# Implementación de Controladores Discretos

## CONTROL POR COMPUTADOR

**Grado en Electrónica, Robótica y Mecatrónica**

# Introducción

- Una vez sintetizado el controlador, su programación requiere tener cuidado con ciertos detalles para evitar problemas de fragilidad.

- Además, se pueden incluir mejoras adicionales, como saturaciones o mecanismos de *anti-windup*.

# Fragilidad

$$C(z)=\frac{N(z)}{(z-1)(z-0.9751)(z-0.8425)(z-0.7897)(z-0.6421)(z-0.5374)(z-0.3144)(z-0.2678)}$$

$$\Downarrow$$

$$C(z)=\frac{N(z)}{z^8\text{-}5.3690\,z^7+12.3351\,z^6\text{-}15.8007\,z^5+12.3079\,z^4\text{-}5.9506\,z^3+1.7376\,z^2\text{-}0.2792\,z+0.0188}$$

$$\Downarrow$$

$$\begin{aligned}u_k &= 5.3690\,u_{k-1}-12.3351\,u_{k-2}+15.8007\,u_{k-3}-12.3079\,u_{k-4}+5.9506\,u_{k-5}\\ &\quad -1.7376\,u_{k-6}+0.2792\,u_{k-7}-0.0188\,u_{k-8}+\ldots\end{aligned}$$

**¡ATENCIÓN!**

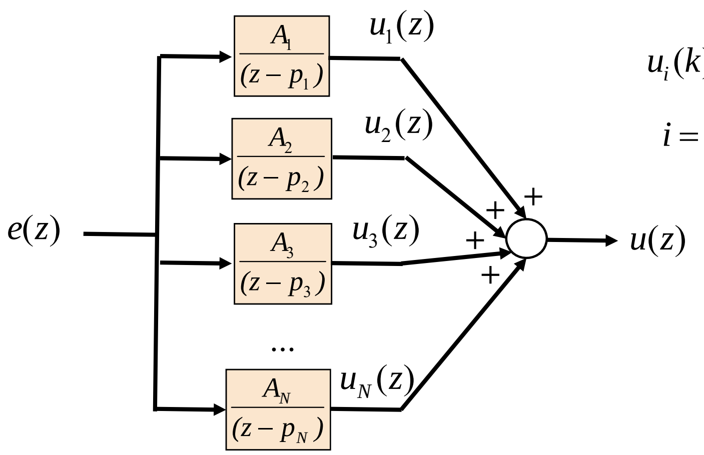
$$¿\,polos\left(z^8\text{-}5.3690\,z^7+12.3351\,z^6\text{-}15.8007\,z^5+12.3079\,z^4\text{-}5.9506\,z^3+1.7376\,z^2\text{-}0.2792\,z+0.0188\right)?$$



# Implementación de controladores

$$\boxed{C(z)=\frac{K_c\,(z-c_1)(z-c_2)(z-c_3)..(z-c_N)}{(z-p_1)(z-p_2)(z-p_3)..(z-p_N)}}$$

- **Paralelo:** $C(z)=\frac{A_1}{(z-p_1)}+\frac{A_2}{(z-p_2)}+\frac{A_3}{(z-p_3)}+\ldots+\frac{A_N}{(z-p_N)}$




$$u_i(k)=p_i\;u_i(k-1)+A_i\;e_i(k-1)$$

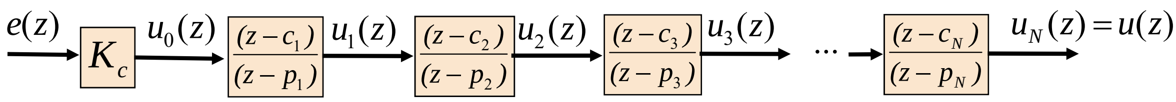
$$i=1,\ldots,N$$

# Implementación de controladores

$$\boxed{C(z)=\frac{K_c\,(z-c_1)(z-c_2)(z-c_3)\ldots(z-c_N)}{(z-p_1)(z-p_2)(z-p_3)\ldots(z-p_N)}}$$

