# General Information

**Prof. Antonella Ferrara** https://scholar.google.com/citations?user=r5JuMskAAAAJ&hl=en

**Course Teaching Material:** KIRO UNIPV https://elearning.unipv.it/

- 504462 - PROCESS CONTROL 2025-26 - PROF.SSA FERRARA ANTONELLA

**Lecture Time-table:** http://www-3.unipv.it/ingserv/orario2526/1sem/insegnamenti/ProCont.html

**Exams:** https://studentionline.unipv.it/esse3/Home.do https://kirotesting.unipv.it/

# Introduction

- Program of the course:

**Advanced SISO control schemes:**
Pre-filters and parallel compensators, two degrees of freedom control schemes, compensation of measurable disturbances, systems with delays and Smith Predictor, Padé approximation, decoupling in the frequency domain, control of open loop unstable systems.

**Advanced MIMO control schemes:**
Decoupling based control schemes, decentralized control, relative gain array.

**PID controllers:**
Features and properties. Rules for the empirical calibration. Wind-up and anti wind-up schemes.

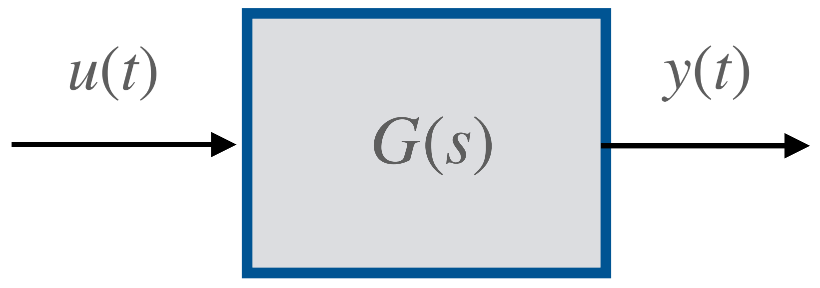
**Digital control:**
Discrete-time systems. The concept of equilibrium for discrete-time systems. Stability of linear time-invariant discrete-time systems. Jury test. Digital control schemes. Zeta transform and its properties. Transfer functions in the z domain. Sampling and aliasing. Choice of the sampling time. Zero-order-Hold. Discretization of continuous-time controllers. Bilinear transformation, Euler and Tustin methods.

# Introduction

- Some of the figures in these slides, kindly provided by McGraw-Hill, are those of the Textbook:

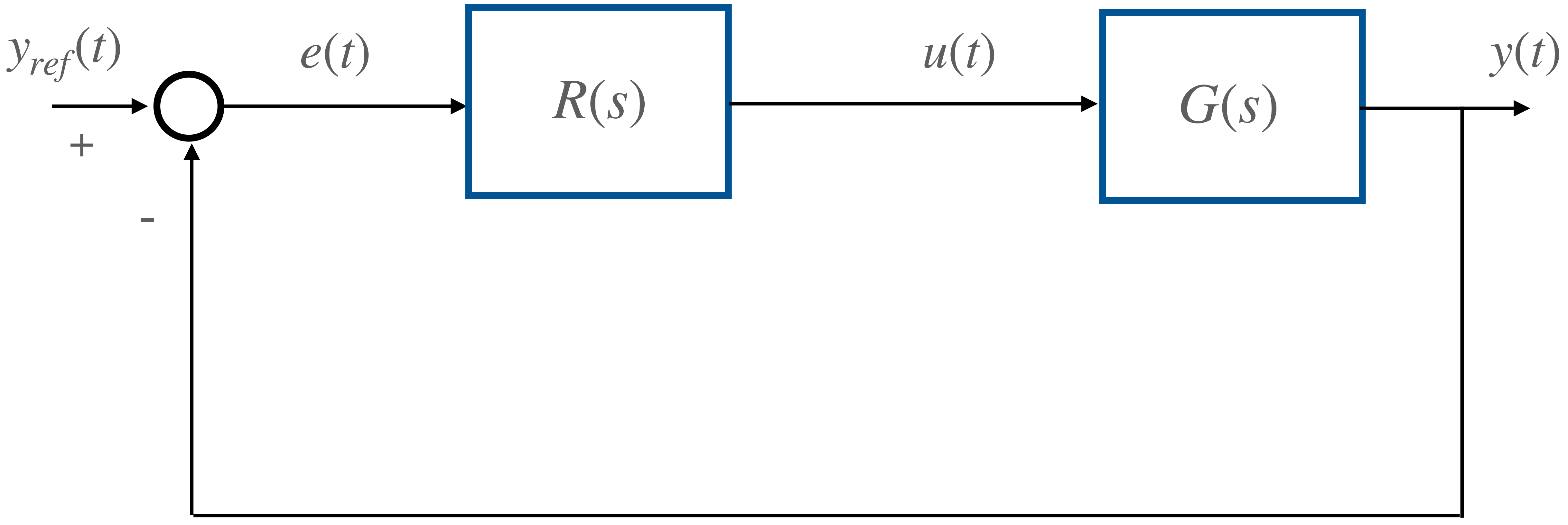
# Design of PID Controllers

Process



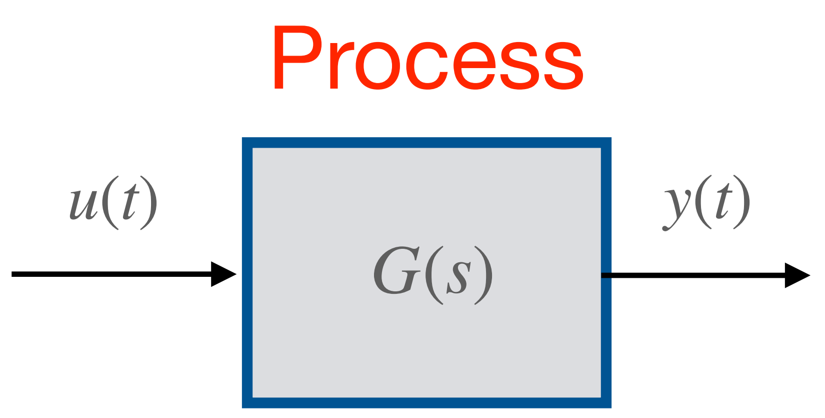

$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$



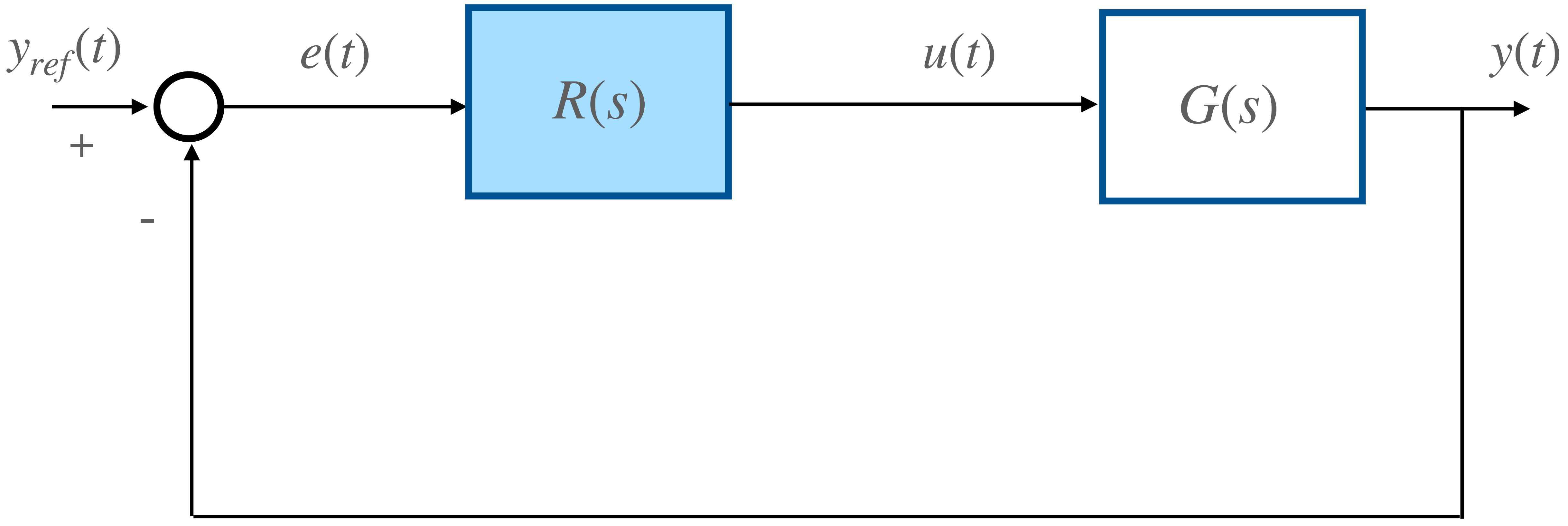

Unitary Feedback Control Scheme

## Design of PID Controllers

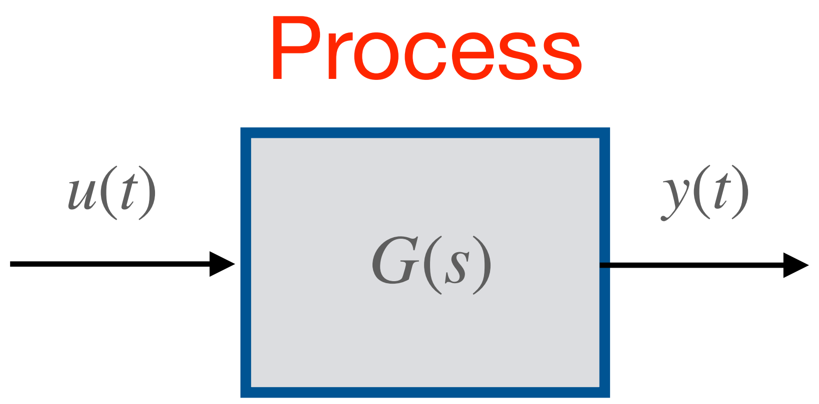



$u(t) \in \mathbb{R}$

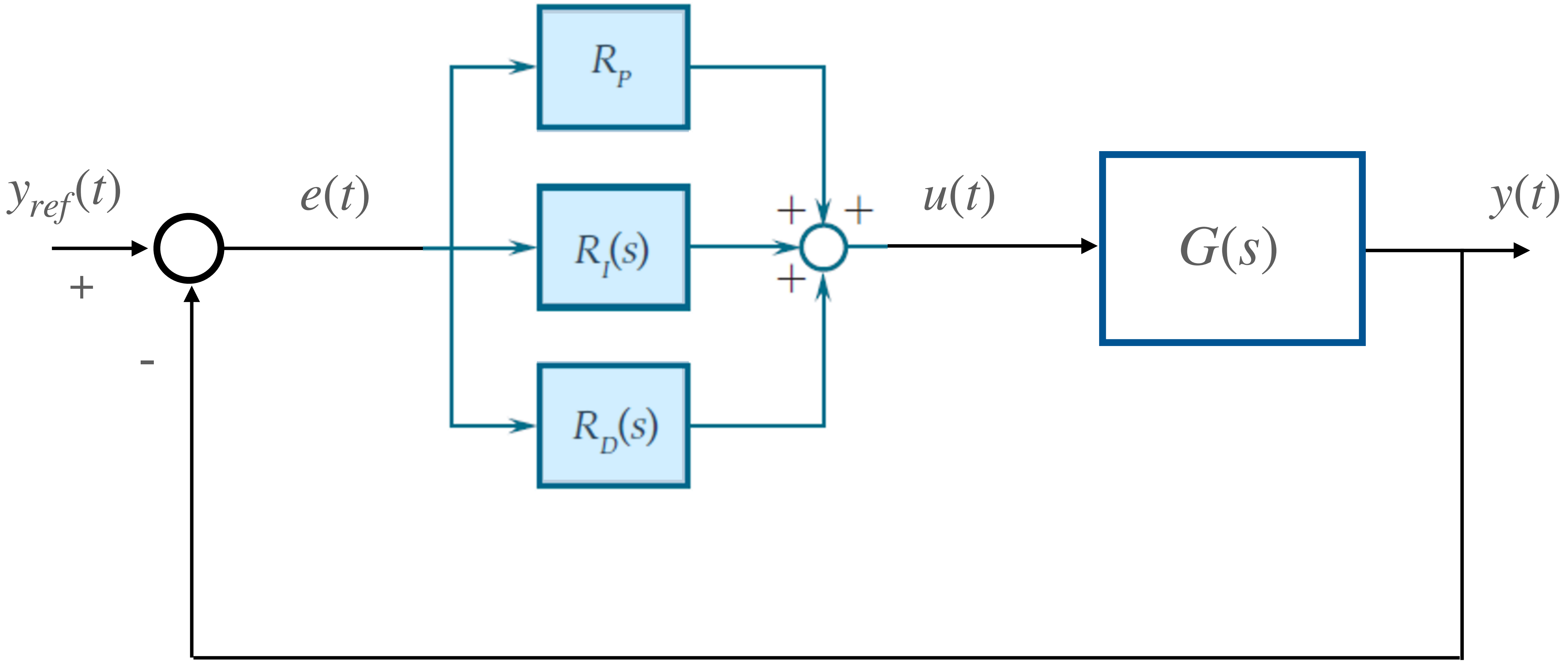
$y(t) \in \mathbb{R}$



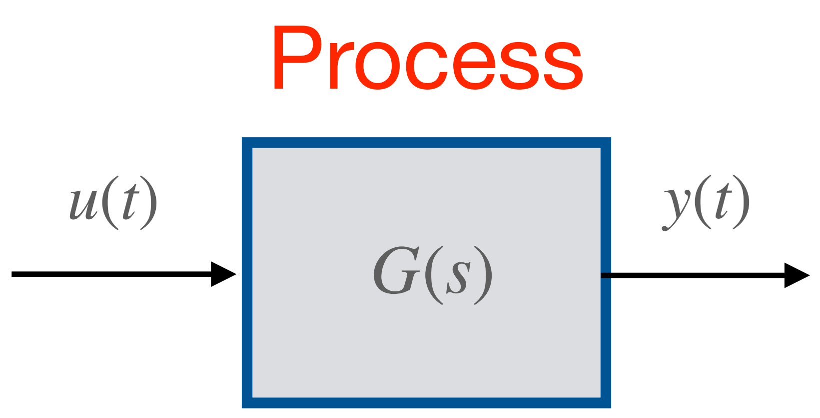
## Design of PID Controllers




$$u(t) \in \mathbb{R}$$
$$y(t) \in \mathbb{R}$$

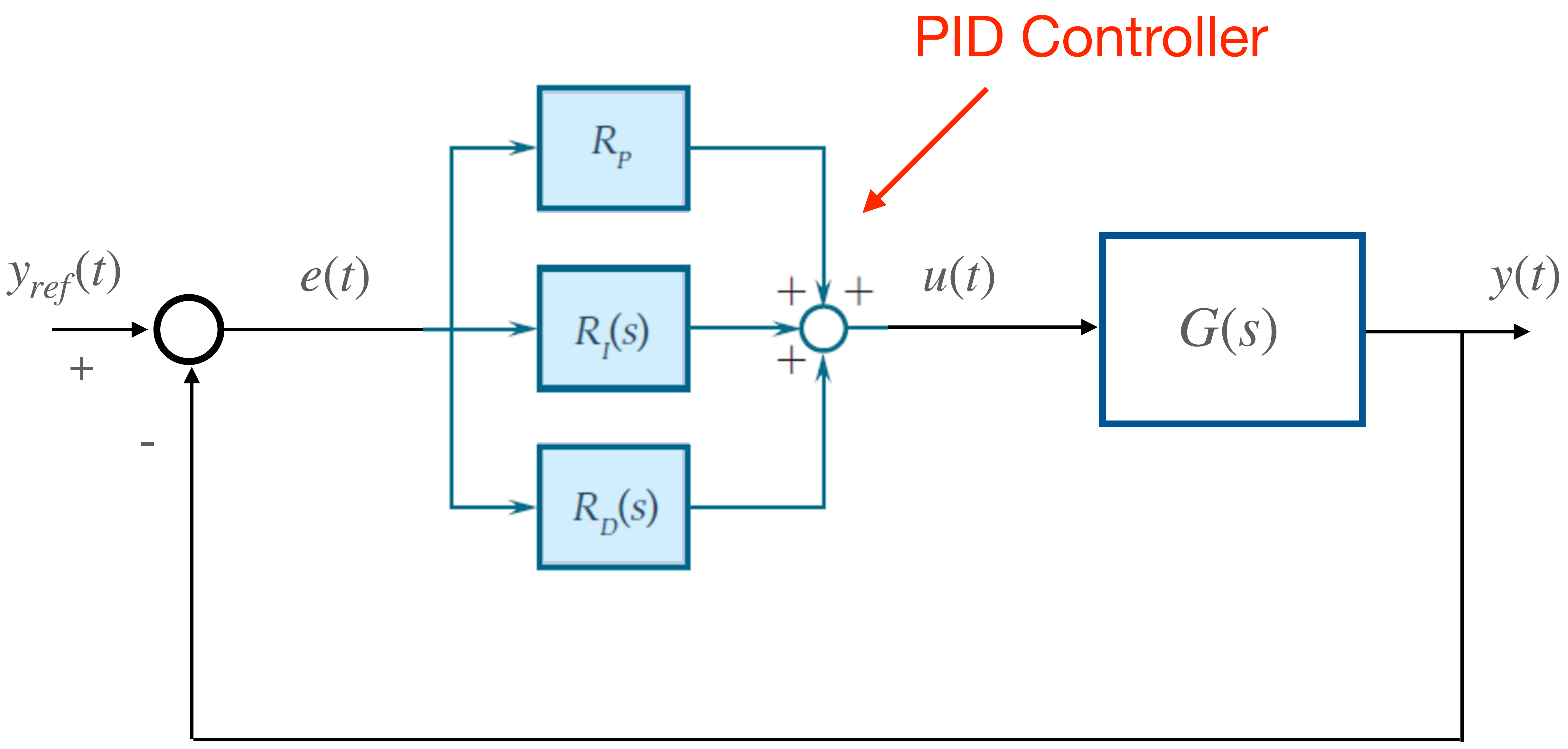


# Design of PID Controllers

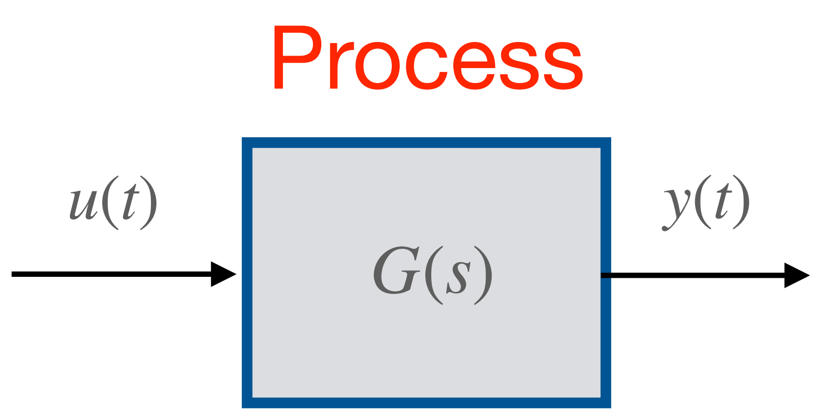



$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$

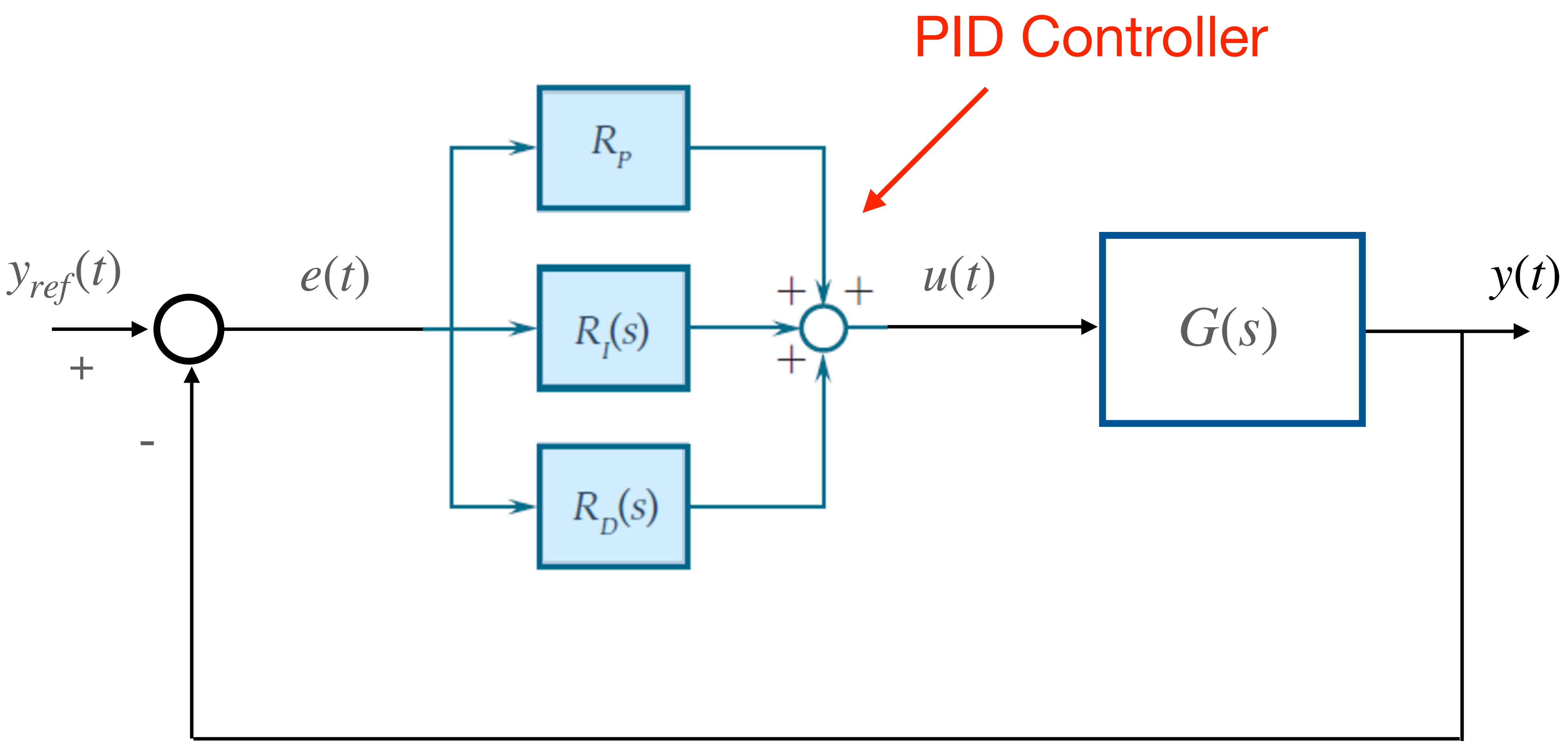


# Design of PID Controllers



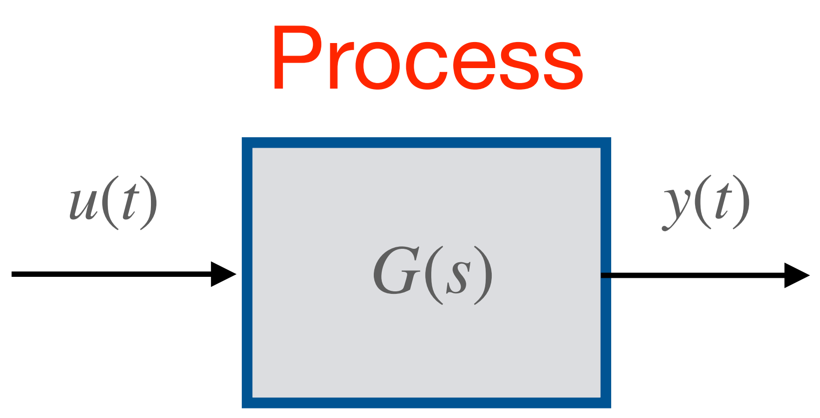

$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$



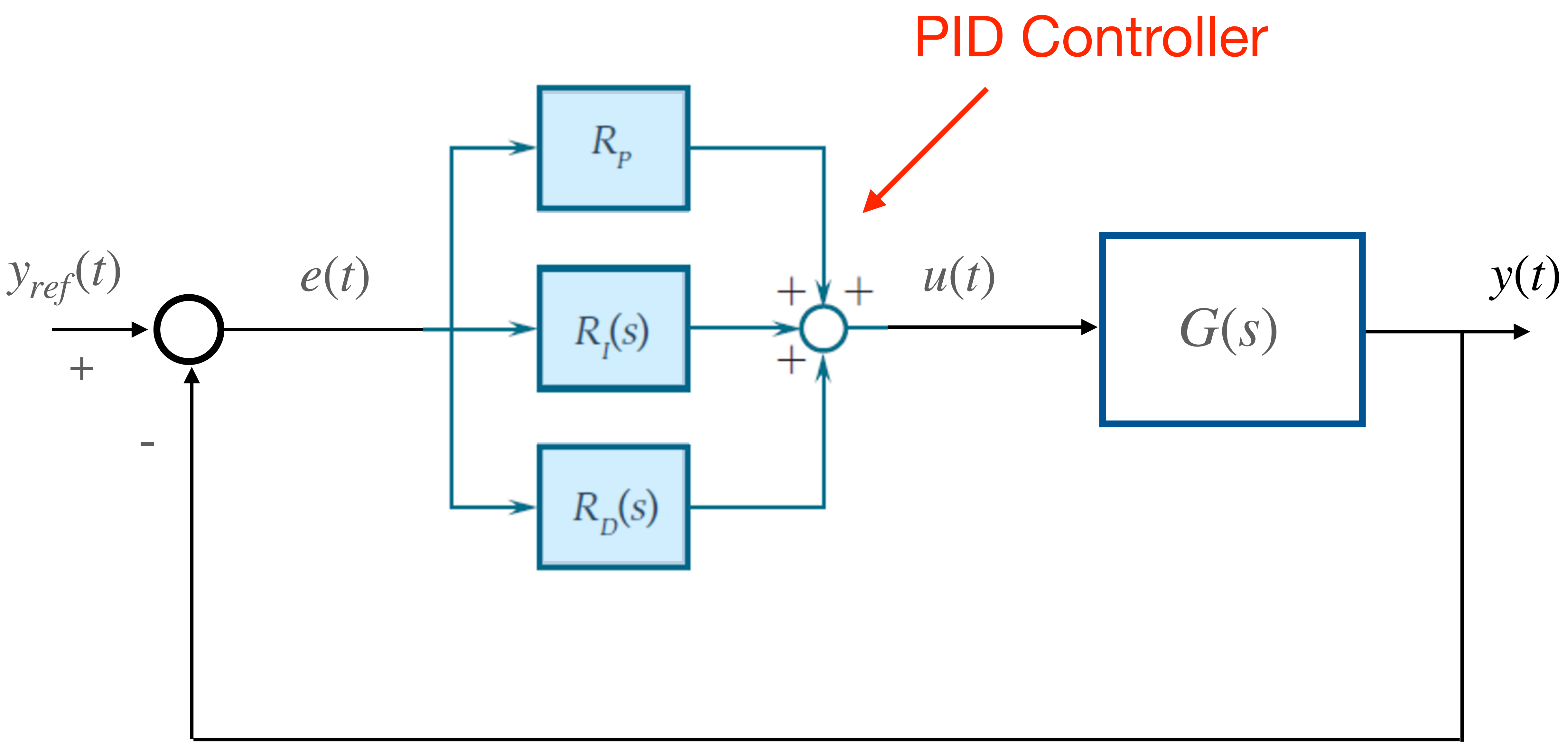

Ideal PID Controller: $u(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt}$

# Design of PID Controllers




$u(t) \in \mathbb{R}$

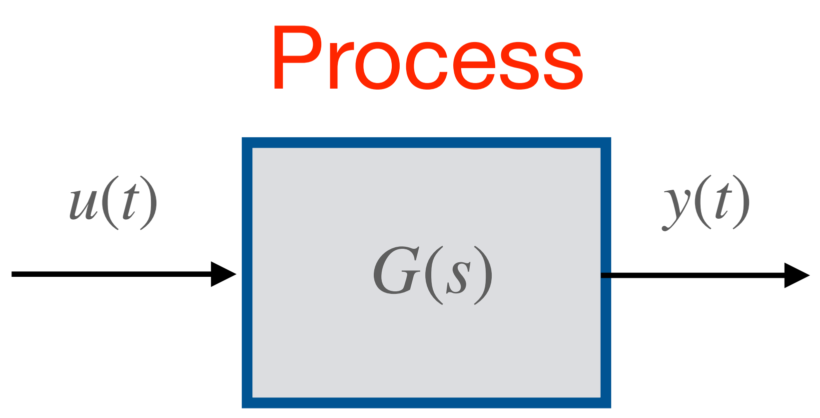
$y(t) \in \mathbb{R}$



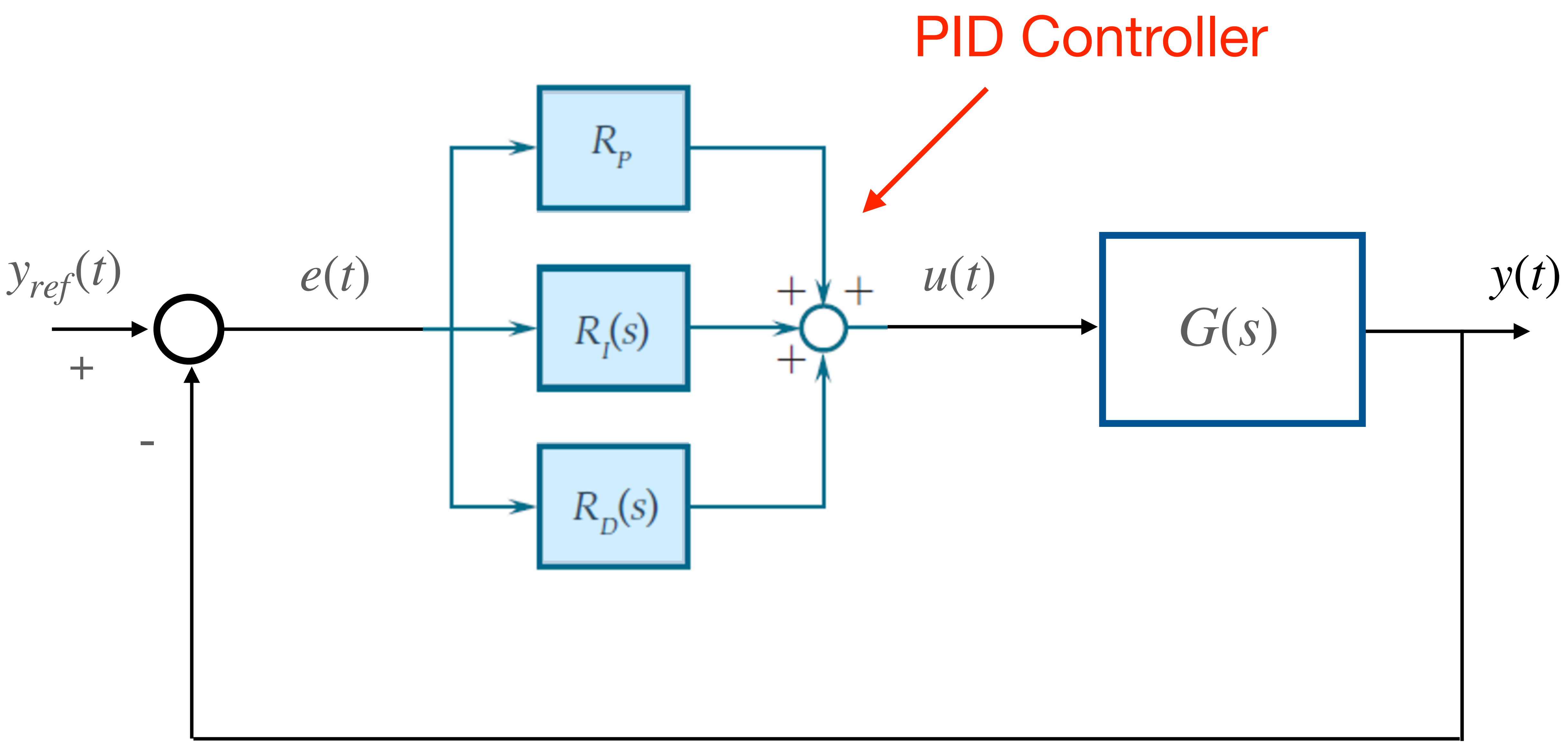

Ideal PID Controller: $u(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt}$

$$\mathscr{L}\{u(t)\} = U(s) = \mathscr{L}\left\{ K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt} \right\} = \left( K_P + \frac{K_I}{s} + K_D s \right) E(s)$$

# Design of PID Controllers




$$u(t) \in \mathbb{R}$$
$$y(t) \in \mathbb{R}$$



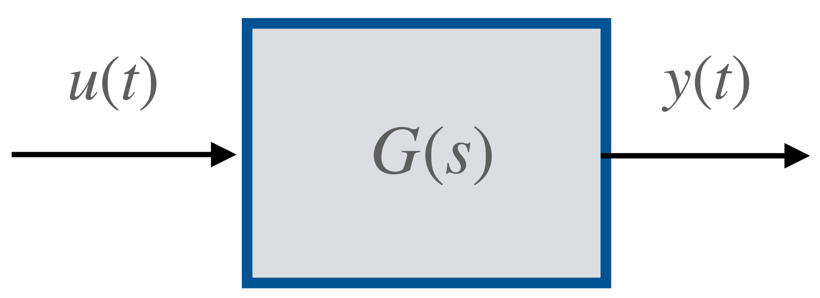

Ideal PID Controller: $u(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \dfrac{de(t)}{dt}$

$$\mathscr{L}\{u(t)\} = U(s) = \mathscr{L}\left\{K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt}\right\} = \left(K_P + \frac{K_I}{s} + K_D s\right) E(s)$$

$$R_{PID_{id}}(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}$$

# Design of PID Controllers

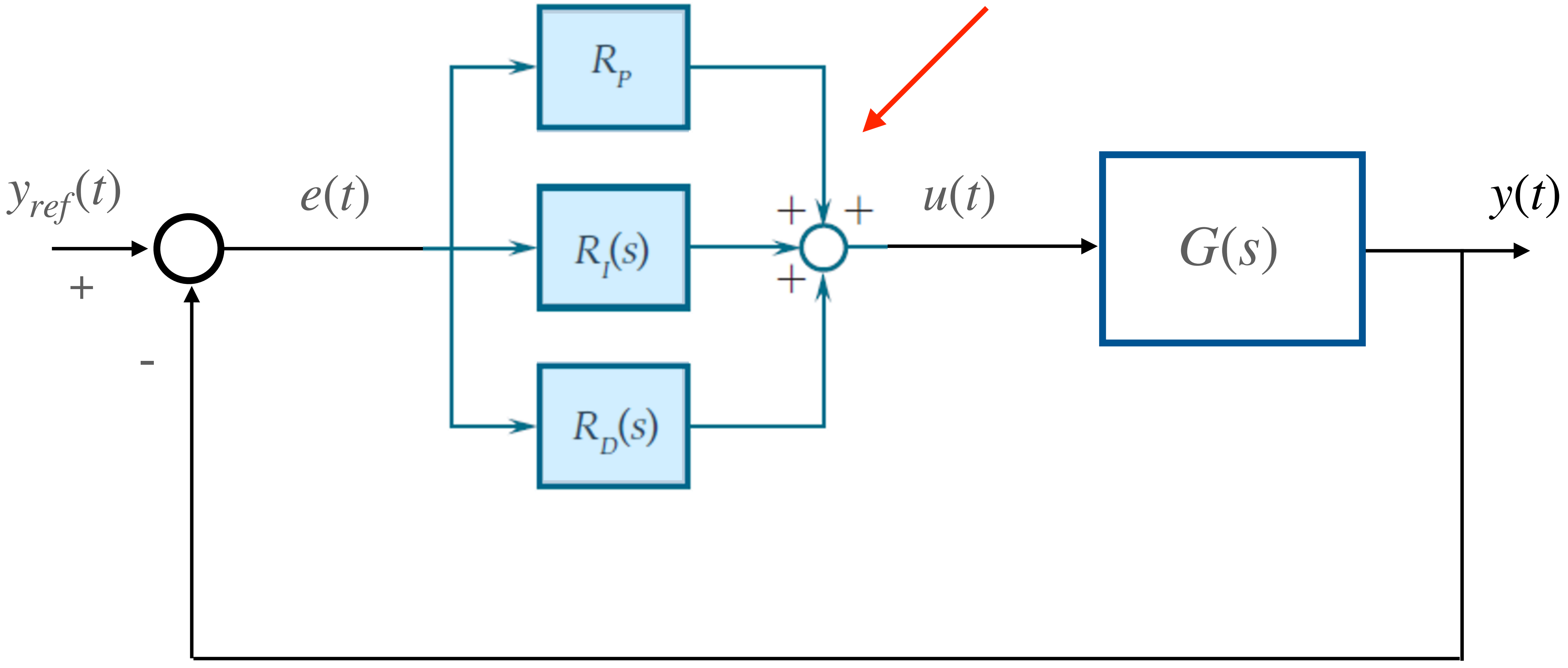
Process




$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$

PID Controller



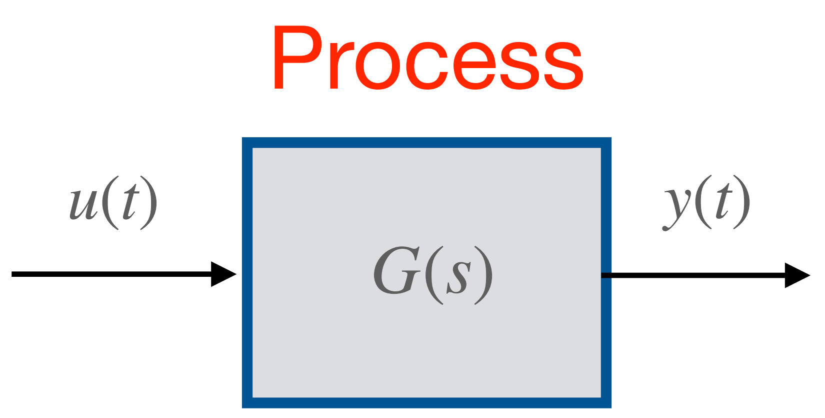

Ideal PID Controller: $u(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \dfrac{de(t)}{dt}$

$$\mathscr{L}\{u(t)\} = U(s) = \mathscr{L}\left\{ K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt} \right\} = \left( K_P + \frac{K_I}{s} + K_D s \right) E(s)$$

