = f(k-1)$$

Comparison between CT and DT systems

CT systems

Transfer function
$G(s) = \frac{b_1 s + b_2}{s^2 + a_1 s + a_2}$

Frequency response

Bode and Nyquist diagrams:
$\omega \in [0, \infty]$

DT systems

Transfer function
$G(z) = \frac{b_1 z + b_2}{z^2 + a_1 z + a_2}$

Similar definition

Similar, with $\omega \in [0, \frac{\pi}{T}]$,
$\frac{\pi}{T_s}$=Nyquist frequency

PID controller design

General criteria

- Effects of increasing a parameter independently:

Parameter	Rise time	Overshoot	Steady-state error	Stability
K_P	Decrease	Increase	Decrease	Degrade
K_I	Decrease	Increase	Eliminate	Degrade
K_D	Minor change	Decrease	No effect in theory	Improve if "small"

- Several design methods available:
  - Ziegler–Nichols' methods $\leftarrow$
  - Pole placement
  - Algebraic design
  - H-infinity tuning
  - Optimization-based $\leftarrow$
  - Empirical tuning by means of simulations
  - Empirical tuning on the real plant (possible refinement of a previous design carried out with other methods).

PID controller design

Ziegler–Nichols' method

- The Ziegler–Nichols' method is a classical tuning technique for PID controllers. Main design steps:
  1. Start with a P controller.
  2. Let K_u be the value of K_P at which the closed-loop system oscillates with a constant amplitude. Let T_u be the oscillation period.
     - ★ K_u can be found either in closed-loop (oscillatory poles, simulations or real system experiments) or in open-loop (Nyquist criterion).
  3. The three parameters of the PID controller can be chosen according to the following table.

controller	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	$0.6K_u$	$2K_P/T_u$	$K_P T_u/8$

PID controller design

Matlab PID tuner

- Matlab PID tuning algorithm based on optimization. Main steps:
  1. Choice of performance targets:
     - ω_c^{des}: desired cross-over frequency.
     - m_ϕ^{des}: desired phase margin.
  2. Choice of algorithm options:

     $\texttt{op} = \texttt{pidtuneOptions}(\text{'DesignFocus'},\texttt{foc},\text{'PhaseMargin'},m_\phi^{des})$

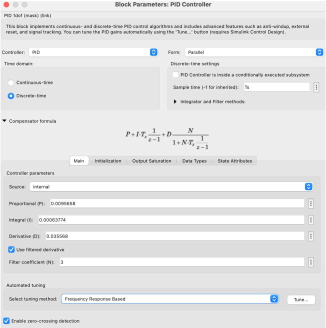
     where $\texttt{foc} \in$ {'reference-tracking', 'disturbance-rejection', 'balanced'}.
  3. Controller synthesis: $\texttt{C} = \texttt{pidtune}(\texttt{Plant}, \texttt{type}, \omega_{\texttt{c}}^{\texttt{des}}, \texttt{op})$

     `C`: controller
     `Plant`: LTI plant model
     `type` $\in$ {'P','PI','PID','PIDF',...}.
- Discrete-time PID design: discretize the model (`c2d` command) and apply the same procedure.

PID controller design

Simulink PID block/tuner

- A PID block is available in Simulink. Two design approaches:
  - Directly insert the parameters and perform trial-and-error tuning.
  - Use the "automated tuning" function, based on optimization, able to perform the design directly in Simulink. Two methods:
    - ★ Transfer Function Based: Time-domain design with slower/faster and aggressive/robust trade-offs.
    - ★ Frequency Response Based: It allows choosing target ω_c and m_ϕ.



A simple example

Ziegler–Nichols

Plant: $G(s) = \frac{1}{(s+1)^3}$.

Osciallating condition

Ku is found as the value of Kp for which the closed-loop transfer function has oscillatory poles.

```
Ku=8;
T_osc=minreal(Ku*G/(1+Ku*G));
lambda=pole(T_osc)
```

Tu is obtained from the imaginary part of the closed-loop oscillatory poles, which corresponds to the oscillation frequency (see mode analysis).

```
Tu=2*pi/1.73
```

The command "pole" shows that that the imaginary part of the oscillatory poles is 1.73.

```
lambda = 6×1 complex
   -3.0000 + 0.0000i
    0.0000 + 1.7321i
    0.0000 - 1.7321i
   -1.0000 + 0.0000i
   -1.0000 - 0.0000i
   -1.0000 + 0.0000i
```

```
Tu = 3.6319
```

A simple example

Ziegler–Nichols

- Four controllers designed using Ziegler–Nichols' method.

P controller

```
Kp=0.5*Ku;
Cp=Kp;
```

PI controller

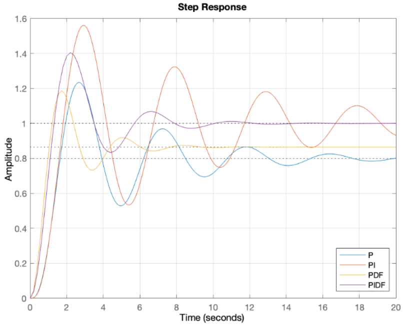
```
Kp=0.45*Ku;
Ki=1.2*Kp/Tu;
Cpi=Kp+Ki/s;
```

PDF controller

```
Kp=0.8*Ku;
Kd=Kp*Tu/8;
Cpdf=Kp+Kd*s/(s/1000+1);
```

PIDF controller

```
Kp=0.6*Ku;
Ki=2*Kp/Tu;
Kd=Kp*Tu/8;
% Kd=Kp*Tu/8*2; % manual tuning
Cpidf=Kp+Ki/s+Kd*s/(s/1000+1);
```



A simple example

Matlab tuner

- **Plant:** $G(s) = \frac{1}{(s+1)^3}$.

Performance targets

```
wc_des=1;   % desired cross-over frequency [rad/s]
pm_des=60;  % desired phase margin [deg]
```

Options

```
op=pidtuneOptions('DesignFocus','balanced',...
    'PhaseMargin',pm_des);
```

PI controller (continuous-time)

With wc_des=1, the PI controller cannot guarantee the desired phase margin, leading to a closed-loop oscillatory behaviour.

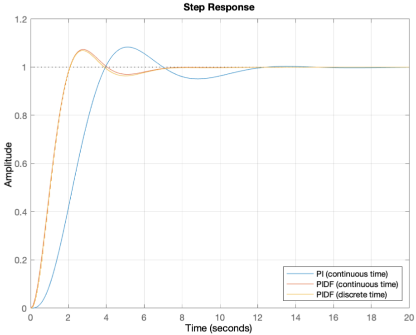
```
Cpi=pidtune(G,'PI',0.5,op);
```

PIDF controller (continuous-time)

```
Cpid=pidtune(G,'PIDF',wc_des,op);
```

PIDF controller (discrete-time)

```
G_dt=c2d(G,0.05);
Cpid_dt=pidtune(G_dt,'PIDF',wc_des,op);
```



Discussion

- PID controllers are effective in many industrial applications.
- The main reason for such a popularity is its simplicity: a standard PID controller is characterized by only 3 or 4 parameters, which can be suitably tuned off-line or on-line.
- Standard PID controllers typically work well when applied to relatively simple linear plants.
- Application to complex nonlinear plants is possible using parameter-varying PID controllers (gain-scheduling approach), requiring however more complicated design and implementation.