PID Controllers

1 PID controllers

In industrial applications, PID controllers, which are mass-produced and have a common structure, are frequently used.

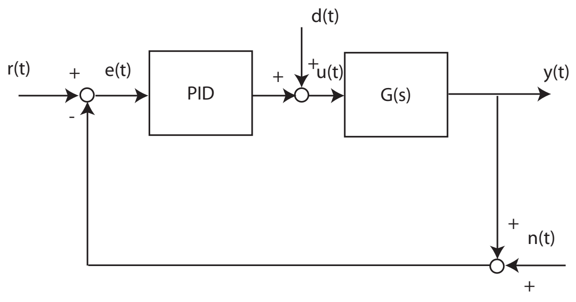

Figura 1: PID Scheme *Anonymous*

These controllers are used exclusively for regulation problems (constant reference, position control problem) and, once the transfer function of the plant has been modeled they allow, through proper tuning, within certain limits of a parameter set, to achieve the desired closed-loop performance. These controllers are called PID controllers because, historically, they were obtained through the appropriate combination of proportional action, integral action, and derivative action. The three actions can therefore be described as follows:

1. *Proportional Action*

$$u(t) = K_p\, e(t) = K_p\, (r(t) - y(t)) \tag{1}$$

K_p is the proportional gain. The advantage of proportional action is related to:

- reduction of the step tracking error and reduction of the regulated output error due to load disturbances on the control channel;
- reduction of the output error due to constant disturbances in the control loop (load disturbances or disturbances at the plant output);

Note that it is not possible to *zero* the steady-state tracking error with the sole use of proportional action. Furthermore, a high value of proportional gain K_p could lead to a regulated output response with a significant overshoot (bad transient behavior). The elimination of the steady-state tracking error (commonly referred to as a *reset procedure*) can also be achieved through the use of a constant *bias* signal, whose amplitude is modulated externally based on considerations about the plant model, the amplitude of the reference signal to be tracked, and the presence of constant disturbances in the control channel. Thus, we have the following proportional control law with manual *reset*:

$$u(t) = K_p\, e(t) + u_b \tag{2}$$

where the signal u_b must be modulated in such a way as to eliminate the tracking error.

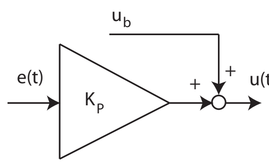

Figura 2: Proportional Controller with Manual Reset

Let us assume for simplicity that there are no disturbances in the system and that the reference is a step input of amplitude R. At steady state, the tracking error will be:

$$e_\infty = \lim_{t \to +\infty} e(t) = \lim_{s \to 0} s\, E(s) = \lim_{s \to 0} s \left(\frac{1}{1 + K_p\, G(s)} \frac{R}{s} - \frac{G(s)}{1 + K_p\, G(s)} \frac{u_b}{s} \right) \tag{3}$$

Assuming there are no integral terms in the plant (otherwise this reasoning would not make sense), we obtain:

$$e_\infty = \frac{R}{1 + K_p\, G(s)} - \frac{G(0)\, u_b}{1 + K_p\, G(0)} = \frac{R - G(0)\, u_b}{1 + K_p\, G(0)} \tag{4}$$

Choosing:

$$u_b = \frac{R}{G(0)} \tag{5}$$

allows for resetting the feedback loop. Note that from (5), the reset signal changes if the amplitude of the external reference varies. The concept of manual reset also makes sense in the presence of disturbances. However, from the analysis of (3)-(5), it is evident that it is *necessary* to know *exactly* the amplitude of the disturbance to be eliminated in the tracking error.

Finally, in designing the proportional action, it is often necessary to increase the gain K_p to reduce the output error caused by constant load disturbances once the transient has subsided. However, high values of K_p can result in transients in the regulated output that exhibit exaggerated overshoot in the presence of constant load disturbances. To address this issue, it is appropriate to weight the reference contribution in the proportional action to reduce its impact on the error signal while emphasizing the contribution from the regulated output (which is necessary to mitigate errors due to load disturbances). Essentially, the proportional component is modified as follows:

$$K_p\, (b\, r(t) - y(t))\,, \quad 0 < b < 1 \tag{6}$$

The proportional action thus includes a reference weighting term (*set-point weighting*) to compensate for lively transients in the presence of constant load disturbances. An effective choice for the parameter b is to set it in the range $b = 0.6 \div 0.8$.

2. *Integral Action*

   The use of integral action means summing (and storing) the past values of the tracking error:

$$u(t) = K_i \int_0^t e(\sigma)\, d\sigma \tag{7}$$

   where K_i is the integral gain. Numerically, as previously stated, this corresponds to an adder which tracks past error values. The advantage lies in its ability to achieve automatic reset of the step tracking error without needing to know the reference amplitude or assess any constant disturbances in the control channel. However, the disadvantage of using only integral action is clear: a pure integrator introduces a phase shift of -90° in the loop's frequency response, leading to significant stability issues (the phase margin is reduced by the lag induced by the integrator).

3. *Derivative Action*

$$u(t) = K_d \frac{d\, e(t)}{d\, t} \tag{8}$$

   Here, K_d is the derivative gain. When combined with proportional action, derivative action provides the controller with basic predictive capability regarding the configuration of the tracking error. We will analyze the advantages and disadvantages of this action shortly.

As mentioned earlier, actions (7)-(8), aside from proportional action (which is defined as the P controller), are never implemented individually but are appropriately combined instead. This leads to the following structures:

1.1 PI Controller

In a standard PI (Proportional + Integral) controller, the structure is as follows:

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\sigma)\, d\sigma \tag{9}$$

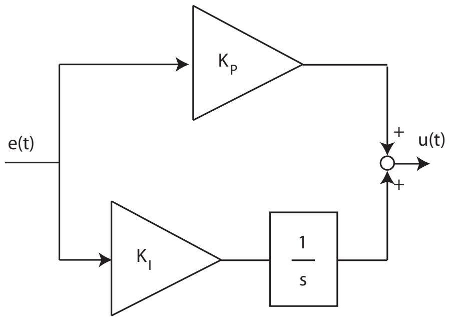

Figura 3: PI Controller

This combines two effects: proportional control of the error and stroring of the tracking error's past history. It is often useful to rewrite the PI in the time domain, highlighting the so-called *integral time constant* or *reset time*, T_i, which is also referred to as *reset time*.

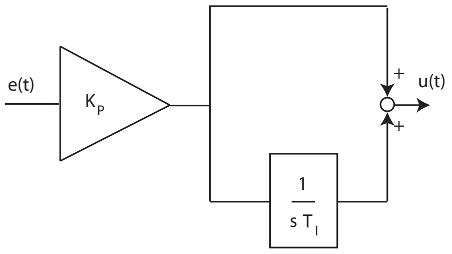

Figura 4: PI Controller highlighting *reset time*

The integral time or reset time T_i measures how quickly the tracking error is stored over time. A high T_i value implies that the integral action is smoother because the adder takes more time to "remember" past tracking error values. Conversely, if T_i is small, the integrator becomes "nervous," quickly retaining the past tracking error values. To better understand the meaning of *reset time*, consider the PI transfer function:

$$C_{PI}(s) = K_p \left(1 + \frac{1}{s\,T_i}\right) = K_p\, \frac{1 + s\,T_i}{s\,T_i} \tag{10}$$

Formally, the PI controller adds a pole at the origin and a zero at $z = -\frac{1}{T_i}$ to the feedback loop. The integral effect of the pole at the origin is useful at low frequencies because it increases the loop gain at these frequencies. The zero is necessary to compensate for the $-90°$ phase lag introduced by the pole at the origin and to maintain uniform magnitude. In general, the zero is placed to the left of the frequency where the mid-frequency region begins to compensate for the phase delay introduced by the pole at the origin (see the figure showing the Bode plot of a PI compensator).

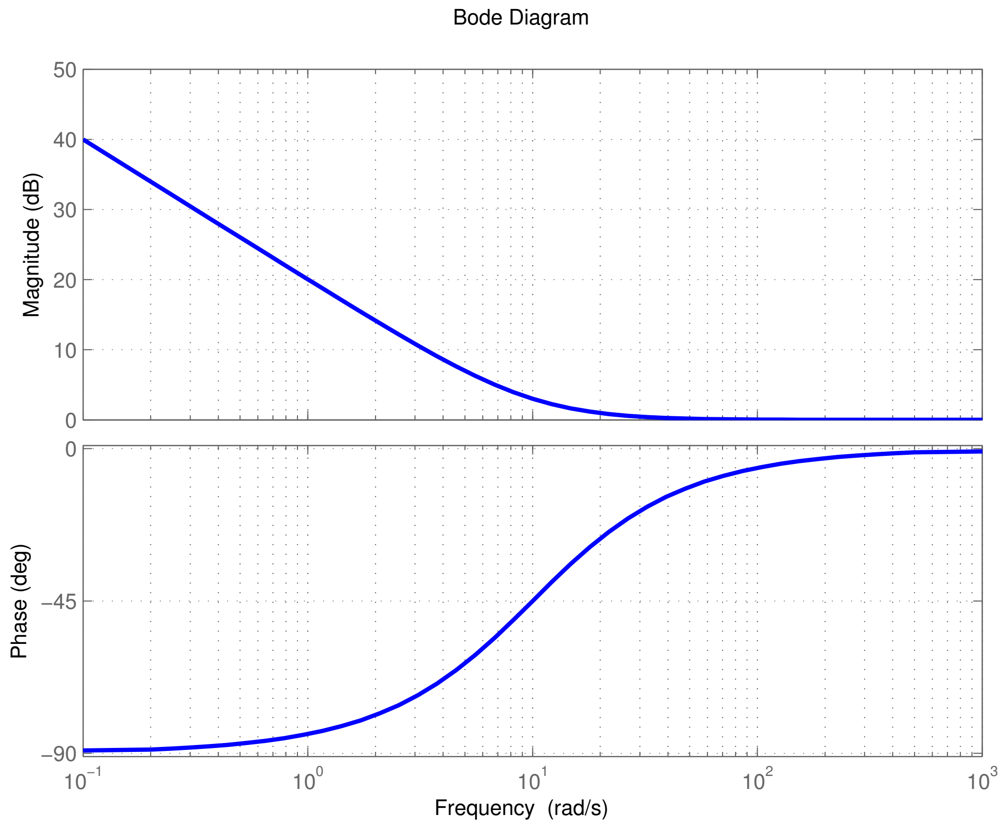

Figura 5: Bode diagram of a PI controller

The phase effect of the pole at the origin is compensated by the zero located at $-\frac{1}{T_i}$, which counters the 90° phase lag with an advance. The effect of this phase advance depends on the location of the zero's break frequency $\frac{1}{T_i}$. If the reset time T_i is large, the zero "anticipates" at lower frequencies, providing more effective compensation but resulting in a "softer" integrator. Conversely, if T_i is small, the compensation occurs at higher frequencies, making the integrator more effective but potentially compromising closed-loop stability.