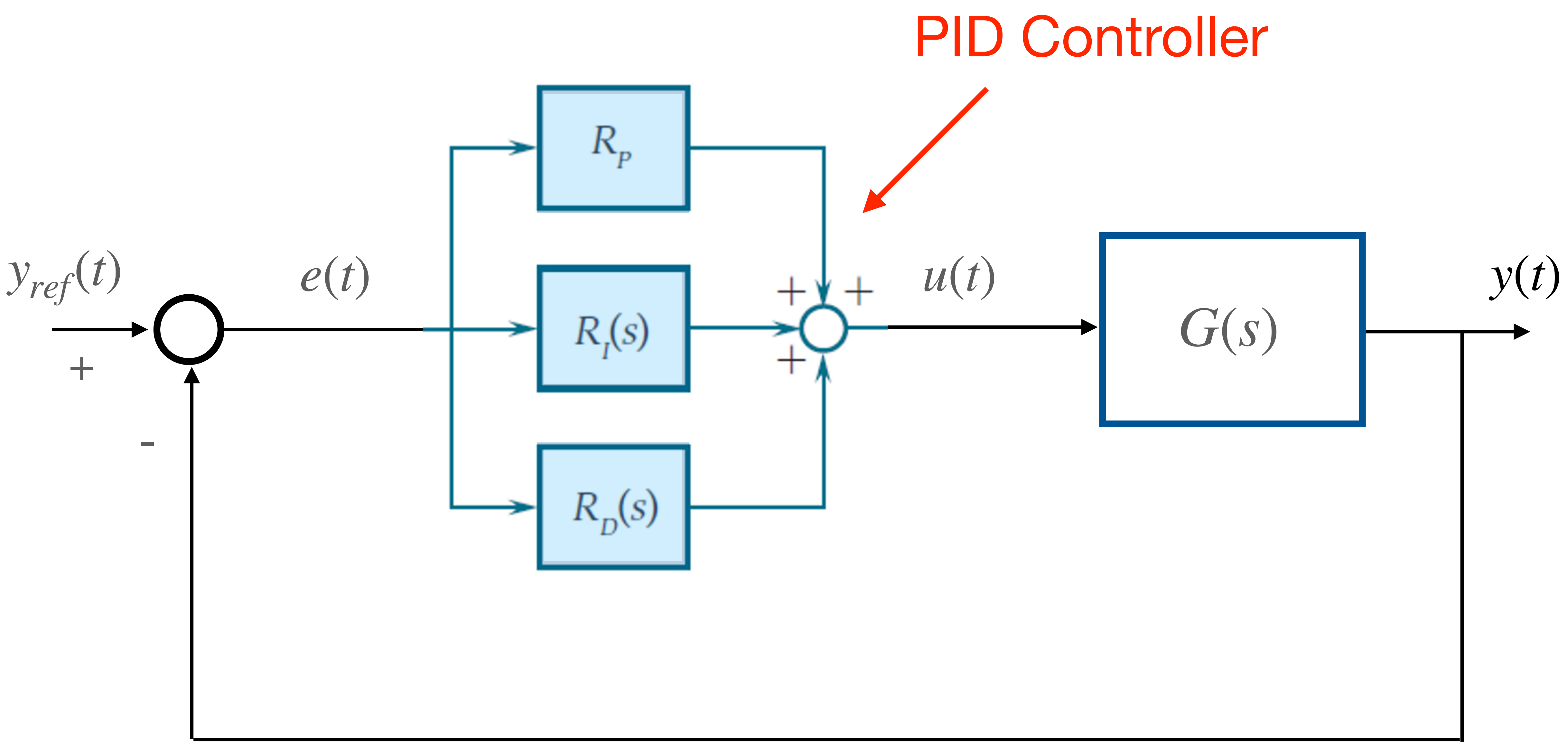
It is not causal

$$R_{PID_{id}}(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}$$

# Design of PID Controllers




$$u(t) \in \mathbb{R}$$
$$y(t) \in \mathbb{R}$$



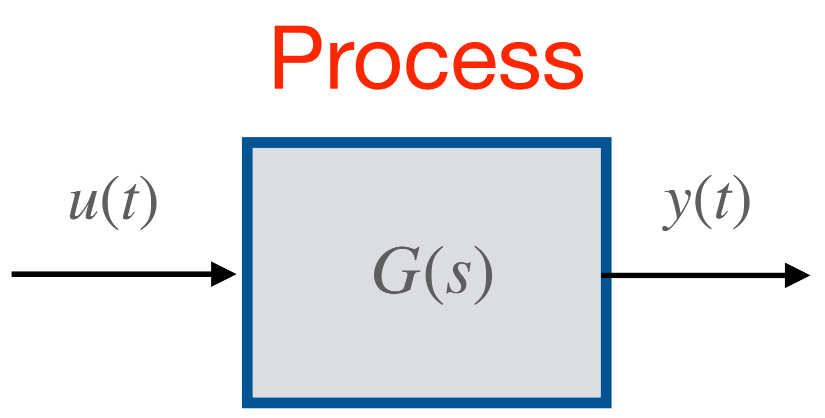

Alternative representation:

$$R_{PID_{id}}(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

Ideal PID Controller: $u(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt}$

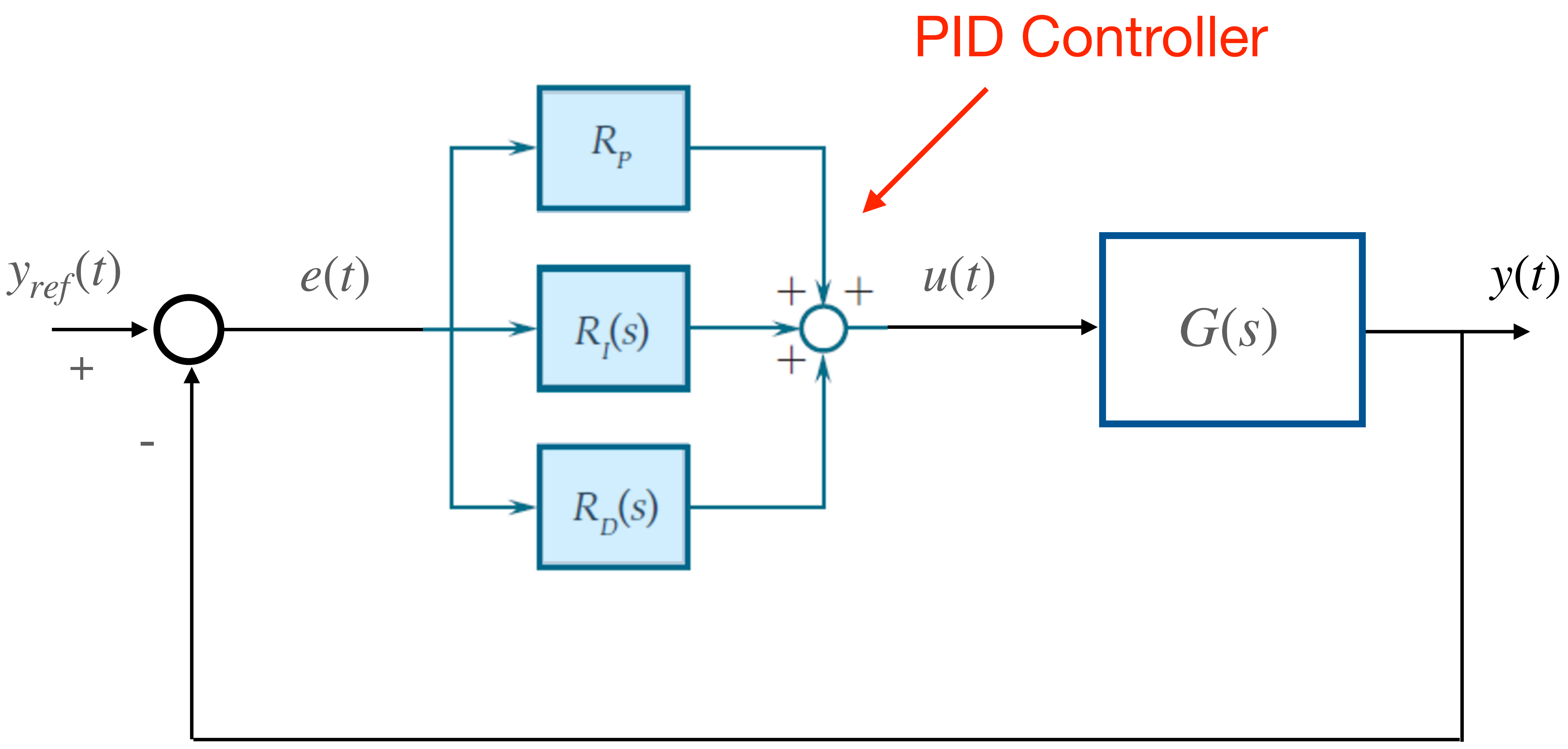
$$\mathcal{L}\{u(t)\} = U(s) = \mathcal{L}\left\{K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt}\right\} = \left(K_P + \frac{K_I}{s} + K_D s\right) E(s)$$

# Design of PID Controllers




$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$




Alternative representation:

$$R_{PID_{id}}(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

Ideal PID Controller: $u(t) = K_P e(t) + K_I \int_0^t e(\tau)\,d\tau + K_D \frac{de(t)}{dt}$

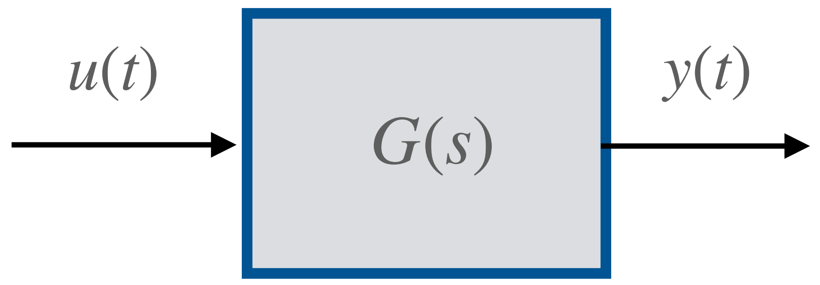
$$\mathscr{L}\{u(t)\} = U(s) = \mathscr{L}\left\{K_P e(t) + K_I \int_0^t e(\tau)\,d\tau + K_D \frac{de(t)}{dt}\right\} = \left(K_P + \frac{K_I}{s} + K_D s\right) E(s)$$

To make it causal

$$R_{PID}(s) = K_P\left(1 + \frac{1}{T_I s} + \frac{T_D s}{1 + \frac{T_D}{N} s}\right)$$

# Design of PID Controllers

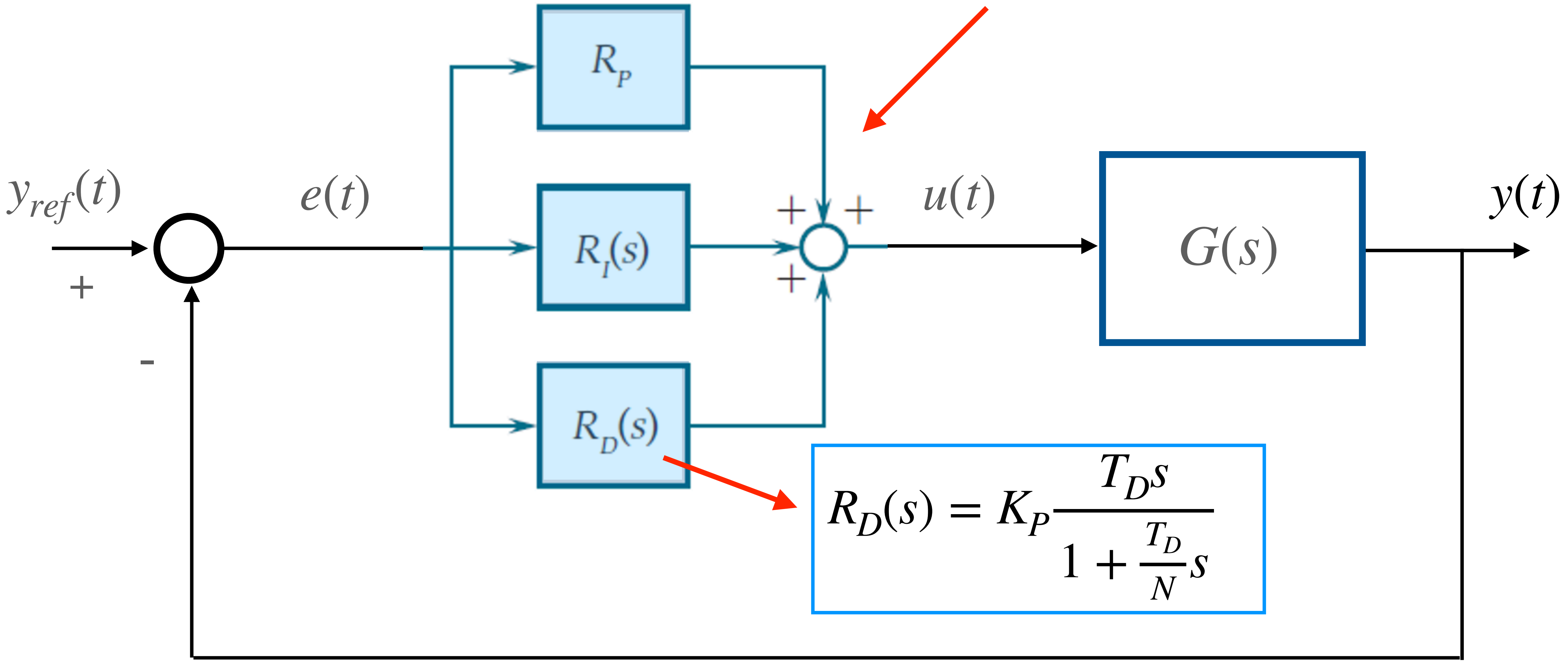
Process




$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$

PID Controller: causal version




Alternative representation:

$$R_{PID_{id}}(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

Ideal PID Controller: $u(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt}$

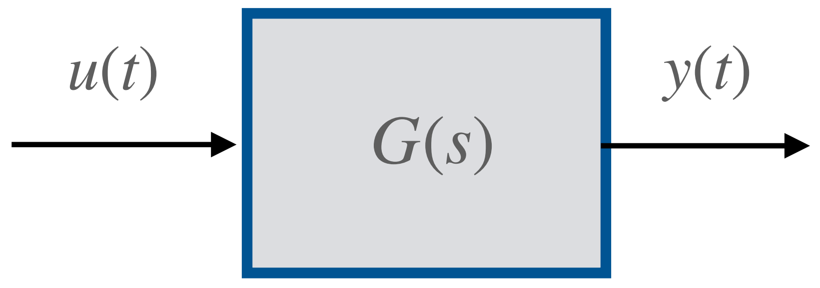
$$\mathscr{L}\{u(t)\} = U(s) = \mathscr{L}\left\{K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de(t)}{dt}\right\} = \left(K_P + \frac{K_I}{s} + K_D s\right) E(s)$$

To make it causal

$$R_{PID}(s) = K_P\left(1 + \frac{1}{T_I s} + \frac{T_D s}{1 + \frac{T_D}{N} s}\right)$$

# Design of PID Controllers

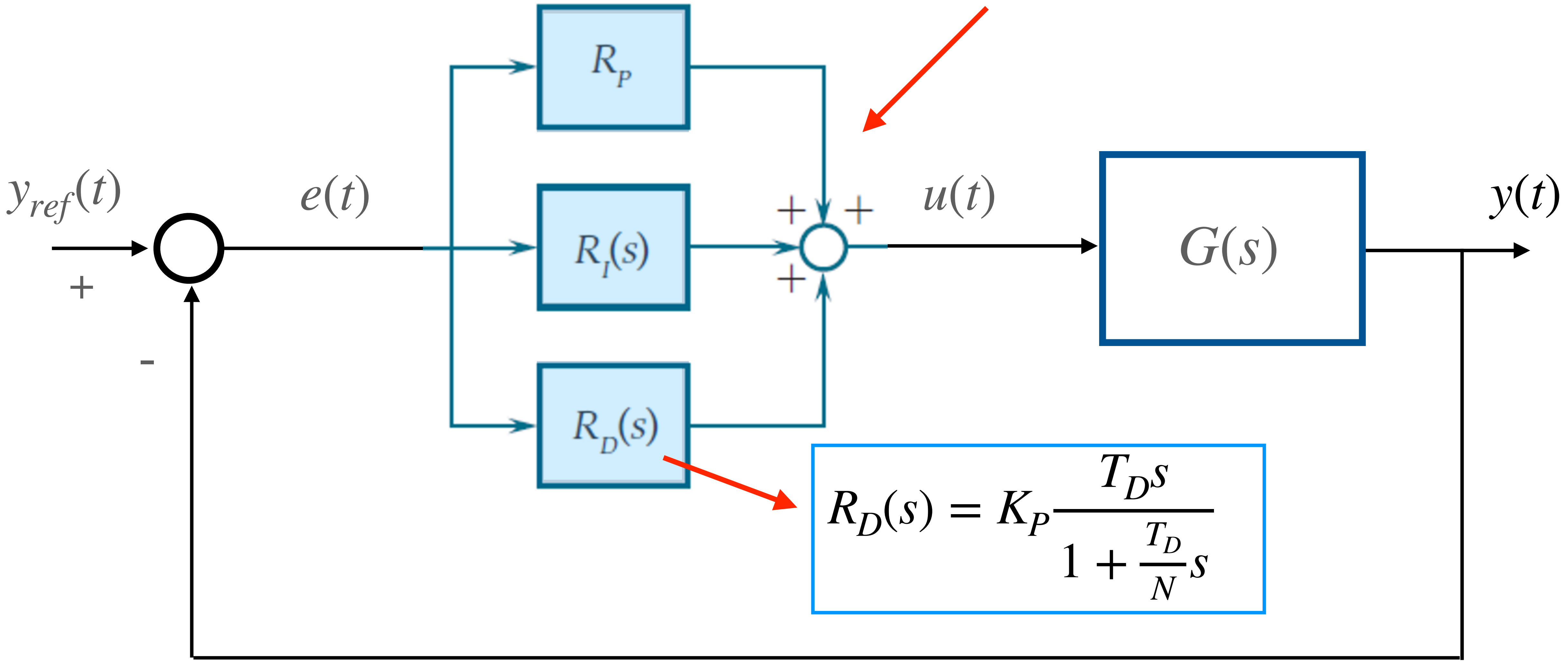
Process




$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$

PID Controller: causal version

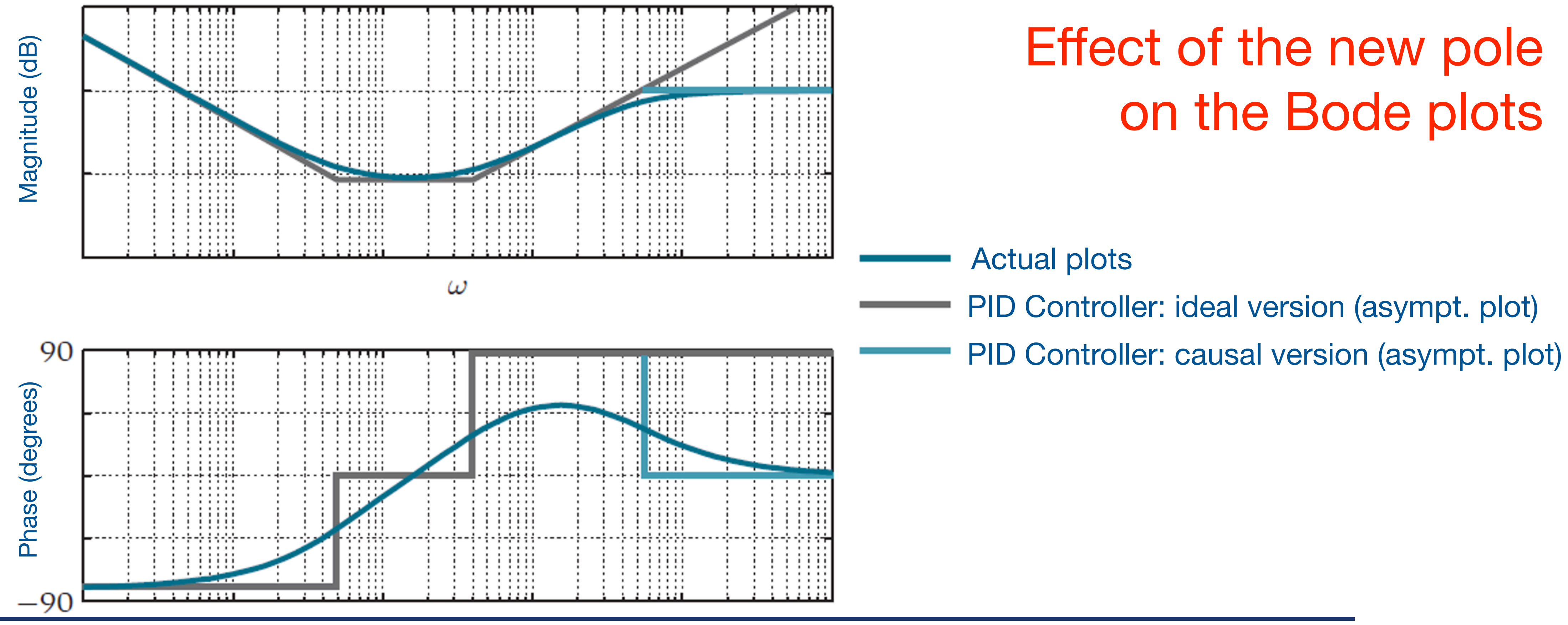



Alternative representation:

$$R_{PID_{id}}(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s = K_P \left(1 + \frac{1}{T_I s} + T_D s\right)$$

It is causal

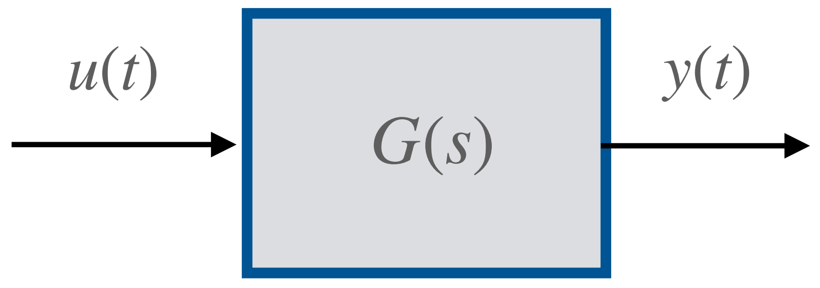
$$R_{PID}(s) = K_P \left(1 + \frac{1}{T_I s} + \frac{T_D s}{1 + \frac{T_D}{N} s}\right)$$




Effect of the new pole on the Bode plots

# Design of PID Controllers

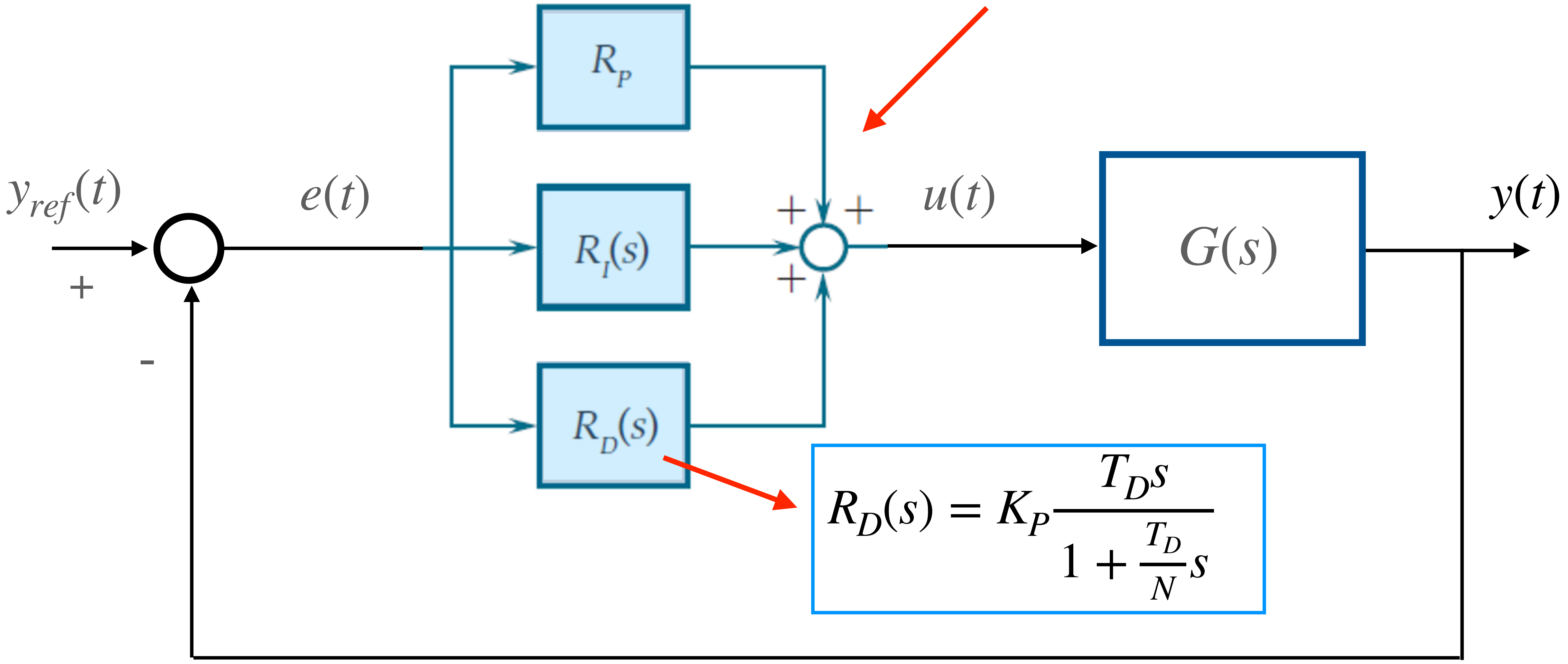
Process




$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$

PID Controller: causal version




Alternative representation:

$$R_{PID_{id}}(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

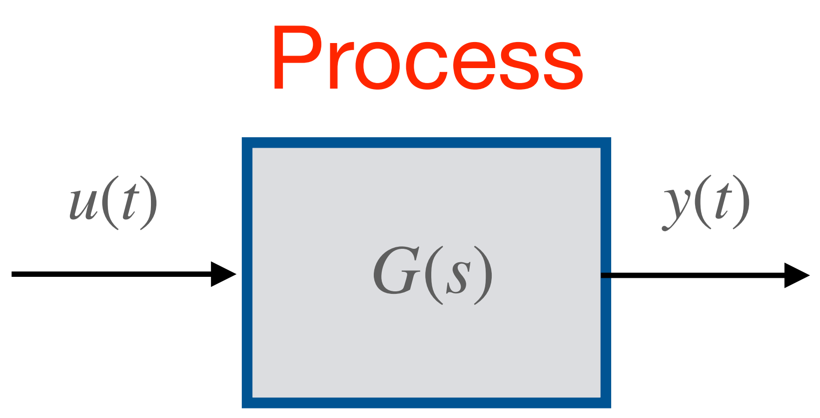
It is causal $\quad R_{PID}(s) = K_P\left(1 + \frac{1}{T_I s} + \frac{T_D s}{1 + \frac{T_D}{N} s}\right)$

Effect of the new pole on the PID zeros:

$$R_{PID}(s) = K_P\left(1 + \frac{1}{T_I s} + \frac{T_D N s}{T_D s + N}\right) = \frac{K_P\left[T_D s^2 + \left(1 + \frac{T_D}{T_I}\right) s + \frac{1}{T_I} N\right]}{s(T_D s + N)}$$

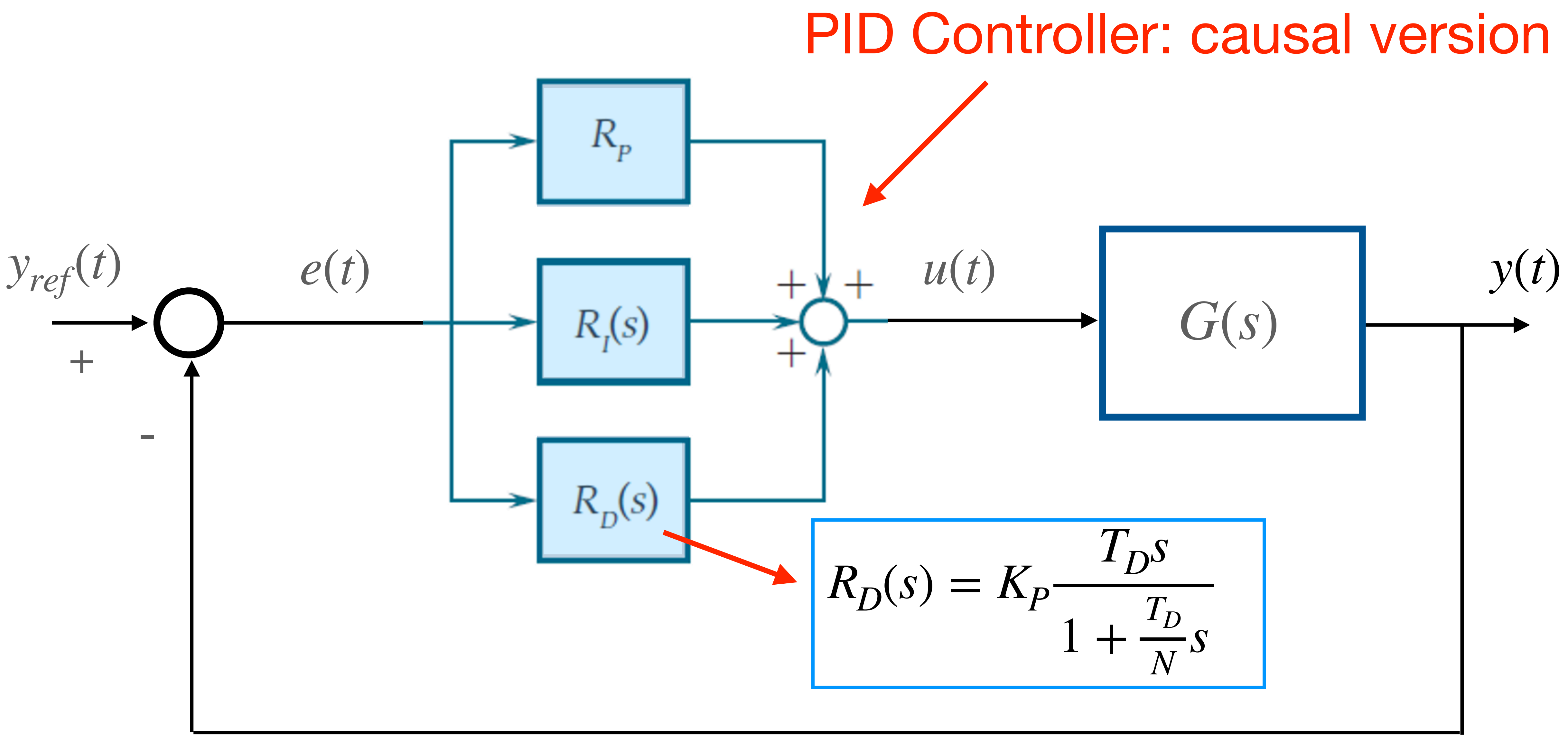
UNIVERSITÀ DI PAVIA

# Design of PID Controllers




$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$




Alternative representation:

$$R_{PID_{id}}(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

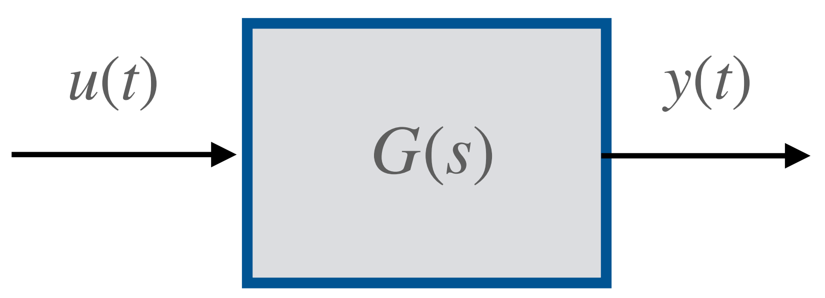
It is causal $\quad R_{PID}(s) = K_P\left(1 + \frac{1}{T_I s} + \frac{T_D s}{1 + \frac{T_D}{N} s}\right)$

Effect of the new pole on the PID zeros:

$$R_{PID}(s) = K_P\left(1 + \frac{1}{T_I s} + \frac{T_D N s}{T_D s + N}\right) = \frac{K_P\left[T_D s^2 + \left(1 + \frac{T_D}{T_I}\right)s + \frac{1}{T_I}N\right]}{s(T_D s + N)}$$

# Design of PID Controllers

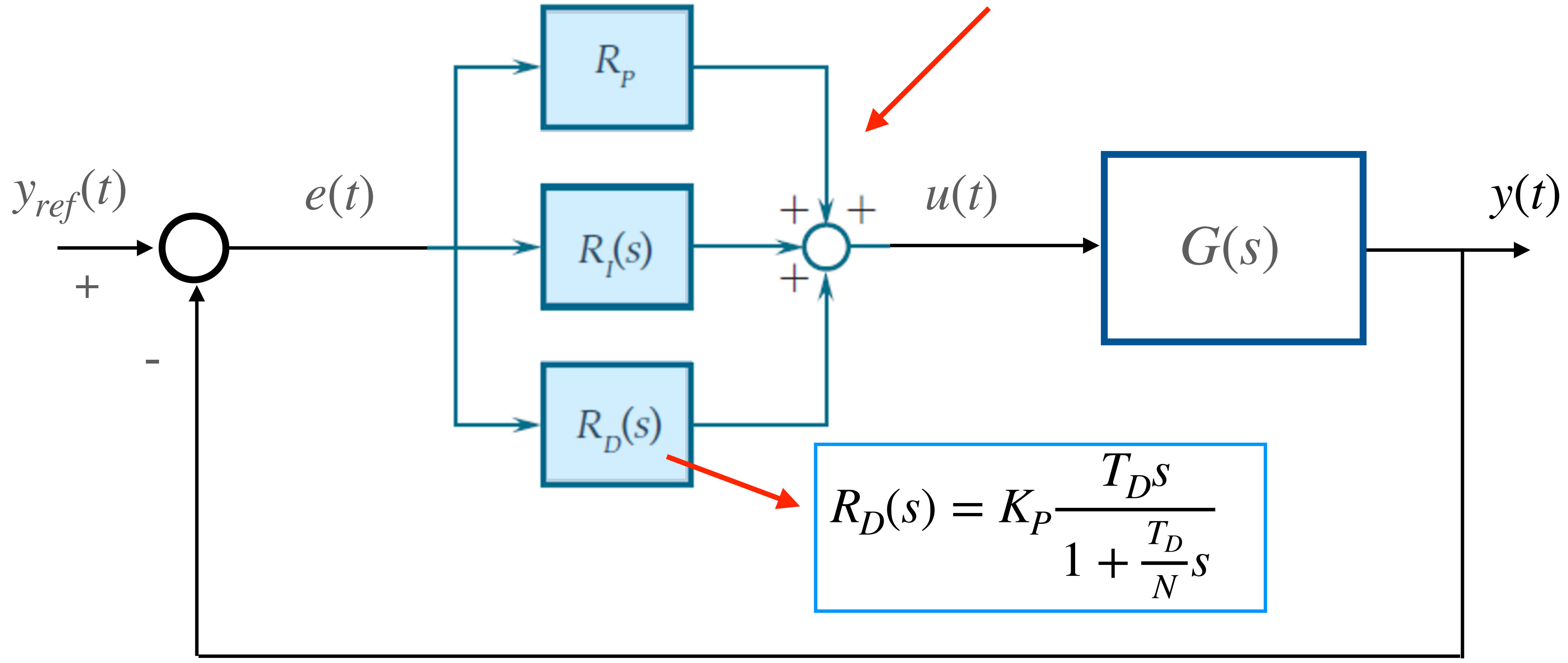
Process




$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$

PID Controller: causal version




$$R_D(s) = K_P \frac{T_D s}{1 + \frac{T_D}{N} s}$$

Alternative representation:

$$R_{PID_{id}}(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s = K_P \left(1 + \frac{1}{T_I s} + T_D s\right)$$

Effect of the new pole on the PID zeros:

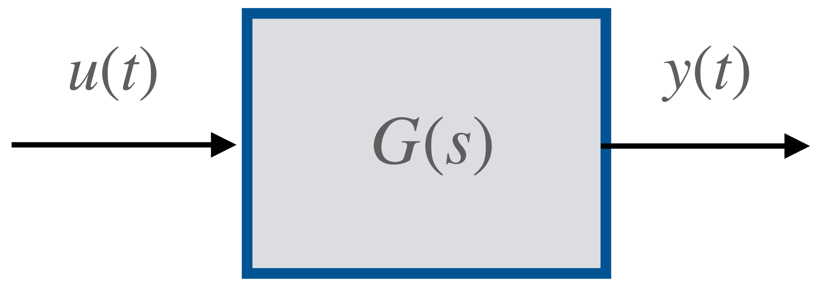
It is causal

$$R_{PID}(s) = K_P \left(1 + \frac{1}{T_I s} + \frac{T_D s}{1 + \frac{T_D}{N} s}\right) \longrightarrow R_{PID}(s) = K_P \left(1 + \frac{1}{T_I s} + \frac{T_D N s}{T_D s + N}\right) = \frac{K_P \left[T_D s^2 + \left(1 + \frac{T_D}{T_I}\right) s + \frac{1}{T_I} N\right]}{s(T_D s + N)}$$

$N \to +\infty$ the zeros are practically the same as before

# Design of PID Controllers

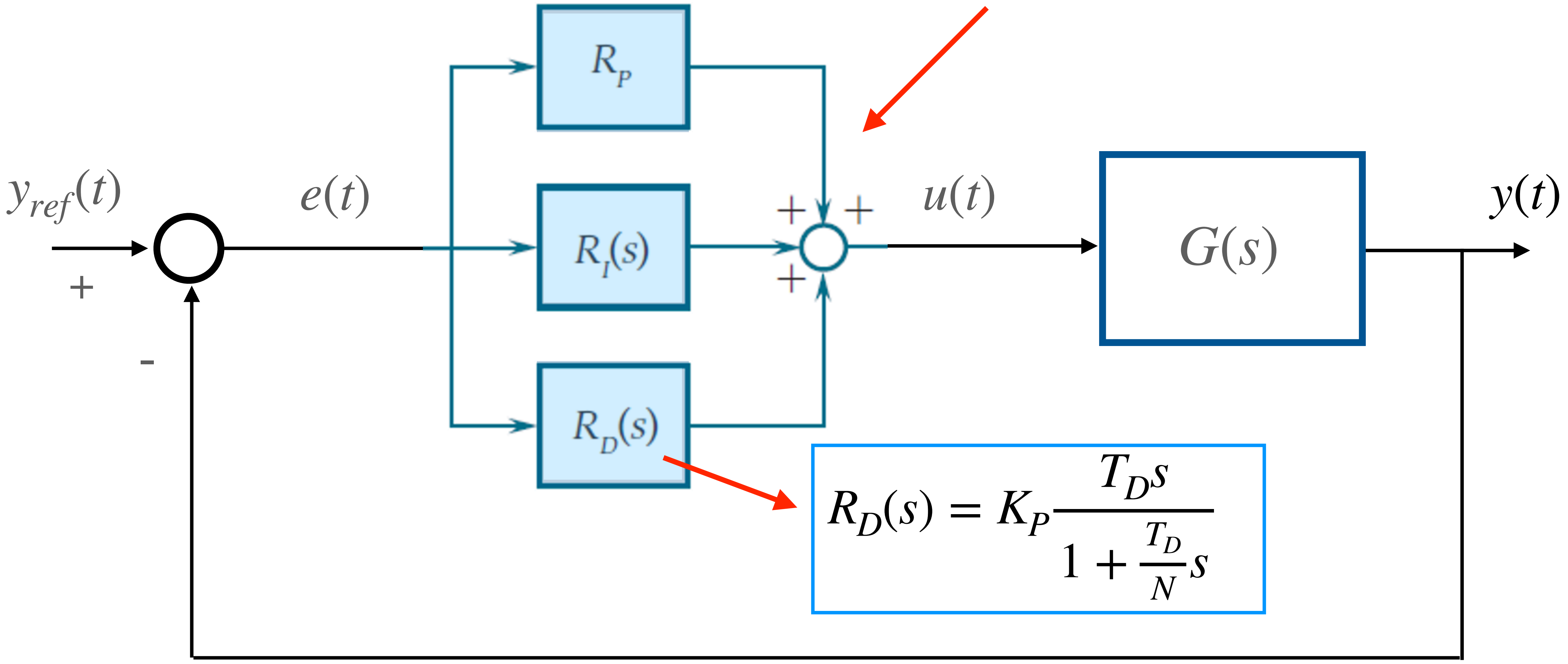
Process




$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$

PID Controller: causal version




$$R_D(s) = K_P \frac{T_D s}{1 + \frac{T_D}{N} s}$$

Alternative representation:

$$R_{PID_{id}}(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s = K_P \left(1 + \frac{1}{T_I s} + T_D s\right)$$