- **Serie:**

$$C(z)=K_c\frac{(z-c_1)}{(z-p_1)}\frac{(z-c_2)}{(z-p_2)}\frac{(z-c_3)}{(z-p_3)}\cdots\frac{(z-c_N)}{(z-p_N)}$$



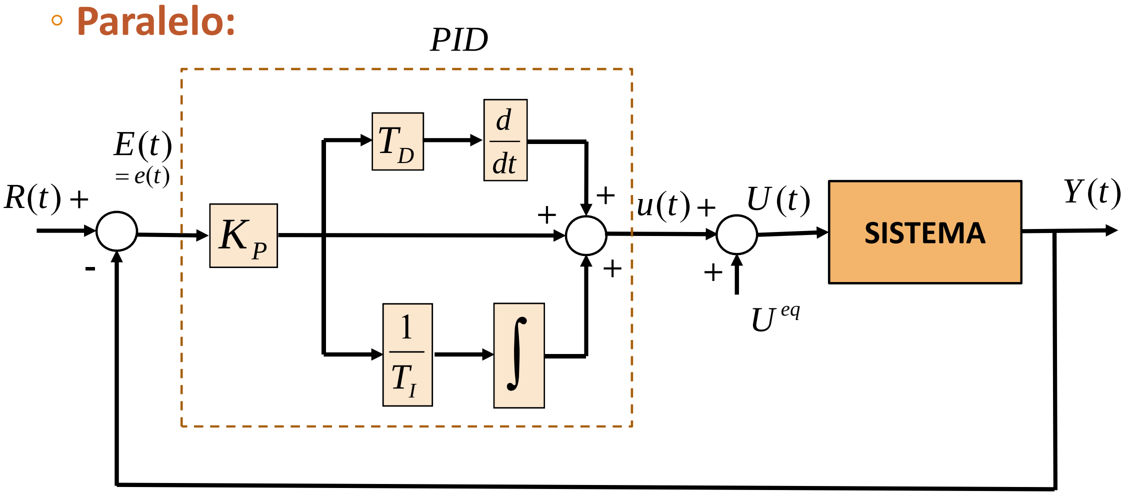

$$u_i(k)=p_i\;u_i(k-1)+u_{i-1}(k)-c_i\;u_{i-1}(k-1)$$

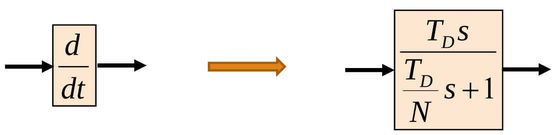
$$i=1,\ldots,N$$



# Implementación de PIDs

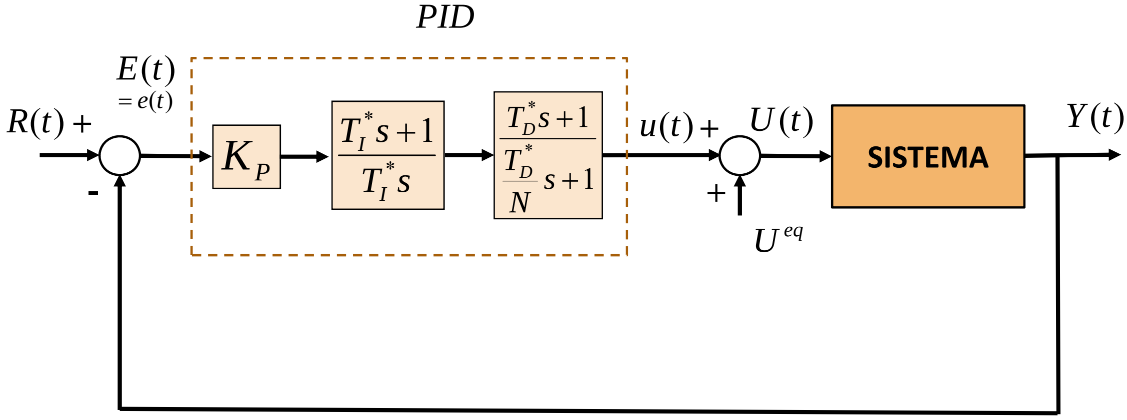
- **Paralelo:**



# Implementación de PIDs

- **Serie:**



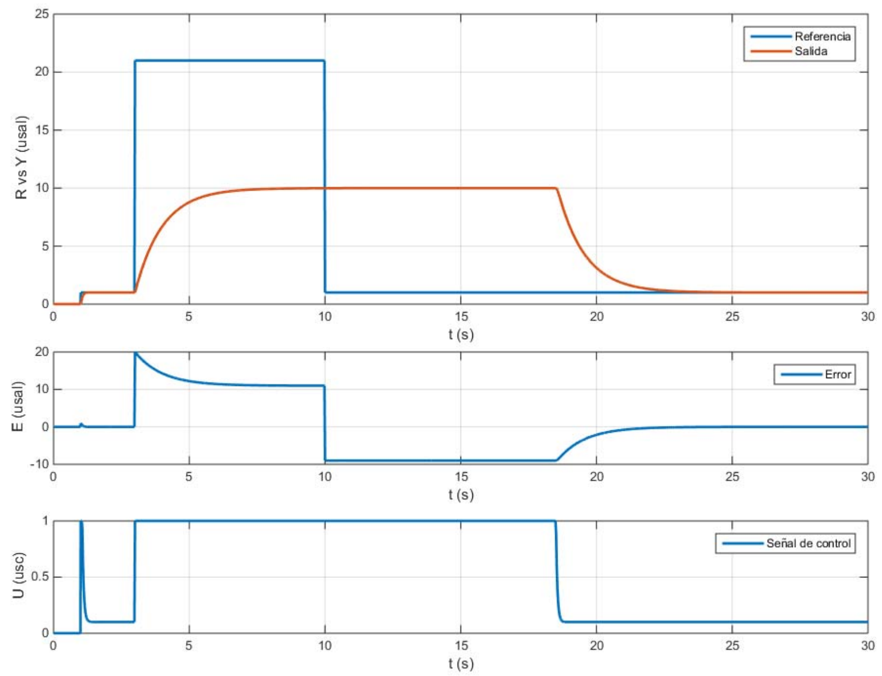

$$PID: K_P \frac{T_I T_D s^2 + T_I s + 1}{T_I s} \quad \Rightarrow \quad \boxed{\begin{array}{c} T_I^* \neq T_I \\ T_D^* \neq T_D \end{array}}$$



# Efecto *windup*

- Todos los sistemas físicos tienen **saturaciones en la señal de control** (y de hecho, se debe saturar dicha señal para proteger al sistemas).
- Cuando la señal de control satura, se pierde la realimentación.
- Si el controlador tiene **efecto integral**, se sigue acumulando error a pesar de que no hay realimentación.
- El exceso de acumulación de error es mayor a medida que el error instantáneo y el tiempo de acumulación lo sean.
- No se vuelve a recuperar la realimentación hasta que el exceso de la acumulación de la integral del error desaparece.
- Puede incluso que el sistema en bucle cerrado se vuelva inestable por este efecto.

