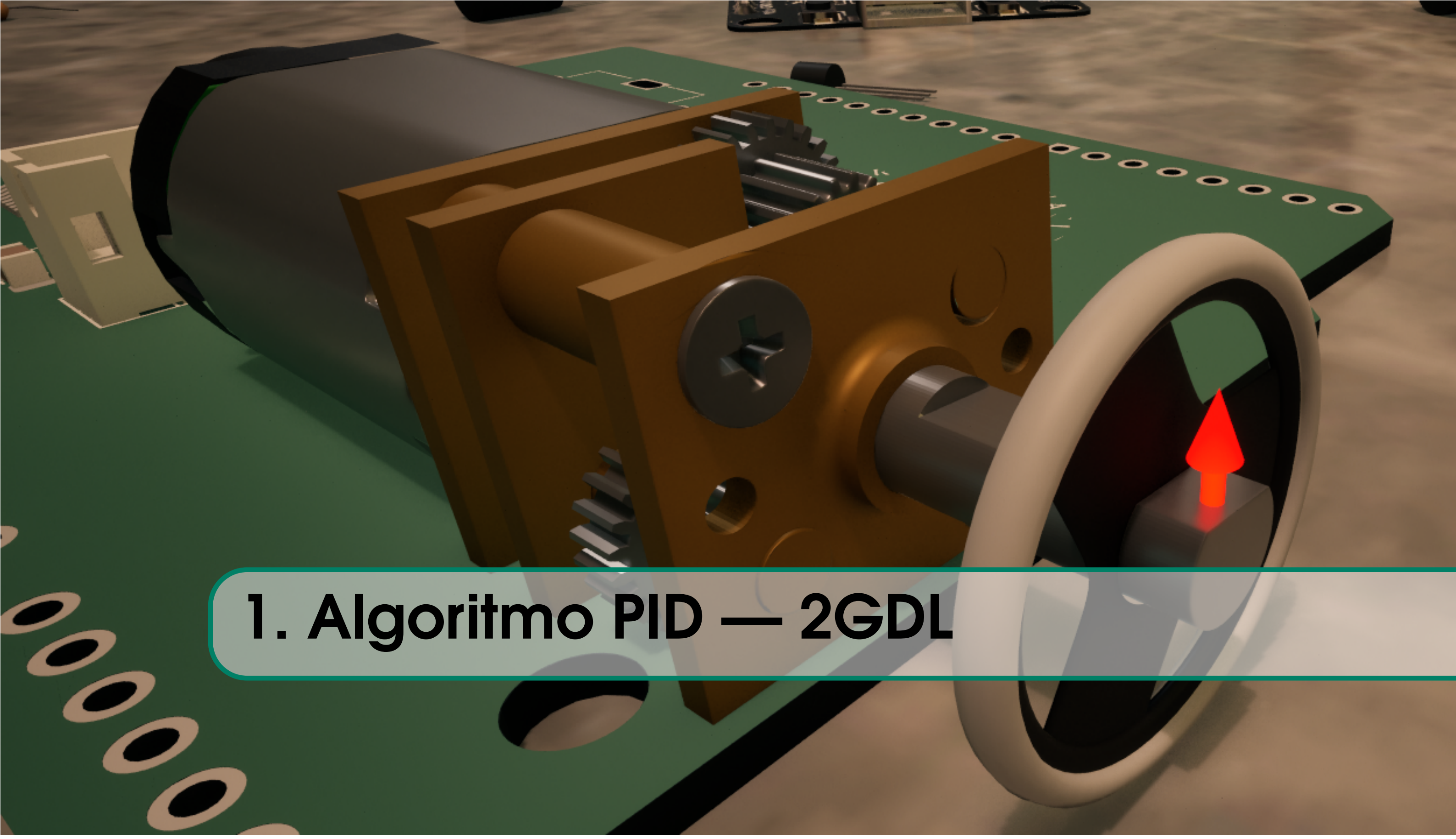


1. Algoritmo PID — 2GDL

1.1 Teoría básica

El controlador PID (Proporcional-Integral-Derivativo) es el control realimentado más usado en los sistemas de ingeniería [1]. Este controlador consta de tres acciones fundamentales: la acción proporcional (P), la cual depende del error presente; la acción integral (I), que depende de los errores en el pasado; y, la acción derivativa (D) que depende de la predicción del error en el sistema. Las acciones integral y derivativa son importantes porque la primera permite eliminar el error de estado estacionario y la segunda permite tener una acción predictiva o anticipativa, permitiendo que un sistema sea más estable.