It is causal $\quad R_{PID}(s) = K_P \left(1 + \frac{1}{T_I s} + \frac{T_D s}{1 + \frac{T_D}{N} s}\right)$ $\longrightarrow$ $R_{PID}(s) = K_P \left(1 + \frac{1}{T_I s} + \frac{T_D N s}{T_D s + N}\right) = \frac{K_P \left[T_D s^2 + \left(1 + \frac{T_D}{T_I}\right) s + \frac{1}{T_I} N\right]}{s(T_D s + N)}$

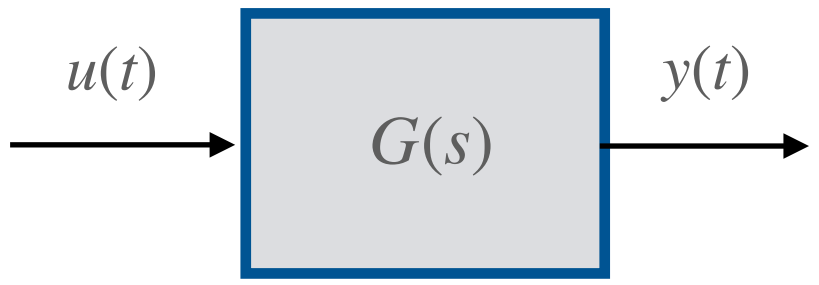
$N \to +\infty$ the zeros are practically the same as before

Effect of the new pole on the PID zeros:

when $T_I = 4T_D$ the zeros coincide in $s = -\frac{1}{2T_D}$

# Design of PID Controllers

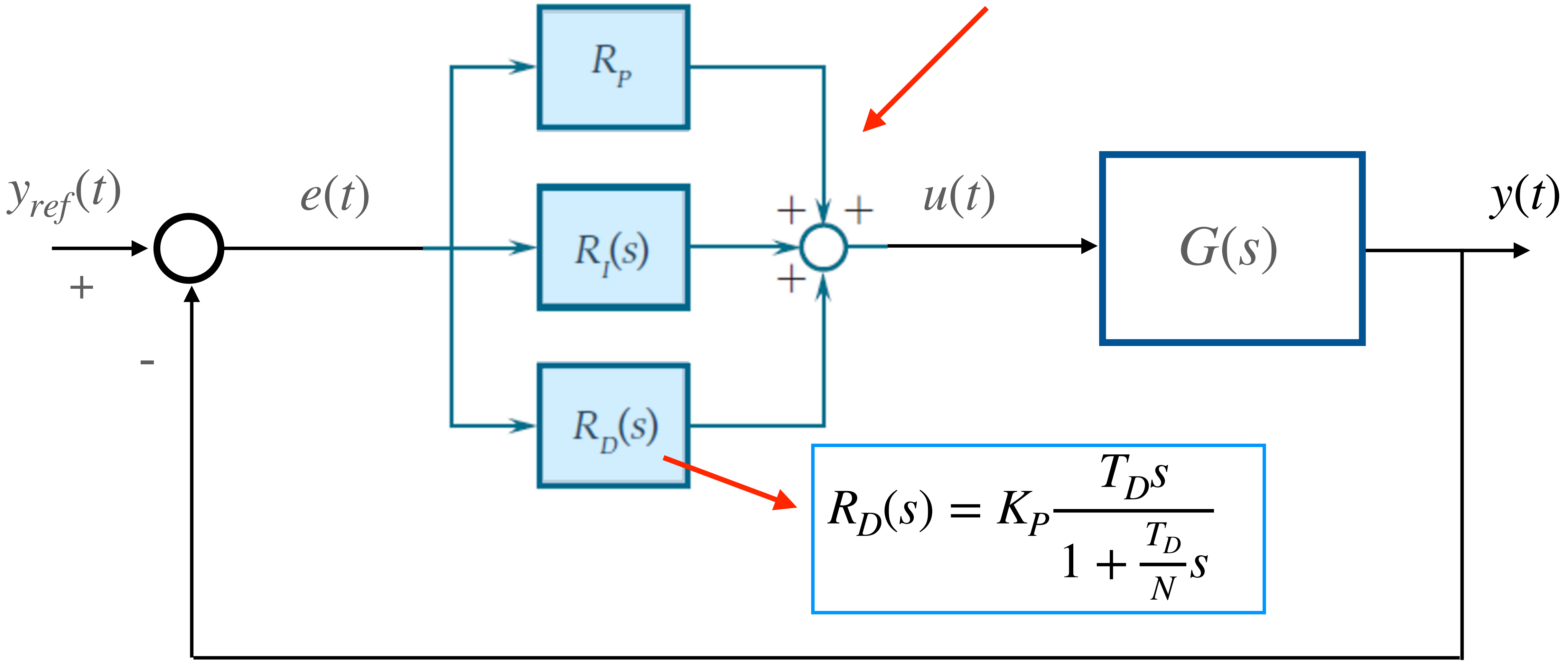
Process




$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$

PID Controller: causal version




$$R_D(s) = K_P \frac{T_D s}{1 + \frac{T_D}{N} s}$$

Alternative representation:

$$R_{PID_{id}}(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s = K_P \left(1 + \frac{1}{T_I s} + T_D s\right)$$

Effect of the new pole on the PID zeros:

choice made to simplify PID parameter tuning

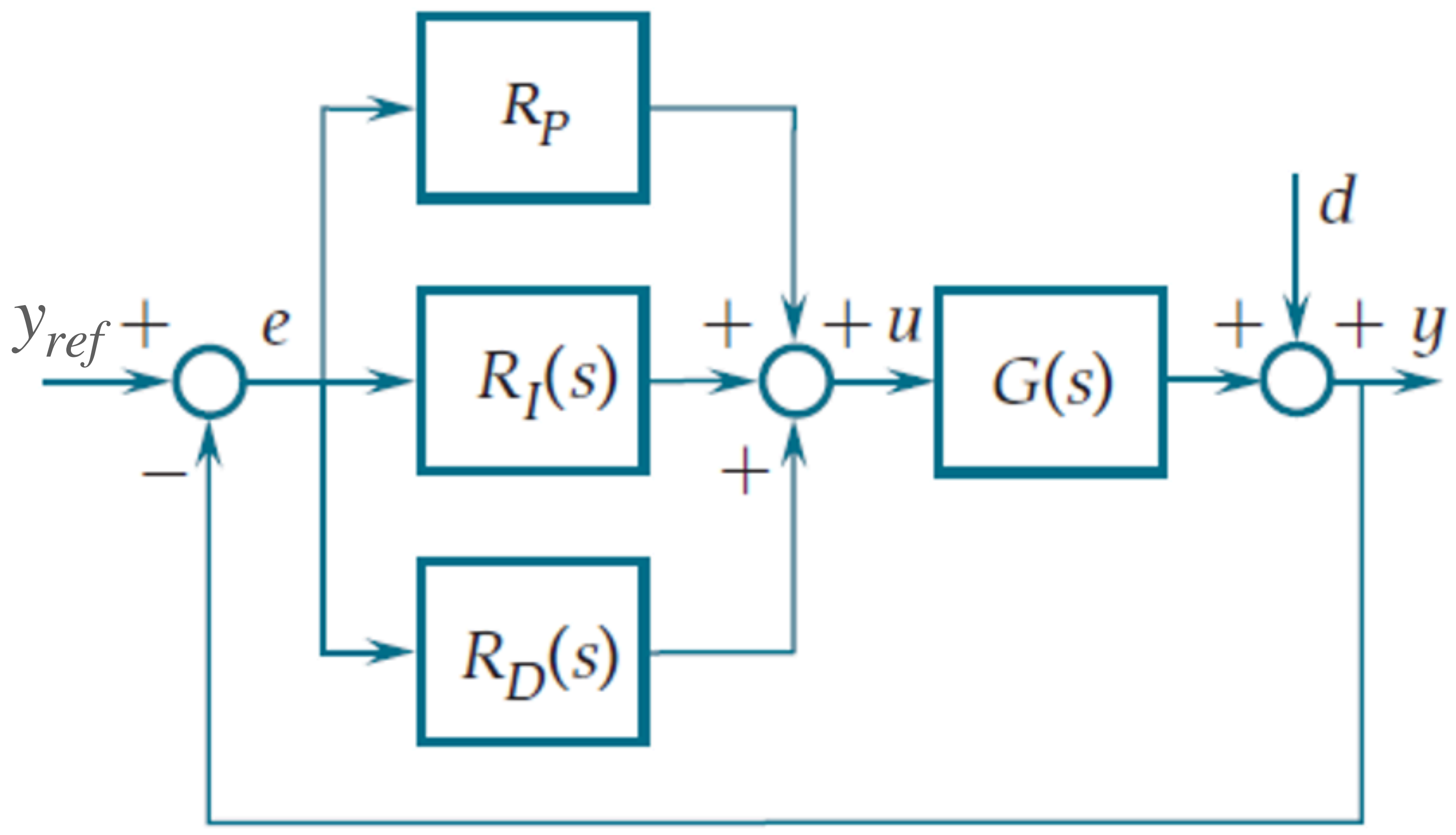
when $T_I = 4T_D$ the zeros coincide in $s = -\dfrac{1}{2T_D}$

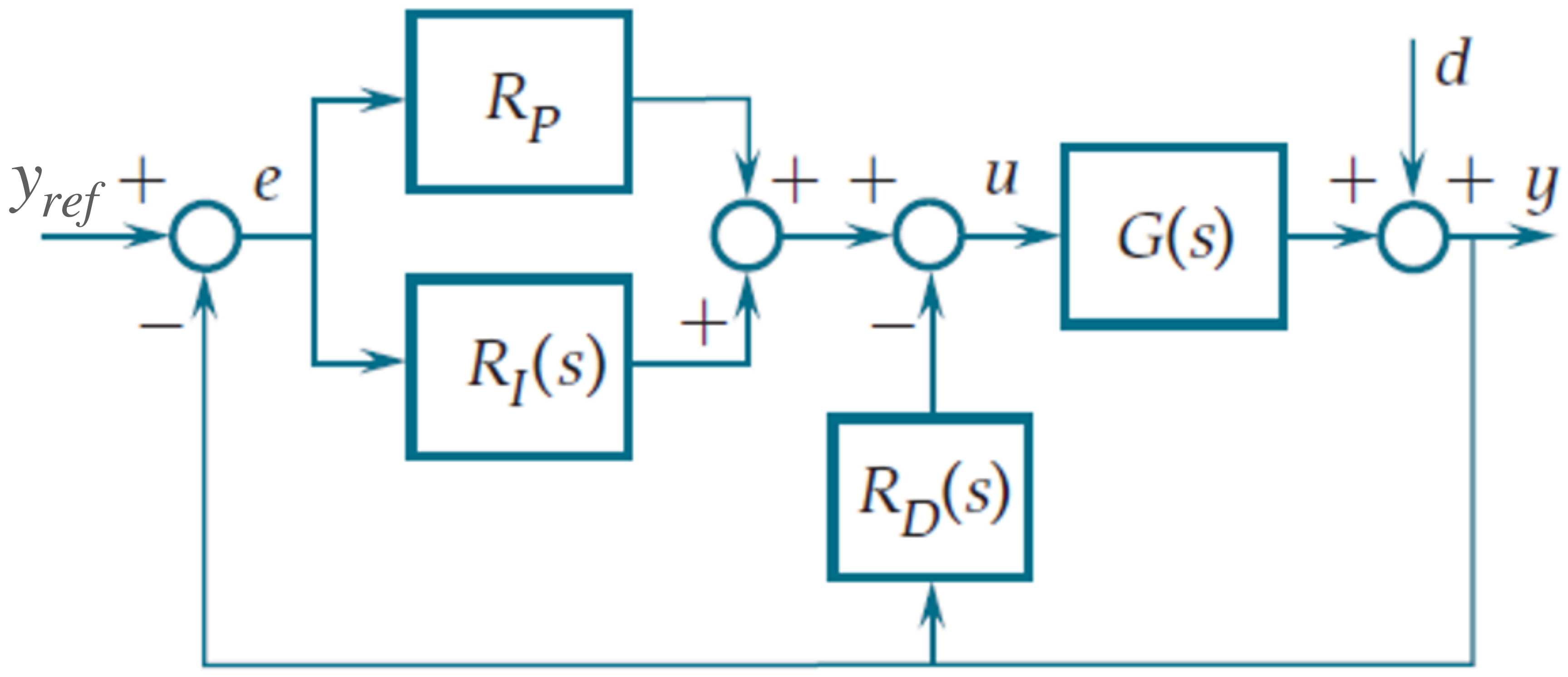
It is causal

$$R_{PID}(s) = K_P \left(1 + \frac{1}{T_I s} + \frac{T_D s}{1 + \frac{T_D}{N} s}\right) \longrightarrow R_{PID}(s) = K_P \left(1 + \frac{1}{T_I s} + \frac{T_D N s}{T_D s + N}\right) = \frac{K_P \left[T_D s^2 + \left(1 + \frac{T_D}{T_I}\right) s + \frac{1}{T_I} N\right]}{s(T_D s + N)}$$

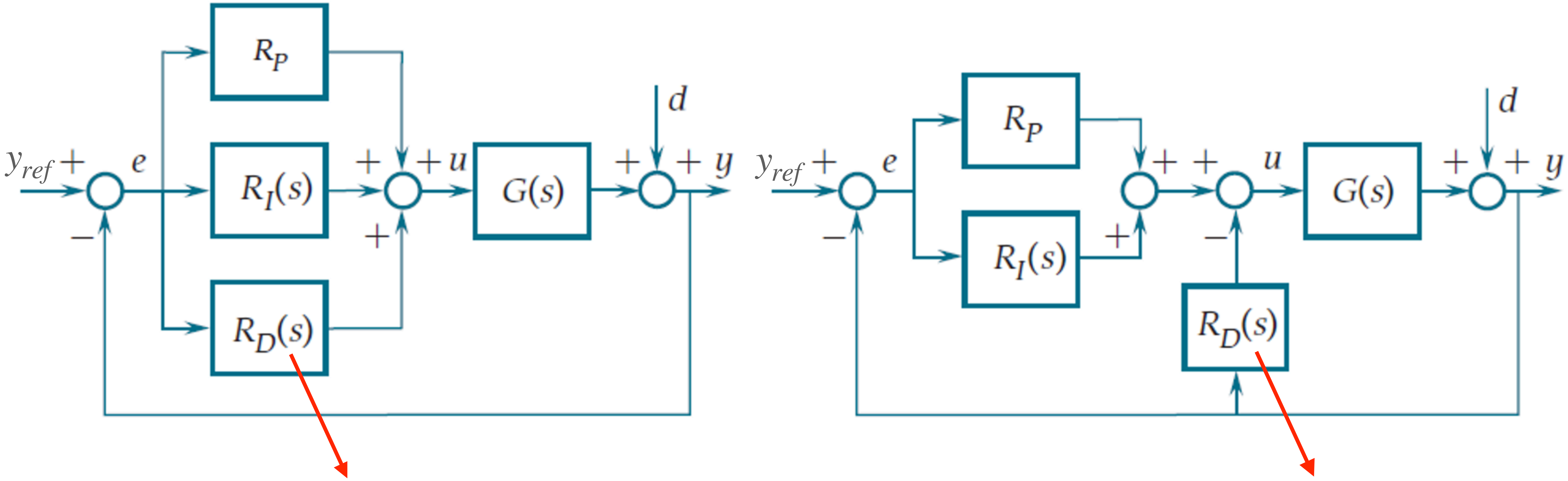
$N \to +\infty$ the zeros are practically the same as before

# Practical Implementation of PID Controllers



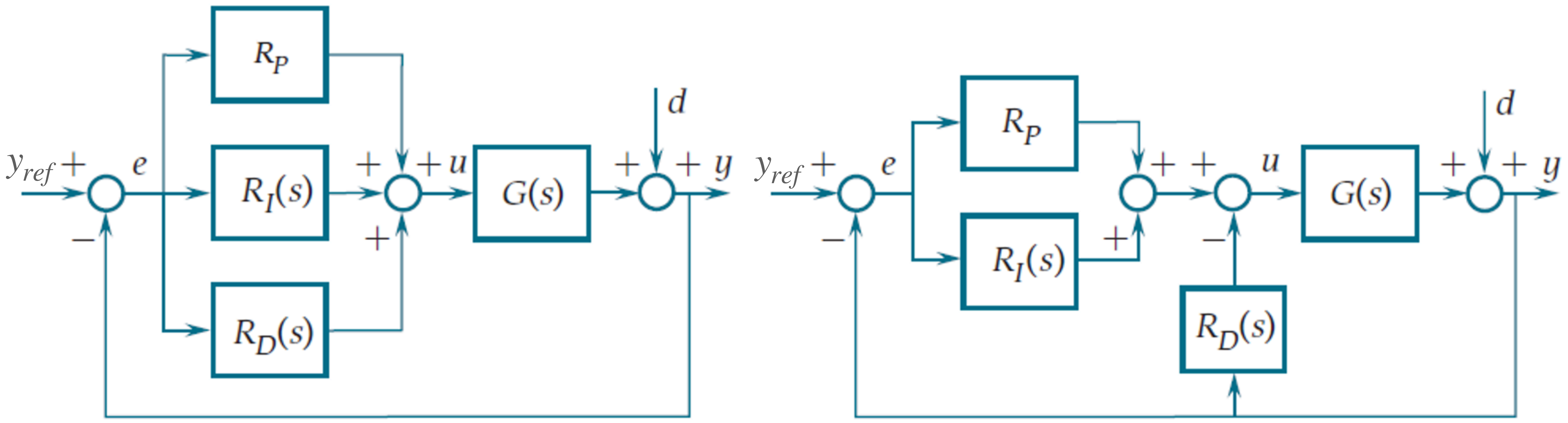
# Practical Implementation of PID Controllers

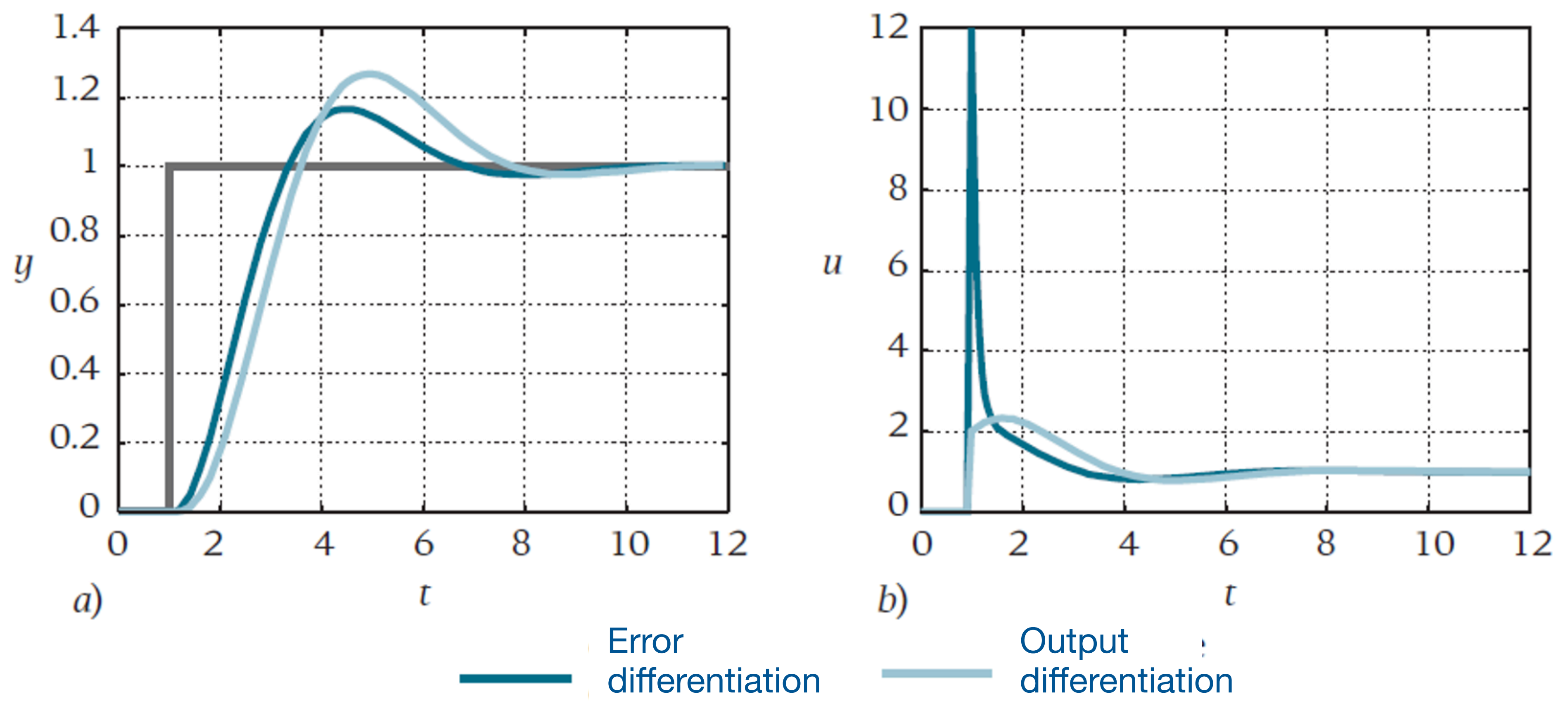



$$R_D(s) = K_P \frac{T_D s}{1 + \frac{T_D}{N} s}$$

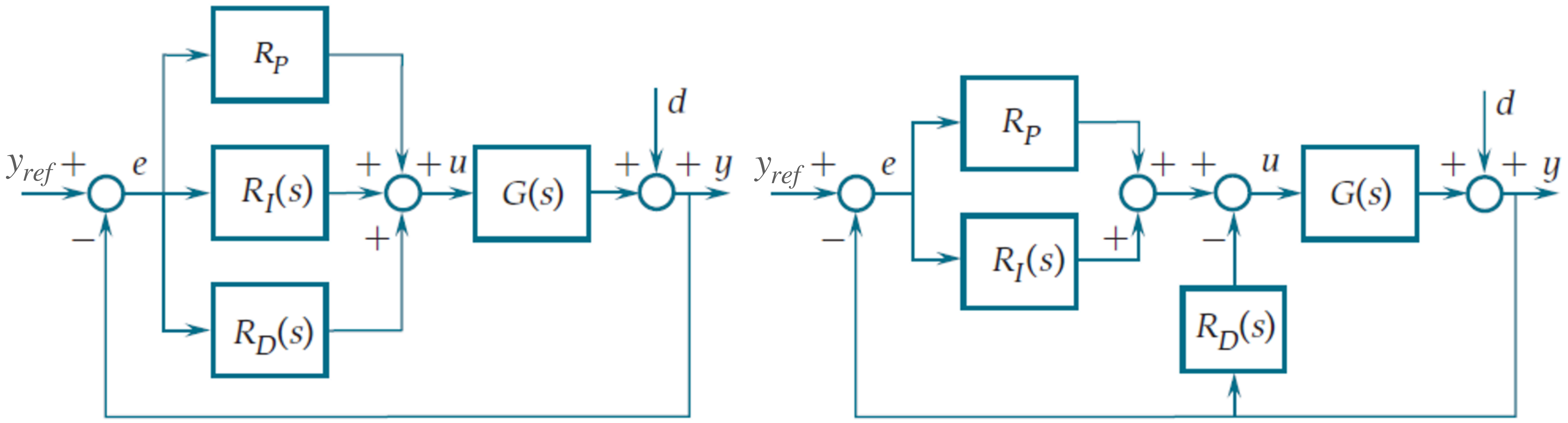
$$R_D(s) = K_P \frac{T_D s}{1 + \frac{T_D}{N} s}$$

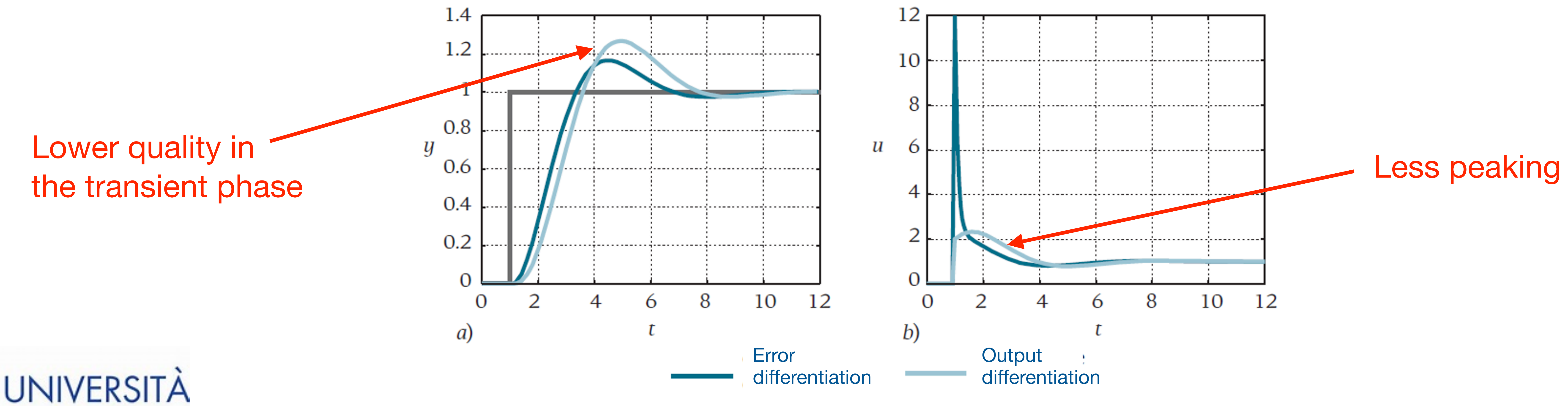
# Practical Implementation of PID Controllers



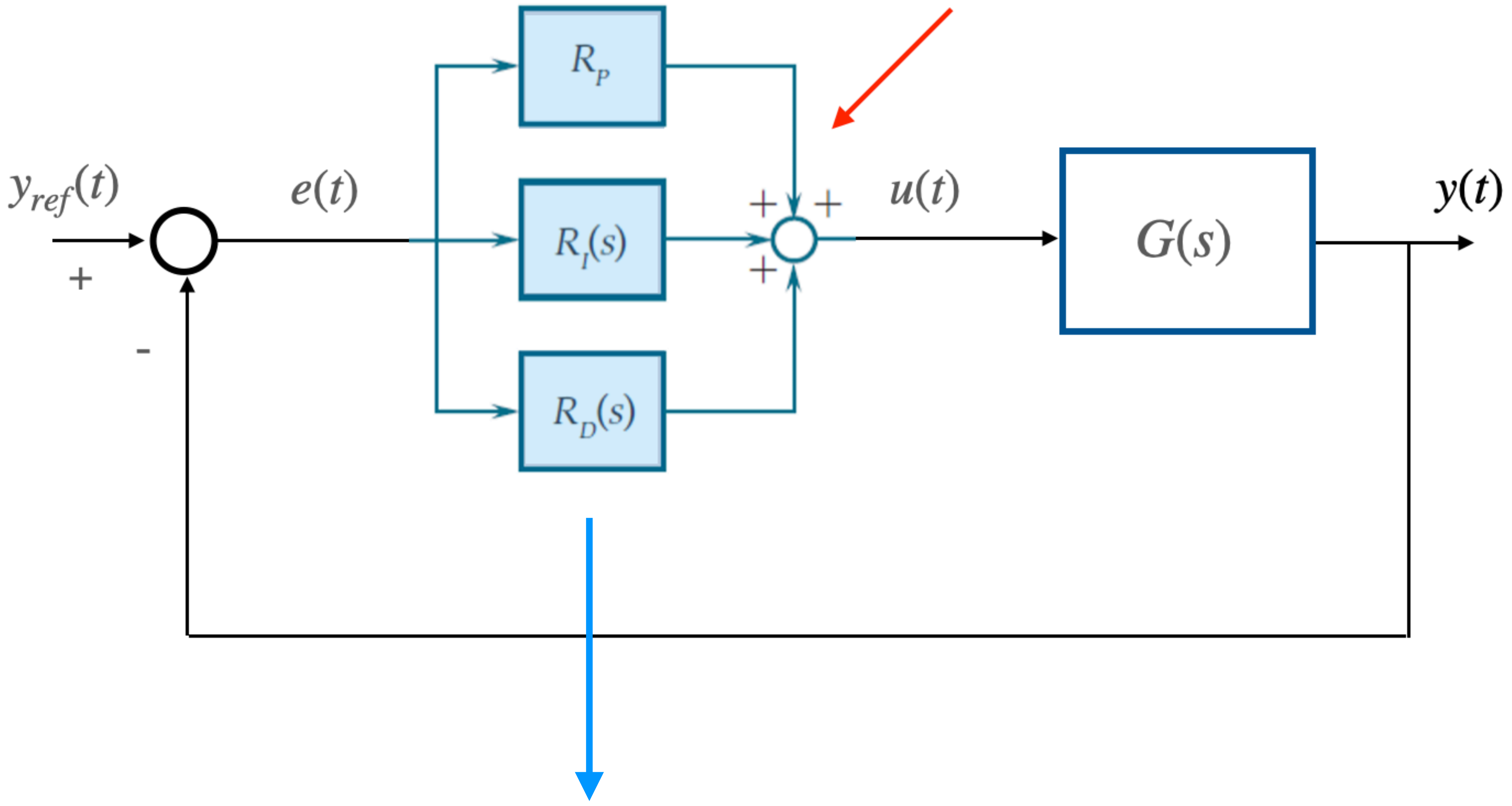
# Practical Implementation of PID Controllers



# Practical Implementation of PID Controllers: Tuning Rules

PID Controller



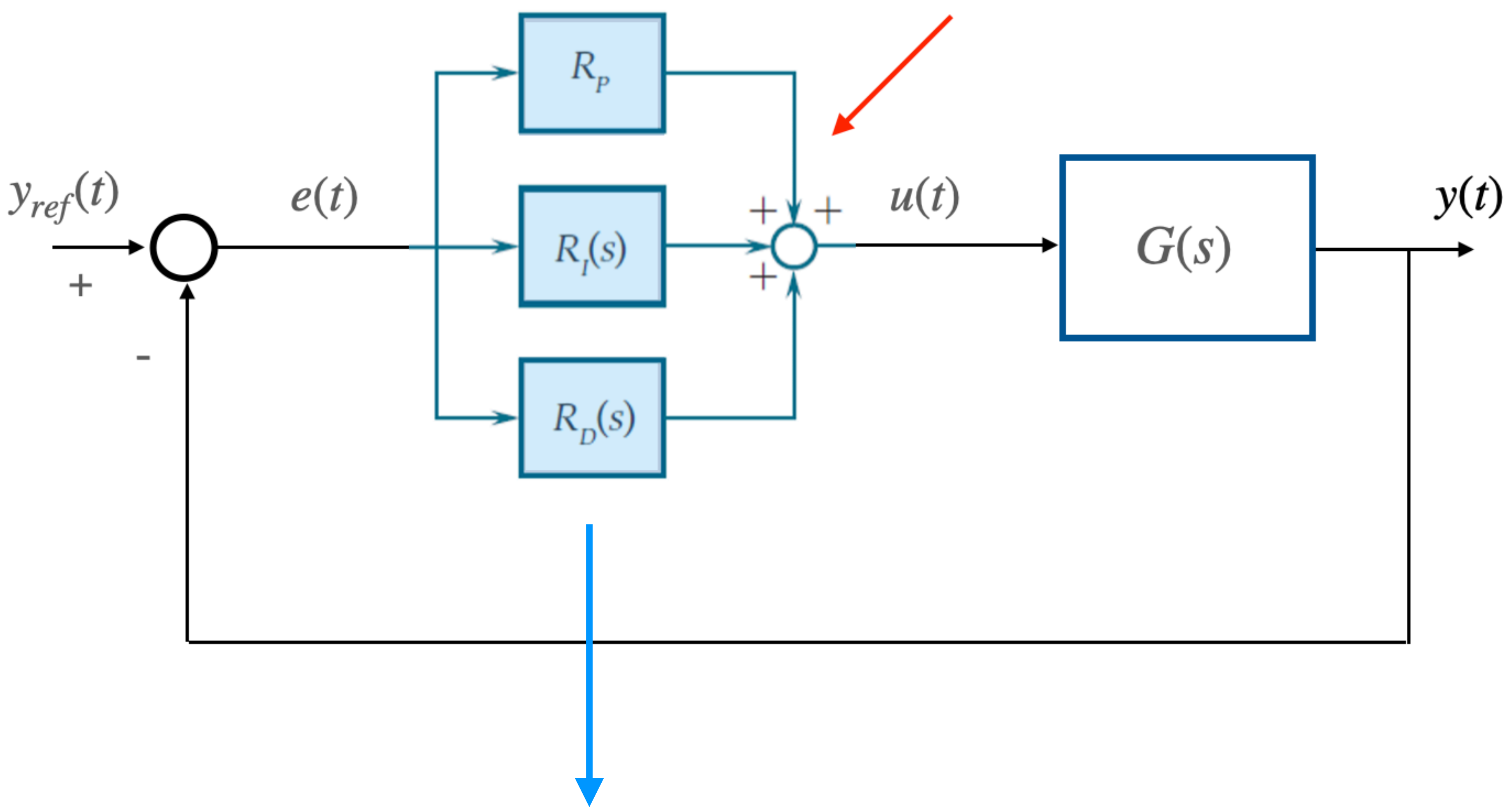

Testing phase:

$$R_P(s) = K_P, \quad R_I(s) = 0, \quad R_D(s) = 0$$

Closed-loop Ziegler-Nichols Method

# Practical Implementation of PID Controllers: Tuning Rules

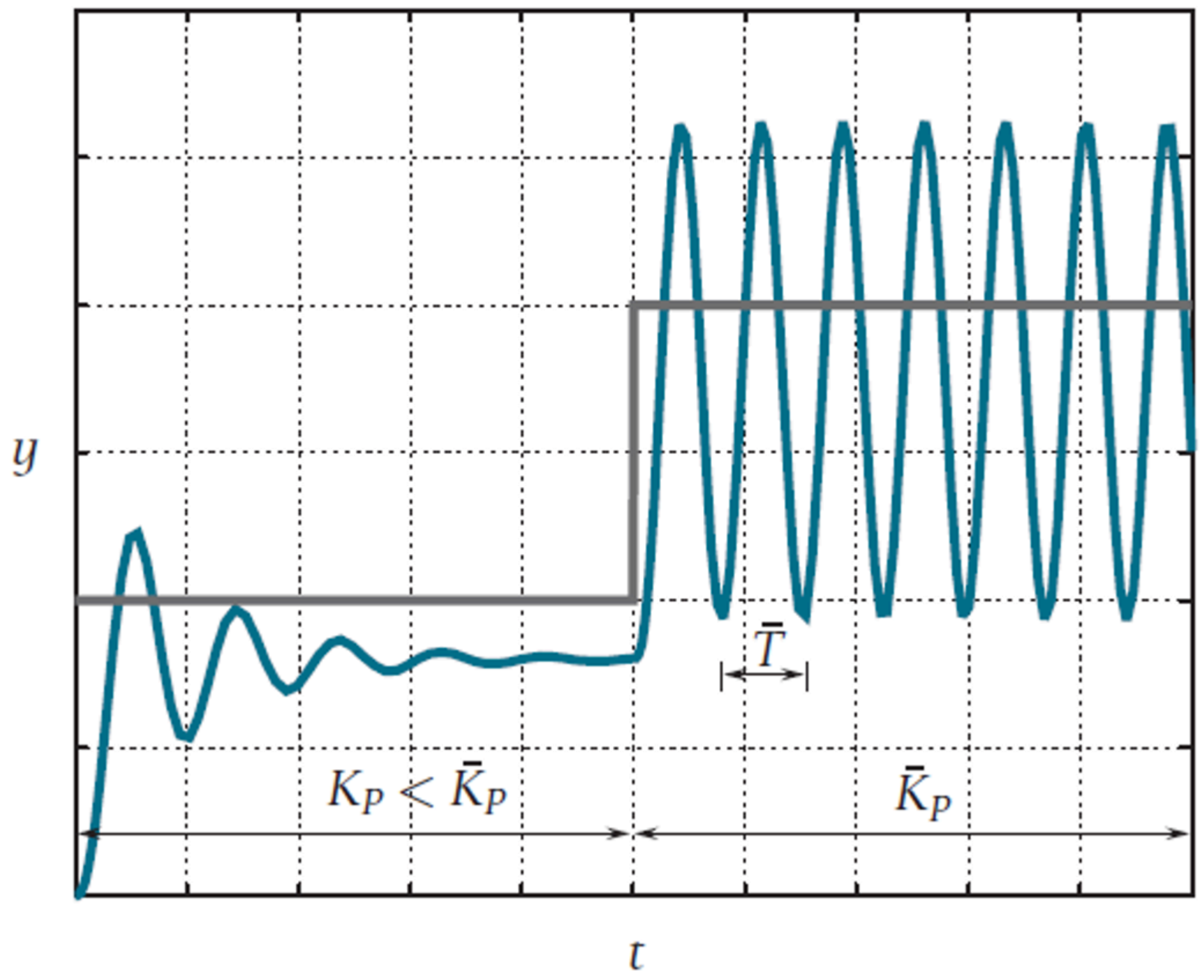
PID Controller




Testing phase:

$$R_P(s) = K_P, \quad R_I(s) = 0, \quad R_D(s) = 0$$

Closed-loop Ziegler-Nichols Method

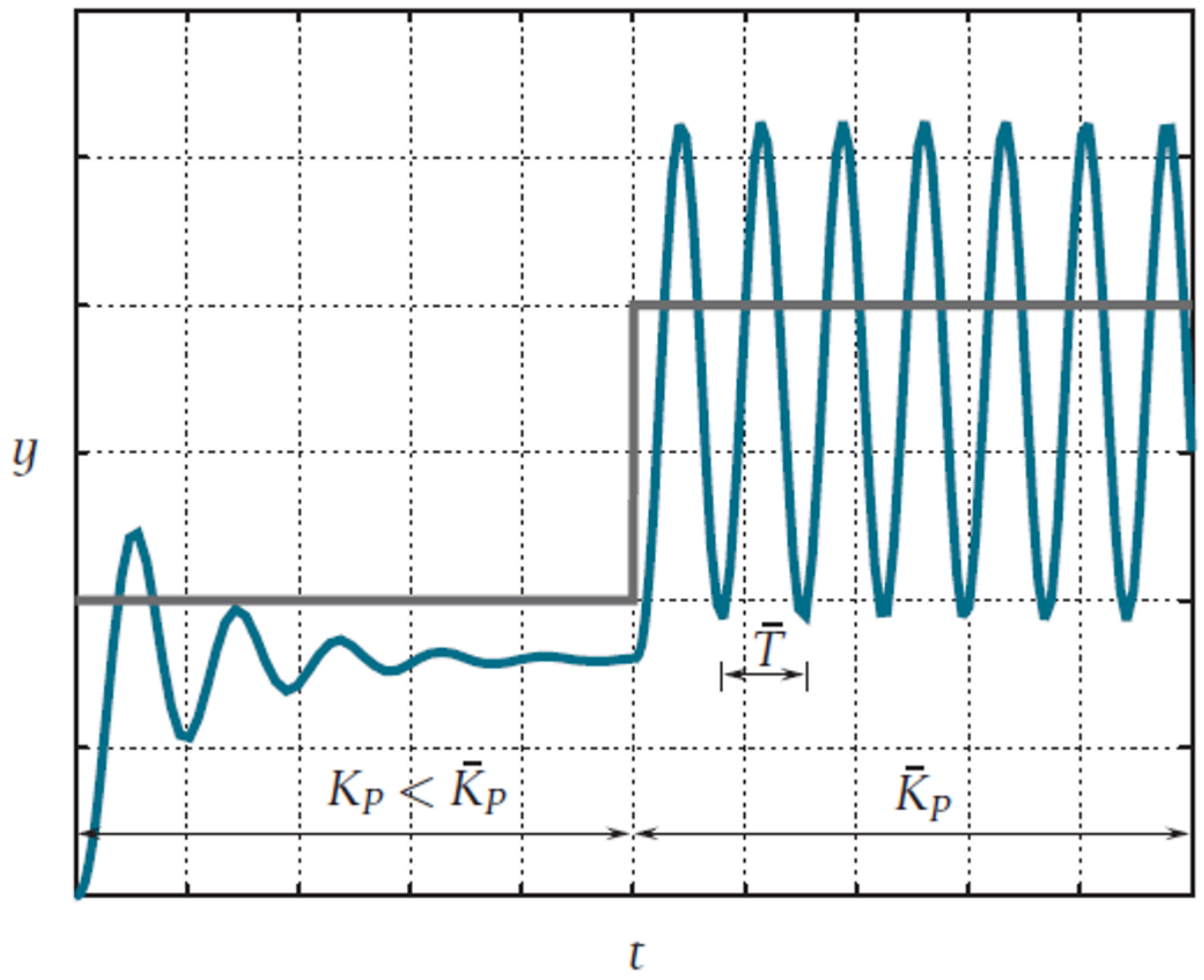


# Practical Implementation of PID Controllers: Tuning Rules

## Closed-loop Ziegler-Nichols Method

| | $K_P$ | $T_I$ | $T_D$ |
|---|---|---|---|
| P | $0.5\overline{K}_P$ | | |
| PI | $0.45\overline{K}_P$ | $0.8\overline{T}$ | |
| PID | $0.6\overline{K}_P$ | $0.5\overline{T}$ | $0.125\overline{T}$ |