From (10), we have:

$$C_{PI}(s) = K_p \frac{1 + s\,T_i}{s\,T_i} = K_p \frac{1}{1 - \frac{1}{s\,T_i + 1}} \tag{11}$$

Thus, the PI controller explicitly highlights the automatic reset signal:

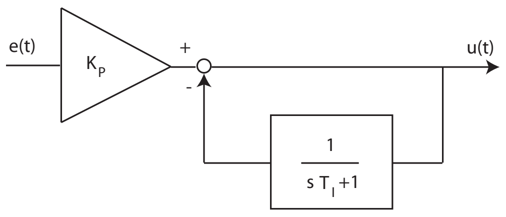

Figura 6: PI controller emphasizing the automatic reset effect

The reset signal is generated by a first-order system whose transfer function is:

$$\frac{1}{s\,T_i + 1} \tag{12}$$

If the integral time T_i (which represents the time constant of the filter in (12)) is large, the reset effect will occur more slowly. Conversely, if T_i is small, the reset effect will be faster. The PI controller can also be viewed as an lag network. This can be understood as follows, starting from an lag network model:

$$\frac{s\,\alpha\,T+1}{s\,T+1} = \alpha\,\frac{s+\frac{1}{\alpha\,T}}{s+\frac{1}{T}} \tag{13}$$

Now, if α is very small (approaching zero), we have $T \gg \alpha\,T$, and (13) can be approximated as:

$$\alpha\,\frac{s+\frac{1}{\alpha\,T}}{s+\frac{1}{T}} \approx \alpha\,\frac{s+\frac{1}{\alpha\,T}}{s} = \alpha\,\frac{\alpha\,T\,s+1}{\alpha\,T\,s} \tag{14}$$

Thus, an lag network is approximately equivalent to a PI controller with a reset time of $\alpha\,T$ and a proportional gain of α. Alternatively, a PI controller can be viewed as an lag network with an added gain modulation operation using K_p.

1.2 PD Controller

A standard PD (Proportional + Derivative) controller has the following structure:

$$u(t) = K_p\,e(t) + K_d\,\frac{d\,e(t)}{d\,t} \tag{15}$$

This combines two primary effects: proportional control based on the error and predictive control achieved through a linear approximation of the signal. It is often useful to rewrite the PD in the time domain, highlighting the so-called *derivative time* or *prediction horizon* T_d:

$$u(t) = K_p\left(e(t) + T_d\,\frac{d\,e(t)}{d\,t}\right) \tag{16}$$

The control signal, for a fixed t, is directly proportional (via K_p) to a linear approximation of the future evolution of the tracking error.

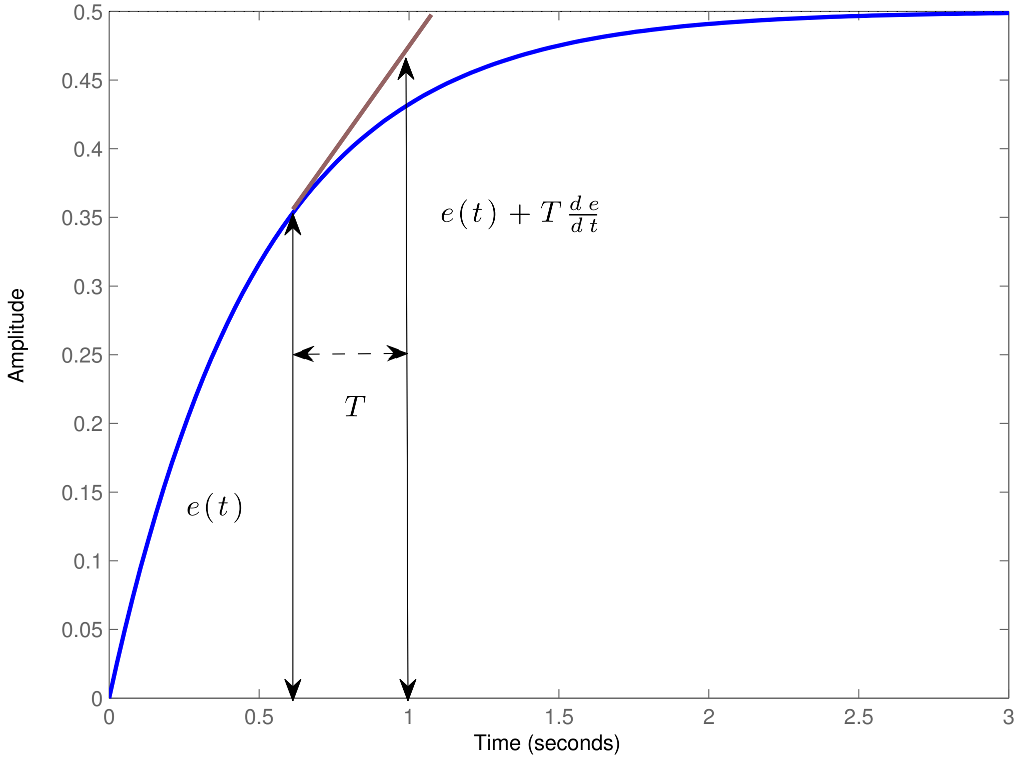

Figura 7: Linear prediction of the tracking error

The predictive capability of the controller is determined by the derivative time T_d: the larger T_d, the further the linear approximation deviates from the future value of the error. Conversely, the smaller T_d, the less effective the predictive capability.

In the Laplace transform domain, we have:

$$U(s) = K_p\ (1 + T_d\, s)\ E(s) \tag{17}$$

A pure PD controller is not physically realizable, as it is described by a transfer function with a zero at $z = -\frac{1}{T_d}$. The issue of physical realizability arises due to the effect of the derivative action on measurement noise present in the feedback channel.

If we assume noise characterized by a high-frequency harmonic:

$$n(t) = A_n\ \sin(\omega\, t) \tag{18}$$

This noise appears in the control channel, and its component processed by the derivative (scaled by K_d) will be:

$$A_n\, \omega\ \cos(\omega\, t) \tag{19}$$

That is, it will become a harmonic with amplitude $A_n\, \omega$. If ω is high, this effect can be significant, creating a problematic situation. To address this, a low-pass filter is added to the derivative structure with two objectives: making the derivative operator physically realizable and compensating for high-frequency measurement noise:

$$U(s) = K_p \left(1 + \frac{T_d\, s}{1 + \frac{T_d}{N}\, s}\right) E(s) \tag{20}$$

where $N = 1 \div 33$.

The PD structure in (20) is effectively a lead compensator with added proportional gain K_p:

$$C_{PD}(s) = K_p \left(1 + \frac{T_d\, s}{1 + \frac{T_d}{N}\, s}\right) = K_p\, \frac{1 + \frac{T_d}{N}\, s + T_d\, s}{1 + \frac{T_d}{N}\, s} = K_p\, \frac{1 + \frac{N+1}{N}\, T_d\, s}{1 + \frac{T_d}{N}\, s} = K_p\, \frac{1 + s\, T}{1 + \alpha\, s\, T} \tag{21}$$

where:

$$T = \frac{N+1}{N}\, T_d, \quad \alpha = \frac{1}{N+1} \tag{22}$$

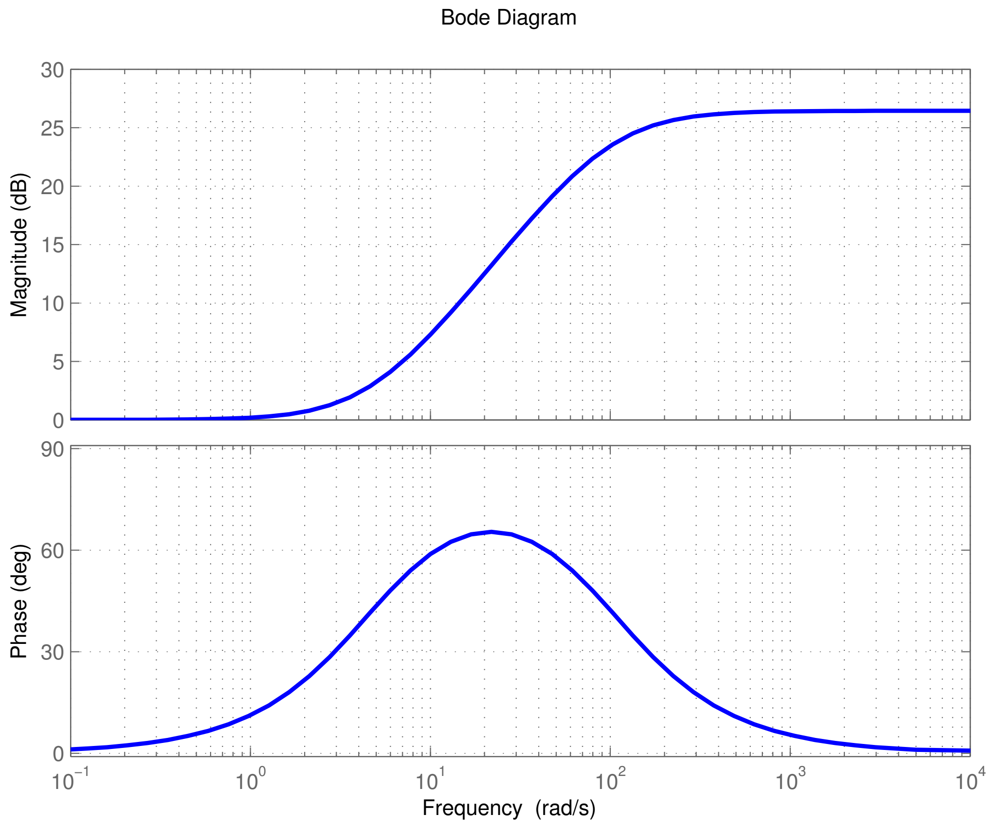

Figura 8: Bode diagram of a PD controller with a realizability pole

The PD controller is described by a zero with a time constant $T_d + \frac{T_d}{N} \approx T_d$ and a faster pole with a time constant $\frac{T_d}{N}$.