# Efecto *windup*



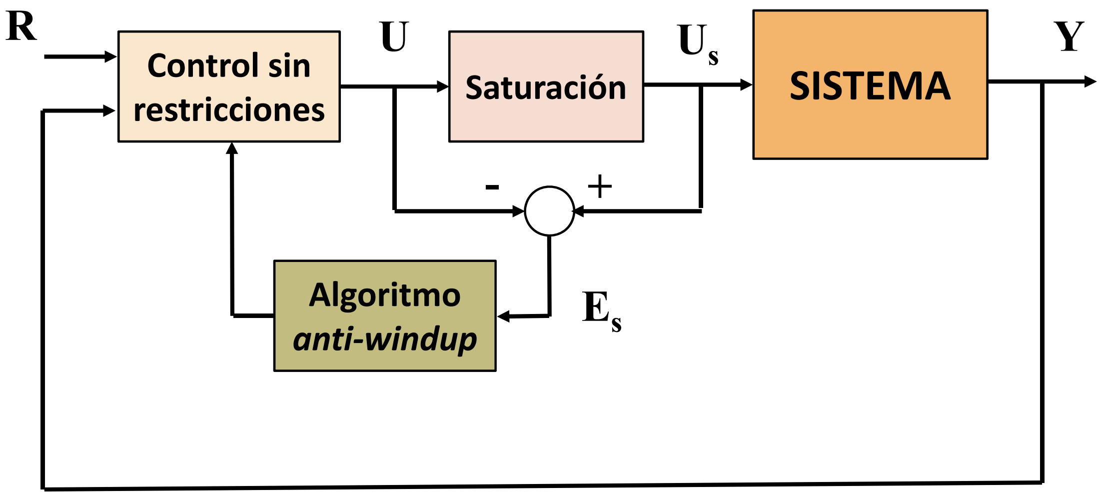



# Algoritmos anti-*windup*

- Pretenden atenuar los efectos del *windup* de manera automática.
- Existen distintos enfoques, como por ejemplo:
  - Modificar la referencia de manera adecuada (no en este curso).
  - Inicializar el control integral a un valor deseado, por ejemplo, al valor justo antes de detectar el problema.
  - Deshabilitar la parte integral del controlador hasta que la salida (variable a controlar) vuelta a la región controlable.
  - Dejar de acumular el término integral por encima de unos límites predetermindos.
  - Realimentar el término integral del controlador para mantener la salida dentro de las límites admisibles.
- Se hace necesario la implementación del controlador en paralelo.

# Algoritmos anti-*windup*

◦ **Esquema general**






# Algoritmos anti-*windup*

◦ **AW1: Algoritmo incremental:**

Se calcula: $\boxed{u_k = u_{k-1} + q_0 e_k + q_1 e_{k-1} + q_2 e_{k-2}}$

Y si hay saturación: $\boxed{u_k = u_{k-1}}$

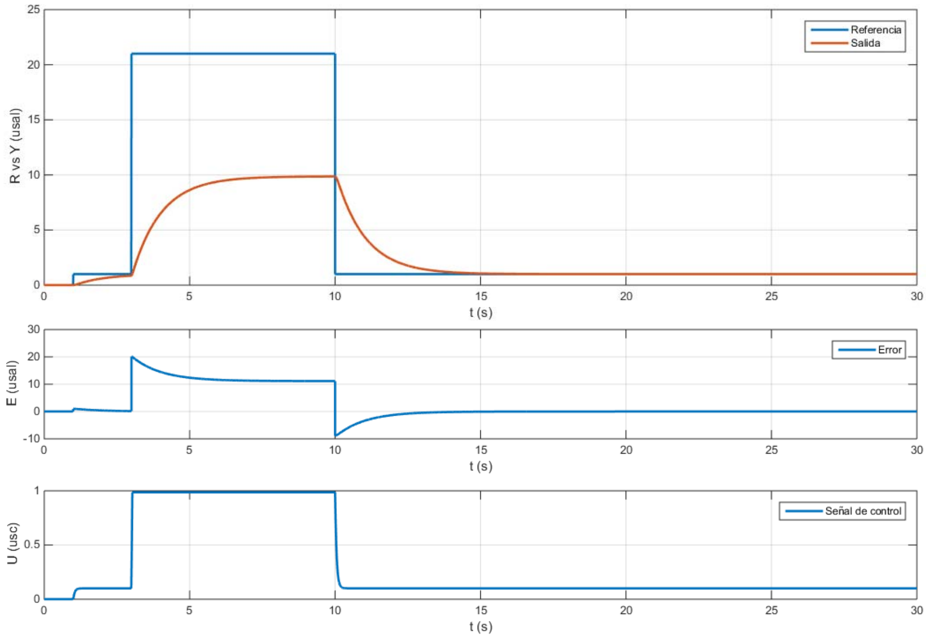
◦ **AW2: Integración condicional a la señal de control**

$$u(t) = K_P\left( e(t) + \frac{1}{T_I}\int_0^t e(\tau)d\tau + T_D\frac{de(t)}{dt}\right)$$