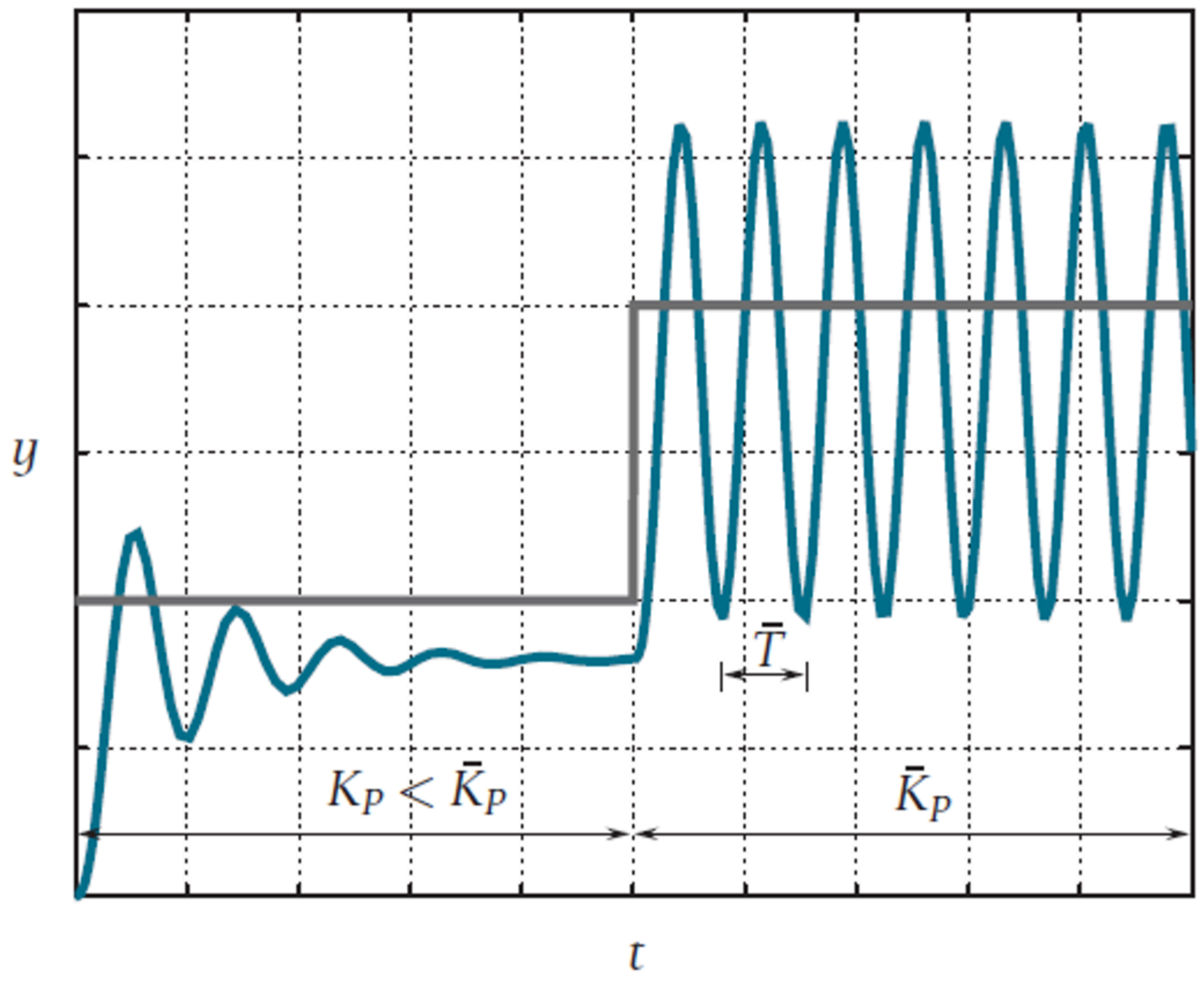
$$R_{PID_{id}}(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$



# Practical Implementation of PID Controllers: Tuning Rules

## Closed-loop Ziegler-Nichols Method

| | $K_P$ | $T_I$ | $T_D$ |
|---|---|---|---|
| P | $0.5\overline{K}_P$ | | |
| PI | $0.45\overline{K}_P$ | $0.8\overline{T}$ | |
| PID | $0.6\overline{K}_P$ | $0.5\overline{T}$ | $0.125\overline{T}$ |



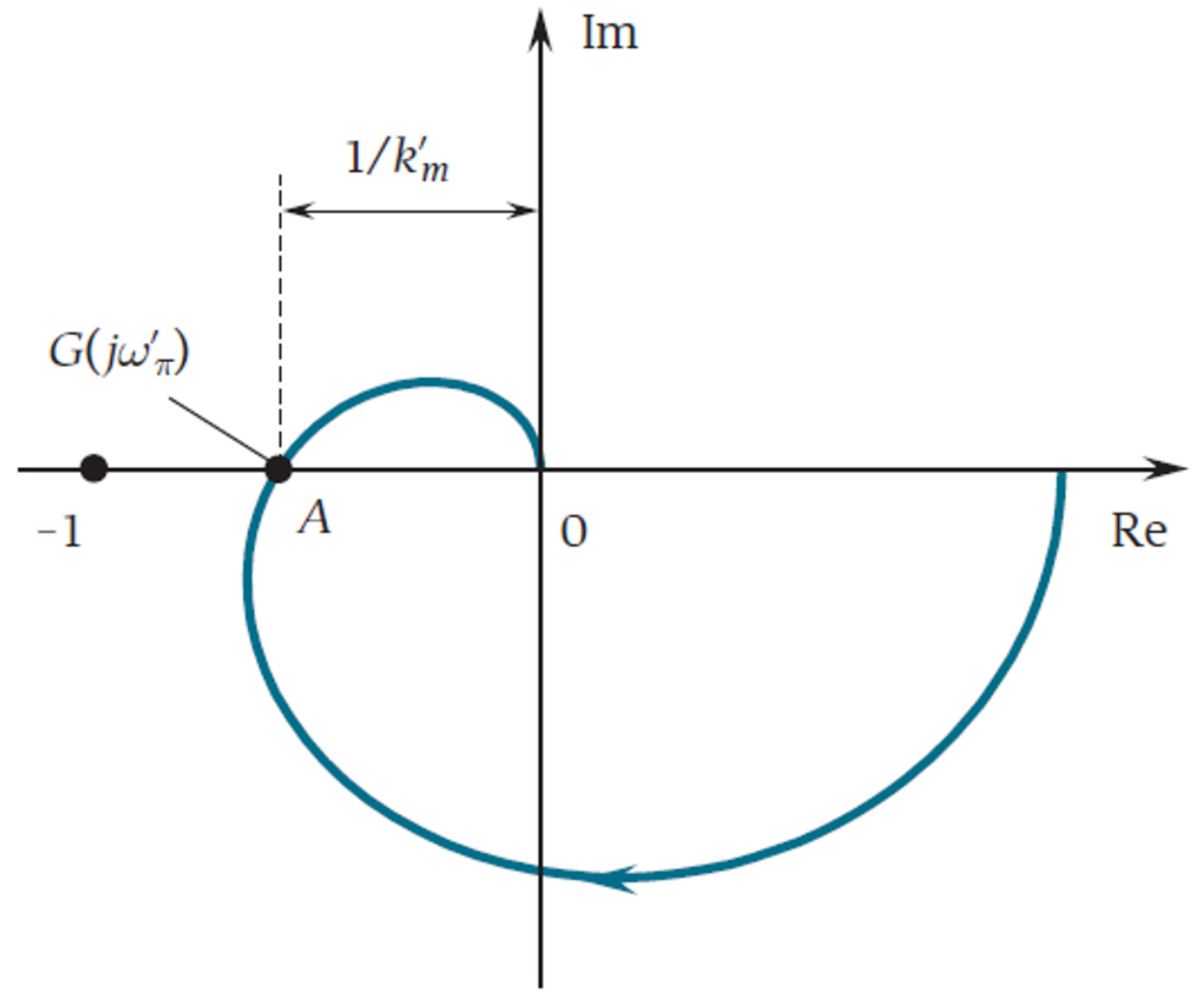

$$R_{PID_{id}}(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

Note that $T_I = 4T_D$ $\rightarrow$ the PID zeros coincide in $s = -\dfrac{1}{2T_D}$

# Practical Implementation of PID Controllers: Tuning Rules

## Closed-loop Ziegler-Nichols Method: Interpretation

| | $K_P$ | $T_I$ | $T_D$ |
|---|---|---|---|
| P | $0.5\overline{K}_P$ | | |
| PI | $0.45\overline{K}_P$ | $0.8\overline{T}$ | |
| PID | $0.6\overline{K}_P$ | $0.5\overline{T}$ | $0.125\overline{T}$ |




$$R_{PID_{id}}(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

$\bar{K}_p = k'_m \;\rightarrow$ Gain Margin of $G(s)$

$\bar{T} = \dfrac{\pi}{\omega'_\pi} \;\rightarrow$ where $\omega'_\pi$ is the pulse corresponding to point $A$

Note that $T_I = 4T_D \;\rightarrow$ the PID zeros coincide in $s = -\dfrac{1}{2T_D}$

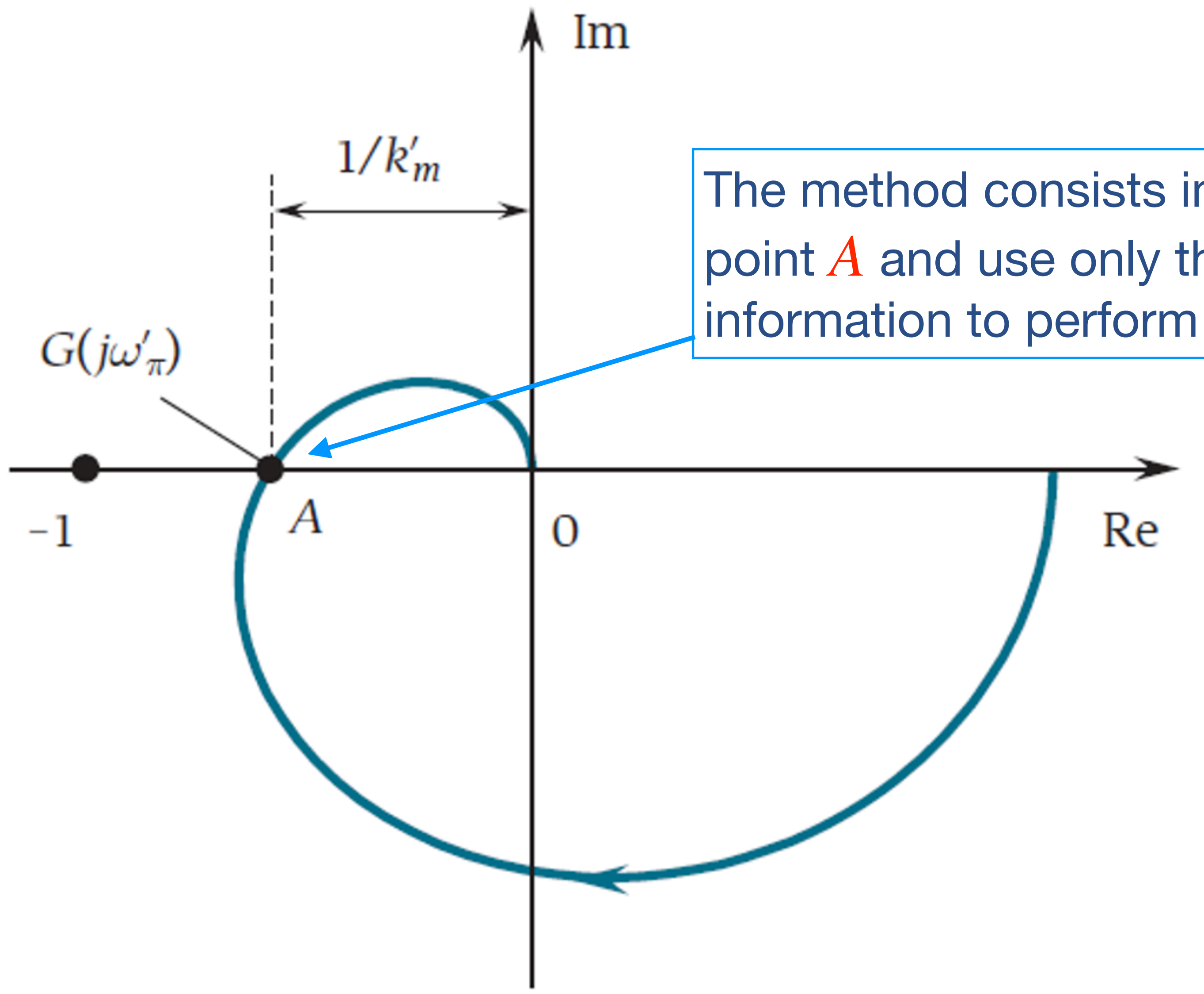
# Practical Implementation of PID Controllers: Tuning Rules

| | $K_P$ | $T_I$ | $T_D$ |
|---|---|---|---|
| P | $0.5\overline{K}_P$ | | |
| PI | $0.45\overline{K}_P$ | $0.8\overline{T}$ | |
| PID | $0.6\overline{K}_P$ | $0.5\overline{T}$ | $0.125\overline{T}$ |

$$R_{PID_{id}}(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

Note that $T_I = 4T_D$ $\rightarrow$ the PID zeros coincide in $s = -\dfrac{1}{2T_D}$

## Closed-loop Ziegler-Nichols Method: Interpretation

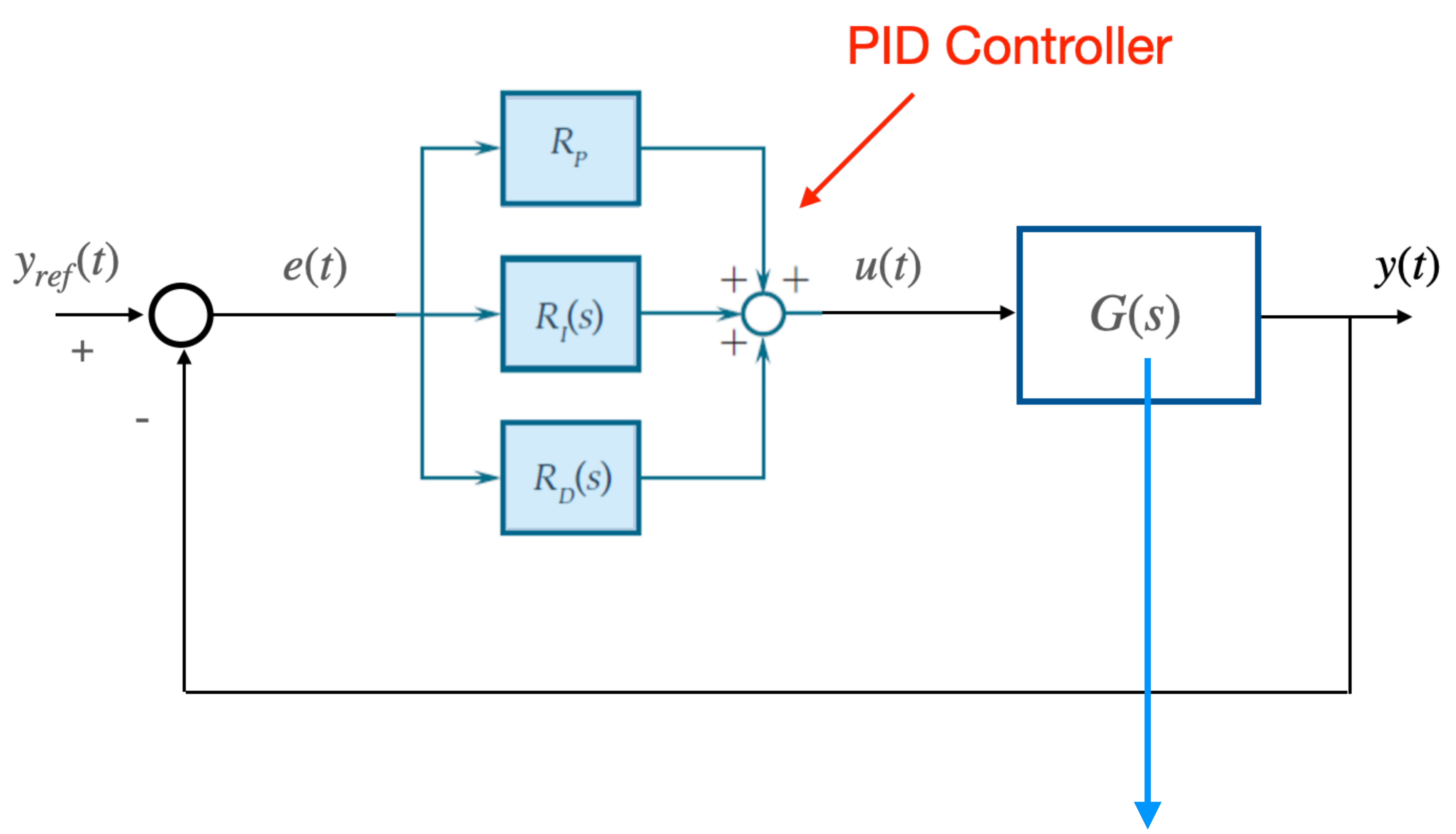



The method consists in finding point $A$ and use only that piece of information to perform the tuning

$\bar{K}_p = k'_m$ $\rightarrow$ Gain Margin of $G(s)$

$\bar{T} = \dfrac{\pi}{\omega'_\pi}$ $\rightarrow$ where $\omega'_\pi$ is the pulse corresponding to point $A$

# Practical Implementation of PID Controllers: Tuning Rules

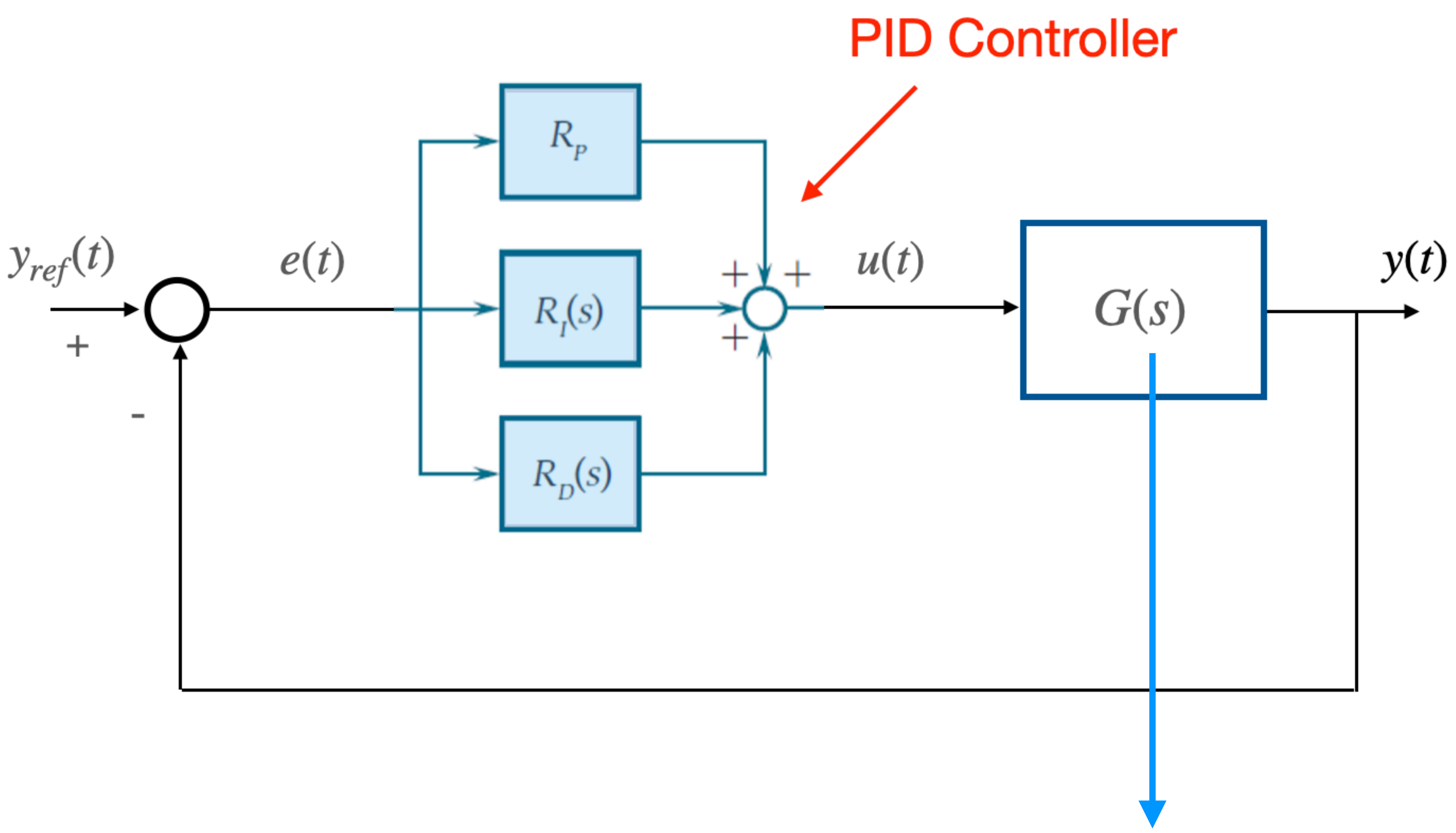



Open-loop Methods

Approximate Model of the Process:

$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1 + Ts}$$

# Practical Implementation of PID Controllers: Tuning Rules




Open-loop Methods

Approximate Model of the Process:

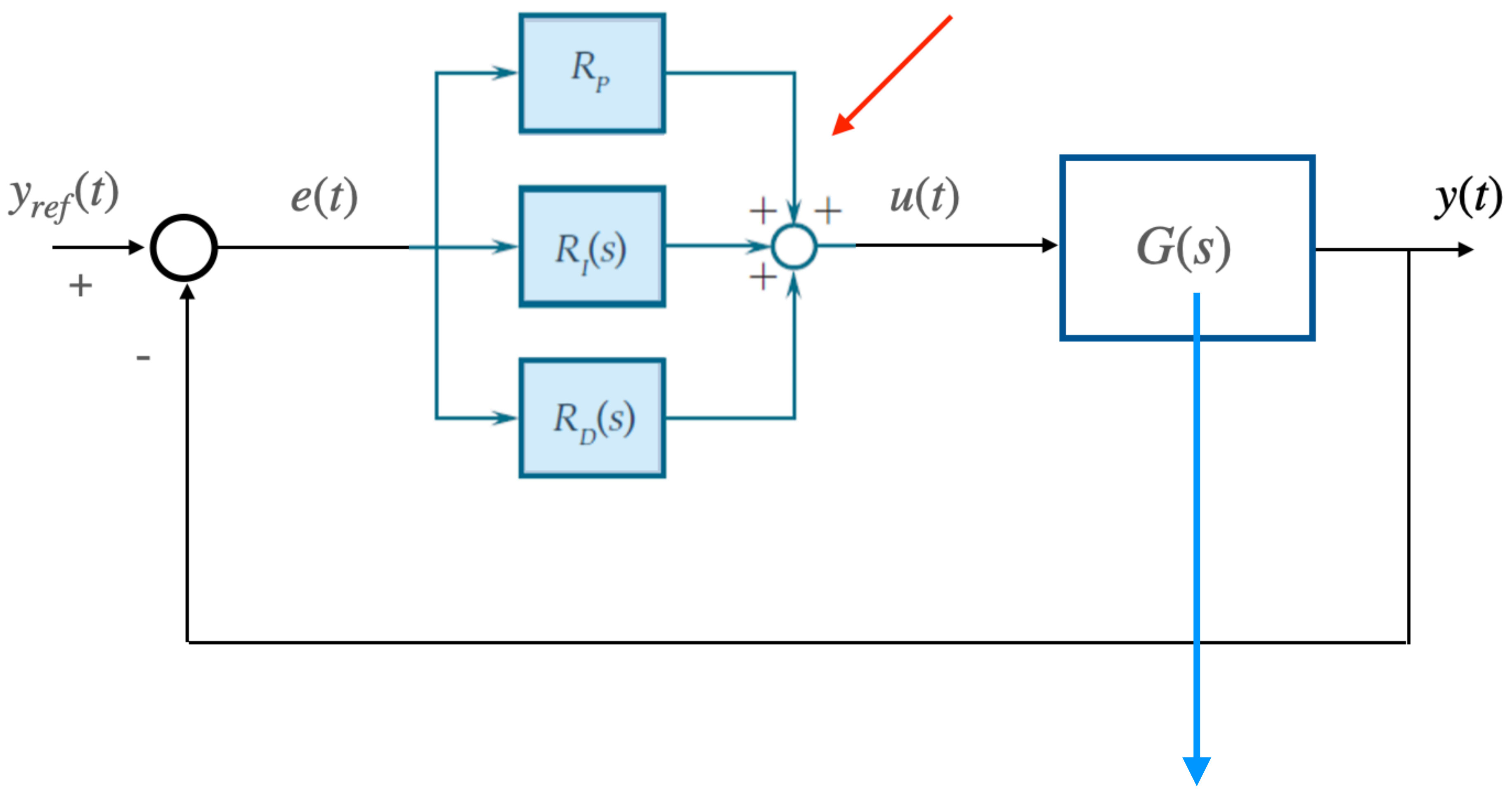
$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1 + Ts}$$

equivalent delay

equivalent time-constant

# Practical Implementation of PID Controllers: Tuning Rules

PID Controller




Approximate Model of the Process:

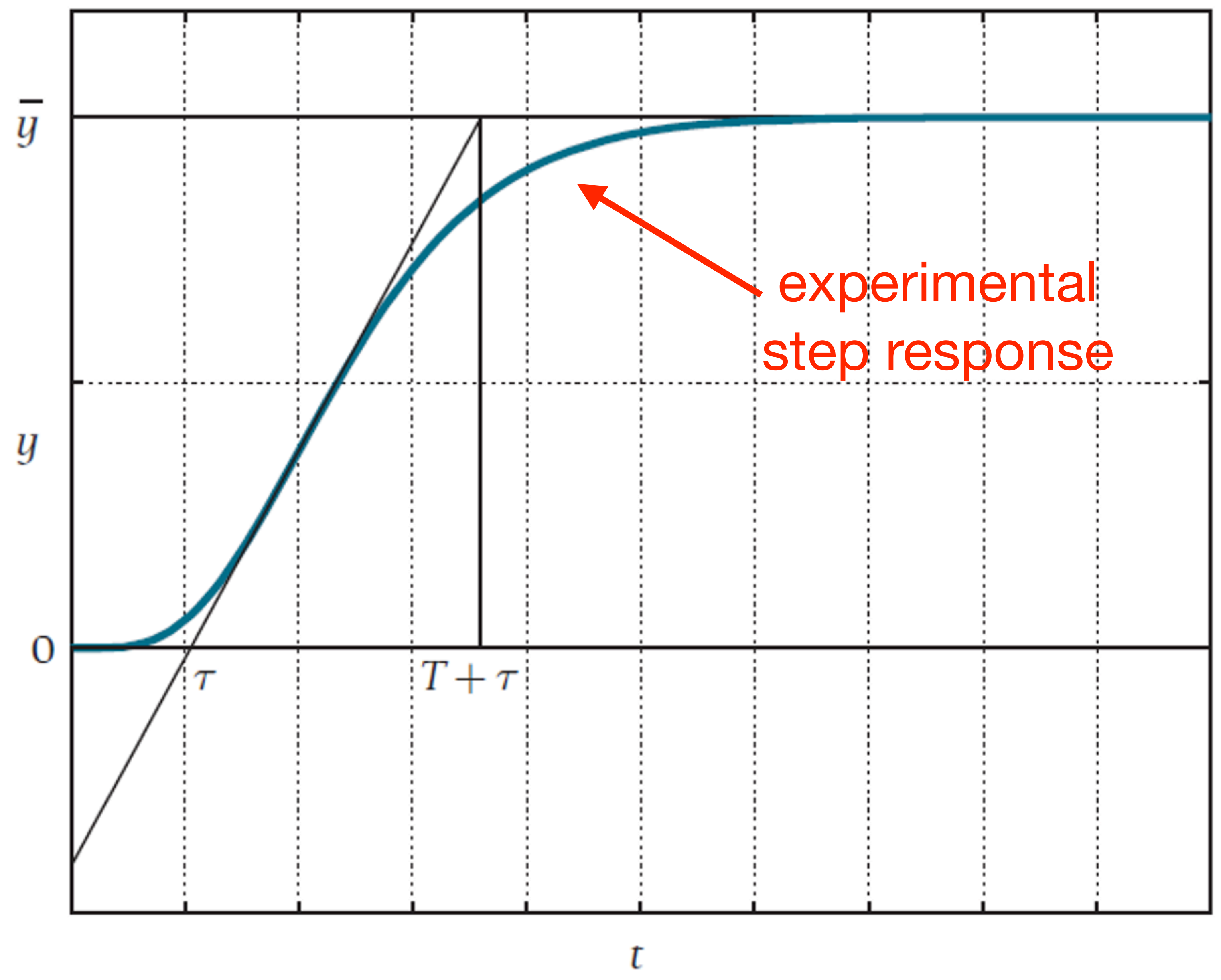
$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1 + Ts}$$

equivalent delay

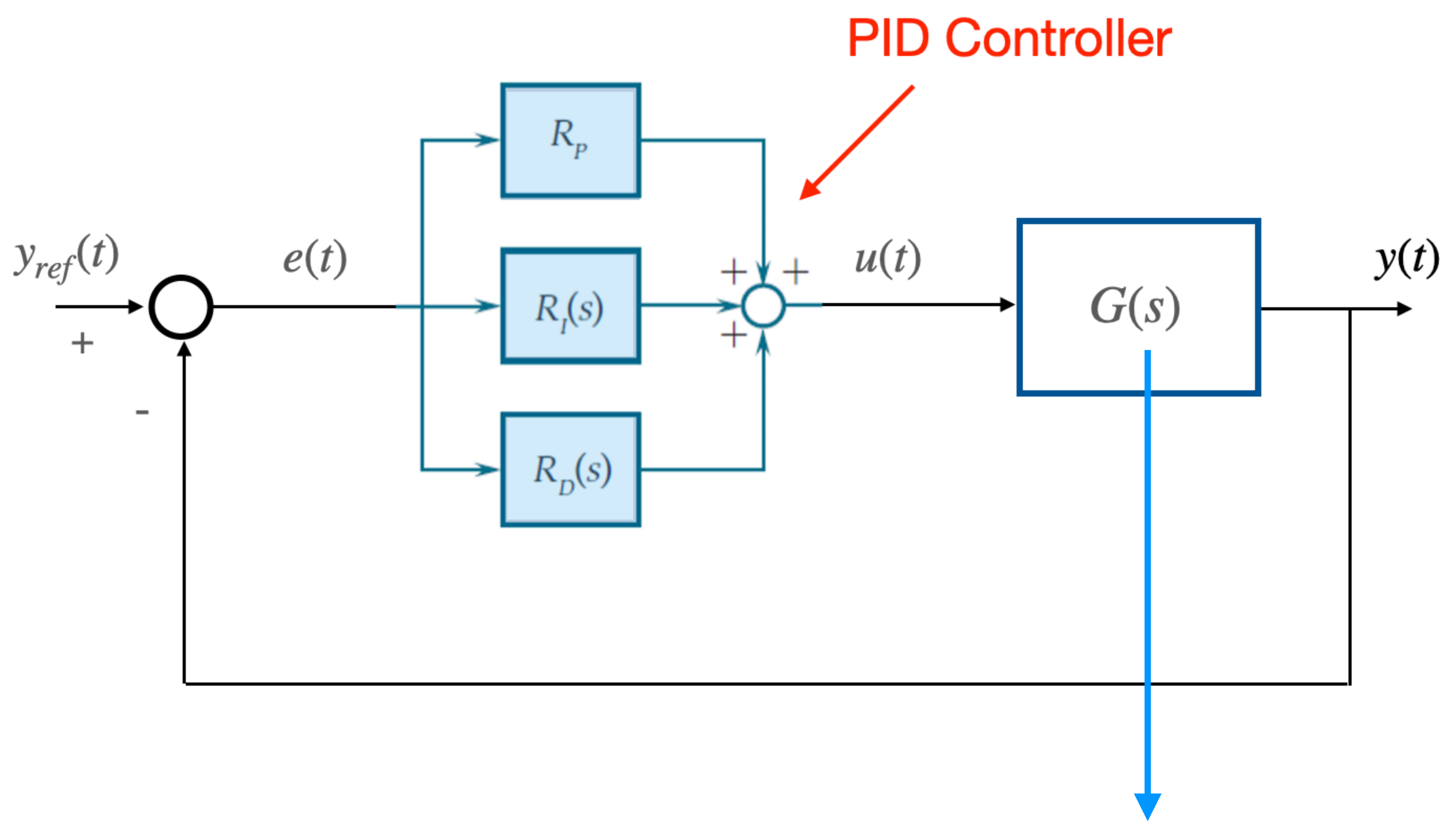
equivalent time-constant

## Open-loop Methods

Parameters $\tau$, $T$ determination via the Tangent Method



# Practical Implementation of PID Controllers: Tuning Rules




Approximate Model of the Process:

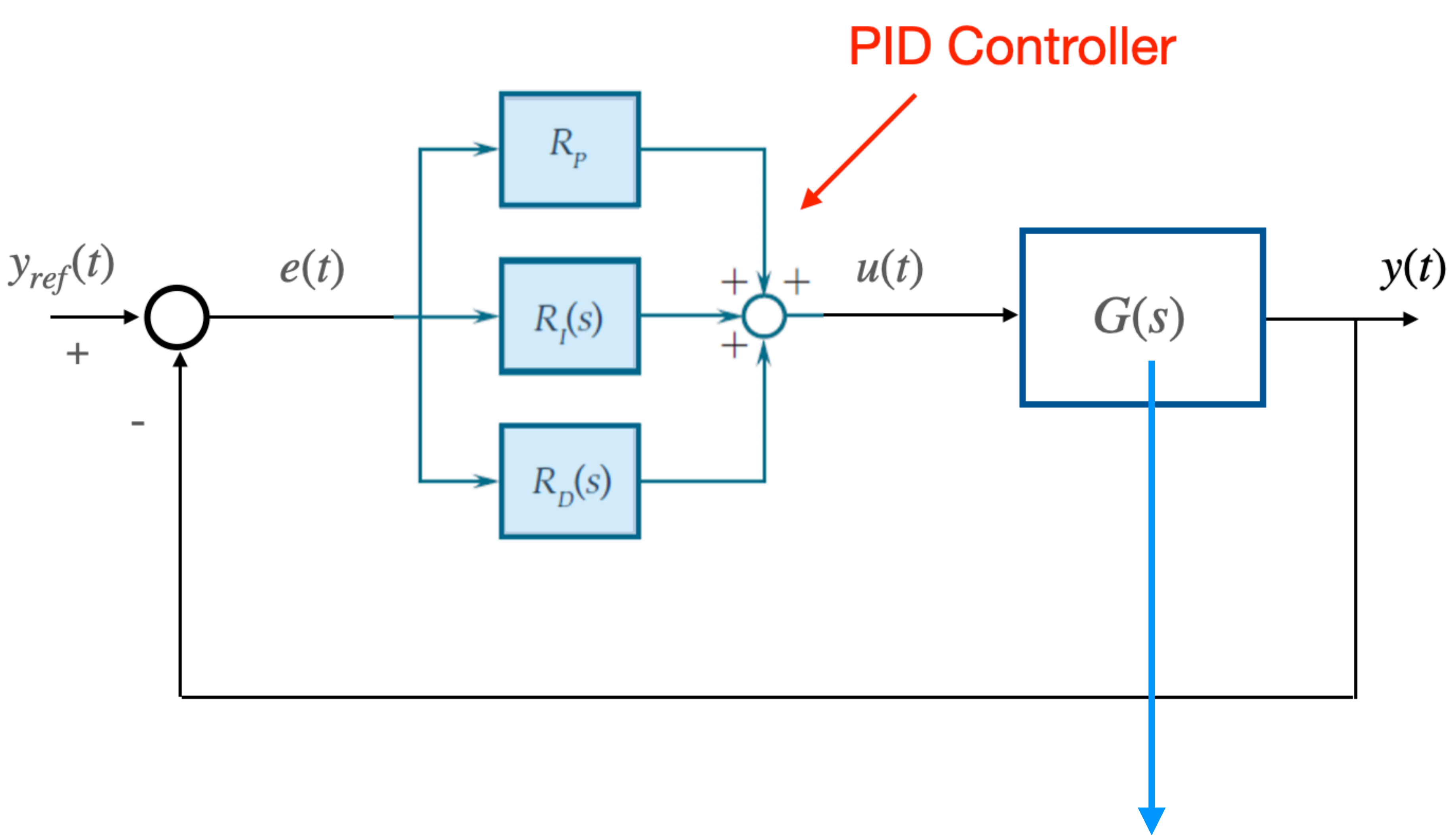
$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1 + Ts}$$