La acción de control en un controlador PID combina las tres acciones así:

$$u = k_p e + k_i \int_0^t e(\tau)d\tau + k_d \frac{de}{dt} = k_p \left(e + \frac{1}{T_i} \int_0^t e(\tau)d\tau + T_d \frac{de}{dt} \right). \tag{1.1}$$

en que:

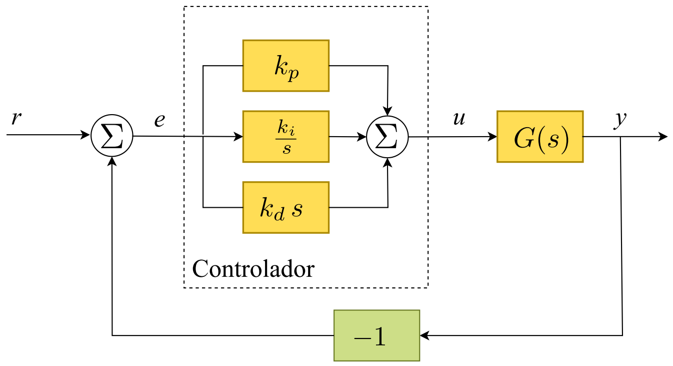
k_p : Constante proporcional

k_i : Constante integral

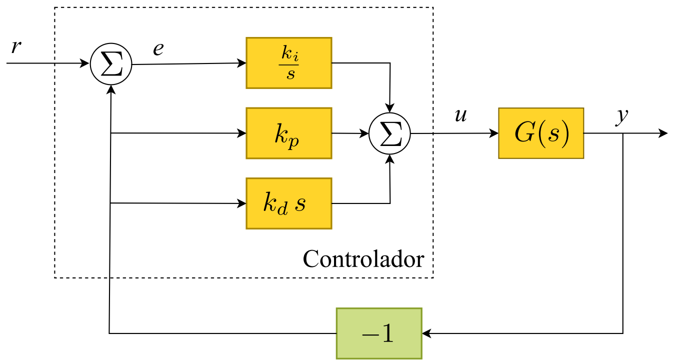
k_d : Constante derivativa

$T_i = kp/k_i$: Constante de tiempo integral,

$T_d = k_d/k_p$: Constante de tiempo derivativa.



(a) PID de un 1-GDL.



(b) PID de 2-GDL.

Figura 1.1: Diagrama de bloques de sistemas en lazo cerrado con controladores PID de uno (a) y dos grados de libertad (b) [1].

En la figura 1.1 se muestra el diagrama de lazo cerrado de un controlador PID de un grado de libertad (1-GDL) y en la figura 1.1b el diagrama de un PID de dos grados de libertad (2-GDL). El controlador PID de 1-GDL tiene una sola entrada e y las tres acciones de control (proporcional, integral y derivativa) actúan sobre el error. El controlador PID de 2-GDL tiene dos entradas, r y y. En este controlador la acción integral actúa sobre el error, mientras que la acción proporcional y la derivativa actúan sobre la salida y.

Comentario 1.1.1 — ¿Cuál configuración es más usada?. La configuración más usada para la implementación de un controlador PID es 2-GDL, puesto que la acción derivativa actúa sobre la salida que es una señal continua y no sobre el error que es una señal discontinua. Las discontinuidades del error hacen que la señal de control tenga cambios muy fuertes al usar un PID de 1-GDL, produciendo, así, un esfuerzo severo en el actuador.

1.2 Implementación computacional del controlador PID-2DOF

En esta sección resumimos brevemente el proceso de implementación computacional de un controlador PID de dos grados de libertad, adaptado de la presentación del libro de

Åström y Murray [2, págs. 11-21—11-23].

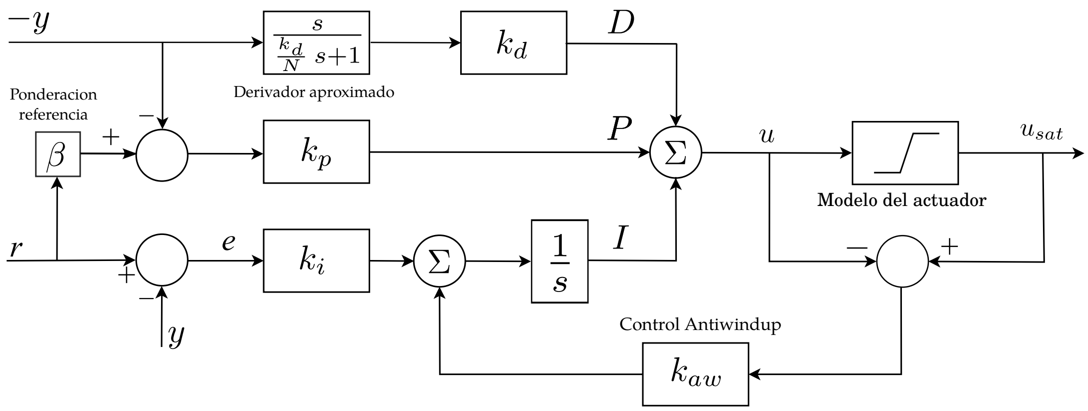


Figura 1.2: Controlador PID de dos grados de libertad con filtro y control *anti-windup*.