1.2.1 The *Derivative Kick*

It is important to note that it is not effective to include the derivative action in the direct path (see the figure below). This is because the reference signal, in PID controllers, is often a step signal that may exhibit first-order discontinuities (e.g., step-level changes). Evaluating the derivative in the direct path can cause impulsive signals ("kicks"):

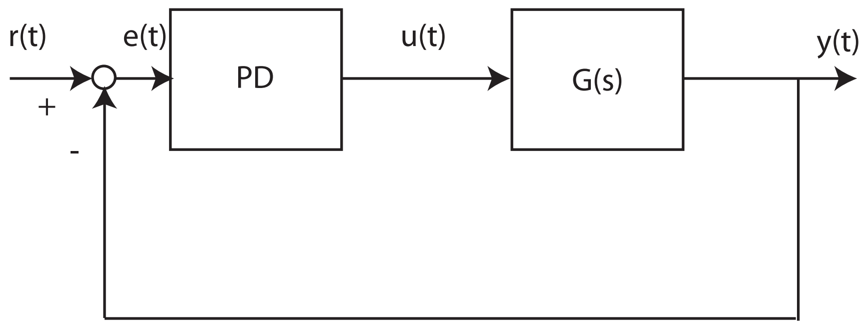

Figura 9: PD control with derivative action in the direct path

To avoid this issue, it is preferable to apply the derivative action directly to the regulated output, which is generally a smooth signal without significant discontinuities (it is the output of a low-pass filter):

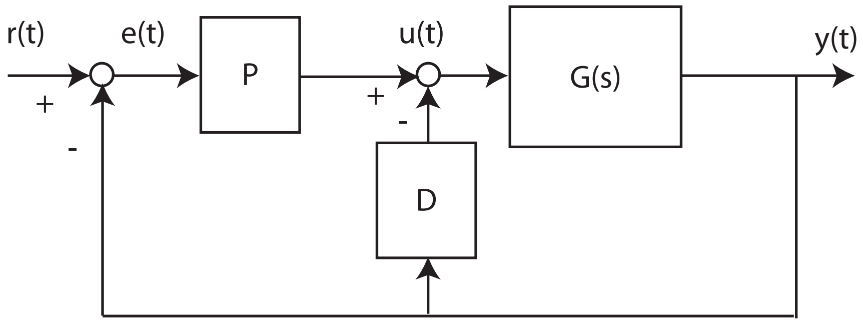

Figura 10: PD control with derivative action on the plant output

A PD controller with derivative action applied to the output can be expressed as:

$$U(s) = K_p\, E(s) + K_p\, \frac{T_d\, s}{1 + \frac{T_d}{N}\, s}\, Y(s) \tag{23}$$

However, it is important to highlight that measurement noise can cause serious issues when implementing derivative action in a standard controller. Consider a signal of the type $n(t) = N\, \sin(\omega\, t)$, representing a potential measurement error. If this noise is processed by the derivative operator, its effect in the direct path will be:

$$K_p\, \frac{d\, n(t)}{d\, t} = K_p\, N\, \omega\, \cos(\omega\, t) \tag{24}$$

Thus, the amplitude of the measurement noise in the control chain depends on the noise frequency. If the frequency is high, the gain $K_p\, N\, \omega$ can become significantly large.

What is the consequence of this effect? It limits the derivative gain, which cannot be increased excessively. As a result, the prediction horizon T_d, which depends on the derivative gain, also cannot be indefinitely extended. Excessive increases in T_d are also impractical due to the divergence between the linear error prediction and the actual error behavior. In essence, the larger the prediction horizon, the greater the discrepancy between the predicted and actual error, increasing the likelihood of making incorrect assumptions ("mistaking apples for oranges").

1.3 PID Controller

Combining the three control actions results in the complete PID controller:

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\sigma)\, d\sigma + K_d\, \frac{d\, e(t)}{d\, t} \tag{25}$$

This can be rewritten to highlight the integral time constant T_i and the derivative time constant T_d:

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\sigma)\, d\sigma + T_d\, \frac{d\, e(t)}{d\, t} \right) \tag{26}$$

Essentially, the PID controller combines the effects of the PI and PD controllers, resembling a saddle network. In the s-domain, this becomes:

$$U(s) = K_p \left(1 + \frac{1}{T_i\, s} + T_d\, s \right) E(s) \tag{27}$$

Thus, the transfer function of the PID in its *pure form* is not physically realizable due to the derivative term. A physically realizable version can be achieved by introducing a low-pass filter to mitigate the effects of measurement noise on the regulated output:

$$U(s) = K_p \left(1 + \frac{1}{T_i\, s} + \frac{T_d\, s}{1 + \frac{T_d}{N}\, s} \right) E(s) \tag{28}$$

The transfer function of the PID controller will then include two poles (one at $s = 0$ and one at $s = -\frac{N}{T_d}$) and two zeros, which—when N is large—are the roots of the polynomial:

$$T_d\, T_i \left(1 + \frac{1}{N} \right) s^2 + \left(T_i + \frac{T_d}{N} \right) s + 1 = 0 \tag{29}$$

These roots can be real or complex conjugates depending on the values of T_d, T_i, N, and thus on the sign of the discriminant:

$$\left(T_i + \frac{T_d}{N}\right)^2 - 4\,T_d\,T_i\,\left(1 + \frac{1}{N}\right) \tag{30}$$

The considerations regarding the placement of the derivative action in the feedback path to avoid derivative kick also apply here. In such a case, the PID controller would become, in the Laplace transform domain:

$$U(s) = K_p\,\left(1 + \frac{1}{T_i\,s}\right)\,E(s) - \frac{K_p\,T_d\,s}{1 + \frac{T_d}{N}\,s}\,Y(s) \tag{31}$$

The Bode diagram of a PID controller, with a low-pass filter applied to the derivative action to compensate for the effects of measurement noise, is shown below:

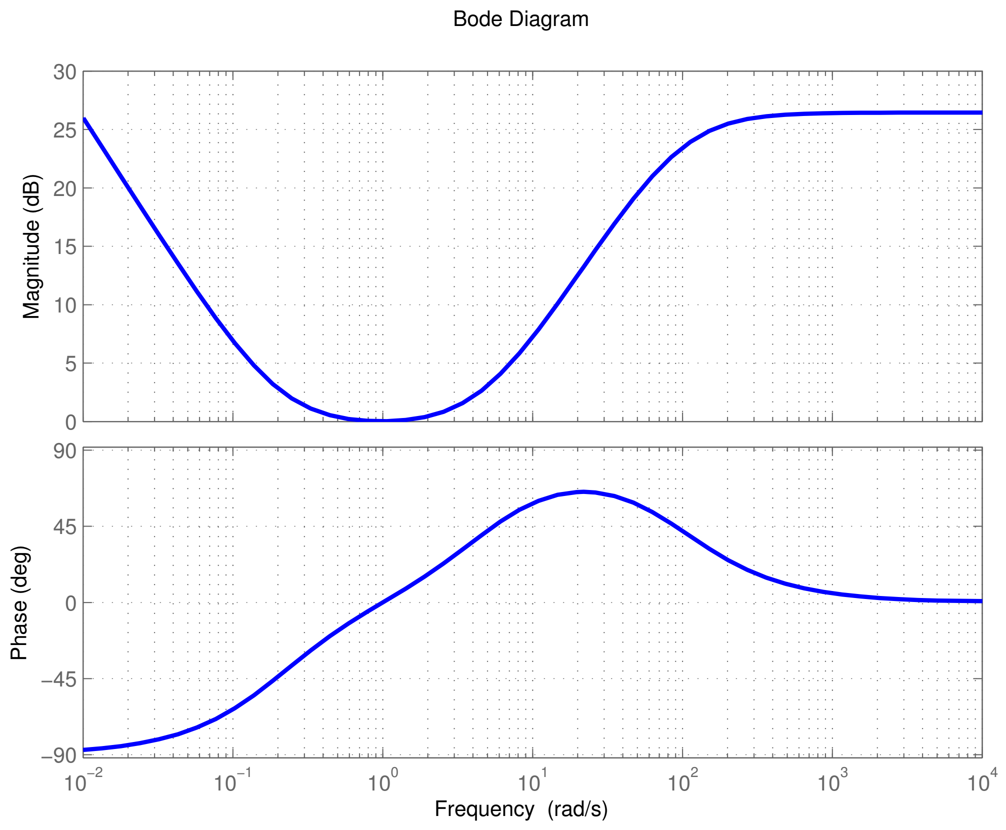

Figura 11: Bode diagram of a PID controller with a low-pass filter on the derivative action to mitigate the effects of measurement noise.

1.4 Tuning Techniques for PID controllers

In this section, we describe some simple methods for tuning PID controllers.