Open-loop Ziegler-Nichols Method

| | $K_P$ | $T_I$ | $T_D$ |
|---|---|---|---|
| P | $\frac{T}{\mu\tau}$ | | |
| PI | $\frac{0.9T}{\mu\tau}$ | $3\tau$ | |
| PID | $\frac{1.2T}{\mu\tau}$ | $2\tau$ | $0.5\tau$ |

$$R_{PID_{id}}(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

# Practical Implementation of PID Controllers: Tuning Rules




Approximate Model of the Process:

$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1 + Ts}$$

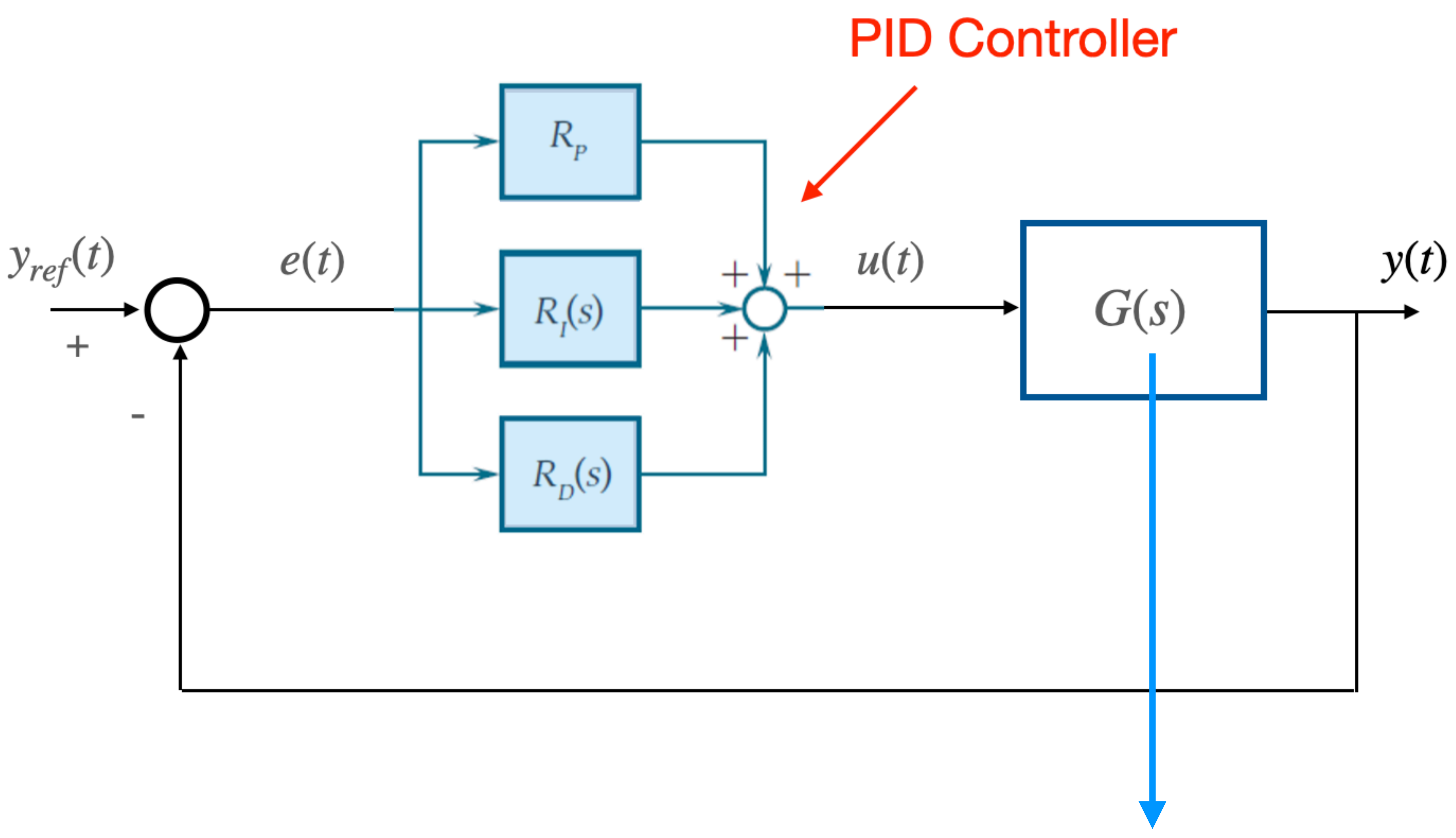
## Open-loop Ziegler-Nichols Method

| | $K_P$ | $T_I$ | $T_D$ |
|---|---|---|---|
| P | $\frac{T}{\mu\tau}$ | | |
| PI | $\frac{0.9T}{\mu\tau}$ | $3\tau$ | |
| PID | $\frac{1.2T}{\mu\tau}$ | $2\tau$ | $0.5\tau$ |

$$R_{PID_{id}}(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

Note that $T_I = 4T_D$ $\rightarrow$ the PID zeros coincide in $s = -\frac{1}{2T_D} = -\frac{1}{\tau}$

# Practical Implementation of PID Controllers: Tuning Rules




Approximate Model of the Process:

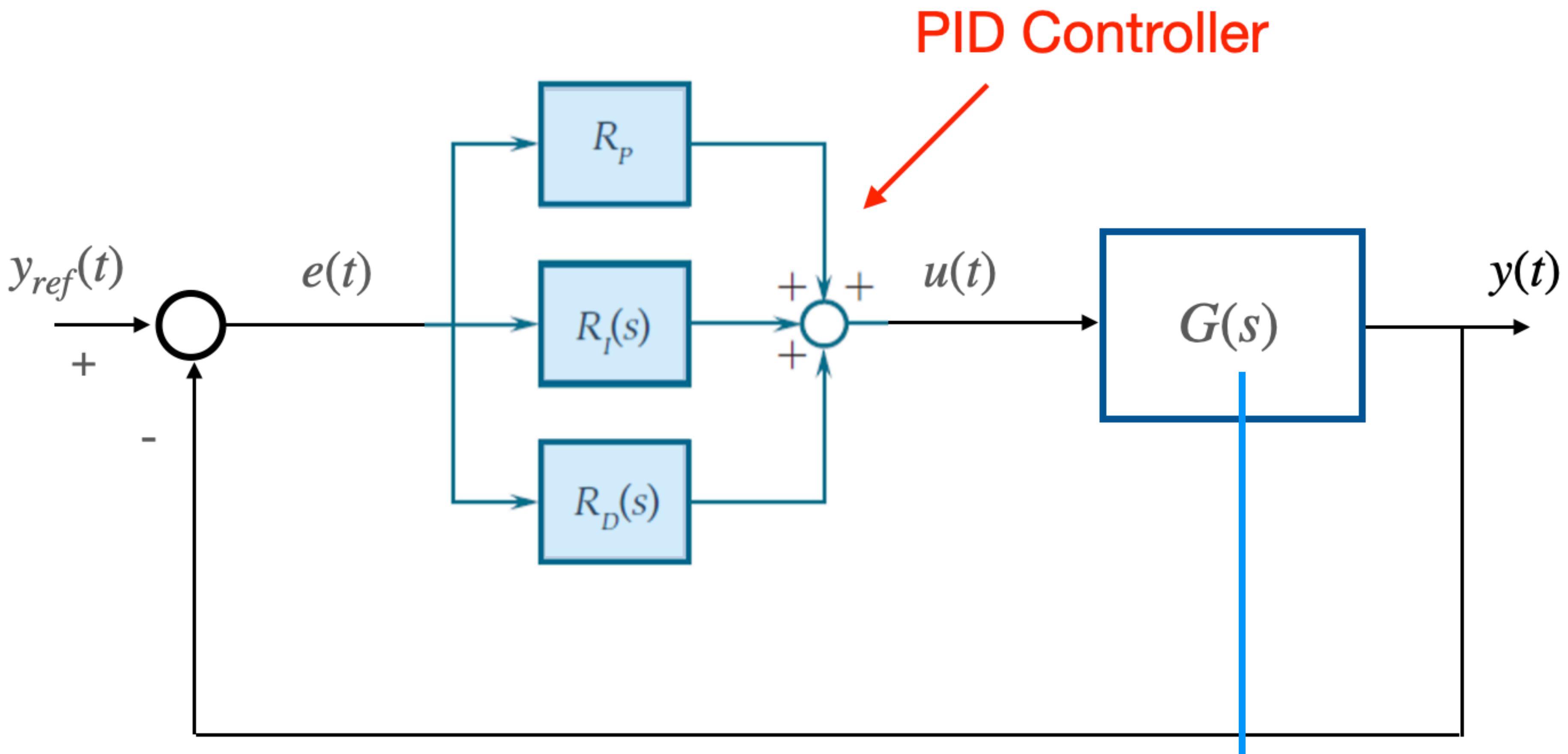
$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1 + Ts}$$

## Open-loop Cohen-Coon Method

| | $K_P$ | $T_I$ | $T_D$ |
|---|---|---|---|
| P | $\frac{3T+\tau}{3\mu\tau}$ | | |
| PI | $\frac{10.8T+\tau}{12\mu\tau}$ | $\tau\frac{30T+3\tau}{9T+20\tau}$ | |
| PID | $\frac{16T+3\tau}{12\mu\tau}$ | $\tau\frac{32T+6\tau}{12\tau}$ | $\frac{4T\tau}{11T+2\tau}$ |

$$R_{PID_{id}}(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

# Practical Implementation of PID Controllers: Tuning Rules




Approximate Model of the Process:

$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1 + Ts}$$

## Open-loop Internal Model Control (IMC) Method

| | $K_P$ | $T_I$ | $T_D$ |
|---|---|---|---|
| PI | $\dfrac{T}{\mu(\tau + T_f)}$ | $T$ | |
| PID | $\dfrac{T + 0.5\tau}{\mu(T_f + 0.5\tau)}$ | $T + 0.5\tau$ | $\dfrac{0.5\tau T}{T + 0.5\tau}$ |

$$R_{PID_{id}}(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

additional parameter: if $T_f$ increases, then $\omega_{BW_{CL}}$ decreases and the phase and gain margins increase

# Practical Implementation of PID Controllers: Tuning Rules

## Comparison:

| | $\varphi_m$ | $k_m$ | $\omega_c$ |
|---|---|---|---|
| Cohen - Coon | 31° | 7.9 | 0.74 |
| IMC ($T_f = 0.4$) | 45° | 10.8 | 0.6 |
| IMC ($T_f = 0.8$) | 53° | 12.8 | 0.5 |
| IMC ($T_f = 1.2$) | 59° | 14.4 | 0.42 |

For system: $G(s) = \dfrac{1}{(1+s)^3}$ approximated as: $G_a(s) = \dfrac{e^{-0.8s}}{1+3.7s}$

## Open-loop Internal Model Control (IMC) Method

| | $K_P$ | $T_I$ | $T_D$ |
|---|---|---|---|
| PI | $\dfrac{T}{\mu(\tau + T_f)}$ | $T$ | |
| PID | $\dfrac{T+0.5\tau}{\mu(T_f+0.5\tau)}$ | $T+0.5\tau$ | $\dfrac{0.5\tau T}{T+0.5\tau}$ |

$$R_{PID_{id}}(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

additional parameter: if $T_f$ increases, then $\omega_{BW_{CL}}$ decreases and the phase and gain margins increase

# Practical Implementation of PID Controllers: Tuning Rules

## Comparison:

| | $\varphi_m$ | $k_m$ | $\omega_c$ |
|---|---|---|---|
| Cohen - Coon | 31° | 7.9 | 0.74 |
| IMC ($T_f = 0.4$) | 45° | 10.8 | 0.6 |
| IMC ($T_f = 0.8$) | 53° | 12.8 | 0.5 |
| IMC ($T_f = 1.2$) | 59° | 14.4 | 0.42 |

For system: $G(s) = \dfrac{1}{(1+s)^3}$ approximated as: $G_a(s) = \dfrac{e^{-0.8s}}{1+3.7s}$

## Open-loop Internal Model Control (IMC) Method

| | $K_P$ | $T_I$ | $T_D$ |
|---|---|---|---|
| PI | $\dfrac{T}{\mu(\tau + T_f)}$ | $T$ | |
| PID | $\dfrac{T+0.5\tau}{\mu(T_f+0.5\tau)}$ | $T+0.5\tau$ | $\dfrac{0.5\tau T}{T+0.5\tau}$ |

$$R_{PID_{id}}(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

additional parameter: if $T_f$ increases, then $\omega_{BW_{CL}}$ decreases and the phase and gain margins increase

Note that: The IMC method produces more moderate control systems w.r.t. those obtained via the Cohen-Coon Method

# Practical Implementation of PID Controllers: Tuning Rules

## Comparison:

| | $\varphi_m$ | $k_m$ | $\omega_c$ |
|---|---|---|---|
| Cohen - Coon | 31° | 7.9 | 0.74 |
| IMC ($T_f = 0.4$) | 45° | 10.8 | 0.6 |
| IMC ($T_f = 0.8$) | 53° | 12.8 | 0.5 |
| IMC ($T_f = 1.2$) | 59° | 14.4 | 0.42 |

For system: $G(s) = \dfrac{1}{(1+s)^3}$ approximated as: $G_a(s) = \dfrac{e^{-0.8s}}{1+3.7s}$

## Open-loop Internal Model Control (IMC) Method

| | $K_P$ | $T_I$ | $T_D$ |
|---|---|---|---|
| PI | $\dfrac{T}{\mu(\tau + T_f)}$ | $T$ | |
| PID | $\dfrac{T+0.5\tau}{\mu(T_f+0.5\tau)}$ | $T+0.5\tau$ | $\dfrac{0.5\tau T}{T+0.5\tau}$ |

$$R_{PID_{id}}(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$

additional parameter: if $T_f$ increases, then $\omega_{BW_{CL}}$ decreases and the phase and gain margins increase

Exercise:
Determine the Control Sensitivity Functions for a PI tuned using Cohen-Coon or the IMC in the table for the considered $G(s)$

Note that: The IMC method produces more moderate control systems w.r.t. those obtained via the Cohen-Coon Method

# Practical Implementation of PID Controllers: Example

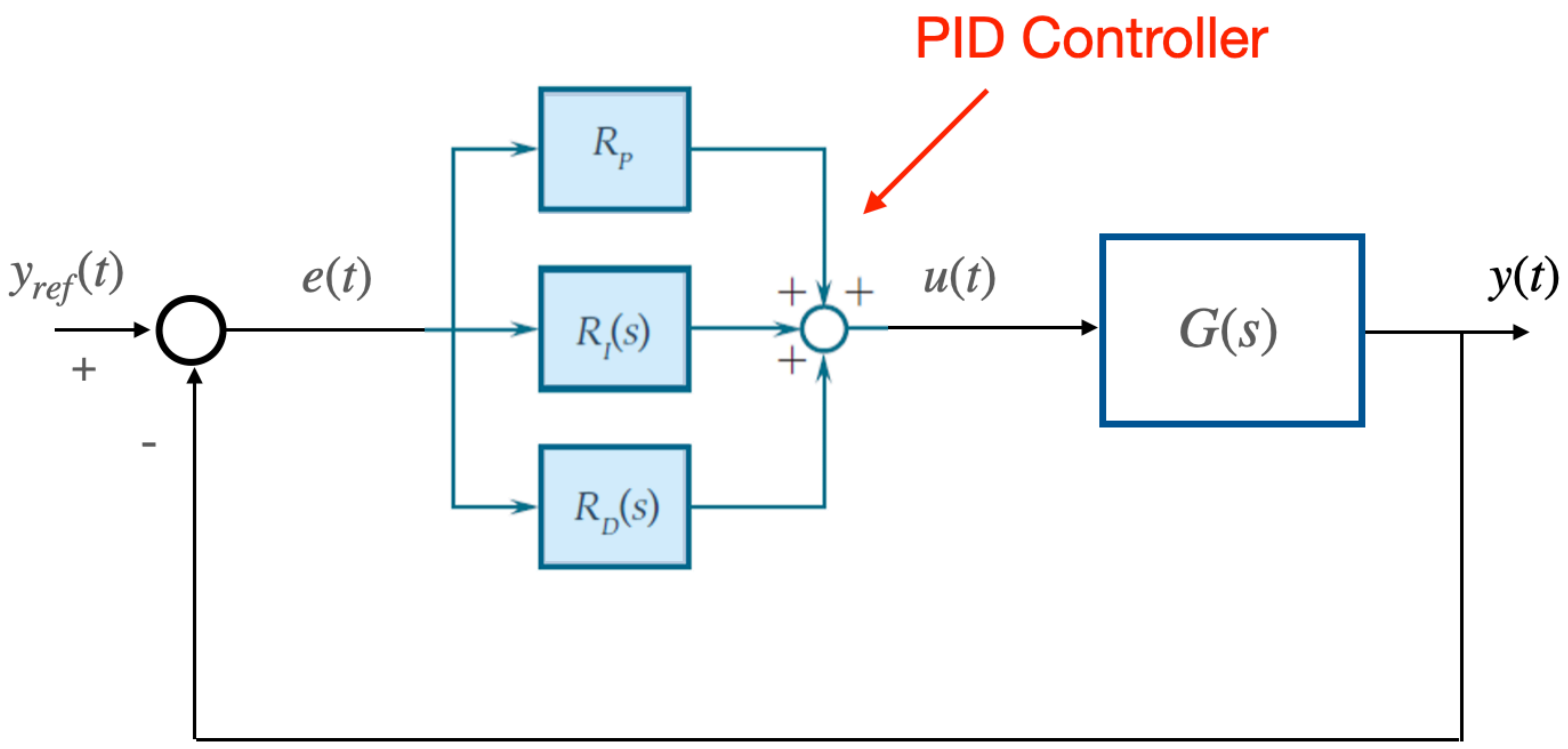
Process model:

$$G(s) = \frac{100}{(s+10)(s+30)(s+5)}$$

/MATLAB Drive/PID_example1.m

```
%% System definition
s = tf('s');
G = 100 / ((s + 10)*(s + 30)*(s+5));
```

Control scheme:



# Practical Implementation of PID Controllers: Example

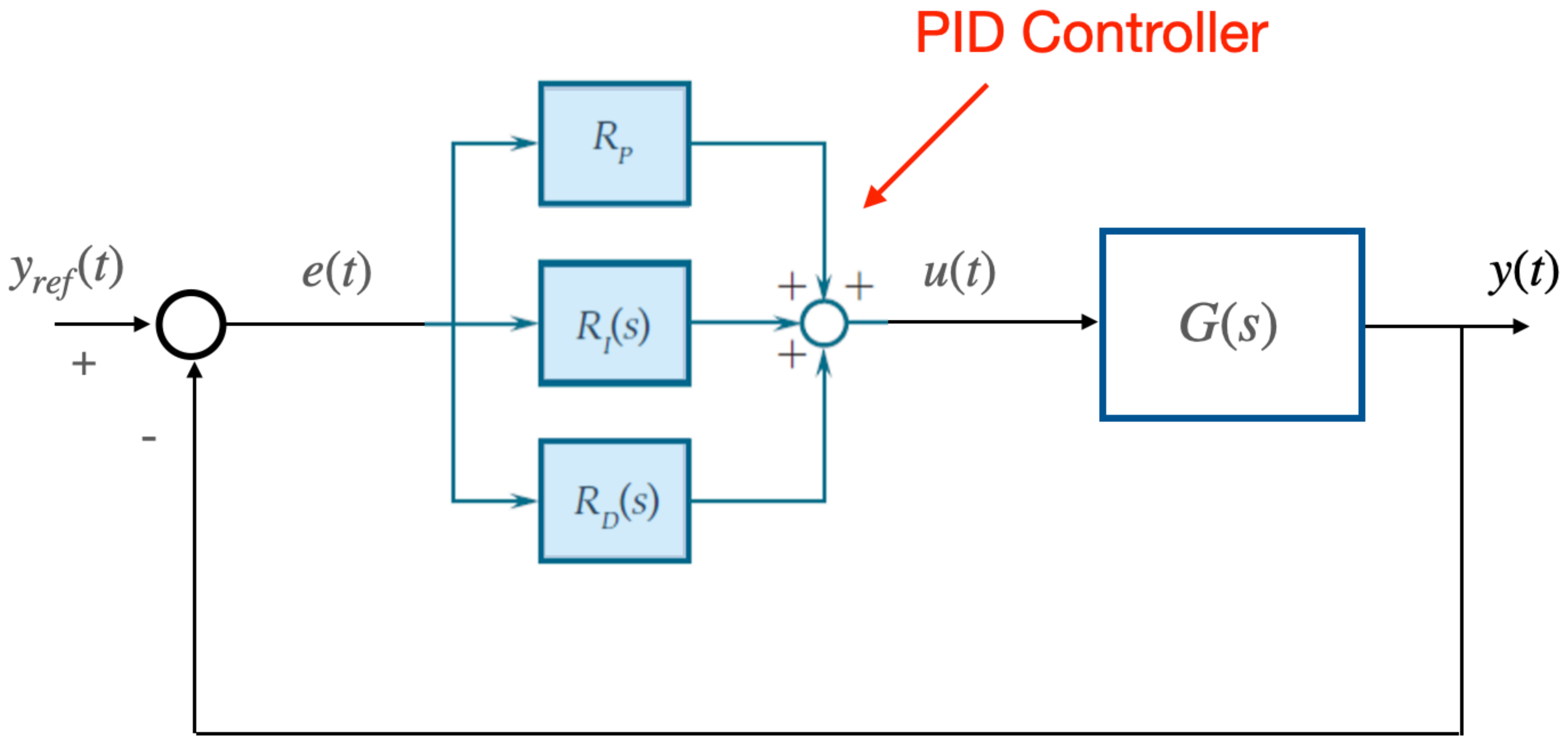
Process model:

$$G(s) = \frac{100}{(s+10)(s+30)(s+5)}$$

/MATLAB Drive/PID_example1.m

```
%% System definition
s = tf('s');
G = 100 / ((s + 10)*(s + 30)*(s+5));
```

Control scheme:




Objective: Compare the performance of the closed loop system with PID designed via CL Ziegler-Nichols rules and PID designed via OL Ziegler-Nichols rules

# Practical Implementation of PID Controllers: Example

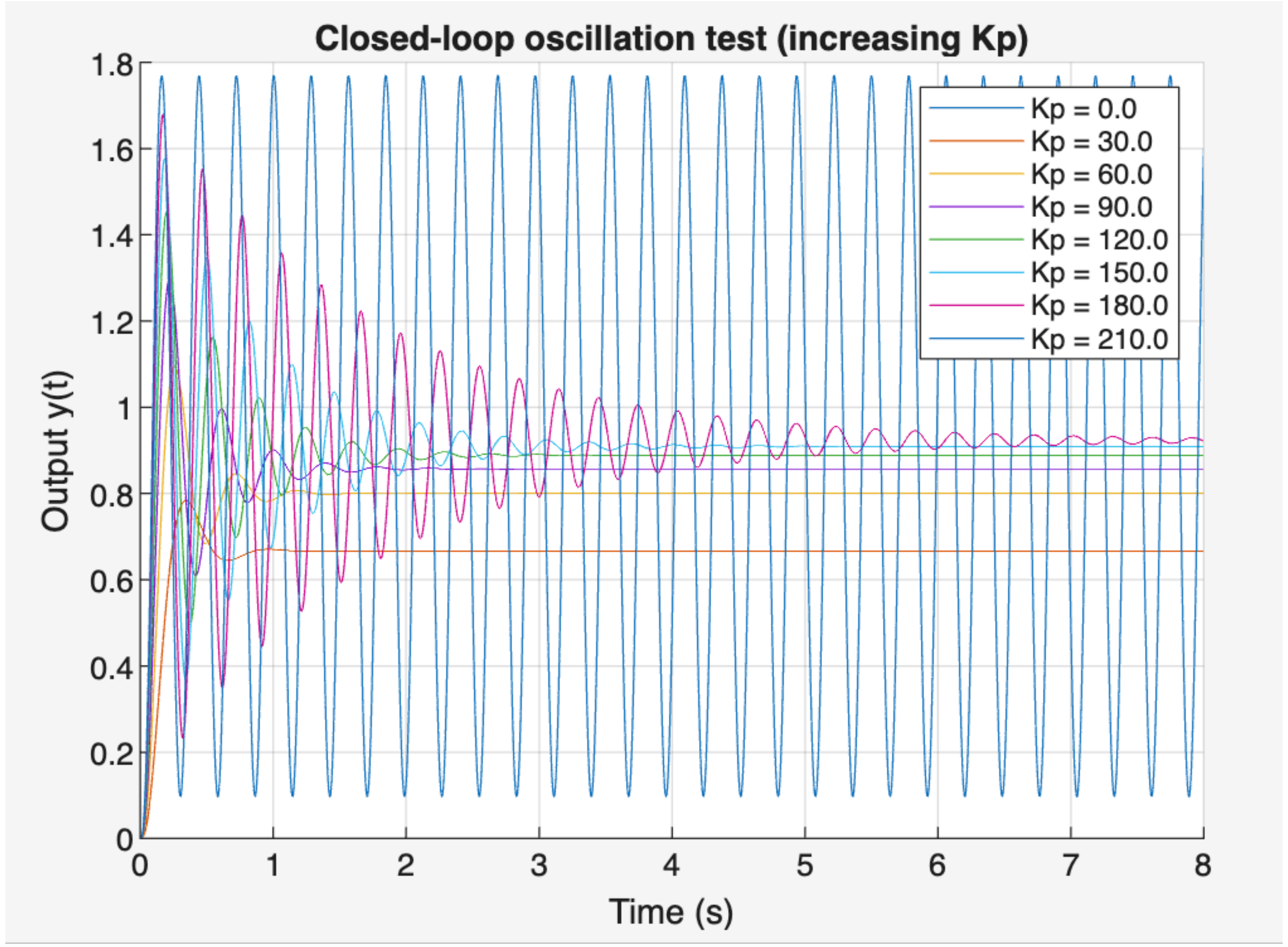
Process model:

$$G(s) = \frac{100}{(s+10)(s+30)(s+5)}$$

/MATLAB Drive/PID_example1.m

```
%% System definition
s = tf('s');
G = 100 / ((s + 10)*(s + 30)*(s+5));
```

Simulation for increasing $K_P$:



# Practical Implementation of PID Controllers: Example

Process model:

$$G(s) = \frac{100}{(s+10)(s+30)(s+5)}$$

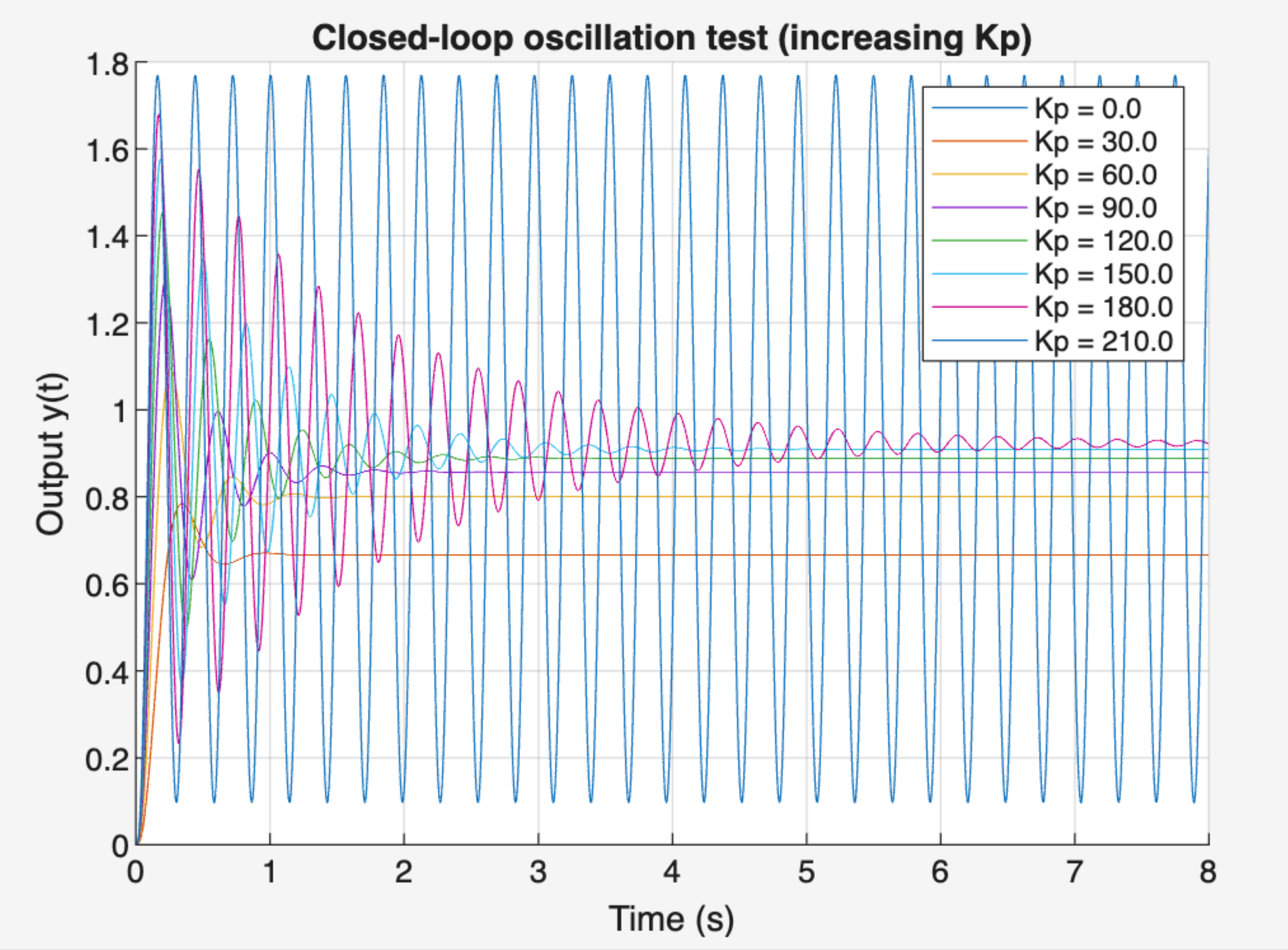
/MATLAB Drive/PID_example1.m

```
%% System definition
s = tf('s');
G = 100 / ((s + 10)*(s + 30)*(s+5));
```

PID calibration:

```
Estimated critical gain Kcr = 210.00
Estimated oscillation period Tcr = 0.281 s

Closed-loop ZN PID (with derivative filter, tau_f=1.000000e-02):
Kp = 126.000, Ti = 0.141, Td = 0.035
```

Simulation for increasing $K_P$:



# Practical Implementation of PID Controllers: Example

Process model:

$$G(s) = \frac{100}{(s+10)(s+30)(s+5)}$$

Approximate Model of the Process:

$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1+Ts}$$

$$\mu = 0.067 \quad \tau = 0.053 \quad T = 0.175$$

## Practical Implementation of PID Controllers: Example

### Process model:

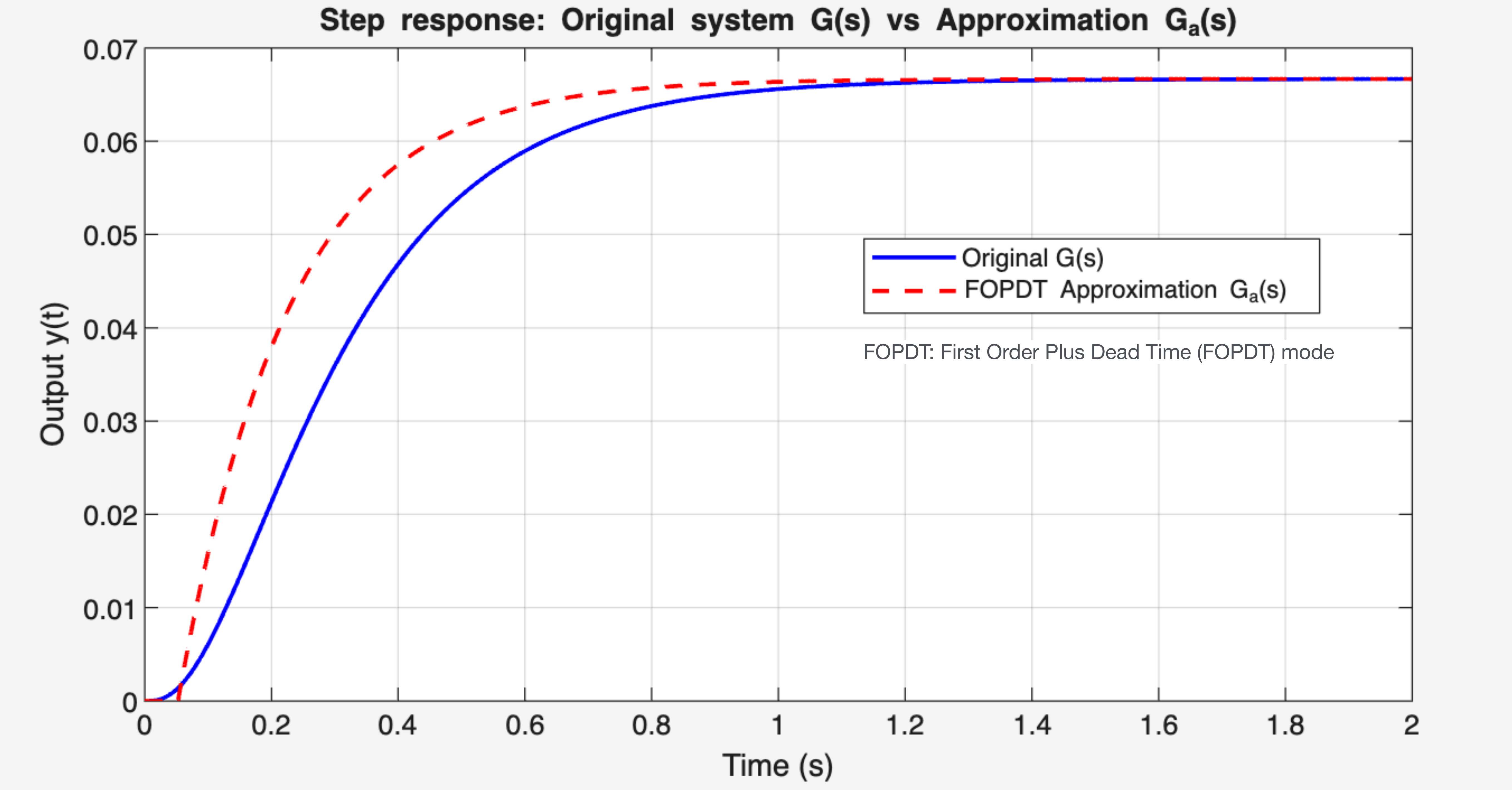
$$G(s) = \frac{100}{(s+10)(s+30)(s+5)}$$

### Approximate Model of the Process:

$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1 + Ts}$$

$$\mu = 0.067 \quad \tau = 0.053 \quad T = 0.175$$

```
%% === COMPARE STEP RESPONSE: REAL G(s) vs APPROXIMATED G_a(s) ===
t_sim = 0:0.001:2;
[yG, tG] = step(G, t_sim);
[yGa, tGa] = step(G_a, t_sim);

figure;
plot(tG, yG, 'b', 'LineWidth', 1.5); hold on;
plot(tGa, yGa, 'r--', 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Output y(t)');
title('Step response: Original system G(s) vs Approximation G_a(s)');
legend('Original G(s)', 'FOPDT Approximation G_a(s)');
grid on;
```



# Practical Implementation of PID Controllers: Example

Process model:

$$G(s) = \frac{100}{(s+10)(s+30)(s+5)}$$

Approximate Model of the Process:

$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1 + Ts}$$

$$\mu = 0.067 \quad \tau = 0.053 \quad T = 0.175$$

Open-loop Ziegler-Nichols Method

| | $K_P$ | $T_I$ | $T_D$ |
|---|---|---|---|
| P | $\frac{T}{\mu\tau}$ | | |
| PI | $\frac{0.9T}{\mu\tau}$ | $3\tau$ | |
| PID | $\frac{1.2T}{\mu\tau}$ | $2\tau$ | $0.5\tau$ |

```
Open-loop ZN PID:
Kp = 60.000, Ti = 0.105, Td = 0.026
```

