Control de Bajo Nivel
Controlador PID

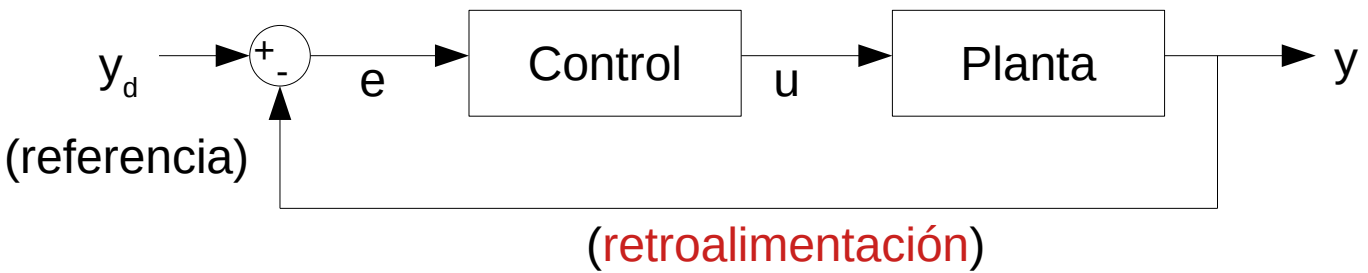
Dr. Alejandro Gutiérrez Giles
alejandro.giles@inaoep.mx
Dr. José Martínez Carranza
carranza@inaoep.mx

Coordinación de Ciencias Computacionales, INAOE

Retroalimentación

- Sistema a controlar

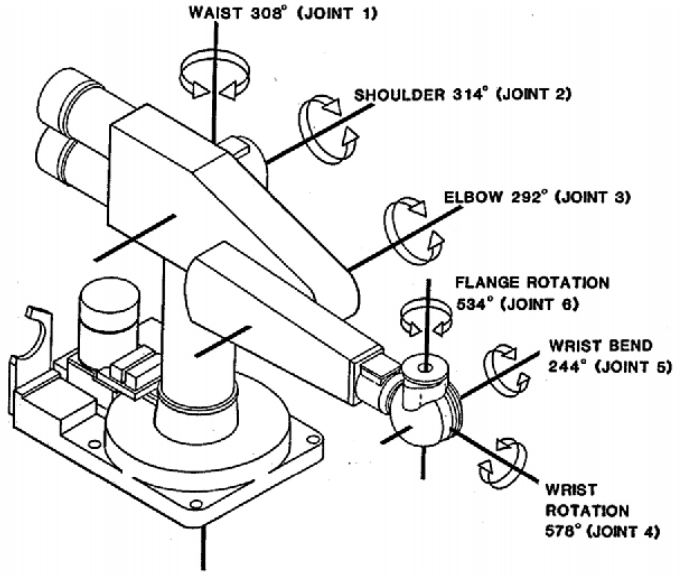
- Objetivo de control: $y \longrightarrow y_d$