Para la implementación del controlador se considera el controlador PID con derivada filtrada, ponderación de la referencia y control *anti-windup*, conforme se ilustra en la figura 1.2. La señal u_a corresponde a la suma de los términos proporcional, integral y derivativo.

La salida del algoritmo de control que debe enviarse al sistema controlado es

$$u_{sat} = sat(u). \tag{1.2}$$

En que la función $sat(u)$ modela los límites energéticos del actuador y se define así:

$$u_{sat} = \begin{cases} u_{max}, & \text{si } u > u_{max} \\ u_{min}, & \text{si } u < u_{min} \\ u, & \text{en otro caso.} \end{cases} \tag{1.3}$$

Implementación digital

El controlador de tiempo continuo ilustrado en la figura 1.2 requiere ser discretizado para su implementación computacional. Supongamos que la discretización se realiza con un periodo de muestreo h, dado en segundos.

El término proporcional $P = k_p(\beta r - y)$ se implementa simplemente reemplazando las variables continuas por las muestras tomadas en el tiempo actual t_k:

$$P_(t_k) = k_p(\beta r(t_k) - y(t_k)). \tag{1.4}$$

El error actual en el momento actual t_k se calcula como:

$$e(t_k) = r(t_k) - y(t_k). \tag{1.5}$$

El término integral se obtiene aproximando la integral por la siguiente ecuación incremental de estado:

$$I(t_{k+1}) = I_(t_k) + k_i\, h\, e(t_k) + \frac{h}{T_{aw}}\left(u_{sat} - u\right)), \tag{1.6}$$

en que $T_{aw} = h/k_{aw}$ es el término *anti-windup*.

Definiendo $b_i = k_i/h$ y $b_r = h/T_{aw}$ en la ecuación (1.6), obtenemos:

$$I(t_{k+1}) = I_(t_k) + b_i e(t_k) + b_r\left(u_{sat} - u\right)), \tag{1.7}$$

A su vez, el término que produce la derivada filtrada está dado por la siguiente ecuación diferencial:

$$\frac{k_d}{N}\frac{dD}{dt} + D = -k_d \dot{y}.$$

Aproximando la derivada en esta ultima ecuación por el método de Euler, obtenemos la siguiente ecuación en diferencias:

$$\frac{k_d}{N}\frac{D(t_k) - D(t_{k-1})}{h} + D(t_k) = -k_d \frac{y(t_k) - y(t_{k-1})}{h}. \tag{1.8}$$

La ecuación (1.8) puede ser reescrita como:

$$D(t_k) = \frac{k_d}{k_d + N\,h} D(t_k - 1) - \frac{k_d\,N}{k_d + N\,h}(y(t_k) - y_{k-1}), \tag{1.9}$$

Definiendo $a_d = k_d/(k_d + N\,h)$ y $b_d = k_d N/(k_d + Nh)$ en la ecuación (1.9), obtenemos la siguiente ecuación en diferencias:

$$D_k = a_d D_{k-1} - b_d(y_k - y_{k-1}). \tag{1.10}$$

Las ecuaciones (1.4),(1.7) y (1.10) se utilizan en el siguiente pseudo-código para implementar un controlador PID de 2-GDL con filtrado de la derivada y control.

```
// define structural parameters
h =
umin =
umax =
deadzone =
// probably others depending on your system


```

```
// define pid paramaters
kp =
ki =
kd =

// define other pid parameters
N =
beta =

//  Compute controller coefficients before control loop
//  if you are not performing self tuning
bi = ki*h
ad = kd/ (kd + N*h)
bd = kd*N/(kd + N*h)
br = h/Taw

// Control algorithm - critical repeating task
for (;;) {
  r = read_reference()    // read setpoint
  y = read_sensor()    // read process variable
  P = kp*(beta*r - y)    // compute proportional part
  D = ad*D - bd*(y-y_ant)    // compute derivative part
  u = P + I + D    // compute temporary output

  //compensateDeadZone(u) if necessary

  usat = sat(u, ulow, uhigh)    // compute actuator saturation
  send_analog_output(usat)    // set analog output
  I = I + bi*(r-y) + br*(usat-u)    // update integral state
  y_ant = y    // update derivative state


  ...Repeat this task when the sampling time has elapsed.

}
```