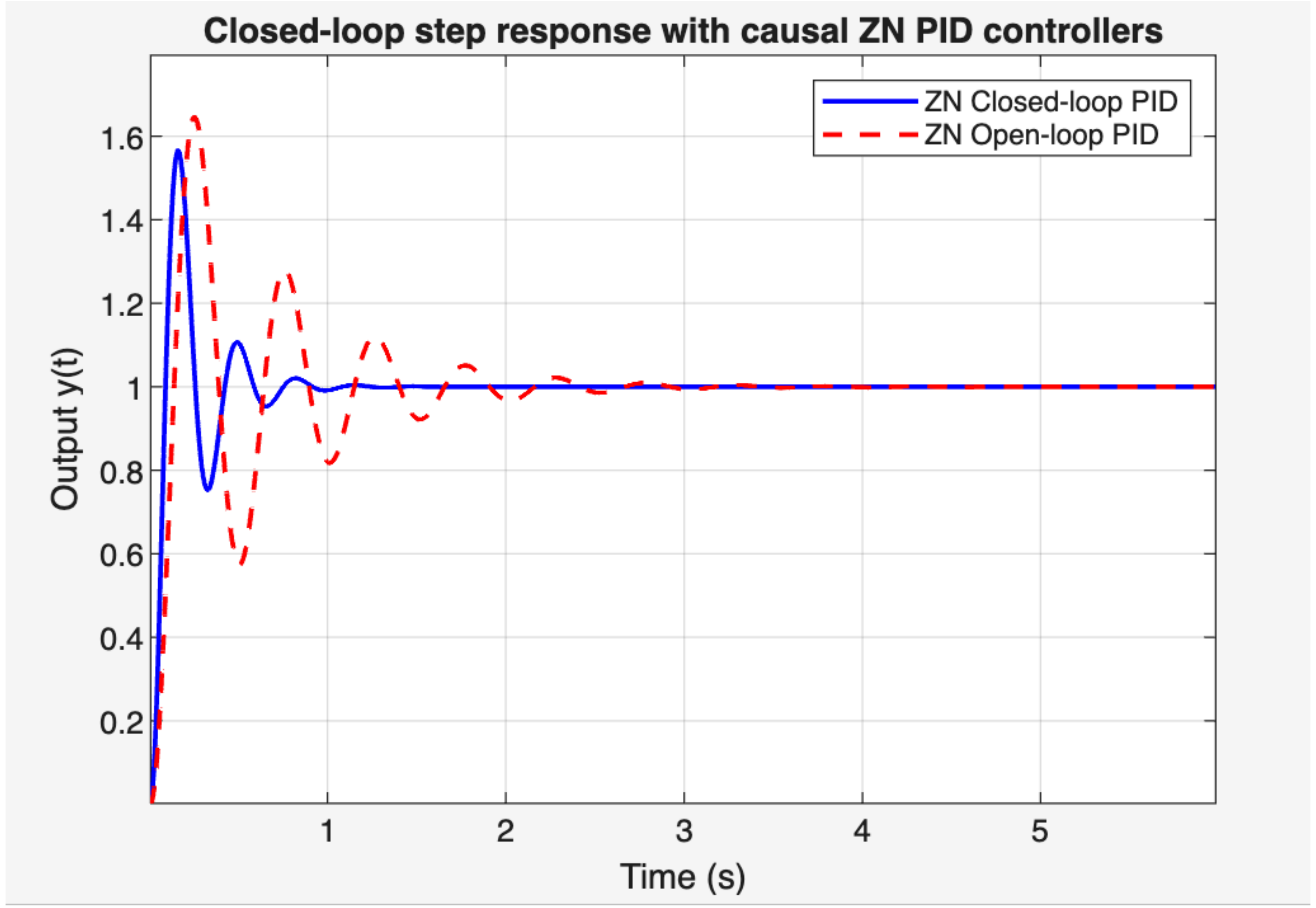
# Practical Implementation of PID Controllers: Example

Example:

/MATLAB Drive/PID_example1.m

```matlab
%% System definition
s = tf('s');
G = 100 / ((s + 10)*(s + 30)*(s+5));

```

```
Estimated critical gain Kcr = 210.00
Estimated oscillation period Tcr = 0.281 s

Closed-loop ZN PID (with derivative filter, tau_f=1.000000e-02):
Kp = 126.000, Ti = 0.141, Td = 0.035

Approximate FOPDT model parameters:
mu = 0.067, tau = 0.053, T = 0.175

Open-loop ZN PID (with derivative filter, tau_f=1.000000e-02):
Kp = 60.000, Ti = 0.105, Td = 0.026
```



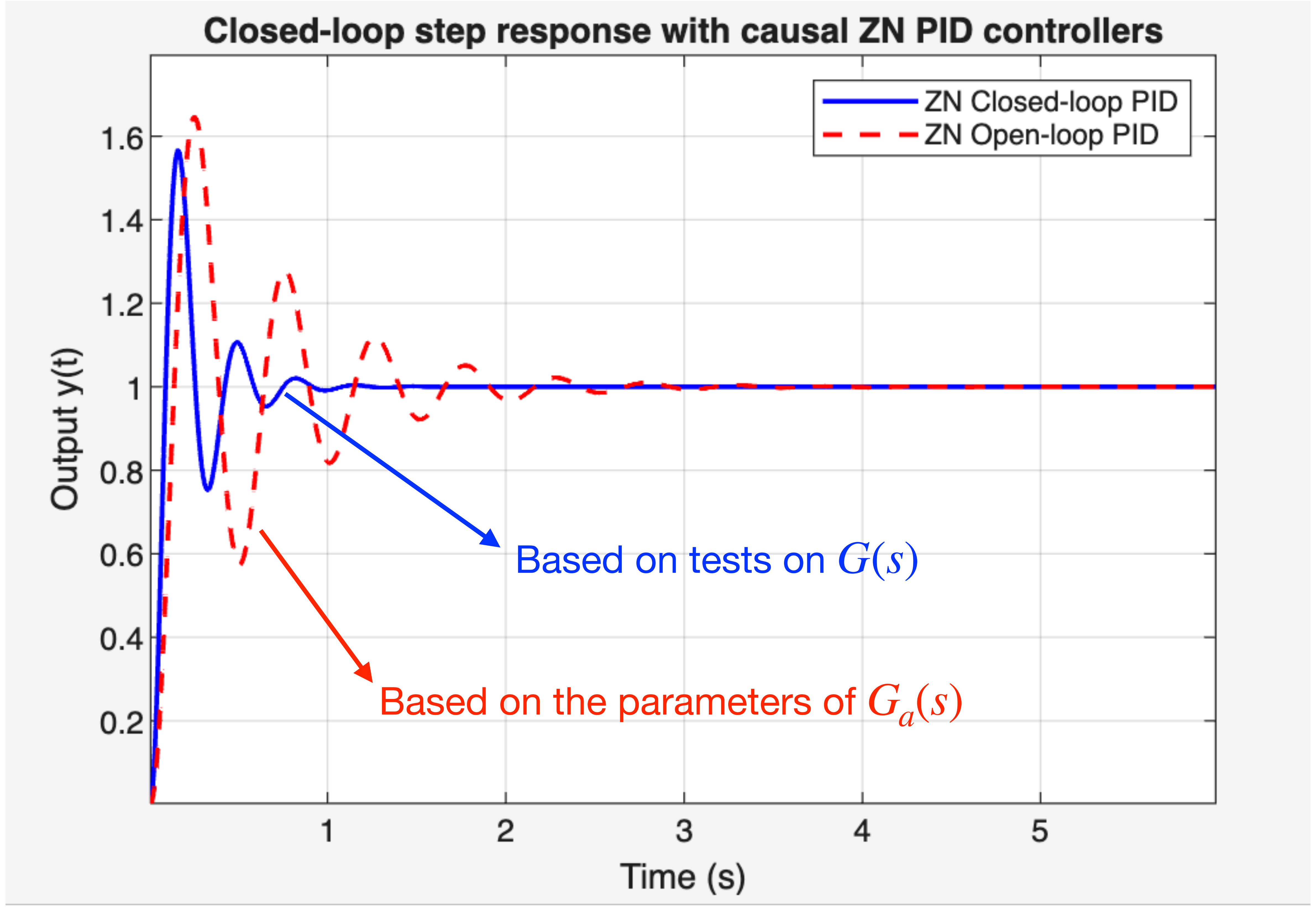
# Practical Implementation of PID Controllers: Example

Example:

/MATLAB Drive/PID_example1.m

```
%% System definition
s = tf('s');
G = 100 / ((s + 10)*(s + 30)*(s+5));
```

```
Estimated critical gain Kcr = 210.00
Estimated oscillation period Tcr = 0.281 s

Closed-loop ZN PID (with derivative filter, tau_f=1.000000e-02):
Kp = 126.000, Ti = 0.141, Td = 0.035

Approximate FOPDT model parameters:
mu = 0.067, tau = 0.053, T = 0.175

Open-loop ZN PID (with derivative filter, tau_f=1.000000e-02):
Kp = 60.000, Ti = 0.105, Td = 0.026
```



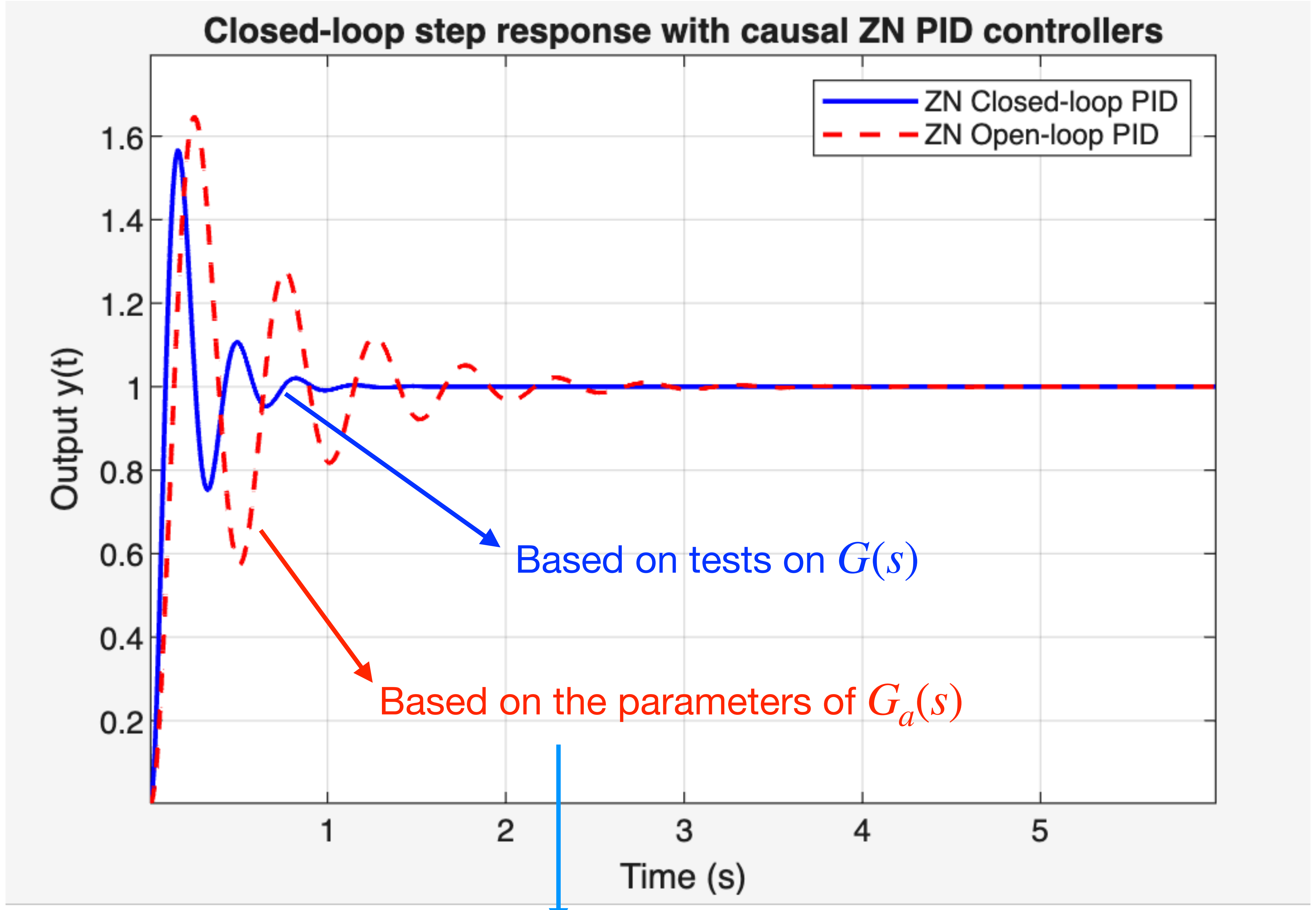
# Practical Implementation of PID Controllers: Example

Example:

/MATLAB Drive/PID_example1.m

```matlab
%% System definition
s = tf('s');
G = 100 / ((s + 10)*(s + 30)*(s+5));

```

```
Estimated critical gain Kcr = 210.00
Estimated oscillation period Tcr = 0.281 s

Closed-loop ZN PID (with derivative filter, tau_f=1.000000e-02):
Kp = 126.000, Ti = 0.141, Td = 0.035

Approximate FOPDT model parameters:
mu = 0.067, tau = 0.053, T = 0.175

Open-loop ZN PID (with derivative filter, tau_f=1.000000e-02):
Kp = 60.000, Ti = 0.105, Td = 0.026
```




in the example we are approximating a 3rd order system with a first order system with delay!

## Practical Implementation of PID Controllers: Example

Process model:

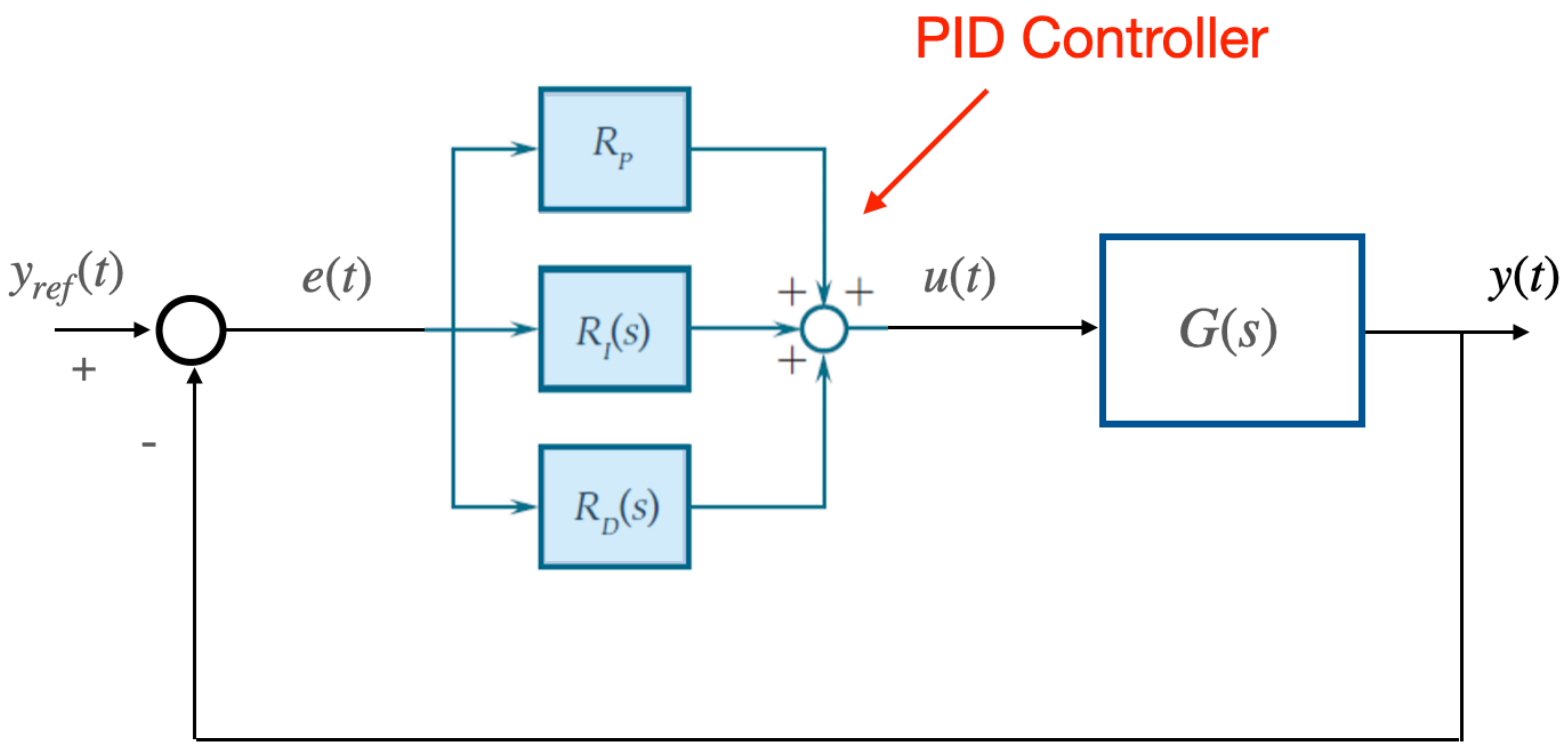
$$G(s) = \frac{100}{(s+10)(s+30)(s+5)}$$

Approximate Model of the Process:

$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1+Ts}$$

$$\mu = 0.067 \quad \tau = 0.053 \quad T = 0.175$$

Control scheme:



# Practical Implementation of PID Controllers: Example

Process model:

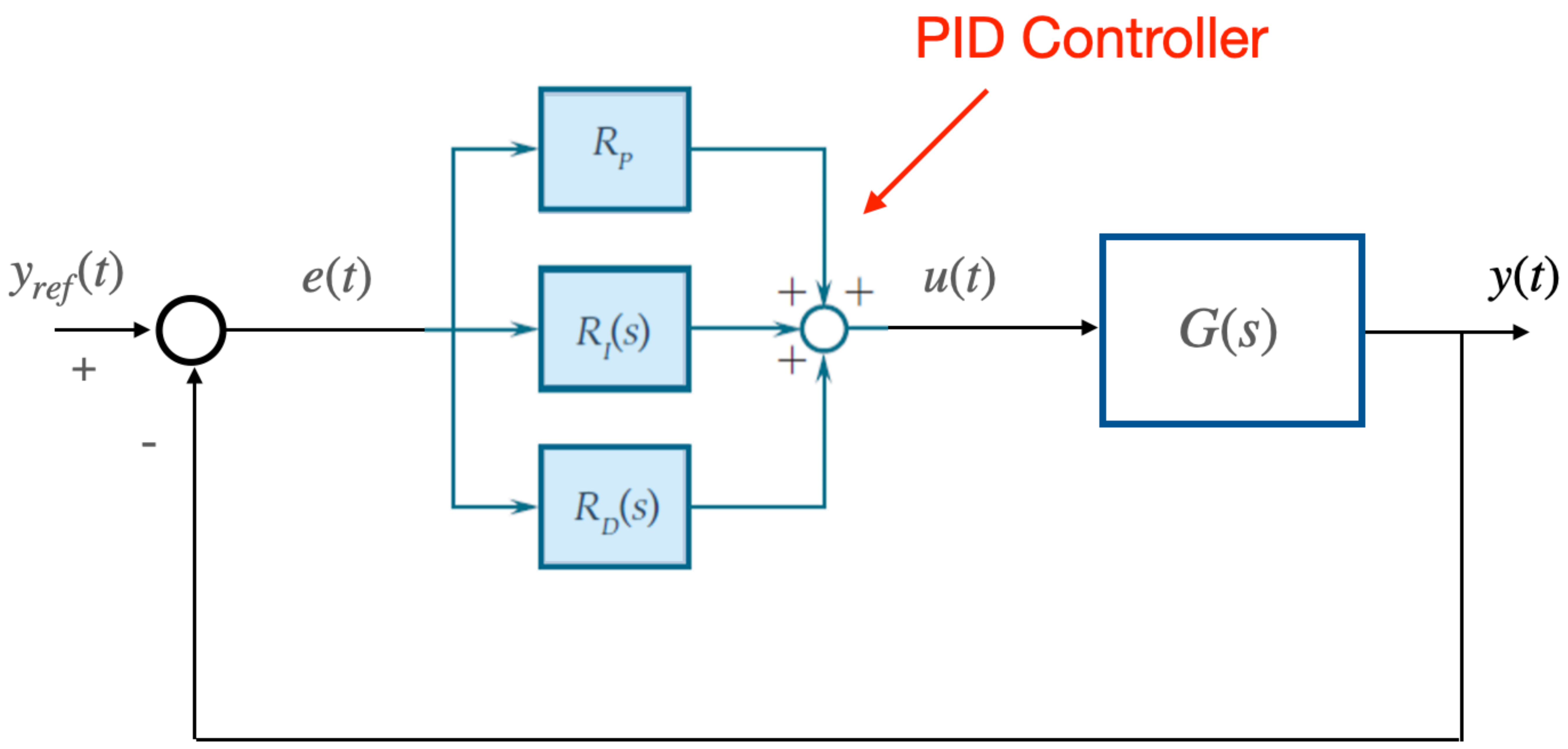
$$G(s) = \frac{100}{(s+10)(s+30)(s+5)}$$

Approximate Model of the Process:

$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1+Ts}$$

$$\mu = 0.067 \quad \tau = 0.053 \quad T = 0.175$$

Control scheme:




**Objective:** Compare the performance of the closed loop system with PID designed via Cohen-Coon rules and PID designed via the IMC method

# Practical Implementation of PID Controllers: Example

Process model:

$$G(s) = \frac{100}{(s+10)(s+30)(s+5)}$$

Approximate Model of the Process:

$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1 + Ts}$$

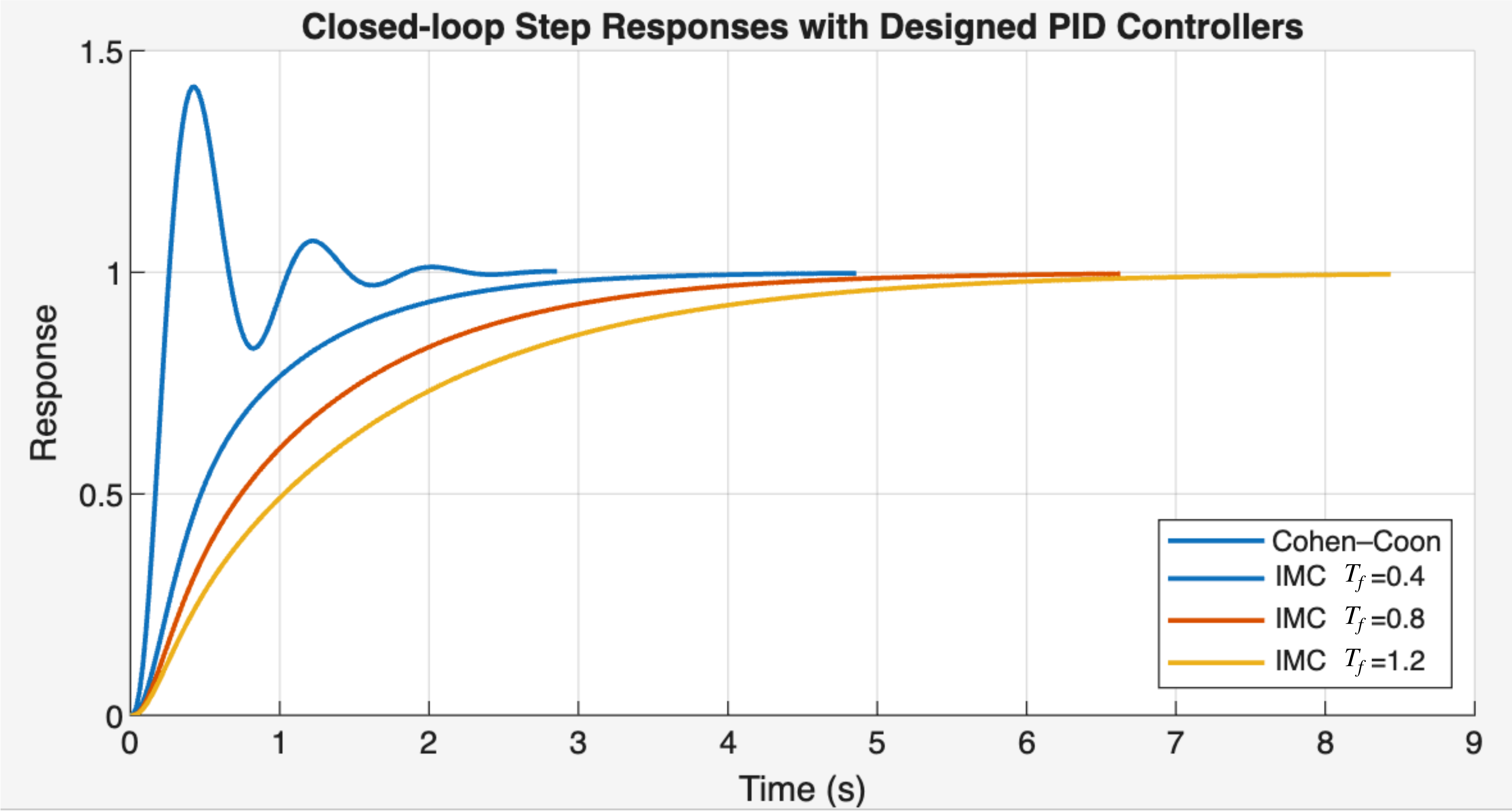
$$\mu = 0.067 \quad \tau = 0.053 \quad T = 0.175$$

Cohen–Coon PID (with derivative filter, tau_f=0.01):
Kp = 23.4620, Ki = 169.5603, Kd = 0.4321

IMC PID ( $T_f$ = 0.4, tau_f=0.01):
Kp = 6.8182, Ki = 20.4545, Kd = 0.6136

IMC PID ( $T_f$ = 0.8, tau_f=0.01):
Kp = 4.4118, Ki = 13.2353, Kd = 0.3971

IMC PID ( $T_f$ = 1.2, tau_f=0.01):
Kp = 3.2609, Ki = 9.7826, Kd = 0.2935



## Practical Implementation of PID Controllers: Example

Process model:

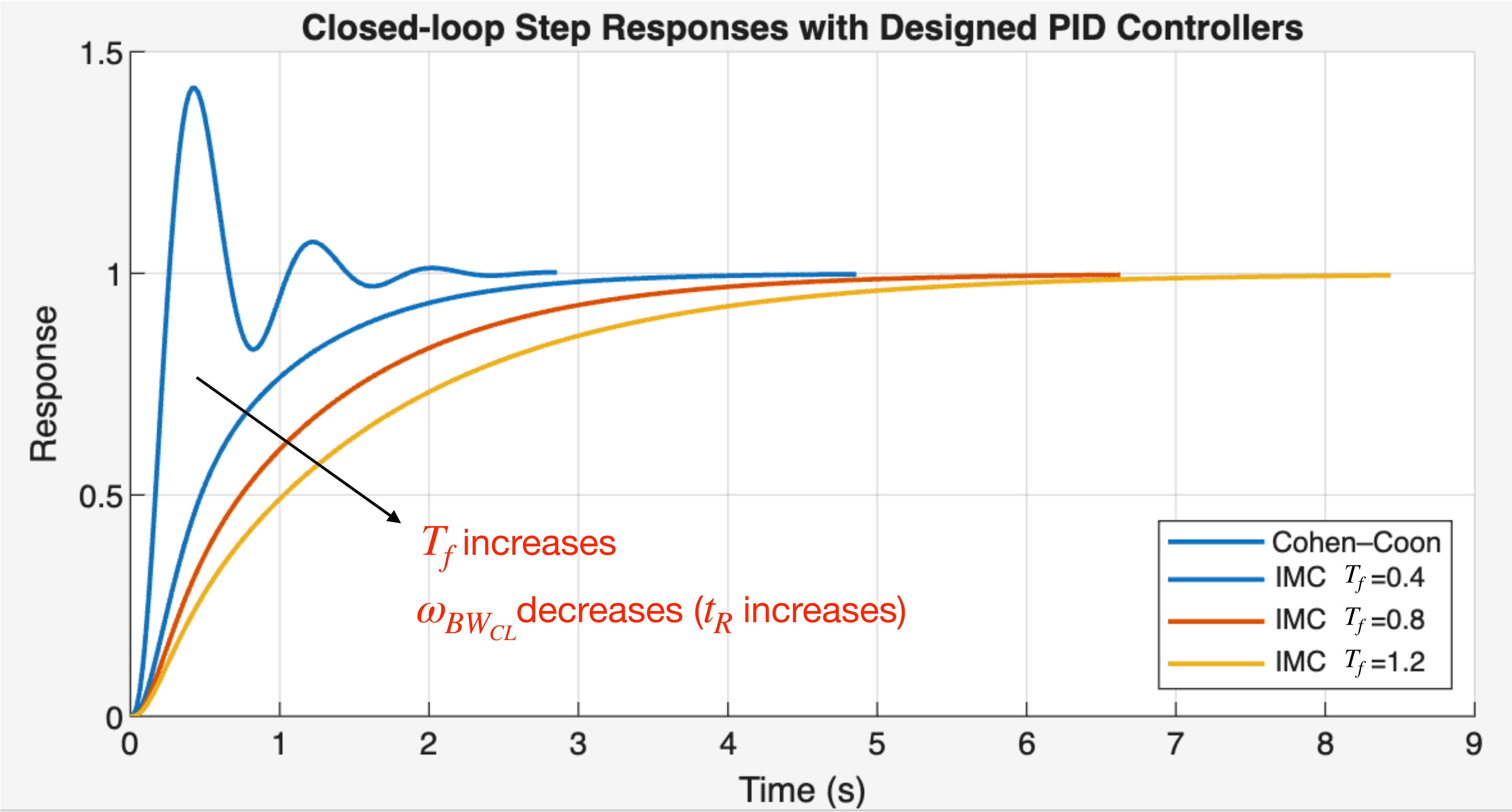
$$G(s) = \frac{100}{(s+10)(s+30)(s+5)}$$

Approximate Model of the Process:

$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1+Ts}$$

$$\mu = 0.067 \quad \tau = 0.053 \quad T = 0.175$$

```
Cohen–Coon PID (with derivative filter, tau_f=0.01):
Kp = 23.4620, Ki = 169.5603, Kd = 0.4321

IMC PID ( Tf = 0.4, tau_f=0.01):
Kp = 6.8182, Ki = 20.4545, Kd = 0.6136

IMC PID ( Tf = 0.8, tau_f=0.01):
Kp = 4.4118, Ki = 13.2353, Kd = 0.3971

IMC PID ( Tf = 1.2, tau_f=0.01):
Kp = 3.2609, Ki = 9.7826, Kd = 0.2935
```




additional parameter: if $T_f$ increases, then $\omega_{BW_{CL}}$ decreases and the phase and gain margins increase

# Practical Implementation of PID Controllers: Example

Process model:

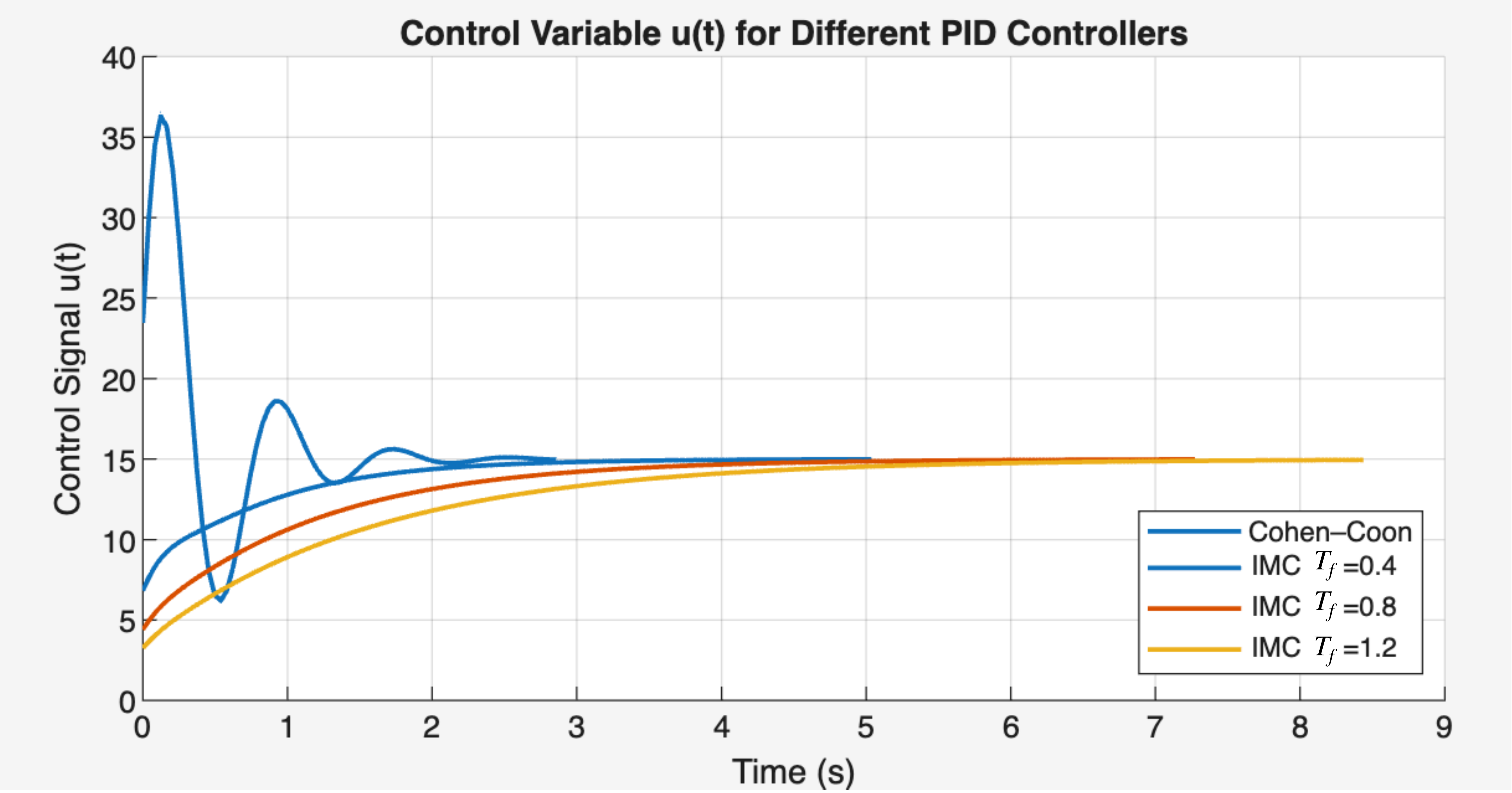
$$G(s) = \frac{100}{(s+10)(s+30)(s+5)}$$

Approximate Model of the Process:

$$G_a(s) = \frac{\mu\, e^{-\tau s}}{1+Ts}$$

$$\mu = 0.067 \quad \tau = 0.053 \quad T = 0.175$$

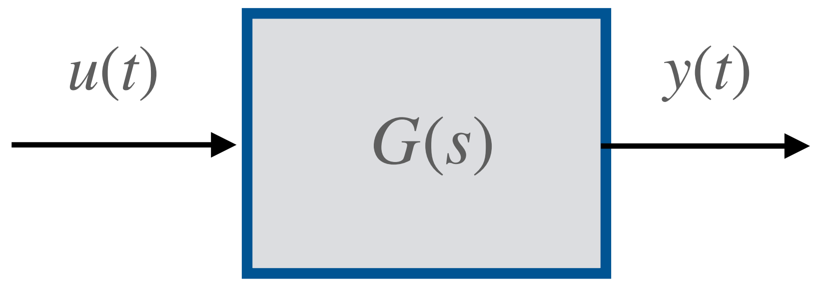
```
Cohen–Coon PID (with derivative filter, tau_f=0.01):
Kp = 23.4620, Ki = 169.5603, Kd = 0.4321

IMC PID ( Tf = 0.4, tau_f=0.01):
Kp = 6.8182, Ki = 20.4545, Kd = 0.6136

IMC PID ( Tf = 0.8, tau_f=0.01):
Kp = 4.4118, Ki = 13.2353, Kd = 0.3971

IMC PID ( Tf = 1.2, tau_f=0.01):
Kp = 3.2609, Ki = 9.7826, Kd = 0.2935
```




Note that: The IMC method produces more moderate control systems w.r.t. those obtained via the Cohen-Coon Method