**Dejar de actualizar la integral cuando se produzca la saturación en la señal de control**

# Algoritmos anti-*windup*

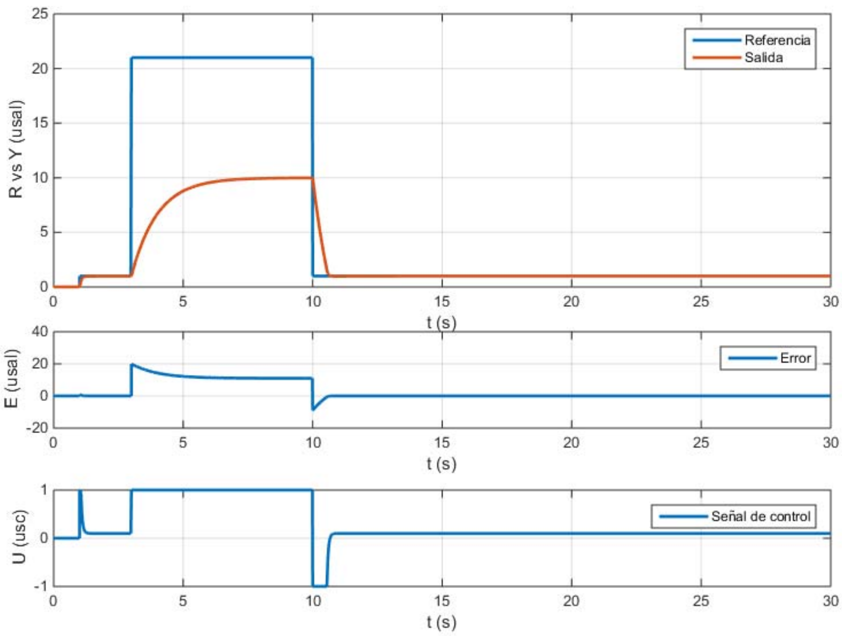
◦ **AW1: Algoritmo incremental:**






# Algoritmos anti-*windup*

◦ **AW2: Integración condicional a la señal de control**



# Algoritmos anti-*windup*

◦ **AW3: Integración condicional a la salida:**

$$u(t) = K_P\left(e(t) + \frac{1}{T_I}\int_0^t e(\tau)d\tau - T_D\frac{dy(t)}{dt}\right) \quad \text{PID modificado}$$

$$u(t) = K_P r(t) + \underbrace{\frac{K_P}{T_I}\int_0^t e(\tau)d\tau}_{I(t)} - K_P\underbrace{\left(y(t) + T_D\frac{dy(t)}{dt}\right)}_{y_p(t)}$$