1.4.1 Model-Based Method

We begin by presenting a simple PID tuning algorithm based on selecting the phase margin ϕ_m at a specified crossover frequency ω_c, assuming the transfer function of the system $G(s)$ is known. The pair $(\omega_c,\ \phi_m)$ can be determined through appropriate conversion of the specifications. Let the PID transfer function be given in its elementary form, adapted for tuning:

$$C(s) = K_p + K_d\,s + K_i/s, \quad K_d = K_p\,T_d, \quad K_i = \frac{K_p}{T_i} \tag{32}$$

If $G(s)$ is the system's transfer function, and we want the open-loop transfer function

$$L(s) = C(s)\,G(s) \tag{33}$$

to achieve, at a specified crossover frequency ω_c, a given positive phase margin ϕ_m (compatible with the specifications):

$$L(j\,\omega_c) = C(j\,\omega_c)\,G(j\,\omega_c) = e^{j(\phi_m - \pi)} \tag{34}$$

the frequency response of the PID controller at ω_c will have the following structure:

$$C(j\,\omega_c) = |C(j\,\omega_c)|\; e^{j\,\theta_C} \tag{35}$$

Note that the maximum phase lead the PID can provide is 90° (since the controller has two zeros and one pole). The system's transfer function, in terms of frequency response, will be:

$$G(j\,\omega_c) = |G(j\,\omega_c)|\; e^{j\,\theta_G} \tag{36}$$

From this, we derive the magnitude and phase conditions:

$$|C(j\,\omega_c)|\;|G(j\,\omega_c)| = 1, \quad |C(j\,\omega_c)| = \frac{1}{|G(j\,\omega_c)|} \tag{37}$$

$$\theta_C + \theta_G = \phi_m - \pi, \quad \theta_C = \phi_m - \pi - \theta_G \tag{38}$$

Next, we express the PID frequency response in terms of the parameters K_p, K_i, and K_d:

$$C(j\,\omega) = K_p + j\,K_d\,\omega + \frac{K_i}{j\,\omega} = K_p + j\left(K_d\,\omega - \frac{K_i}{\omega}\right) \tag{39}$$

We rewrite $C(j\,\omega)$ using Euler's formula:

$$C(j\,\omega) = |C(j\,\omega)|\,\cos(\theta_C) + j\;|C(j\,\omega)|\,\sin(\theta_C) \tag{40}$$

This yields the following equations for the real and imaginary parts:

$$\begin{cases} K_p & = \quad |C(j\,\omega)|\,\cos(\theta_C) \\ K_d\,\omega - \dfrac{K_i}{\omega} & = \quad |C(j\,\omega)|\,\sin(\theta_C) \end{cases} \tag{41}$$

At ω_c, we know $|C(j\,\omega_c)| = \frac{1}{|G(j\,\omega_c)|}$ and

$$\theta_C = \phi_m - \pi - \theta_G \tag{42}$$

Thus:

$$\begin{cases} K_p & = \quad \frac{1}{|G(j\,\omega_c)|}\,\cos(\phi_m - \pi - \theta_G) \\ K_d\,\omega_c - \dfrac{K_i}{\omega_c} & = \quad \frac{1}{|G(j\,\omega_c)|}\,\sin(\phi_m - \pi - \theta_G) \end{cases} \tag{43}$$

This results in two equations with three unknowns: K_p, K_i, and K_d. To solve them, a practical choice is to impose the following relation between the integral and derivative times: $T_i = 4\,T_d$.

1.4.2 Model-Free Calibration Methods

These methods are mostly based on heuristics and empirical reasoning, and they are directly applicable to BIBO-stable systems whose step response is generally monotonic, increasing, and relatively slow in terms of transients. The figure shows a step response and a polar diagram of a plant to be regulated with a PID controller, consistent with the described scenario.

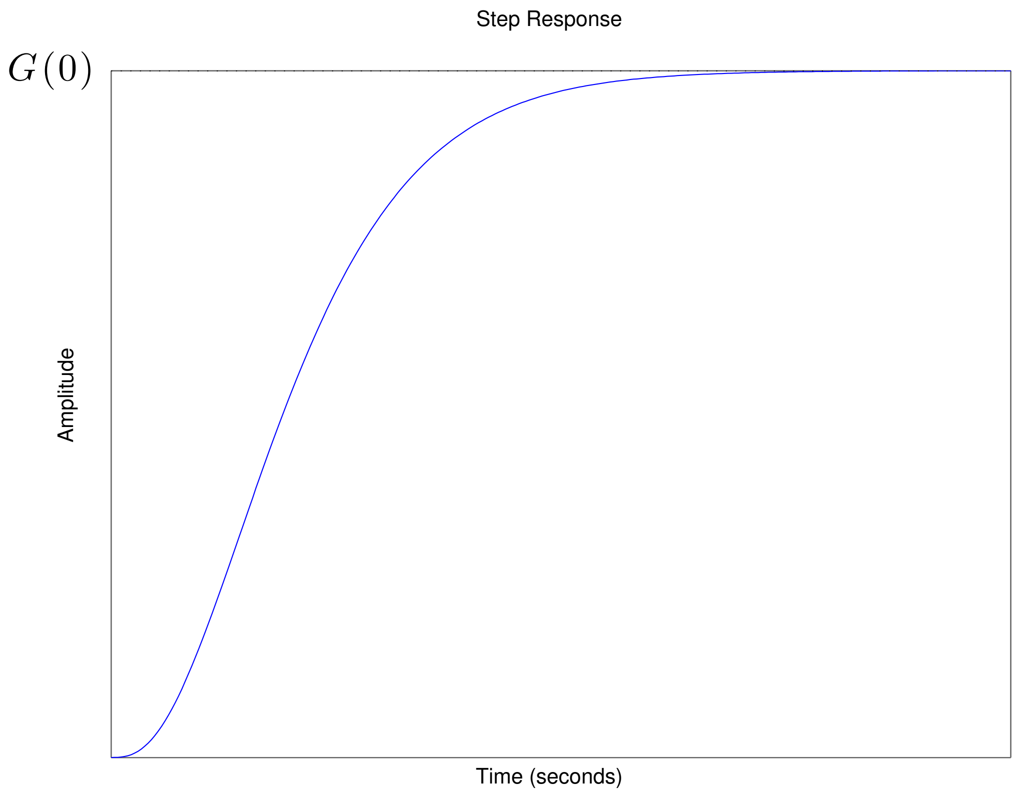

Figura 12: Step response for a process to be regulated using a PID

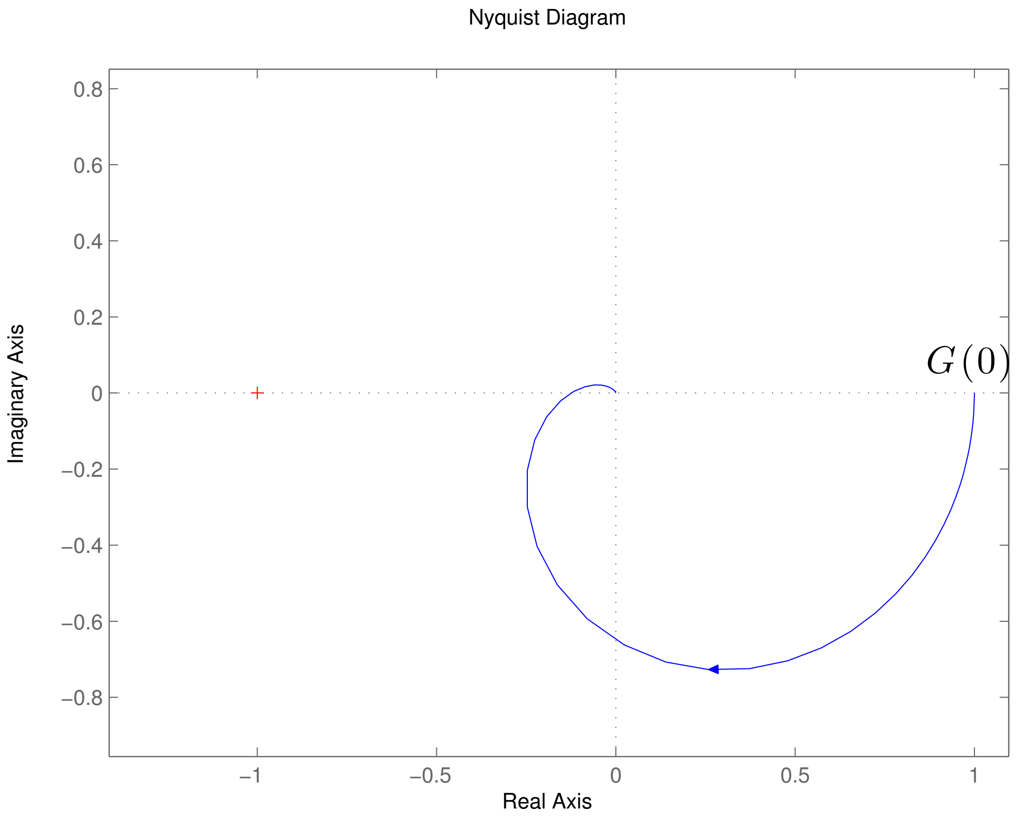

Figura 13: ...and its polar diagram (for positive frequencies)

Clearly, the steady-state value of the step response is the static gain of the system $G(0)$, which corresponds to the low-frequency point on the polar diagram of $G(s)$. Essentially, the methods described here are meaningful for types of systems that can be modeled using transfer functions with negative real poles and high relative degree. Although this may seem restrictive, it can be shown that systems with step responses like (12) are prevalent in numerous practical applications. Moreover, in the vast majority of industrial applications, the transfer function is not known, and the controller must be designed based on the experimental observation of the step response, which often resembles the graph in Figure (12). For this reason, the methods we propose are applicable starting from the experimental observation of the step response and do not require analytical information about the plant (i.e., explicit knowledge of $G(s)$ is not required).

1.4.3 Closed-Loop Method

The simplest and, in some respects, raw method is the *Ziegler-Nichols closed-loop method*. Consider the following schematic:

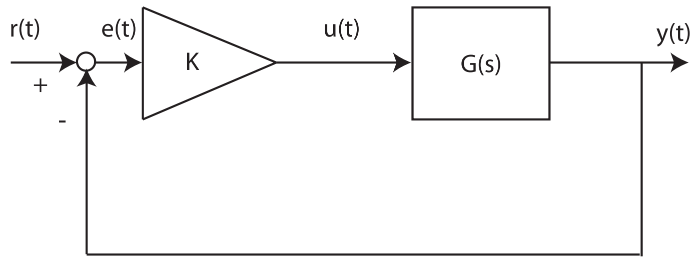

Figura 14: Ziegler-Nichols closed-loop method

Experimentally, the critical gain K_c is determined as the gain at which the polar curve $K_cG(j\omega)$ passes through the critical point $-1 + j0$ (see figure).