# PID Controllers: Wind-up Effect

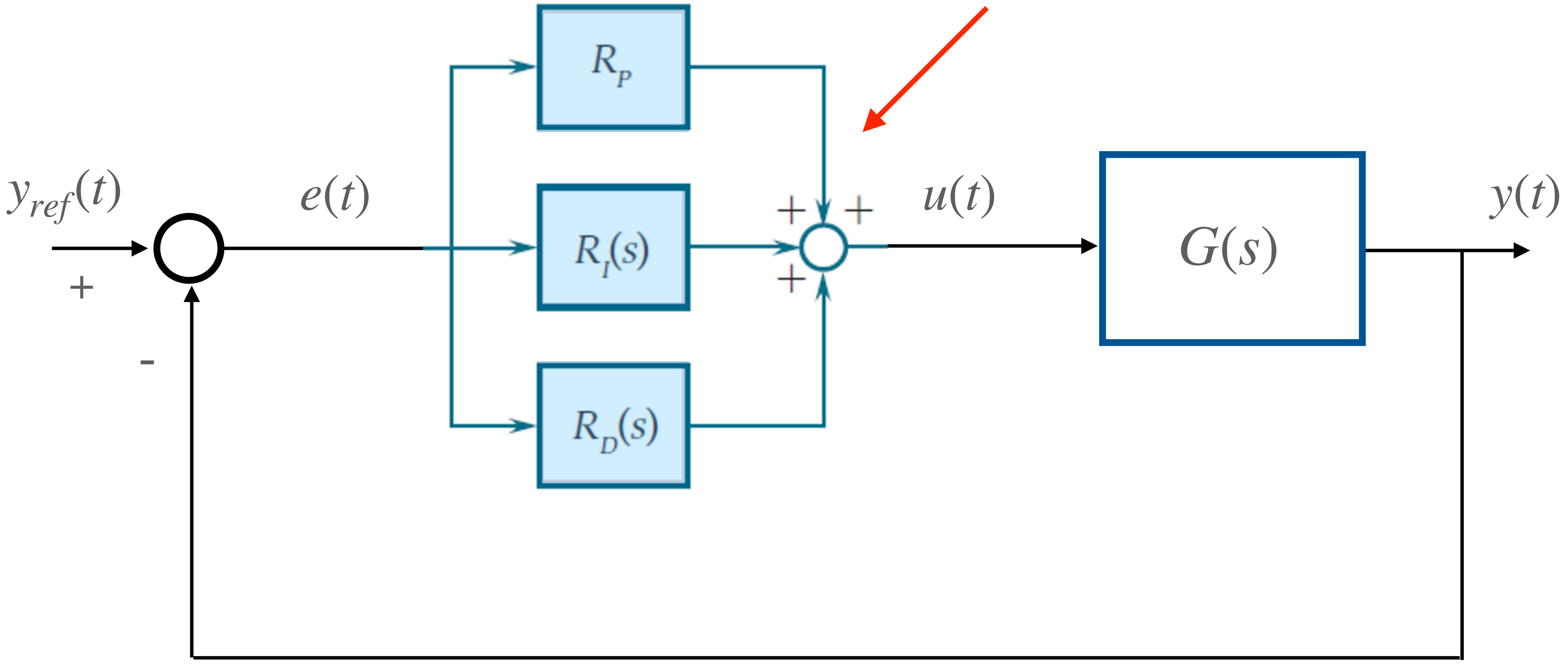
Process




$u(t) \in \mathbb{R}$

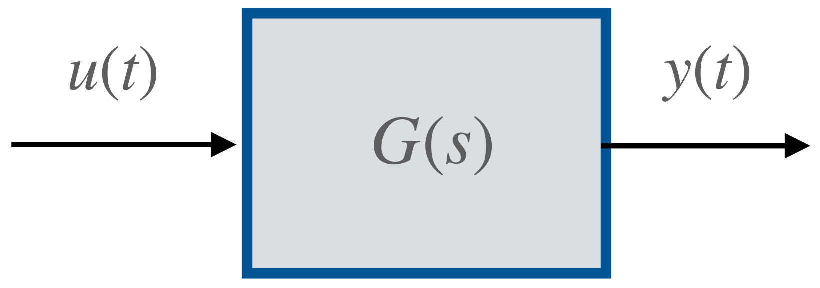
$y(t) \in \mathbb{R}$

PID Controller



# PID Controllers: Wind-up Effect

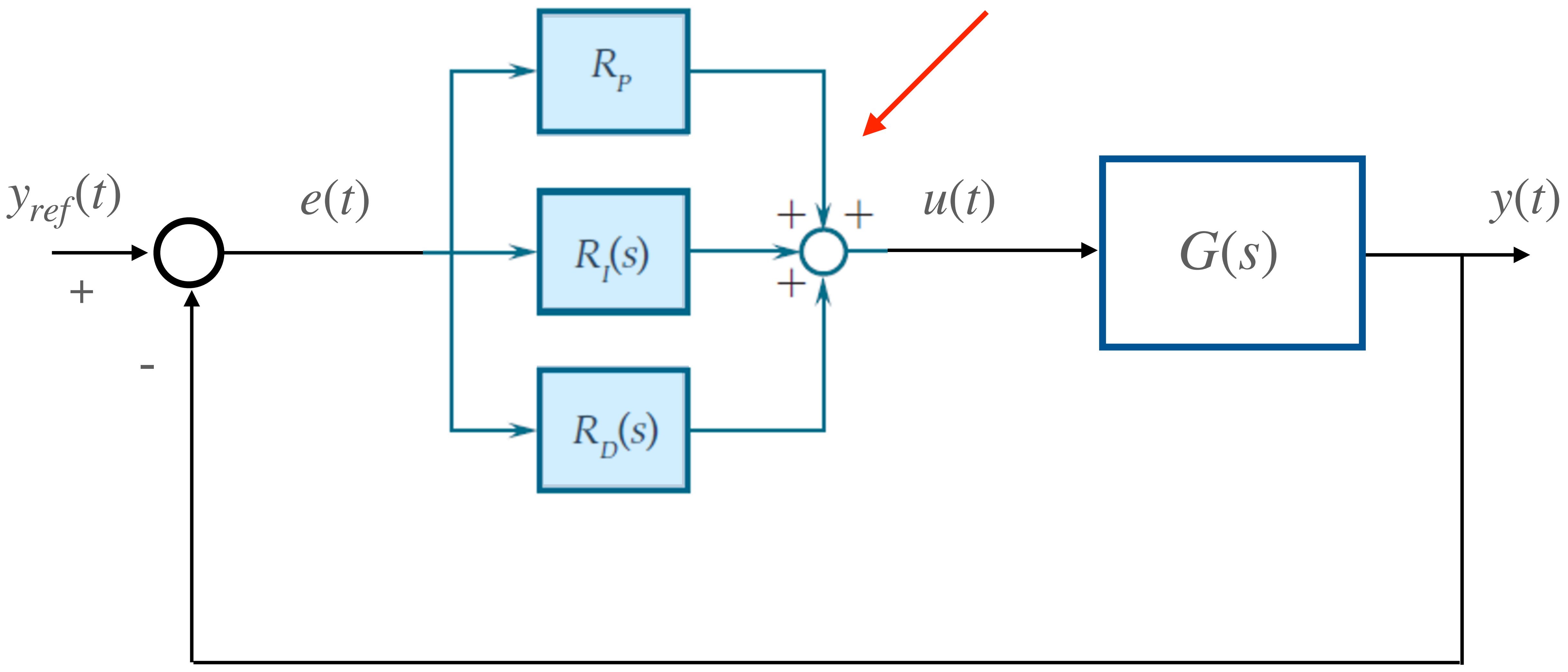
Process



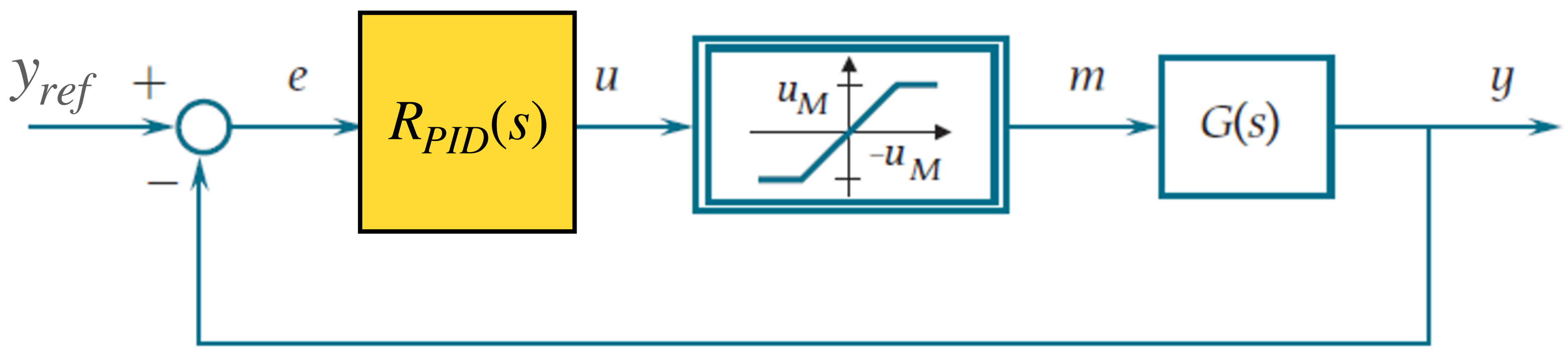

$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$

PID Controller

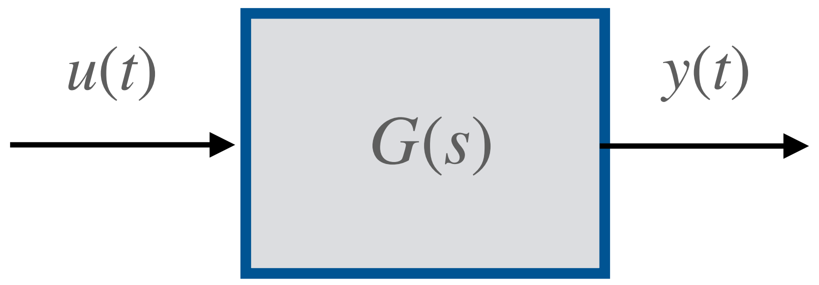



Realistic situation



# PID Controllers: Wind-up Effect

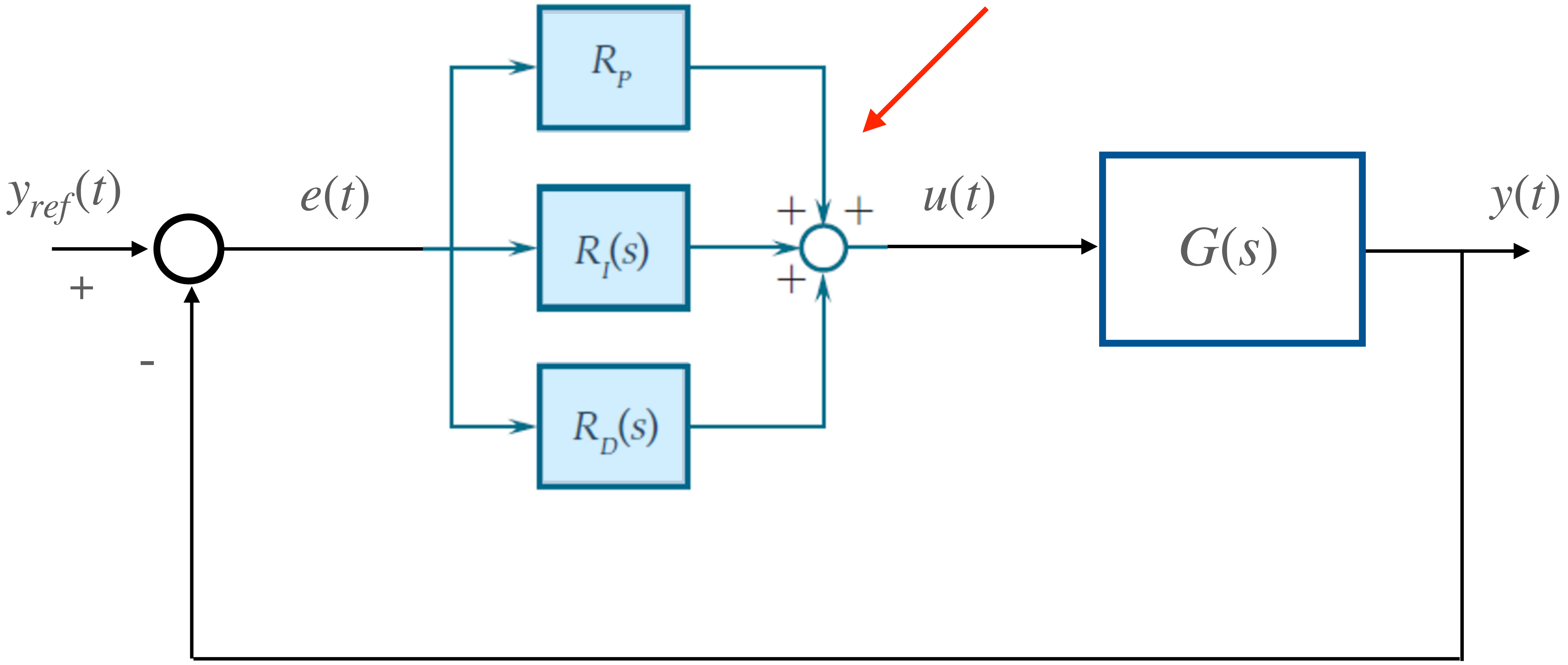
Process



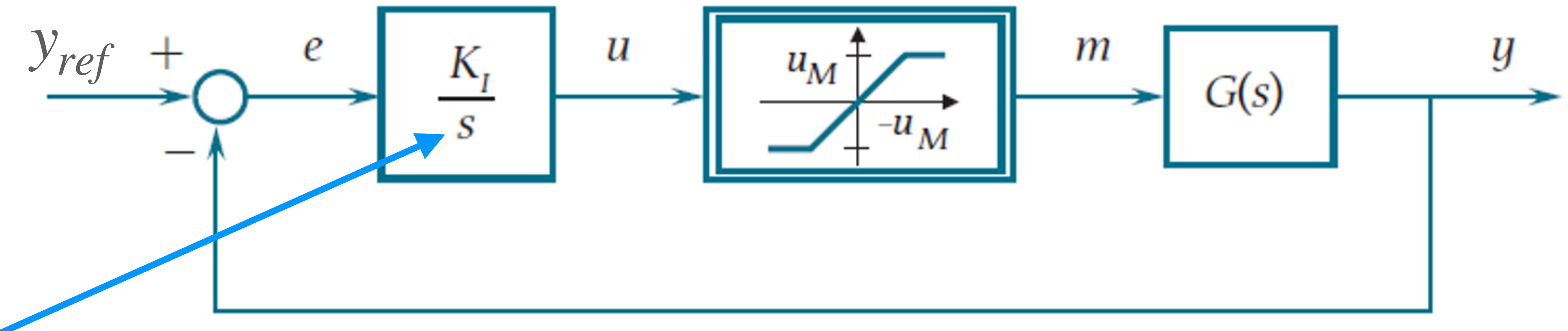

$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$

PID Controller



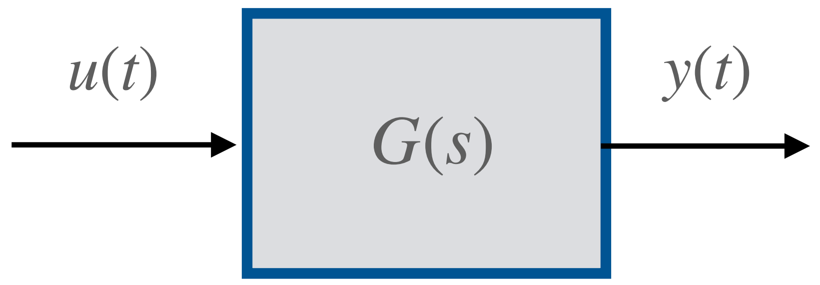

Realistic situation




Only the integral component is considered

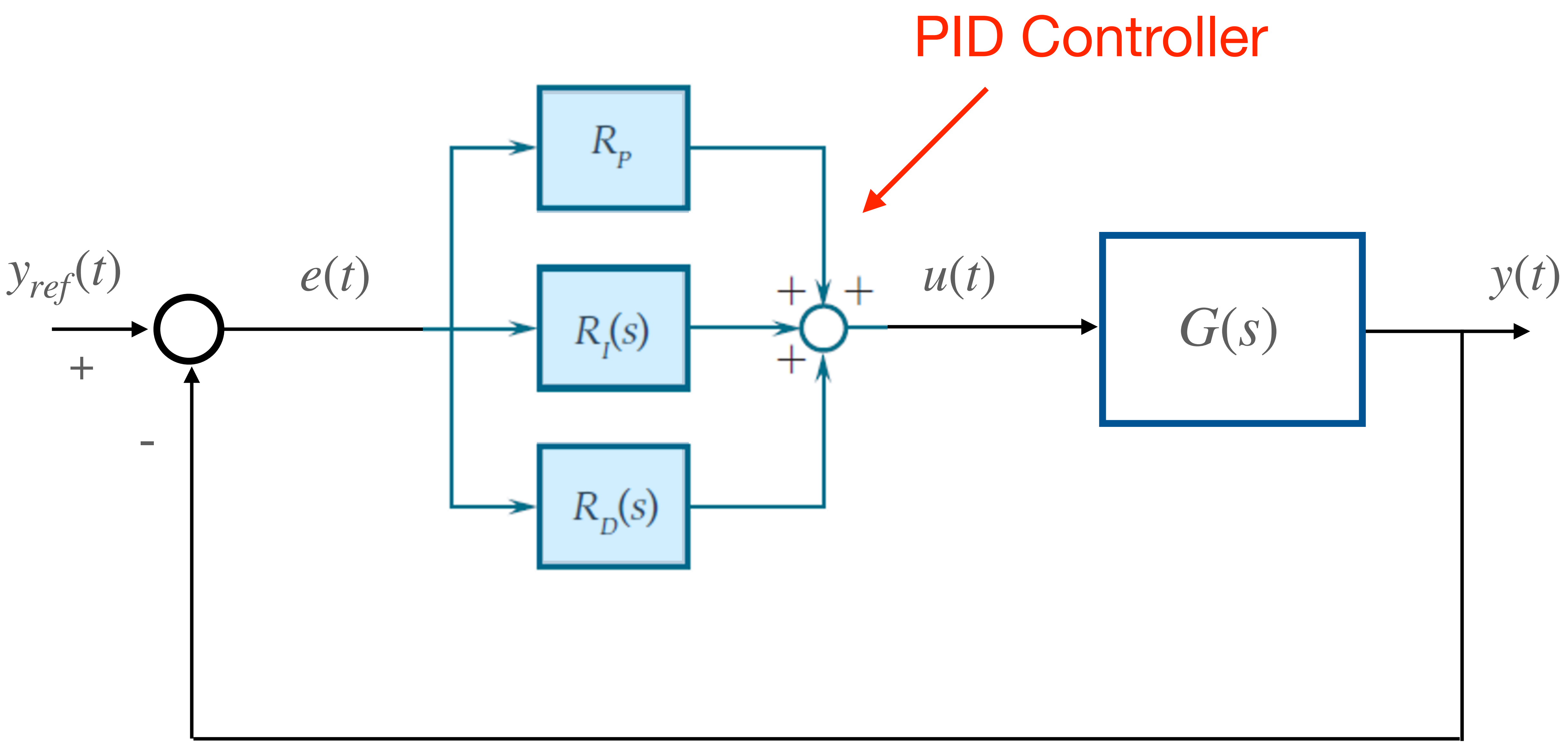
# PID Controllers: Wind-up Effect

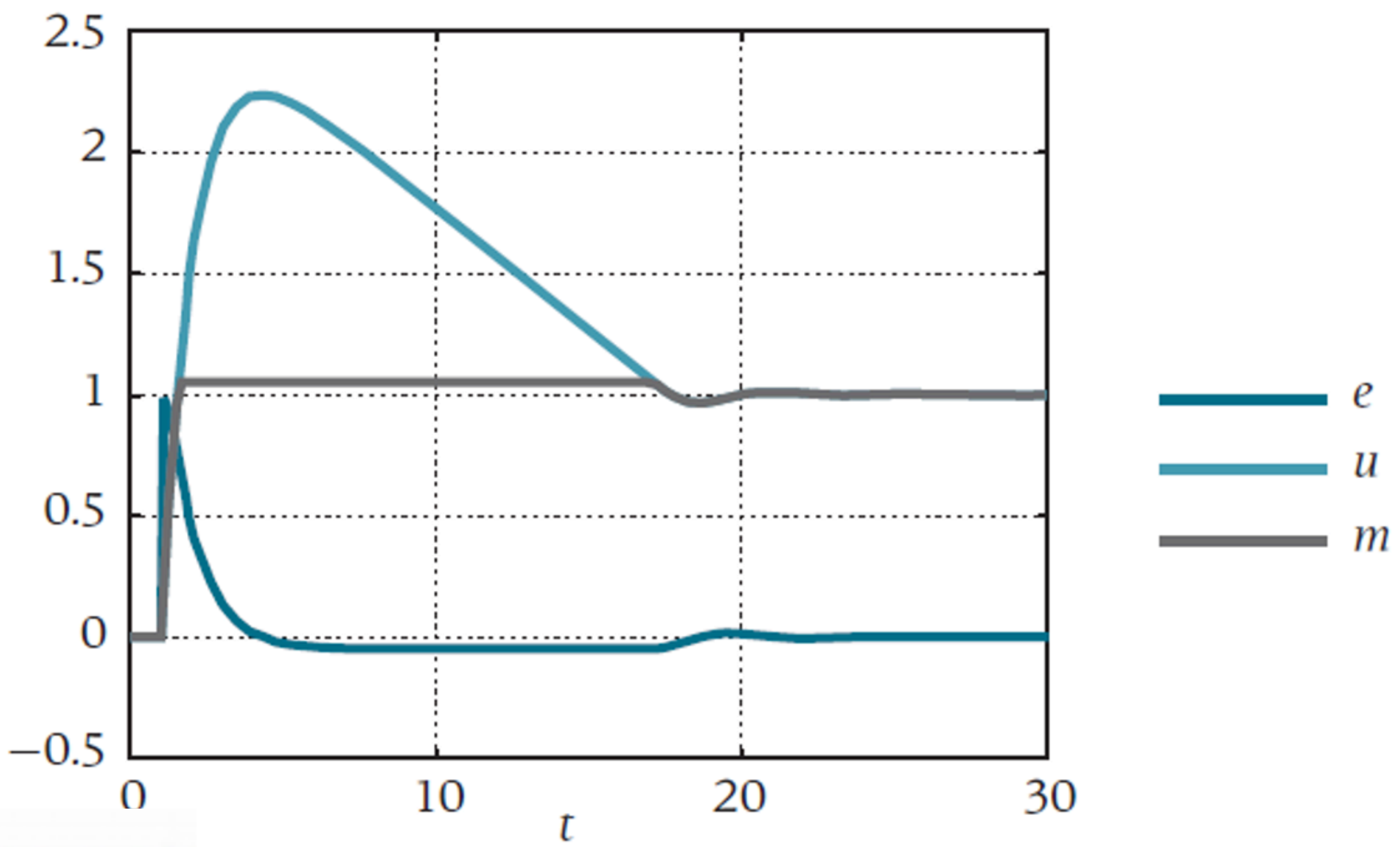
Process



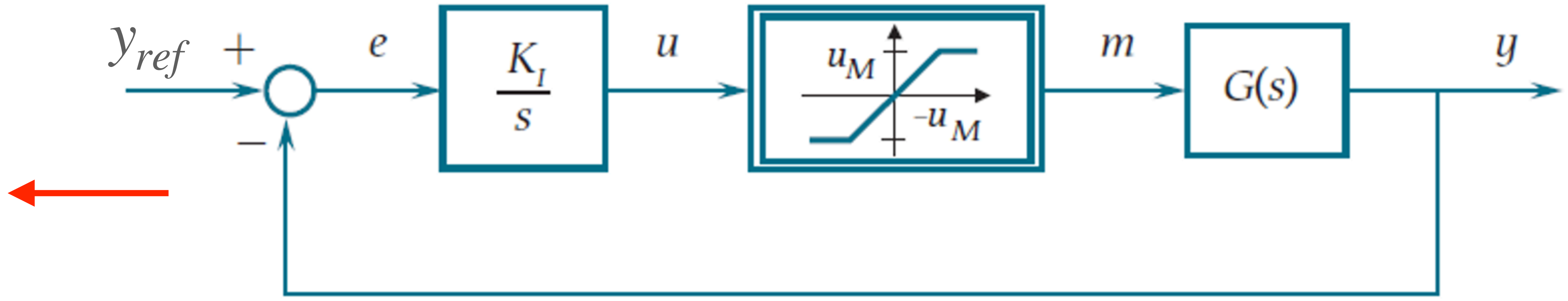

$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$



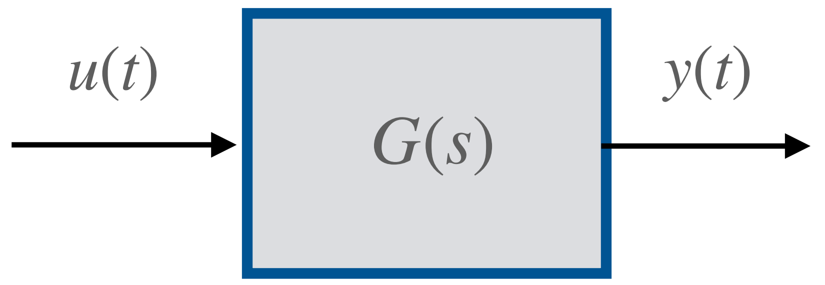




Realistic situation



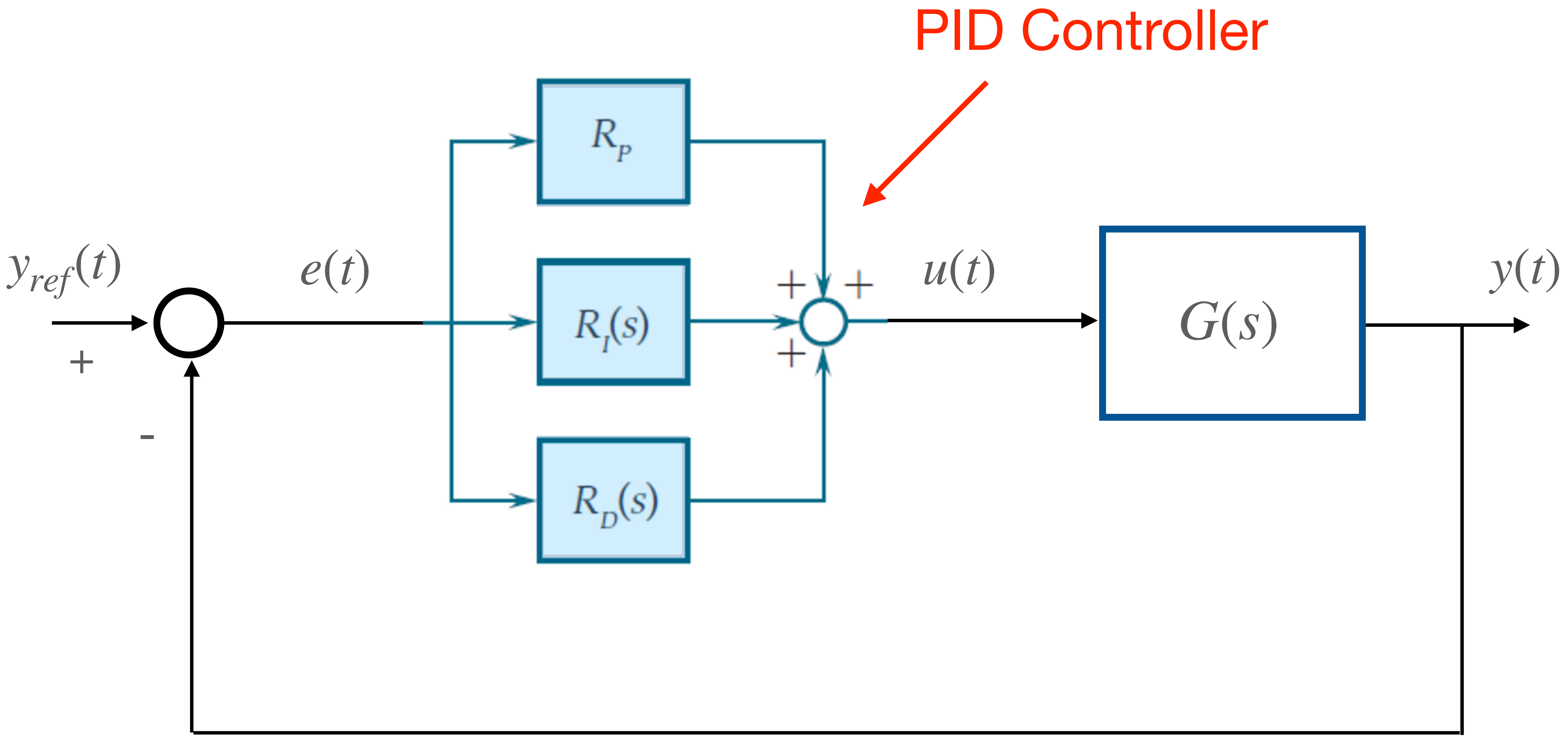
# PID Controllers: Wind-up Effect

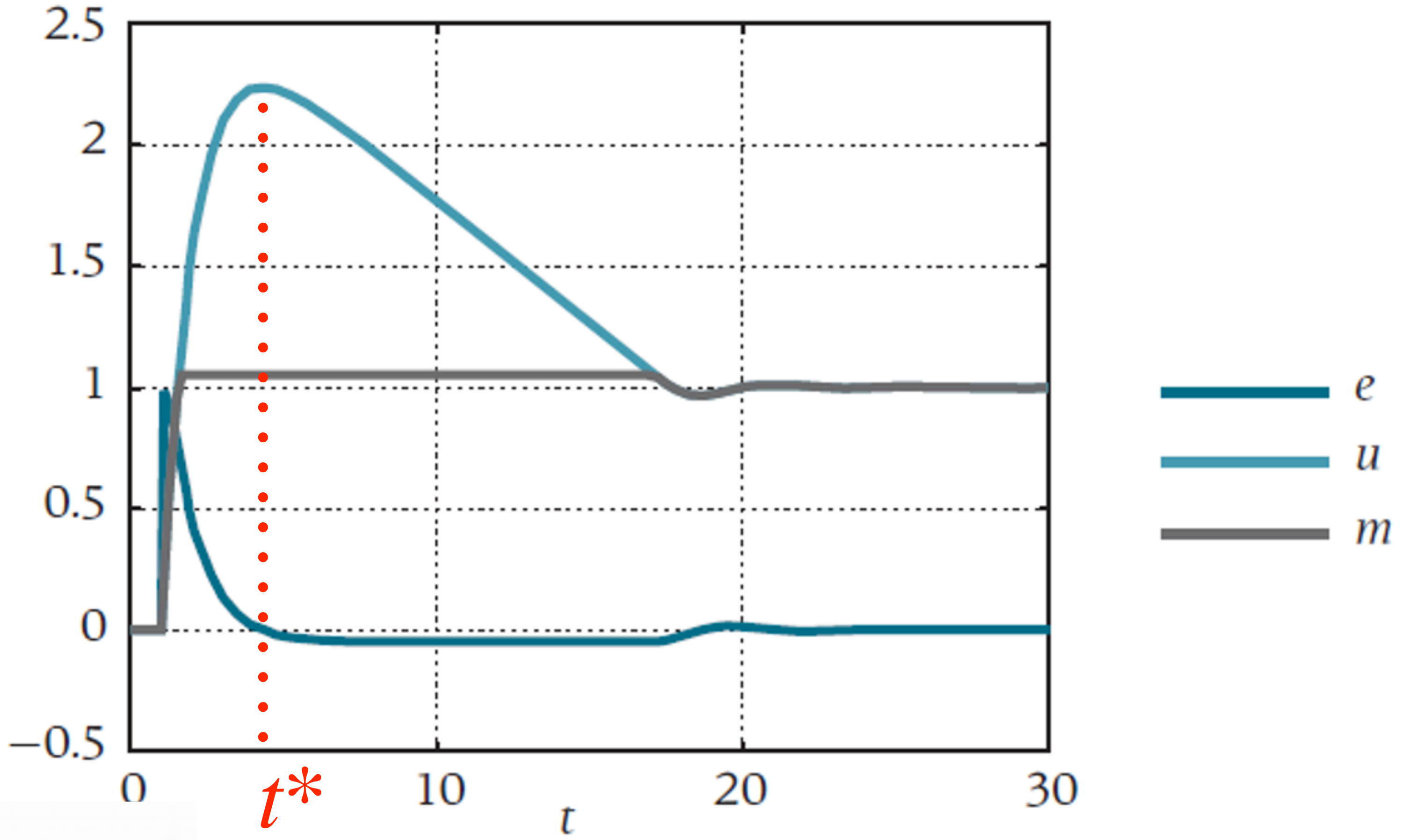
Process



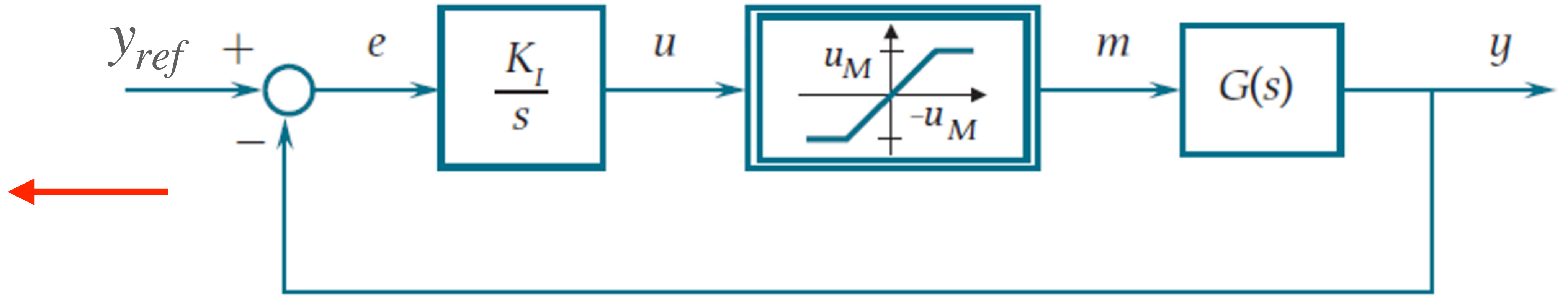

$u(t) \in \mathbb{R}$

$y(t) \in \mathbb{R}$



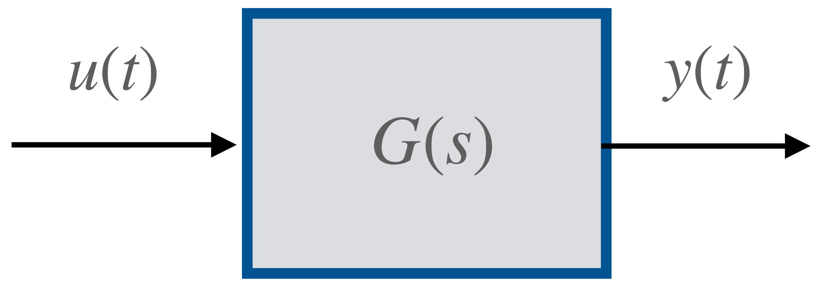




Realistic situation



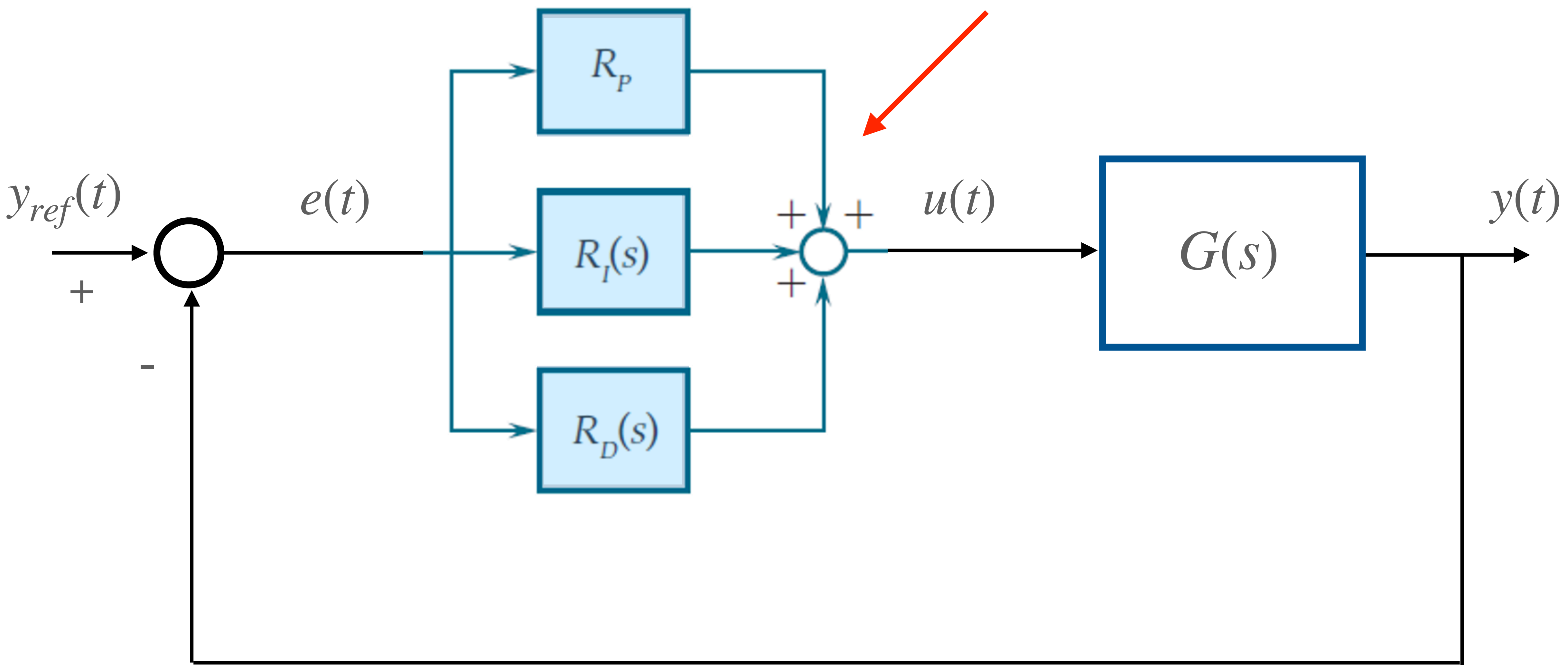
# PID Controllers: Wind-up Effect

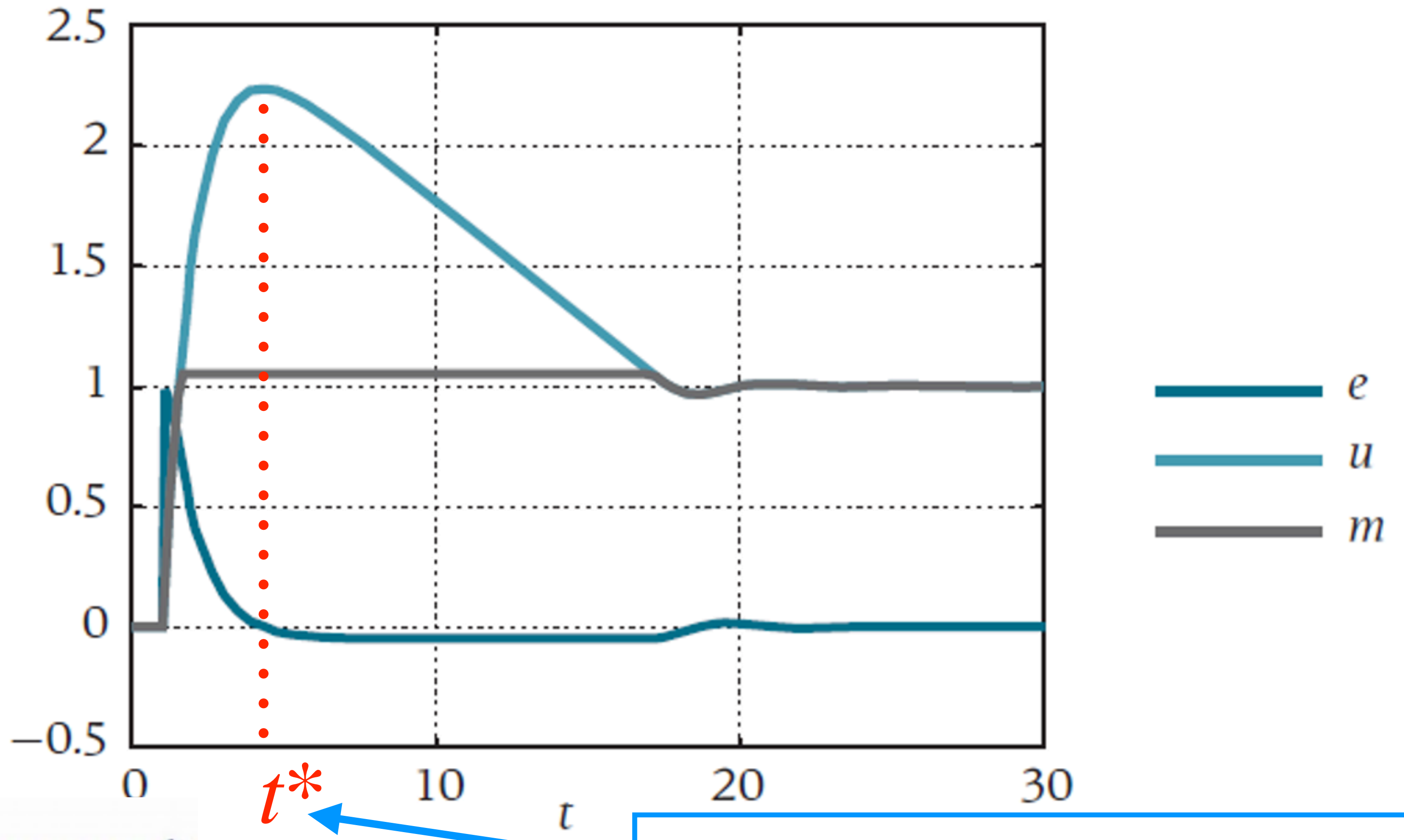
Process



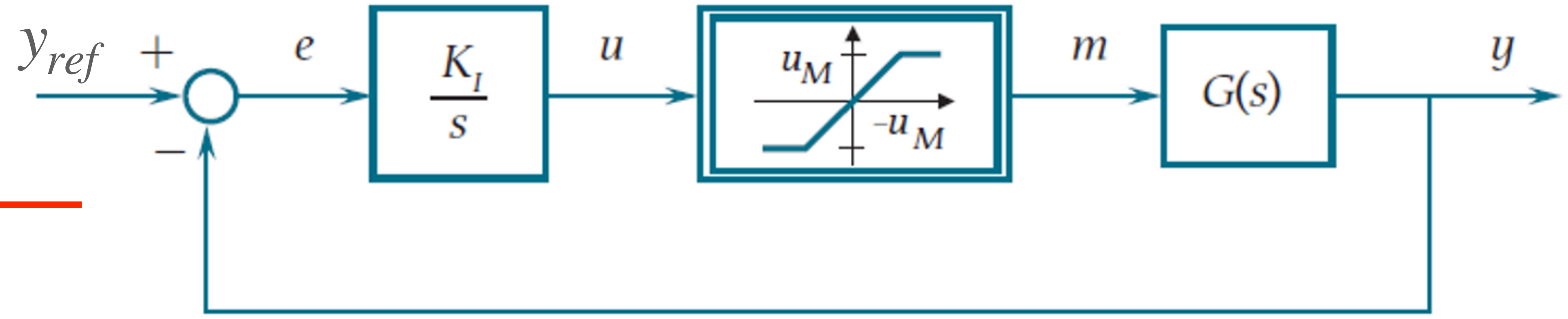

$$u(t) \in \mathbb{R}$$
$$y(t) \in \mathbb{R}$$

PID Controller



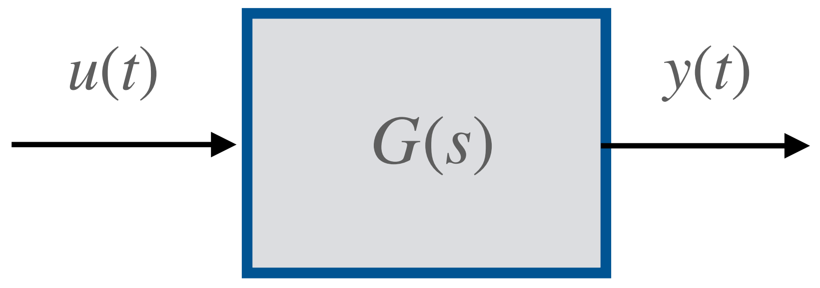

Realistic situation




Ideal situation: the control action exits the saturation mode when $e$ changes its sign (no useless waiting time due to wind-up)