$y_p(t)$ : predicción de la salida a $t=t+T_d$

**Banda de la salida** para la cual se estima que **no habrá saturación**:

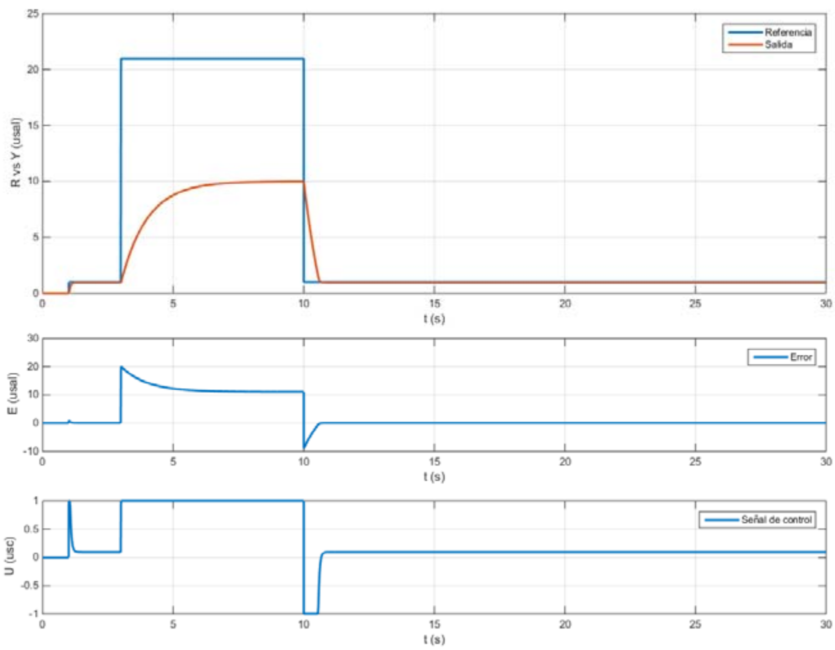
$$y_{\max}(t) = r(t) + \frac{I(t) - u_{\min}}{K_P} \qquad y_{\min}(t) = r(t) + \frac{I(t) - u_{\max}}{K_P}$$