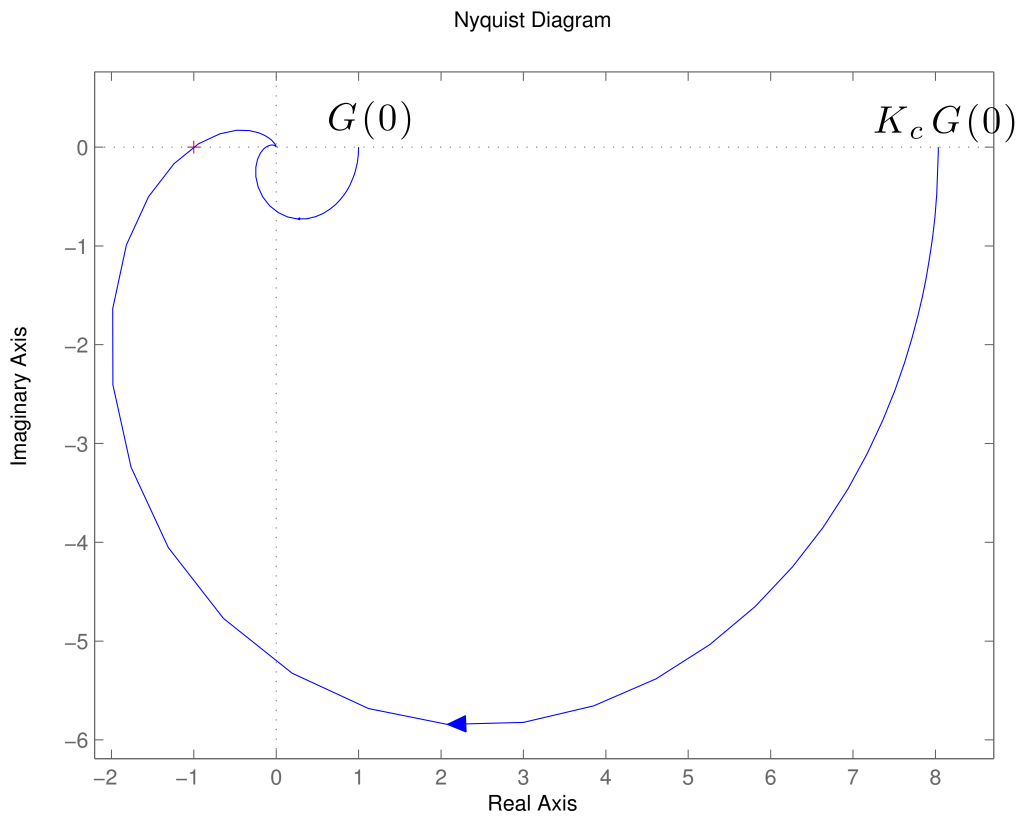

Figura 15: Ziegler-Nichols closed-loop method: polar diagram and critical gain K_c

The figure clearly shows how the critical gain K_c represents the configuration where closed-loop stability transitions to instability. Obviously, it is important to emphasize that the critical gain is precisely the amplitude margin of $G(s)$, and at K_c, the feedback system generates sustained oscillations. If ω_{cr} is the frequency of these oscillations, their period is given by:

$$T_{cr} = \frac{2\pi}{\omega_{cr}} \tag{44}$$

The fundamental parameters of this PID calibration method are therefore K_c and T_{cr}.

1.4.4 Open-Loop Method or Process Curve Method

A less schematic representation of the process is based on approximating the step response, also known as the *process curve*, with the response of a first-order model that has an initial delay τ, followed by a time interval T where the response converges exponentially to its steady-state value $G(0) = K_o$ (see figure).

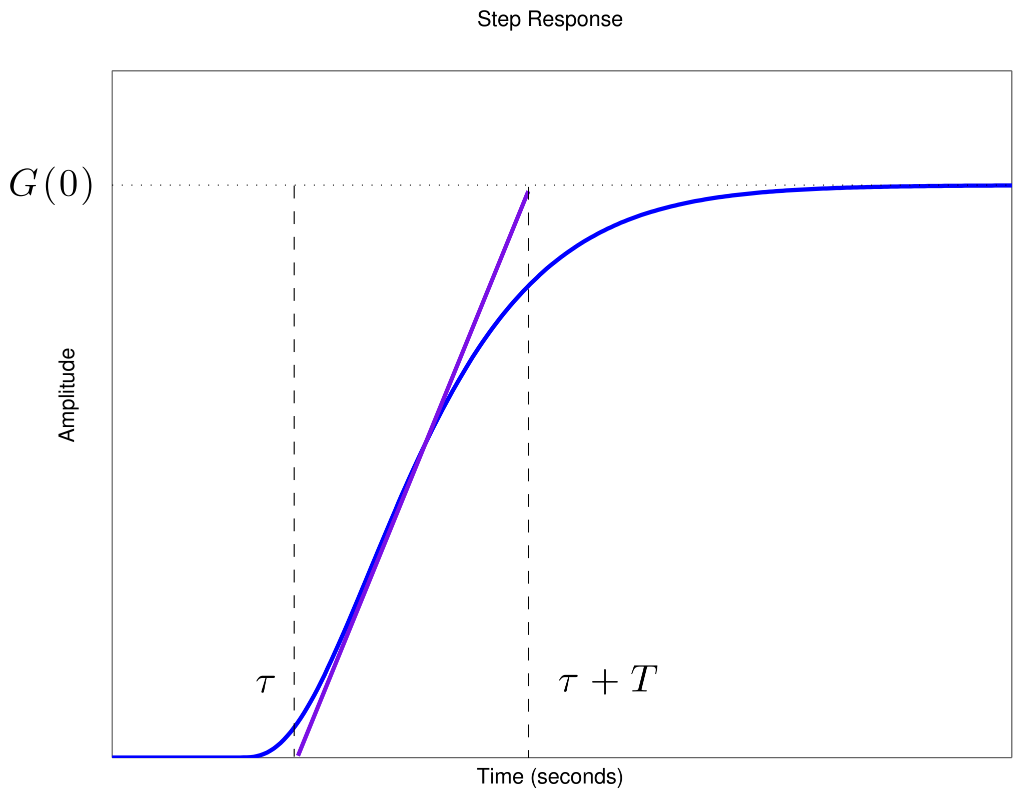

Figura 16: Process curve and first-order model with delay

We then refer to a model of the type:

$$\frac{\mu e^{-s\tau}}{1 + sT} \tag{45}$$

This is known as the *second Ziegler-Nichols method* or the *process curve method*, and the fundamental parameters of this PID calibration method are μ, τ, and T.

1.4.5 Model Approximation Methods Based on the Process Curve

Let us consider a process curve $y(t)$ and assume it is approximated by a first-order + delay model:

$$G_p(s) = \frac{\mu e^{-s\tau}}{1 + sT} \tag{46}$$

For open-loop techniques, we propose two fairly intuitive methods for estimating the model parameters μ, τ, and T from the process curve.

Tangent Method By inspection, the model gain μ can be derived as the steady-state value of the process curve. Observing the tangent line at the inflection point of the process curve, let τ and η represent the x- and y-intercepts of the tangent, respectively. The model delay is estimated as τ, and the time constant T of the pole is estimated as the time required for the tangent to intersect the steady-state value μ. A simple similarity relation leads to the conclusion:

$$T = \frac{\mu\tau}{\eta} \tag{47}$$

Area Method The *area method* is more robust and applicable even if the process curve lacks inflection points. It involves calculating the area A_1 between the steady-state value and the curve:

$$A_1 = \int_0^{+\infty} (K_o - y(t))\, dt \tag{48}$$

For a first-order model with delay, this area is equal to that of a rectangle with height K_o and base $T_1 = T + \tau$:

$$A_1 = T_1 \mu = (T + \tau)\, \mu \tag{49}$$

The base T_1 can then help calculate T and τ based on additional area calculations.

Given A_1 and μ, we can derive the sum of the model's time constant and the delay time. We now calculate the area under the process curve in the time interval $[0, T_1)$.

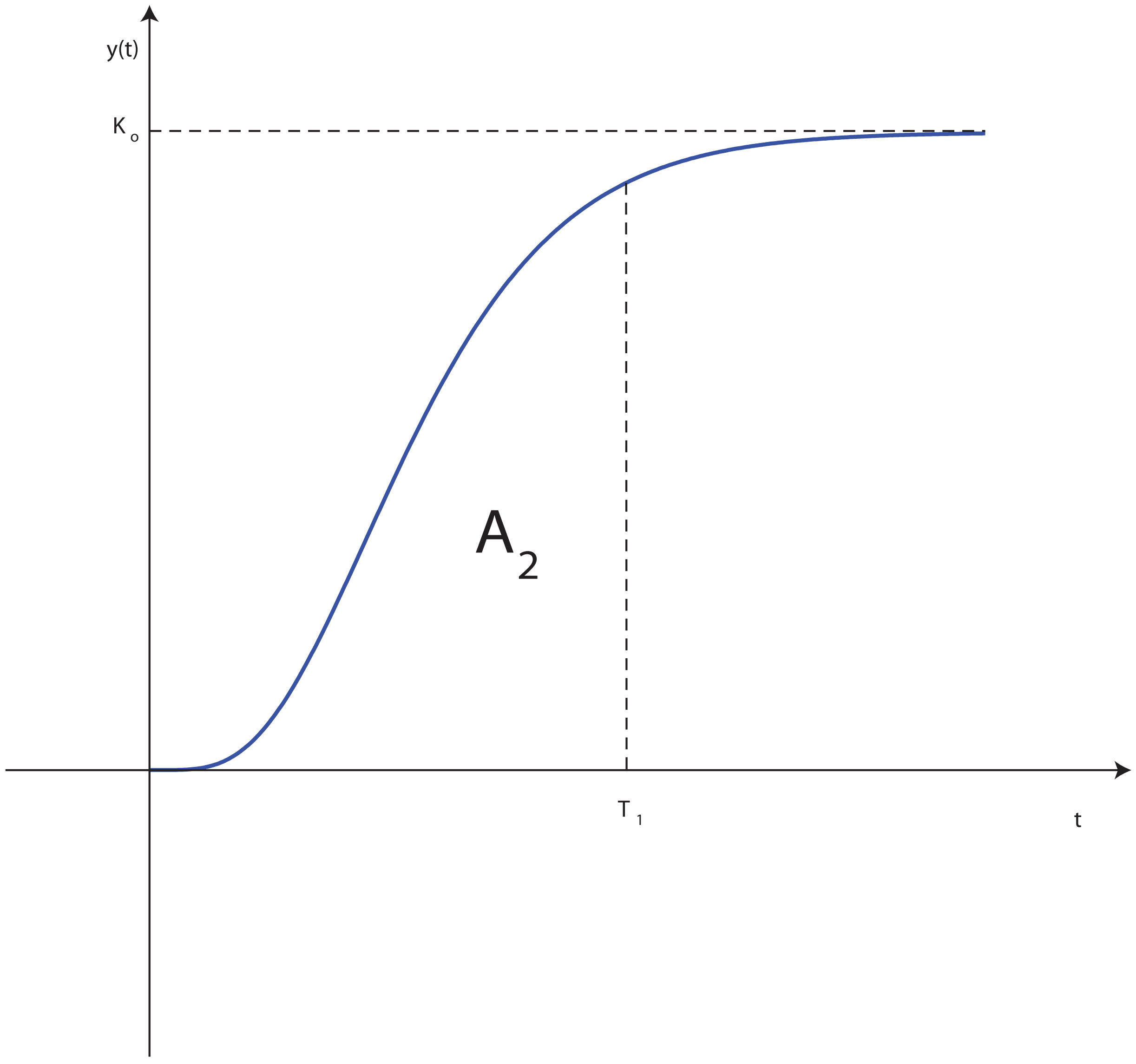

Figura 17: Process curve: area method (Area under the process curve in the interval $[0, T_1)$)