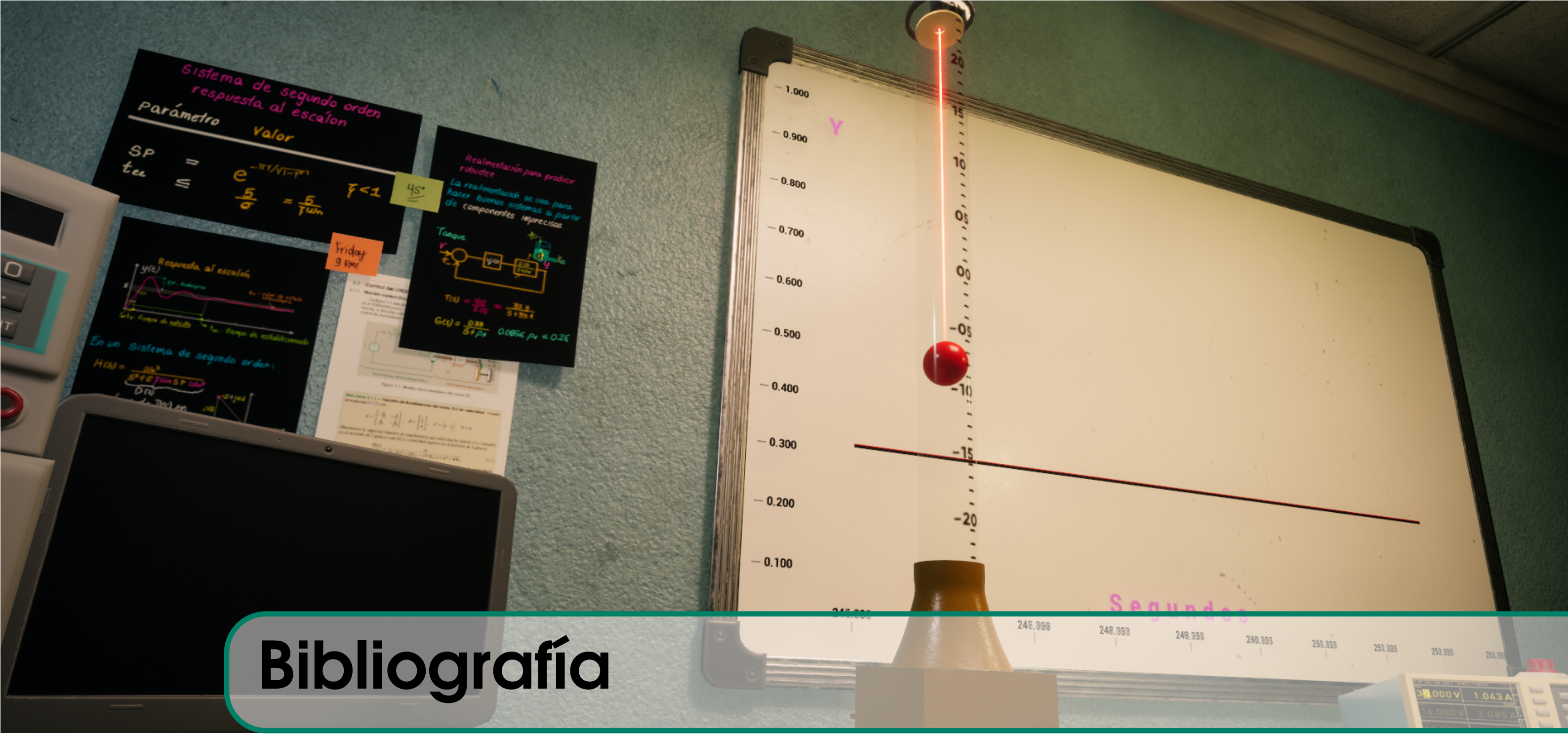
1.3 Notas de implementación

El factor N que aparece en la ecuación (1.9) determina el ancho de banda del derivador aproximado usado en el PID. Normalmente se usa un valor entre 5 y 20. Note que un valor más alto de N implica que el derivador tiene más ancho de banda, se parece más a un derivador ideal y, consecuentemente, tiene un mayor efecto estabilizante, pero también *es más susceptible al ruido*. Así que su valor debe ser ajustado con cuidado y depende del ruido presente en el sistema de control.

El valor de la constante de *anti-windup* se puede elegir como:

$$T_{aw} = 0.9\,h.$$

Siendo h el tiempo de muestreo elegido para el sistema.



Bibliografía

Libros

[ÅH06] K.J. Åström y T. Hägglund. *Advanced PID Control*. ISA-The Instrumentation, Systems, y Automation Society, 2006. ISBN: 9781556179426. URL: https://books.google.com.co/books?id=XcseAQAAIAAJ (véanse páginas 1, 2).

[ÅM21] Karl Johan Åström y Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. 2.ª edición. Princeton University Press, 2021. ISBN: 978-0-691-19398-4. URL: http://press.princeton.edu/ (véase página 3).

Articulos

Online