**Actualizar *I(t)* sólo cunado $y_p(t)$ esté dentro de la banda de la salida**



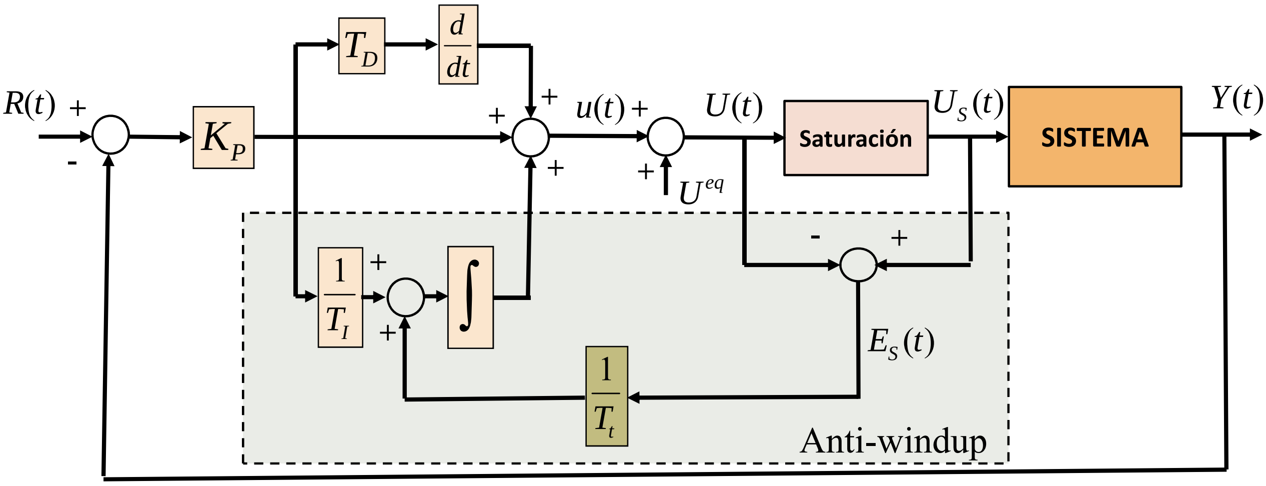
# Algoritmos anti-*windup*

◦ **AW3: Integración condicional a la salida**



# Algoritmos anti-*windup*

◦ **AW4: Realimentación del término integral:**



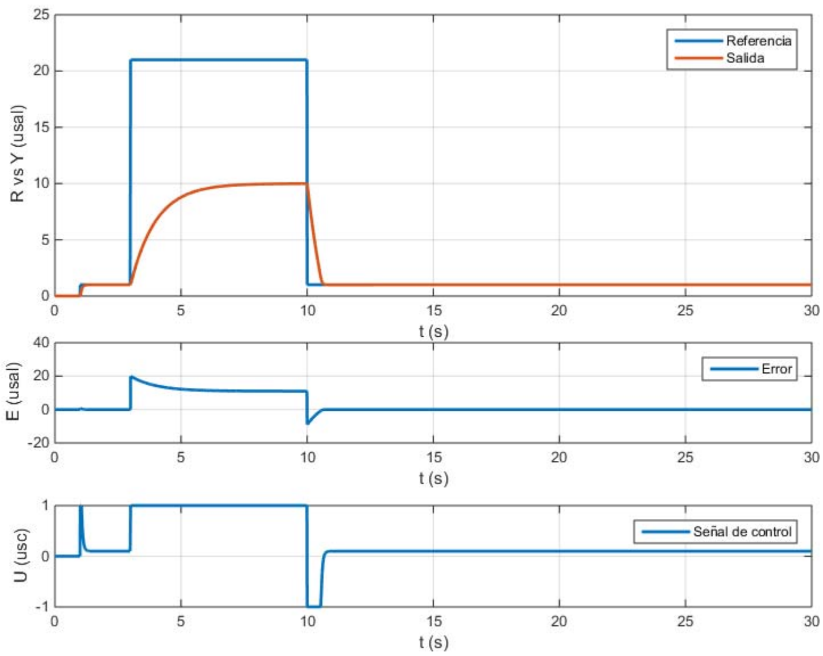

$T_t$: Parámetro de diseño

- Requiere implementación en paralelo
- ¿Sería posible a un controlador que no sea de tipo PID?



# Algoritmos anti-*windup*

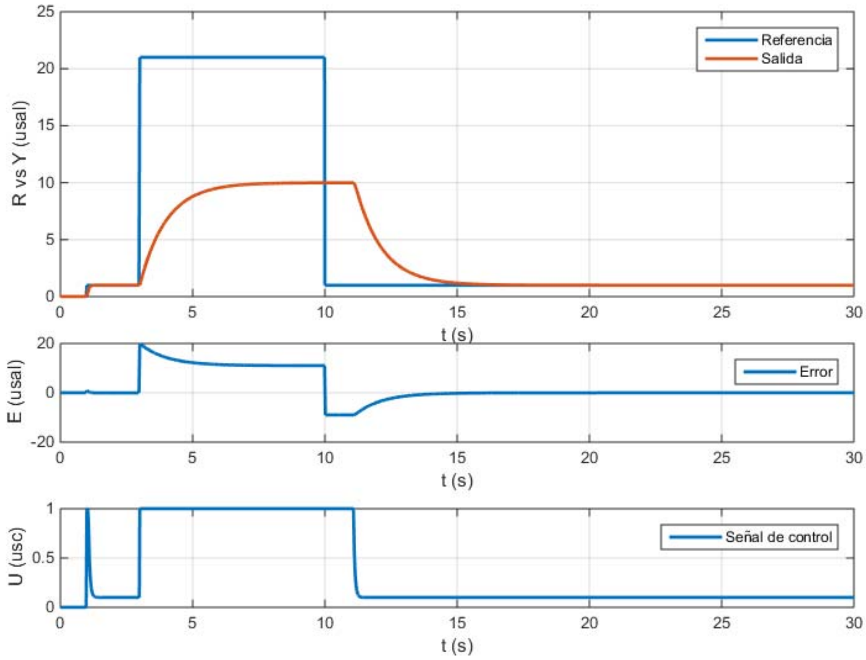
◦ **AW4: Realimentación del término integral**




$T_t = 1$ s

# Algoritmos anti-*windup*

- **AW4: Realimentación del término integral**




$$T_t = 3 \text{ s}$$