We have:

$$A_2 = \int_0^{T_1} y(t)\, dt \tag{50}$$

This area, which can be shown for a first-order model with delay, corresponds to the area of a rectangle with height μ and base $\frac{T}{e}$ (where $e = 2.718281828459046$ is Euler's number). Thus, we have:

$$A_2 = \frac{T}{e}\,\mu, \quad T = \frac{A_2\, e}{\mu} \tag{51}$$

Finally, the delay time is derived as:

$$\tau = T_1 - T \tag{52}$$

The area method, therefore, offers greater robustness and applicability compared to the tangent method.

Regarding the closed-loop tuning technique, the challenge is determining the critical gain K_c and the period of self-sustained oscillations T_{cr} without driving the feedback loop to the edge of instability, which could irreparably damage the system. One can refer to the following schematic, where the PID controller is replaced by a two-state ON-OFF relay without hysteresis, as shown in the figure:

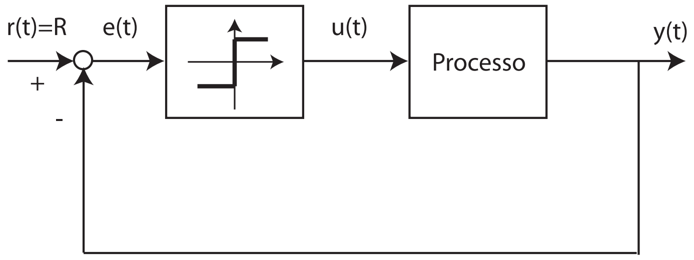

Figura 18: Ziegler-Nichols closed-loop method: schematic for determining K_c and T_{cr}

The operating logic is simple: when the error $e(t)$ (input to the relay) is positive, the system input $u(t)$ (output of the relay) switches to the high value (ON); when $e(t)$ is negative, $u(t)$ switches to the low value (OFF). The experiment is conducted by initially setting the relay output $u(t)$ to its maximum value, as shown in the figure:

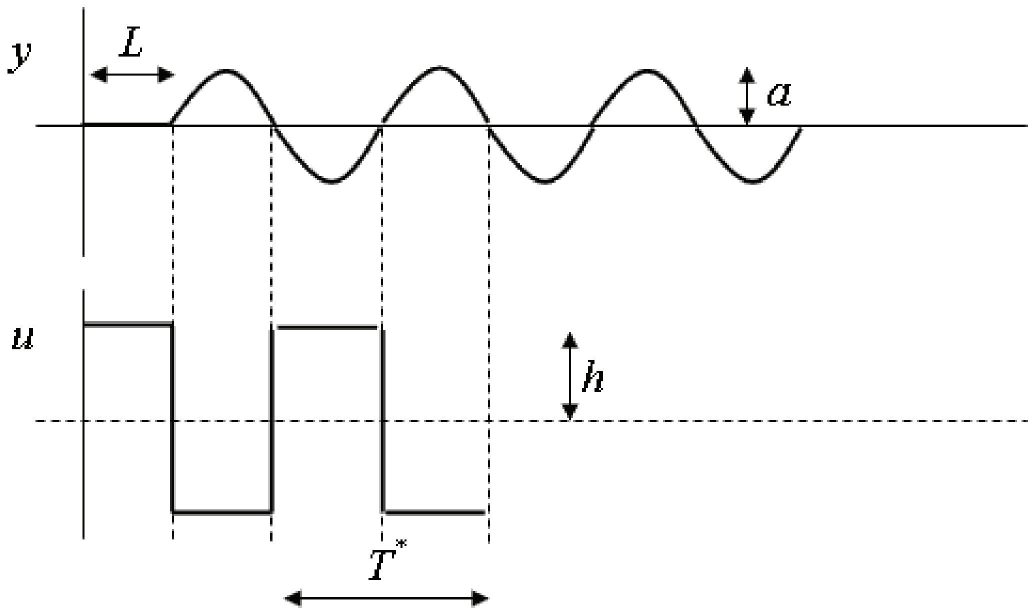

Figura 19: Ziegler-Nichols closed-loop method: behavior of $y(t)$ and $u(t)$ with relay control

The dynamics of the process filter the input signal $u(t)$. The output $y(t)$ starts rising (after a potential delay L) until $e(t)$ becomes negative, causing the relay to switch to its minimum value. After a certain time, the output $y(t)$ decreases until $e(t)$ becomes positive again, causing the relay to switch back to its maximum value. At this point, the cycle repeats, generating a controlled sustained oscillation. In other words, an oscillation is imposed on the output, assuming it matches the one at the system's stability margin. Let T^* denote the oscillation period of the square wave, h its amplitude, and a the amplitude of $y(t)$ oscillations around the bias value. Then, we have:

$$K_c = \frac{4\,h}{\pi\,a}, \quad T_{cr} = T^*, \quad \omega_{cr} = \frac{2\,\pi}{T_{cr}} \tag{53}$$

The validity of (53) can be formally proven, but due to the complexity of the necessary arguments, the proof is omitted.

1.4.6 Ziegler-Nichols Tuning Formulas

Once we have determined the process model parameters using either the closed-loop or open-loop method, we can select the PID controller parameters (K_p, T_i, T_d). The formulas we present are empirically derived to produce a step response in the feedback system optimized according to a damping ratio or *decay rate* (the ratio of the amplitudes of two successive peaks) of $\frac{1}{4}$, also known as the *quarter decay rate.*

Regarding the actual step response configuration, the closed-loop method achieves more robust results, ensuring that the controlled system's step response adheres to the *quarter decay rate* type. For the open-loop method, the actual step response approximates the *quarter decay rate* structure more closely as the process curve matches the first-order model with delay. The following table shows the Ziegler-Nichols formulas for standard P, PI, PD, and PID controllers:

	P	PI	PD	PID
K_p	$0.5\,K_c \simeq \frac{T}{\mu\,\tau}$	$0.45\,K_c \simeq \frac{0.9\,T}{\mu\,\tau}$	$0.5\,K_c \simeq \frac{1.2\,T}{\mu\,\tau}$	$0.6\,K_c \simeq \frac{1.2\,T}{\mu\,\tau}$
T_i		$0.85\,T_{cr} \simeq 3.3\,\tau$		$0.5\,T_{cr} \simeq 2\,\tau$
T_d			$0.2\,T_{cr} \simeq 0.3\,\tau$	$0.12\,T_{cr} \simeq 0.5\,\tau$
Estimated Closed-Loop Period	$T_{cr} \simeq 4\,\tau$	$1.4\,T_{cr} \simeq 5.7\,\tau$		$0.85\,T_{cr} \simeq 3.4\,\tau$

Tabella 1: Ziegler-Nichols Tuning Formulas

2 PID Discretization Algorithm

We consider a general PID controller, assuming the proportional term includes *set-point weighting* and the derivative term acts only on the regulated output to avoid the *derivative kick* phenomenon. Additionally, the derivative term is filtered by a low-pass filter to mitigate the effects of high-frequency measurement noise on the feedback loop:

$$U(s) = K_p\,\left(b\,R(s) - Y(s)\right) + \frac{K_p}{T_i\,s}\,E(s) - \frac{K_p\,T_d\,s}{1 + \frac{T_d}{N}\,s}\,Y(s) \tag{54}$$

Now we aim to discretize the relationship (54) to construct a suitable algorithm. Representing (54) in the time domain, we have:

$$u(t) = P(t) + I(t) + D(t) \tag{55}$$

Here, $P(t)$, $I(t)$, and $D(t)$ represent the proportional, integral, and derivative contributions to the control signal, respectively, corresponding to the individual terms in (54) in the Laplace transform domain. Assume that (55) is uniformly discretized, sampled at intervals t_k with $t_k = k\,h$, $k \in \mathbb{N}$, where h is the sampling time or step size of the algorithm. Thus, we have:

$$u(t_k) = P(t_k) + I(t_k) + D(t_k), \quad k \in \mathbb{N} \tag{56}$$

2.1 Proportional Contribution

The proportional contribution to the control action is straightforward to discretize:

$$P(t_k) = K_p\,\left(b\,r(t_k) - y(t_k)\right), \quad k \in \mathbb{N} \tag{57}$$

2.2 Integral Contribution

The integral contribution is characterized as follows:

$$I(t) = \frac{K_p}{T_i} \int_0^t (r(\sigma) - y(\sigma)) \, d\sigma \tag{58}$$

Or equivalently, applying the derivative operator:

$$\frac{d\,I}{d\,t} = \frac{K_p}{T_i} \, (r(t) - y(t)) \tag{59}$$

To discretize (59), focusing on the derivative, we consider three approaches:

2.2.1 Forward Differences

Assume t_k is the current time and t_{k+1} is the next time step. The derivative is approximated using forward finite differences (Euler's forward method):

$$\frac{d\,I}{d\,t} \approx \frac{I(t_{k+1}) - I(t_k)}{h} \tag{60}$$

Substituting into (59), we have:

$$\frac{I(t_{k+1}) - I(t_k)}{h} = \frac{K_p}{T_i} \, (r(t_k) - y(t_k)) \tag{61}$$

Or equivalently:

$$I(t_{k+1}) = I(t_k) + \frac{K_p\, h}{T_i} \, (r(t_k) - y(t_k)) \, , \quad k \in \mathbb{N} \tag{62}$$