# PID Controllers: Wind-up Effect

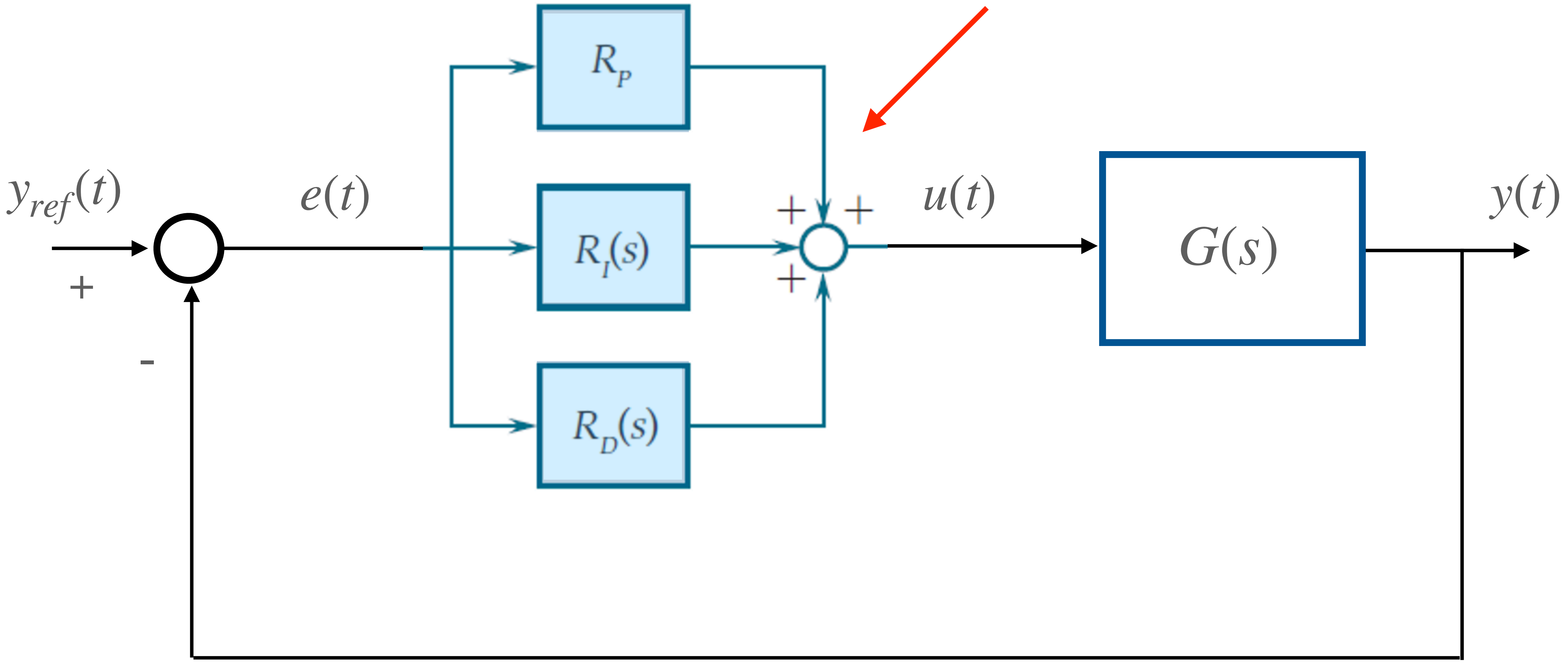
Process

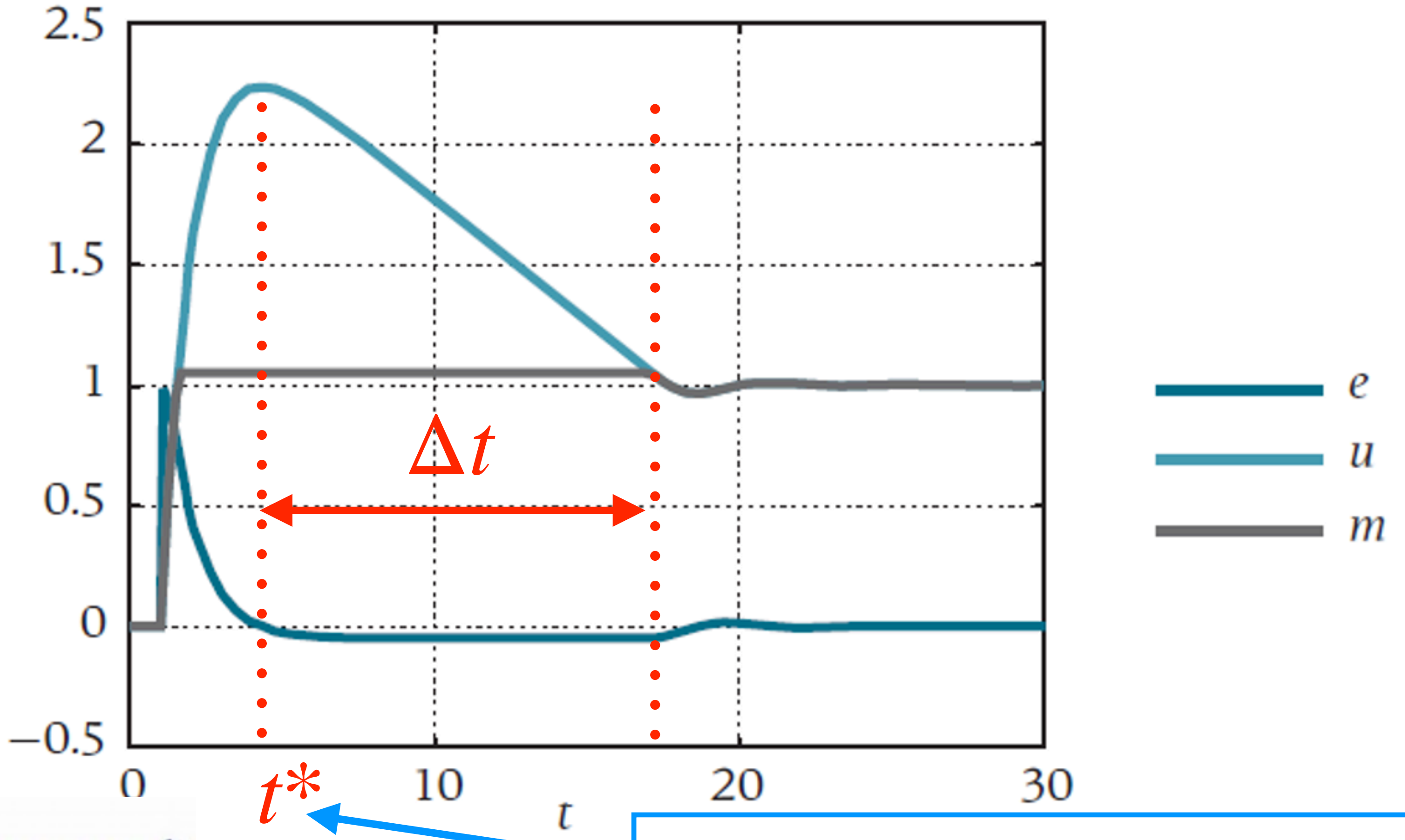



$u(t) \in \mathbb{R}$

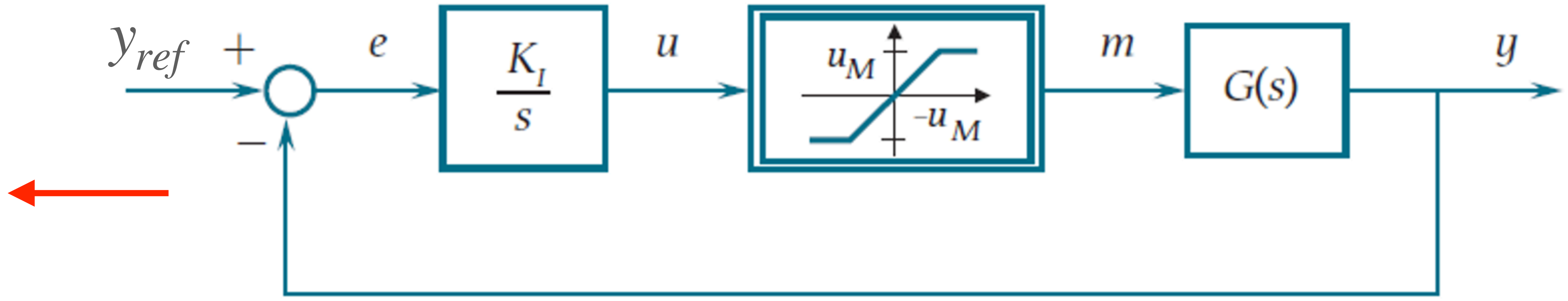
$y(t) \in \mathbb{R}$

PID Controller



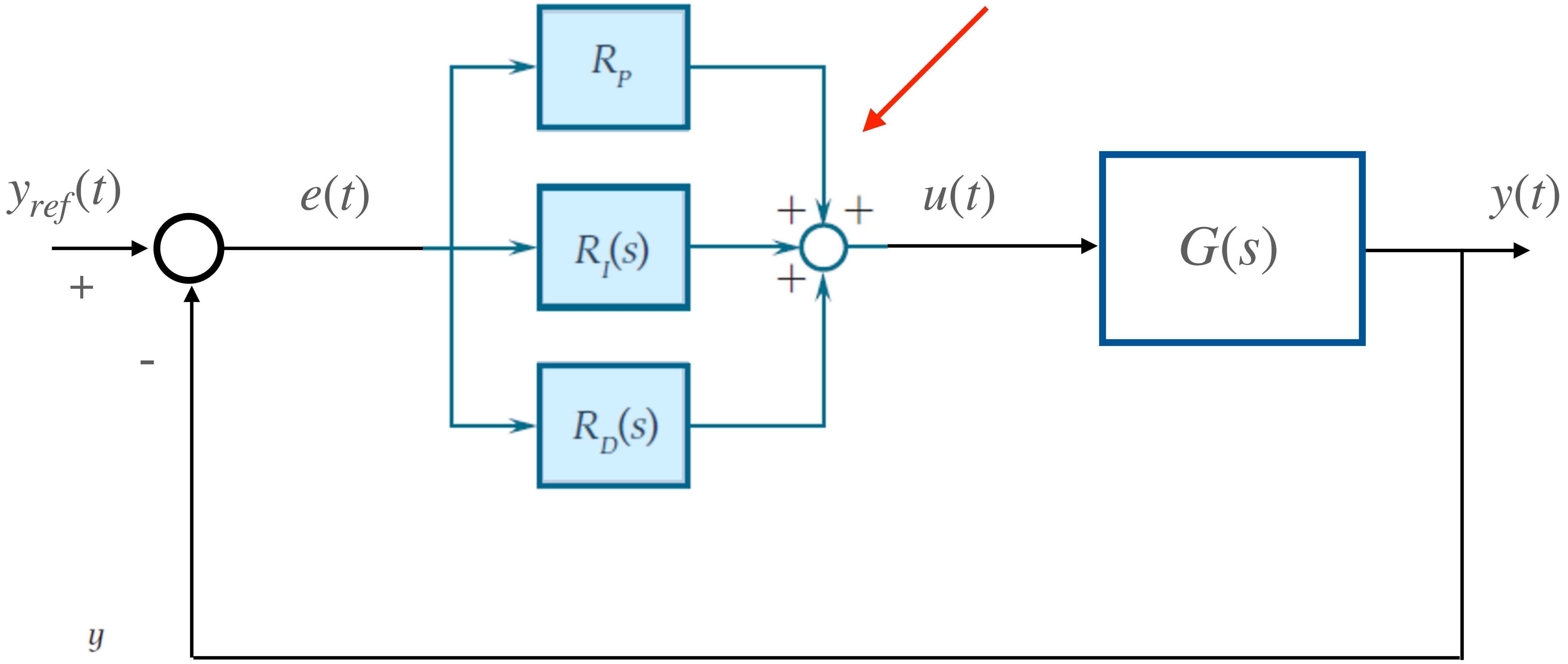

Realistic situation



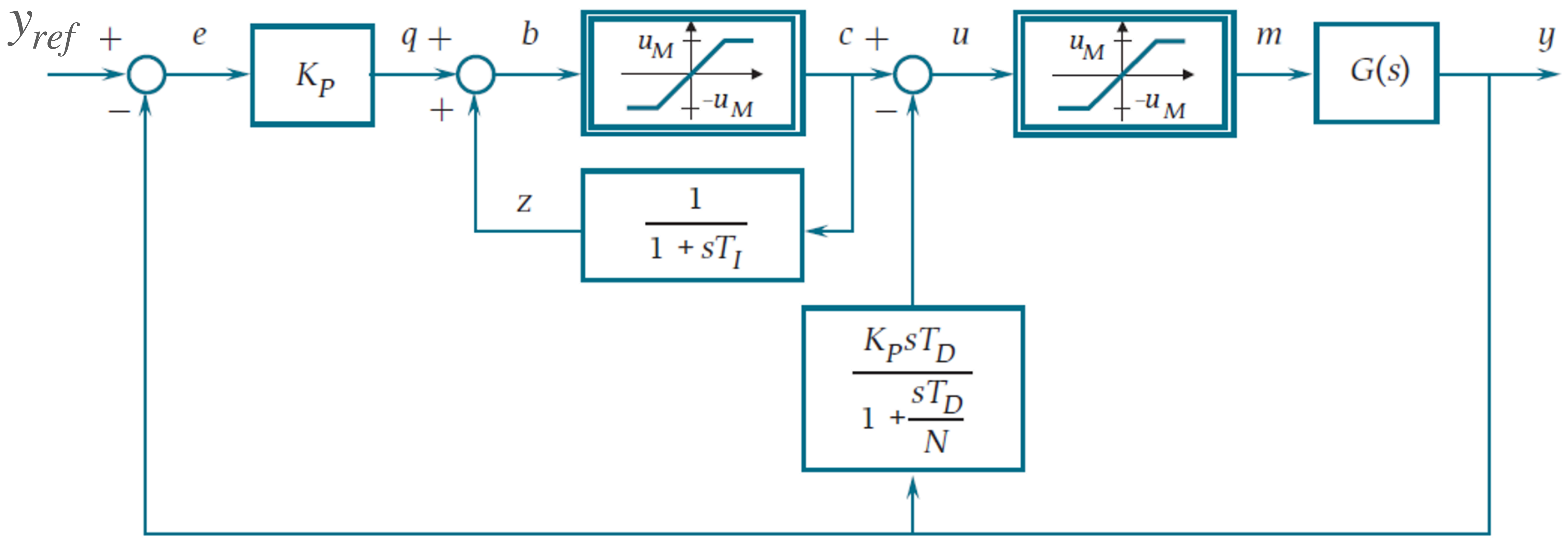

Ideal situation: the control action exits the saturation mode when $e$ changes its sign (no useless waiting time due to wind-up)

# PID Controllers: Wind-up Effect

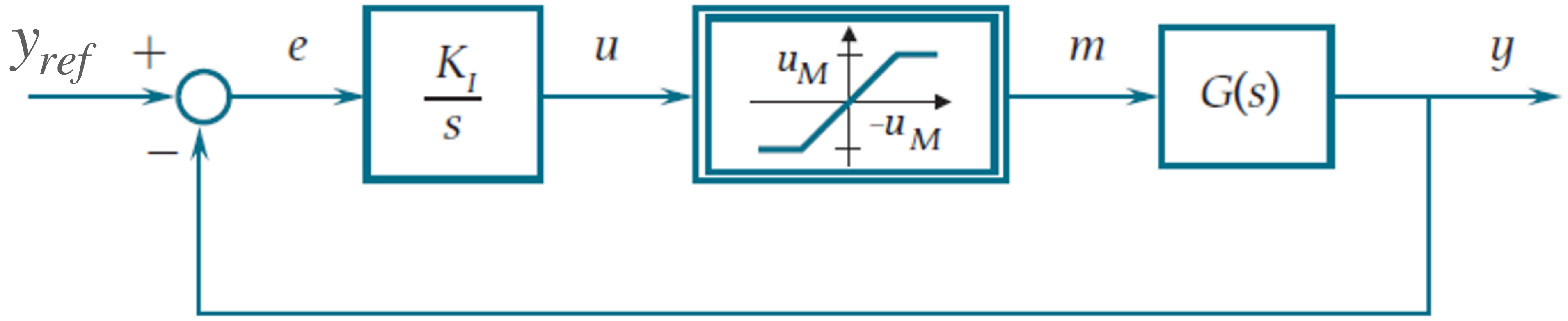
PID Controller



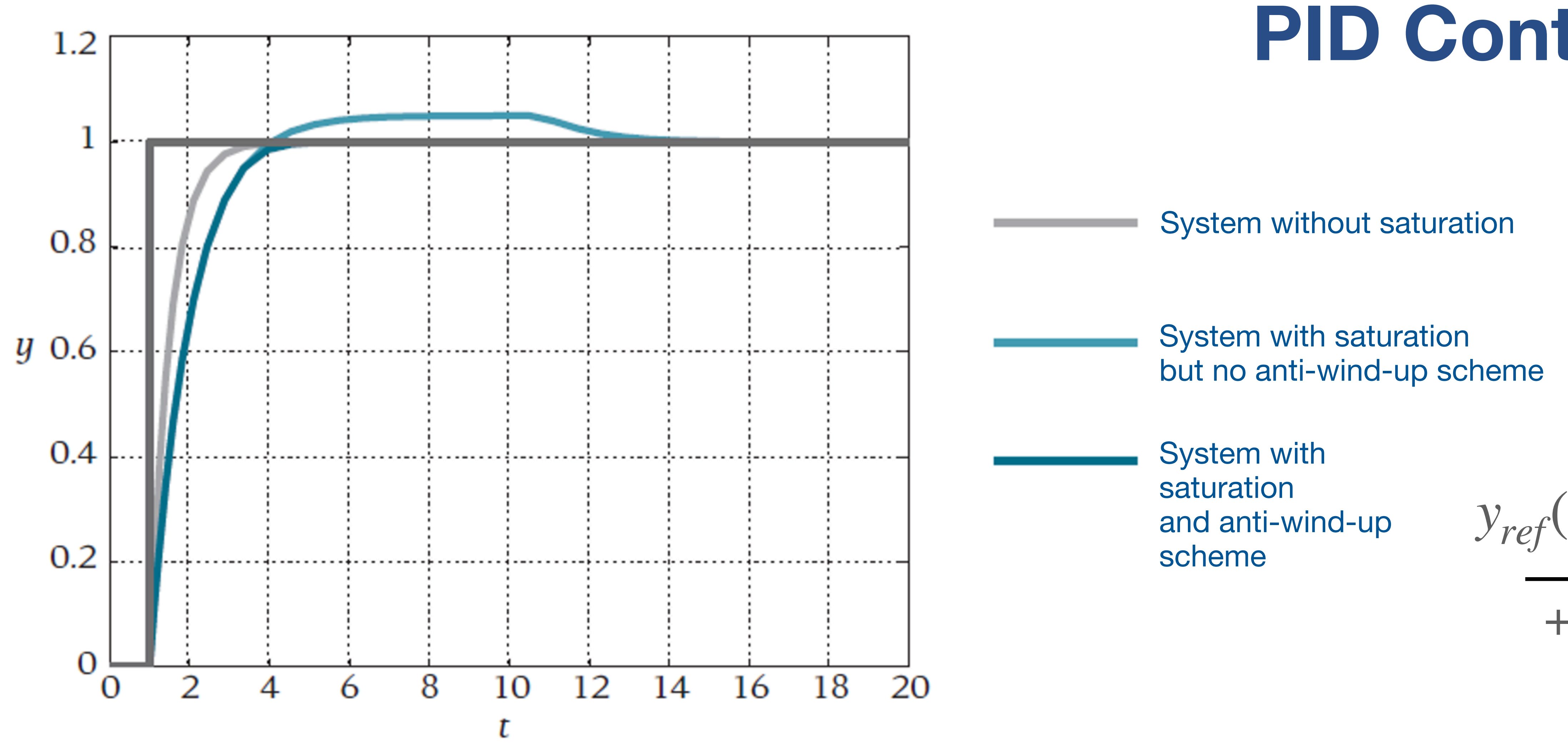

Solution: Anti-wind-up scheme



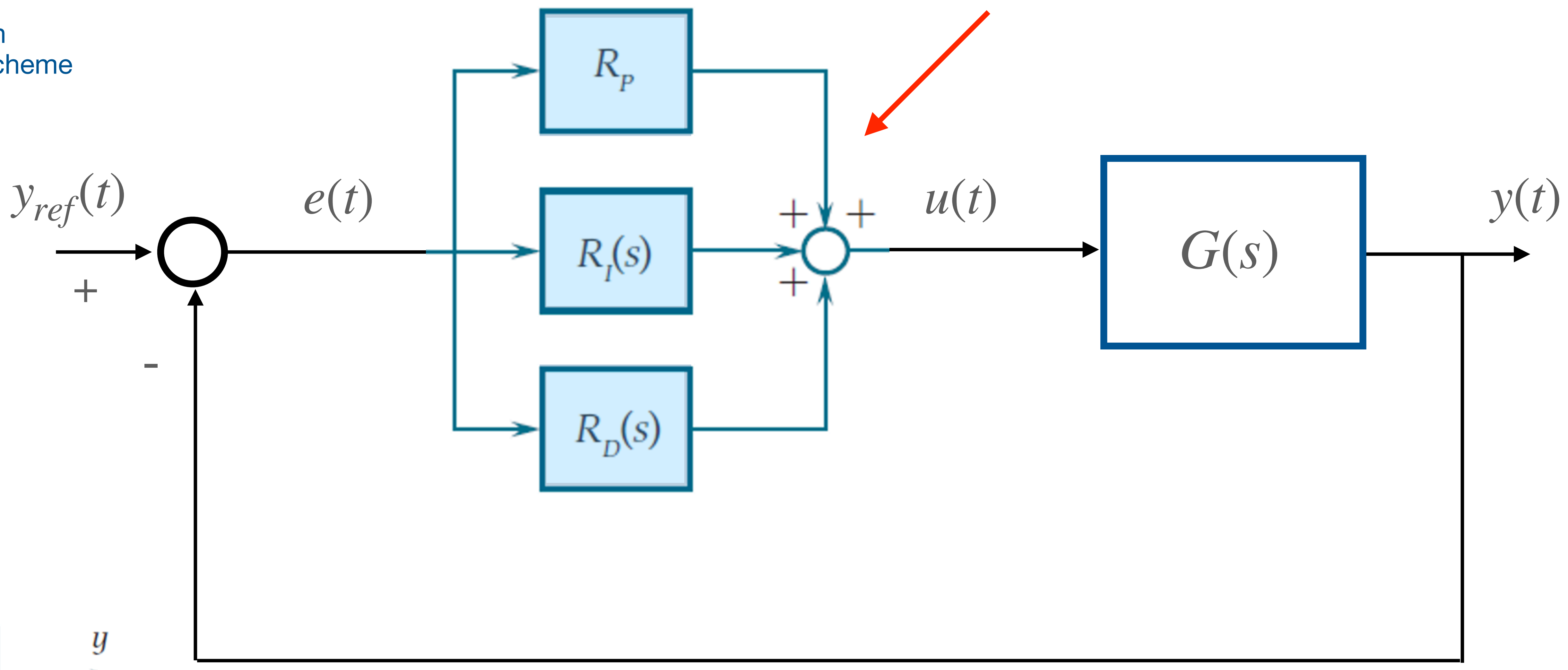

Realistic situation



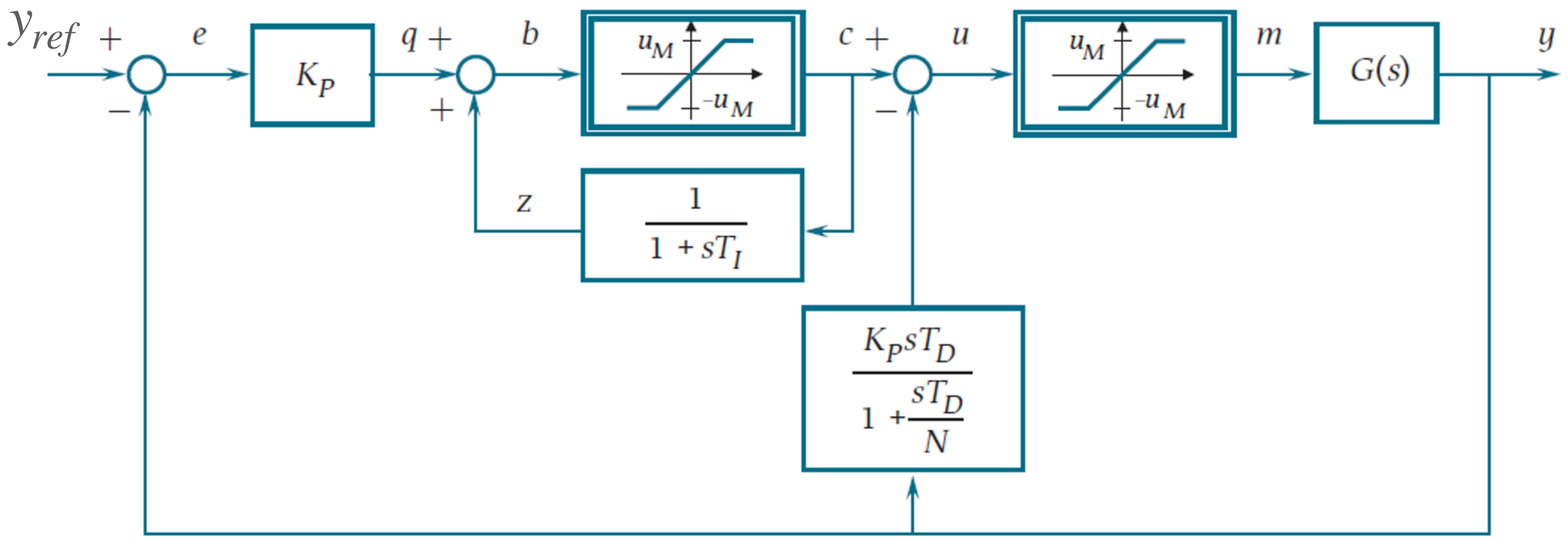
# PID Controllers: Wind-up Effect



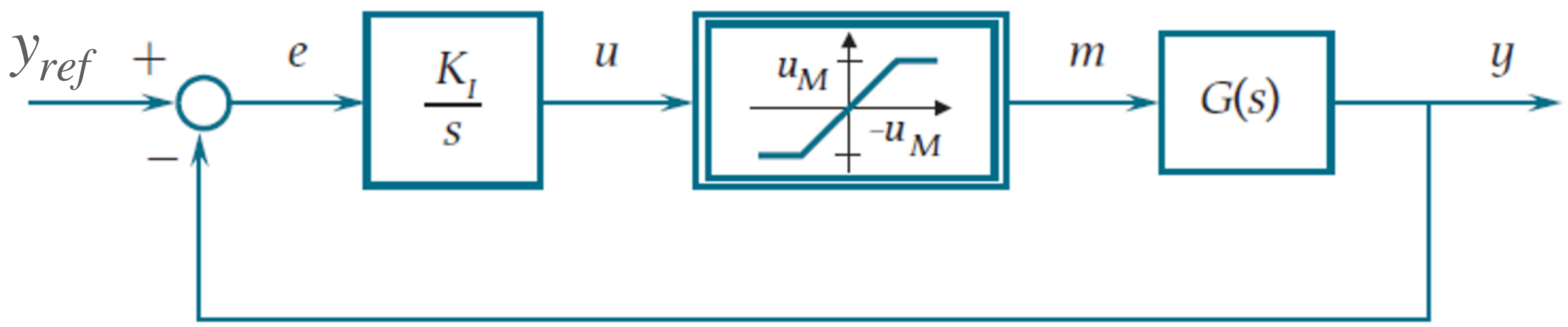

PID Controller




## Solution: Anti-wind-up scheme




## Realistic situation

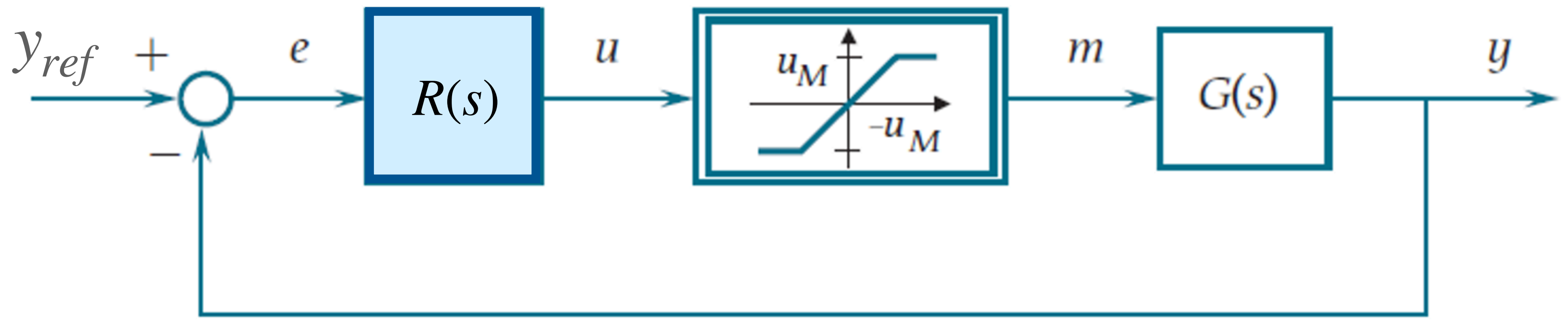


# PID Controllers: Wind-up Effect

## Generic integral controller

$$R(s) = \frac{N_R(s)}{D_R(s)}, \quad D_R(0) = 0$$

## Realistic situation

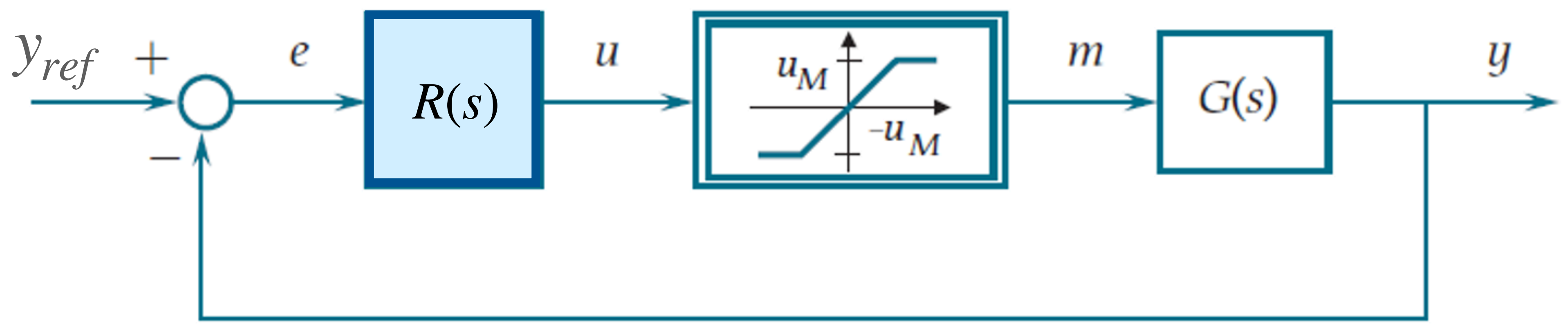


## PID Controllers: Wind-up Effect

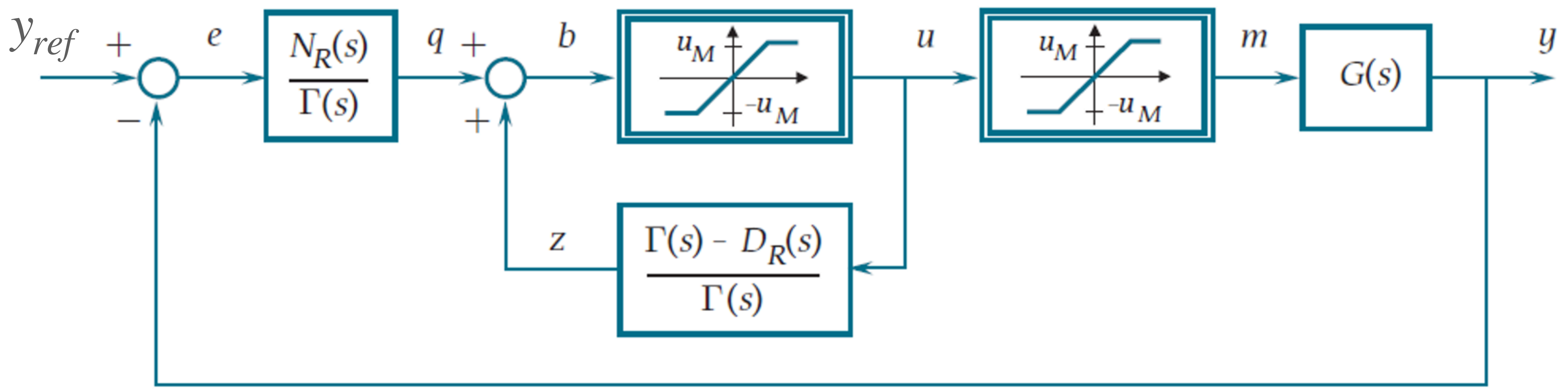
Generic integral controller

$$R(s) = \frac{N_R(s)}{D_R(s)}, \quad D_R(0) = 0$$

Realistic situation




Solution: Anti-wind-up scheme for generic integral controller

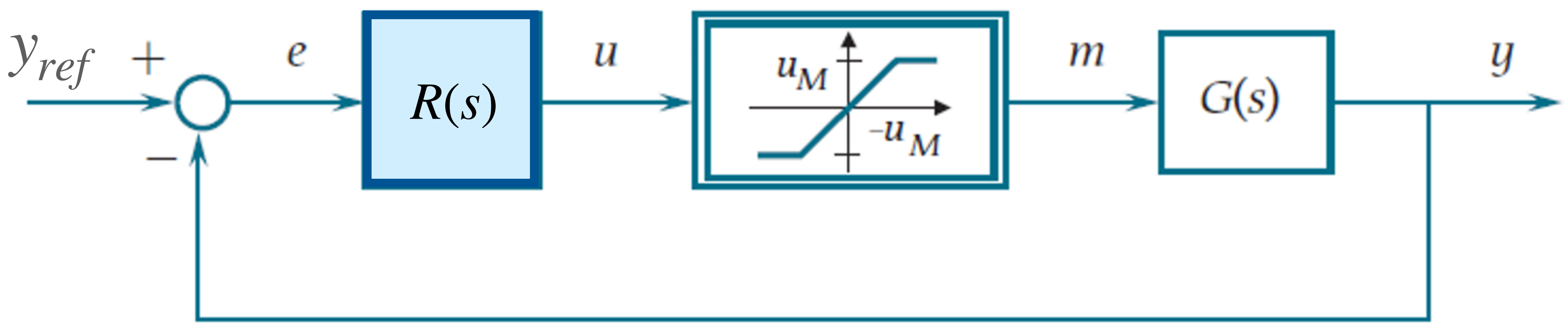


# PID Controllers: Wind-up Effect

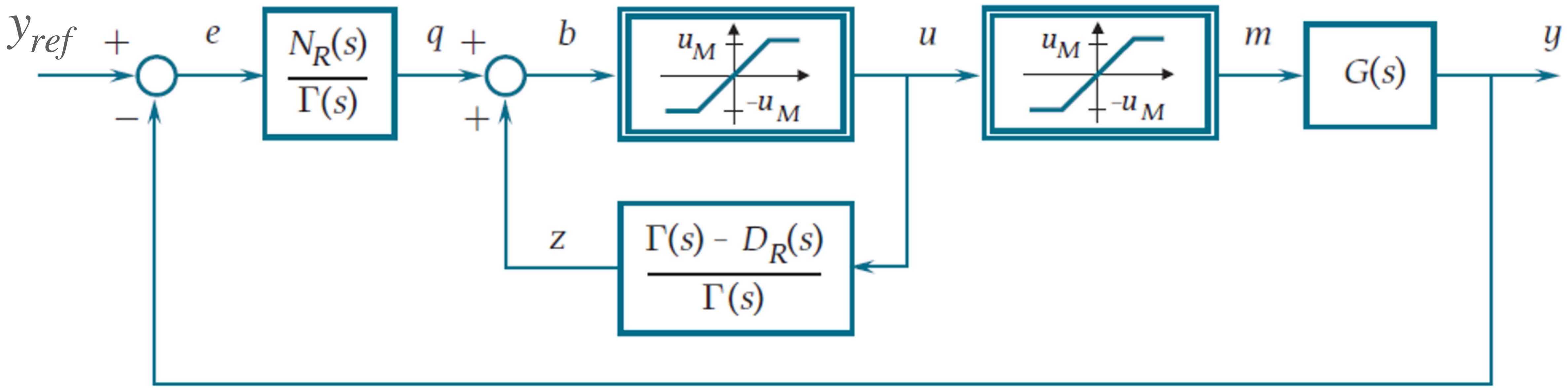
## Generic integral controller

$$R(s) = \frac{N_R(s)}{D_R(s)}, \quad D_R(0) = 0$$

## Realistic situation




## Solution: Anti-wind-up scheme for generic integral controller




## Alternative (accessible signal downstream of saturation)