2.2.2 Backward Differences

Suppose that t_k is the current time instant and t_{k-1} is the previous time instant. The derivative is approximated using the backward finite difference (backward Euler method):

$$\frac{d\,I}{d\,t} \approx \frac{I(t_k) - I(t_{k-1})}{h} \tag{63}$$

and equation (59) becomes:

$$\frac{I(t_k) - I(t_{k-1})}{h} = \frac{K_p}{T_i} \, (r(t_k) - y(t_k)) \tag{64}$$

or equivalently:

$$I(t_k) = I(t_{k-1}) + \frac{K_p\, h}{T_i} \, (r(t_k) - y(t_k)) \, , \quad k \in \mathbb{N} \tag{65}$$

If we want to align (65) with (62), we substitute t_{k-1} with t_k and t_k with t_{k+1}. Then equation (65) becomes:

$$I(t_{k+1}) = I(t_k) + \frac{K_p\, h}{T_i} \, (r(t_{k+1}) - y(t_{k+1})) \, , \quad k \in \mathbb{N} \tag{66}$$

2.2.3 Tustin's Method

Suppose that t_k is the current time instant and t_{k+1} is the next time instant. The derivative is approximated using the forward finite difference (forward Euler method):

$$\frac{d\,I}{d\,t} \approx \frac{I(t_{k+1}) - I(t_k)}{h} \tag{67}$$

and the term:

$$\frac{K_p}{T_i} \, (r(t) - y(t)) \tag{68}$$

is approximated by the average of the value at the next time instant and the value at the current time instant:

$$\frac{K_p}{T_i} \frac{1}{2} \left((r(t_{k+1}) - y(t_{k+1})) + (r(t_k) - y(t_k)) \right) \tag{69}$$

Substituting this into (59), we obtain:

$$\frac{I(t_{k+1}) - I(t_k)}{h} = \frac{K_p}{T_i} \frac{1}{2} \left((r(t_{k+1}) - y(t_{k+1})) + (r(t_k) - y(t_k)) \right) \tag{70}$$

or equivalently:

$$I(t_{k+1}) = I(t_k) + \frac{K_p\, h}{T_i} \frac{1}{2} \left((r(t_{k+1}) - y(t_{k+1})) + (r(t_k) - y(t_k)) \right), \quad k \in \mathbb{N} \tag{71}$$

2.2.4 Discretization Methods and Area Calculation

The three discretization algorithms define three different ways to approximate the integral of the tracking error

$$\int_0^t (r(\sigma) - y(\sigma))\, d\sigma \tag{72}$$

on the action $I(t)$. Let us now consider (72) over a single sampling interval $[t_k,\ t_{k+1})$, resulting in

$$\int_{t_k}^{t_{k+1}} (r(\sigma) - y(\sigma))\, d\sigma \tag{73}$$

Thus, the contribution in (72) is approximated by the finite sum over k of all the individual segments in (73)

$$\sum_k \int_{t_k}^{t_{k+1}} (r(\sigma) - y(\sigma))\, d\sigma \tag{74}$$

Now consider the three discretization schemes discussed

$$I(t_{k+1}) = I(t_k) + \frac{K_p}{T_i}\, h\, (r(t_k) - y(t_k)), \quad k \in \mathbb{N} \tag{75}$$

$$I(t_{k+1}) = I(t_k) + \frac{K_p}{T_i}\, h\, (r(t_{k+1}) - y(t_{k+1})), \quad k \in \mathbb{N} \tag{76}$$

$$I(t_{k+1}) = I(t_k) + \frac{K_p}{T_i}\, h\, \frac{1}{2} \left((r(t_{k+1}) - y(t_{k+1})) + (r(t_k) - y(t_k)) \right), \quad k \in \mathbb{N} \tag{77}$$

and observe that

$$h\, (r(t_k) - y(t_k)) \tag{78}$$

$$h\, (r(t_{k+1}) - y(t_{k+1})) \tag{79}$$

$$h\, \frac{1}{2} \left((r(t_{k+1}) - y(t_{k+1})) + (r(t_k) - y(t_k)) \right) \tag{80}$$

represent three areas. The first, (78), derived from forward discretization, is the area of a rectangle with base $(r(t_k) - y(t_k))$ (current tracking error) and height h (sampling step), as shown in the figure below:

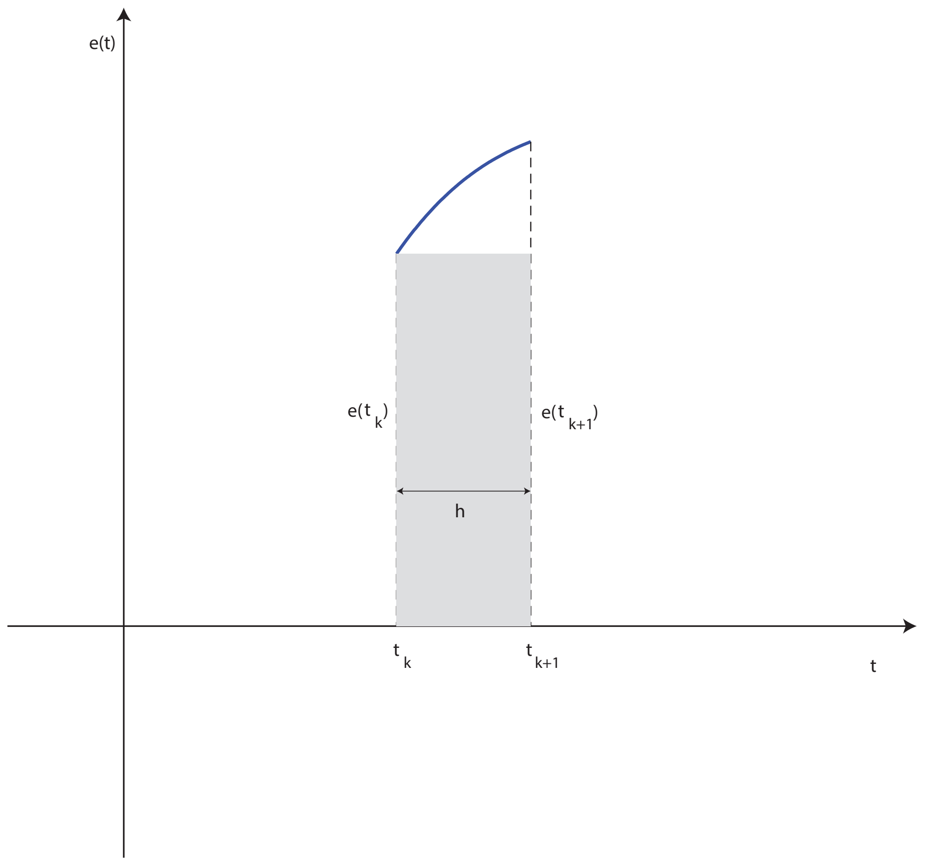

Figura 20: Area under the tracking error in a sampling interval, approximation using forward differences.

The second, (79), derived from backward discretization, is the area of a rectangle with base $(r(t_{k+1}) - y(t_{k+1}))$ (next tracking error) and height h (sampling step), as shown in the following figure:

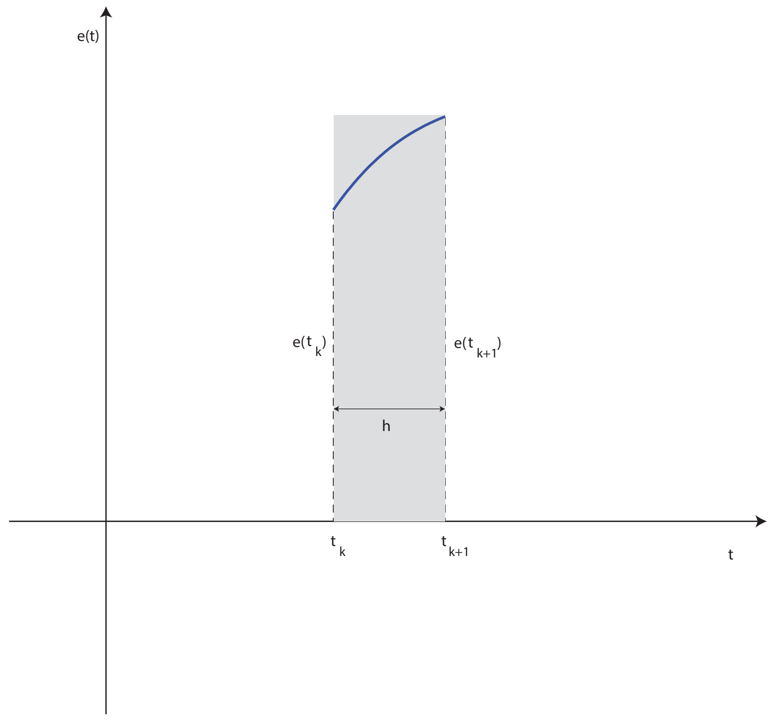

Figura 21: Area under the tracking error in a sampling interval, approximation using backward differences.

The third, (80), derived from the Tustin method, is the area of a trapezoid with bases $(r(t_{k+1}) - y(t_{k+1}))$ (next tracking error) and $(r(t_k) - y(t_k))$ (current tracking error), and height h (sampling step), as shown in the figure below:

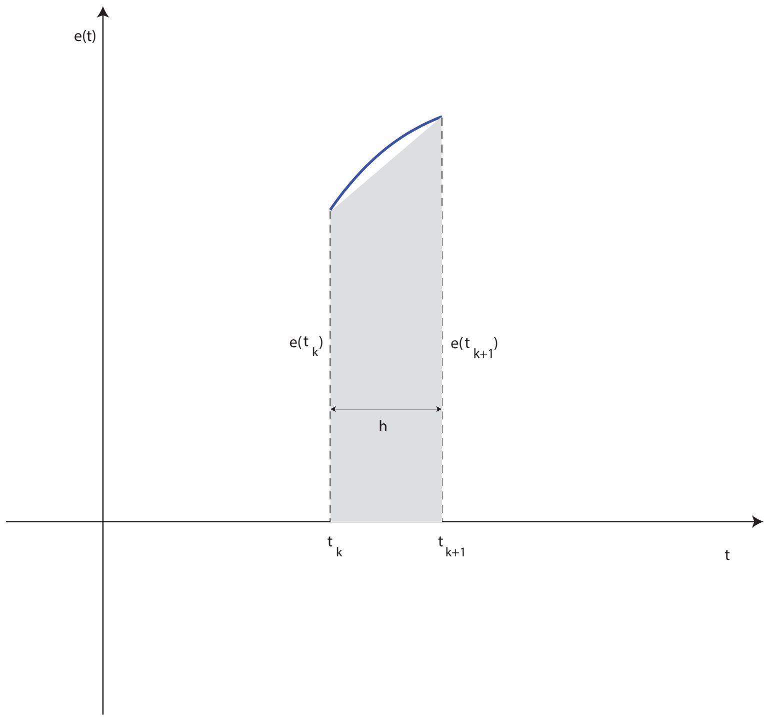

Figura 22: Area under the tracking error in a sampling interval, approximation using the Tustin method.

Due to this specific type of approximation, the Tustin method is also known as the Trapezoidal Method or Trapezoid Rule.

2.2.5 Discretization of the Derivative Contribution

The input-output relationship for the derivative contribution, considering that this contribution is designed to avoid the phenomenon of derivative kick and the effect of measurement noise, is given by

$$D(s) = -\frac{K_p\, T_d\, s}{\frac{T_d}{N}\, s + 1}\, Y(s) \tag{81}$$

which in the time domain corresponds to the following differential equation:

$$\frac{T_d}{N}\,\frac{d\,D}{d\,t} + D(t) = -K_p\, T_d\, \frac{d\,y}{d\,t} \tag{82}$$

Now we discretize the differential equation using the forward difference method, yielding:

$$\frac{T_d}{N}\,\frac{D(t_{k+1}) - D(t_k)}{h} + D(t_k) = -K_p\, T_d\, \frac{y(t_{k+1}) - y(t_k)}{h} \tag{83}$$

If we explicitly solve equation (83) for $D(t_{k+1})$ and move all other terms to the second side, we have:

$$D(t_{k+1}) = \left(1 - \frac{N\,h}{T_d}\right) D(t_k) - K_p\, N\, \left(y(t_{k+1}) - y(t_k)\right) \tag{84}$$

This represents the discretized algorithm for the derivative action using the forward difference method. It is worth noting that this scheme may not always be numerically stable. An unsuitable choice of algorithm parameters N, h, T_d may lead (not always, but potentially) to the following condition:

$$1 - \frac{N\,h}{T_d} > 1 \tag{85}$$

causing equation (84) to lose stability. Instead, if we use the backward difference method, we have:

$$\frac{T_d}{N}\,\frac{D(t_k)-D(t_{k-1})}{h}+D(t_k)=-K_p\,T_d\,\frac{y(t_k)-y(t_{k-1})}{h} \tag{86}$$

Solving equation (86) for $D(t_k)$ and rearranging, we get:

$$D(t_k)=\frac{T_d}{N\,h+T_d}\,D(t_{k-1})-\frac{K_p\,T_d\,N}{N\,h+T_d}\,\left(y(t_k)-y(t_{k-1})\right) \tag{87}$$

If we replace t_k with t_{k+1} and t_{k-1} with t_k, we obtain the discretization scheme for the derivative action using the backward difference method:

$$D(t_{k+1})=\frac{T_d}{N\,h+T_d}\,D(t_k)-\frac{K_p\,T_d\,N}{N\,h+T_d}\,\left(y(t_{k+1})-y(t_k)\right) \tag{88}$$

Note that equation (88) is always stable because the coefficient:

$$\frac{T_d}{N\,h+T_d} \tag{89}$$

is always less than unity. Using the Tustin (or trapezoidal) method gives:

$$\frac{T_d}{N}\,\frac{D(t_{k+1})-D(t_k)}{h}+\frac{1}{2}\left(D(t_{k+1})+D(t_k)\right)=-K_p\,T_d\,\frac{y(t_{k+1})-y(t_k)}{h} \tag{90}$$

By explicitly solving equation (90) for $D(t_{k+1})$, we get:

$$D(t_{k+1})=\frac{T_d-2\,N\,h}{T_d+2\,N\,h}\,D(t_k)-\frac{2\,K_p\,T_d\,N}{T_d+2\,N\,h}\,\left(y(t_{k+1})-y(t_k)\right) \tag{91}$$

This is the discretization scheme for the derivative action using the trapezoidal method. Note that equation (91) is always stable because:

$$-1<\frac{T_d-2\,N\,h}{T_d+2\,N\,h}<1 \tag{92}$$

In summary, the PID algorithm has a structure of the form:

$$u(t_k)=P(t_k)+I(t_k)+D(t_k) \tag{93}$$

where $P(t_k)=K_p\,(b\,r(t_k)-y(t_k))$, and the integral and derivative contributions are derived using the following recurrence equations:

$$I(t_{k+1})=I(t_k)+\alpha_1\,\left(r(t_{k+1})-y(t_{k+1})\right)+\alpha_2\,\left(r(t_k)-y(t_k)\right) \tag{94}$$

$$D(t_{k+1})=\beta_1\,D(t_k)+\beta_2\,\left(y(t_{k+1})-y(t_k)\right) \tag{95}$$

where the coefficients α_1, α_2, β_1, β_2 vary depending on the chosen discretization algorithm, as shown in the following table:

Algorithm	α_1	α_2	β_1	β_2
Forward Euler	0	$\frac{K_p\,h}{T_i}$	$1-\frac{N\,h}{T_d}$	$-K_p\,N$
Backward Euler	$\frac{K_p\,h}{T_i}$	0	$\frac{T_d}{N\,h+T_d}$	$-\frac{K_p\,T_d\,N}{N\,h+T_d}$
Tustin	$\frac{K_p\,h}{2\,T_i}$	$\frac{K_p\,h}{2\,T_i}$	$\frac{T_d-2\,N\,h}{T_d+2\,N\,h}$	$-\frac{2\,K_p\,T_d\,N}{T_d+2\,N\,h}$

3 Overview of the Saturation Effect of the Integral Action

The presence of an integral effect in a standard controller (or in a controller derived from a cascade composition of standard corrective networks) has the drawback that, if the tracking error is constant, the control signal (due to the integrator component) experiences a constant drift (it behaves as a ramp) and thus grows indefinitely in magnitude. However, actuators exhibit inevitable saturation effects, meaning they cannot "accept" excessively large control signals as input (e.g., a valve in a tap or an electric motor). Therefore, it is necessary to prevent the saturation of the control action, as prolonged saturation could result in performance degradation and even *feedback detachment.* In such cases, the plant would continuously receive a constant signal at its input, which is the output of the saturator block. This phenomenon is known in the literature as *integral wind-up*, and various methods exist to compensate for it. Here, we introduce a simple scheme known as *reset anti-wind-up.* The desaturation scheme is illustrated in the following figure:

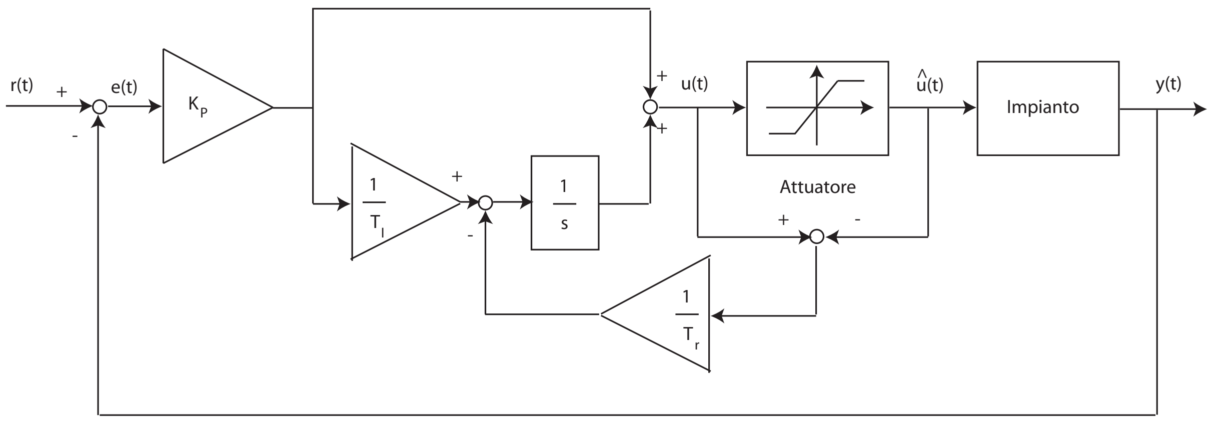

Figura 23: Simple desaturation scheme for the integral action in a PI controller

We provide an intuitive explanation of the figure without delving too deeply into details. We observe that $u(t)$ is the control signal generated by the controller, and $\hat{u}(t)$ is the signal processed by the actuator, which follows the input-output relation or *characteristic*:

$$\hat{u}(t) = \begin{cases} u(t) & \text{if} \quad |u(t)| \leq u_{\text{MAX}} \\ u_{\text{MAX}} & \text{if} \quad u(t) > u_{\text{MAX}} \\ -u_{\text{MAX}} & \text{if} \quad u(t) < -u_{\text{MAX}} \end{cases} \tag{96}$$

Considering the scheme in the figure, the control action is expressed as:

$$u(t) = K_p\left(r(t) - y(t)\right) + \frac{K_p}{T_i}\int_0^t \left(r(\sigma) - y(\sigma)\right)\, d\sigma - \frac{1}{T_r}\int_0^t \left(u(\sigma) - \hat{u}(\sigma)\right)\, d\sigma \tag{97}$$

The third term is a corrective component, also known as the desaturation or anti-reset term, which is transparent (equal to zero) when $u(t) = \hat{u}(t)$, i.e., when the actuator operates in the linear region. However, when the actuator enters saturation, a discrepancy arises between $u(t)$ and $\hat{u}(t)$, making the third term in (97) non-zero. Note that this desaturation term has an opposite sign to the integral contribution and is modulated by the so-called reset time T_r, which determines the magnitude of the desaturation effect (smaller T_r results in a more effective desaturation action, and vice versa). The goal of the third term is clear: to the integrator (more or less quickly depending on the value of the reset time).

Of course, the trade-off when using this anti-reset scheme is that the performance of the controlled system will degrade, with the degradation increasing as T_r decreases (faster integrator numerical discharge), which reduces the quality of the adder's memory $\frac{1}{s}$.