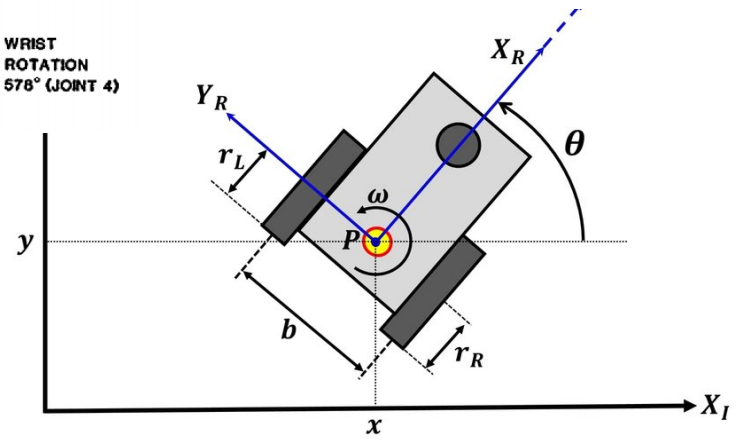


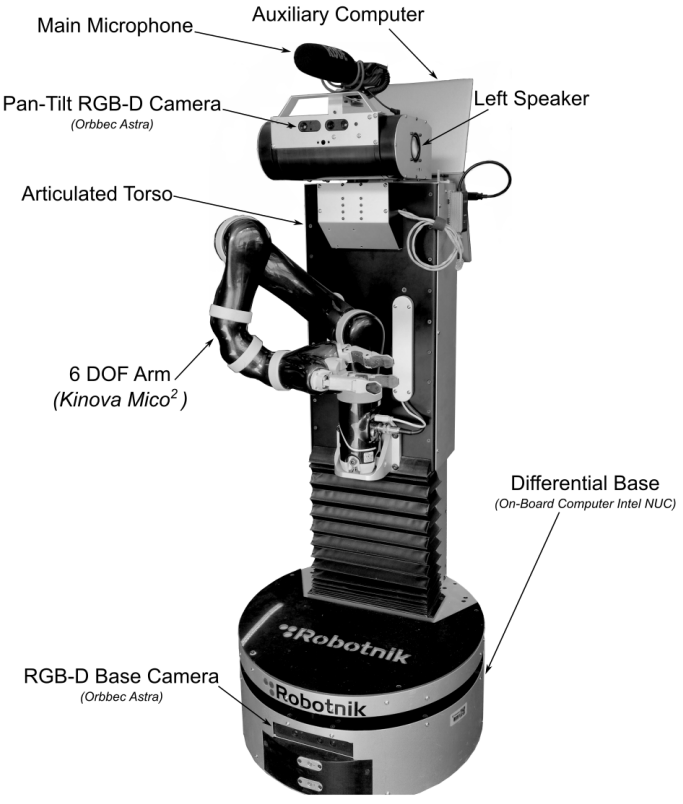
Grados de Libertad

- **Grados de libertad (GLD)**: Número mínimo de variables para describir la configuración de un mecanismo.
- **Entradas de control (u)**: Variables que se pueden manipular independientemente para cambiar el estado de un sistema.
- **Estados (x):** Variables utilizadas para describir completamente el comportamiento dinámico del sistema.
- **Salidas (y)**: Mediciones de algunos (o todos) los estados del sistema.
- **Completamente actuados (u>=GDL)** o **subactuados (u<GDL)**

Grados de Libertad





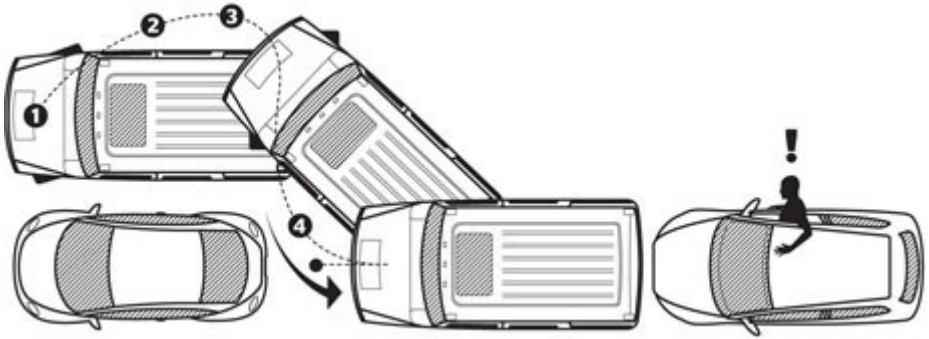


Robots con restricciones

Restricciones en velocidad (Pfaffianas): $A(q)\dot{q}=0$

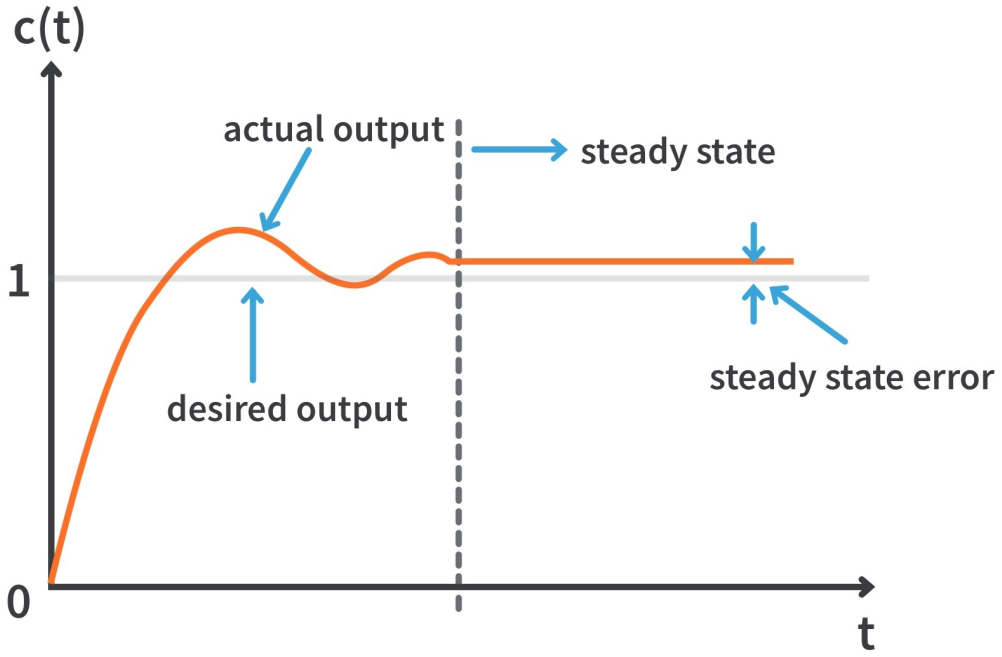
- **Holonómicas**: pérdida de grados de libertad (restricción a un subespacio)

- **No holonómicas**: sin pérdida de grados de libertad



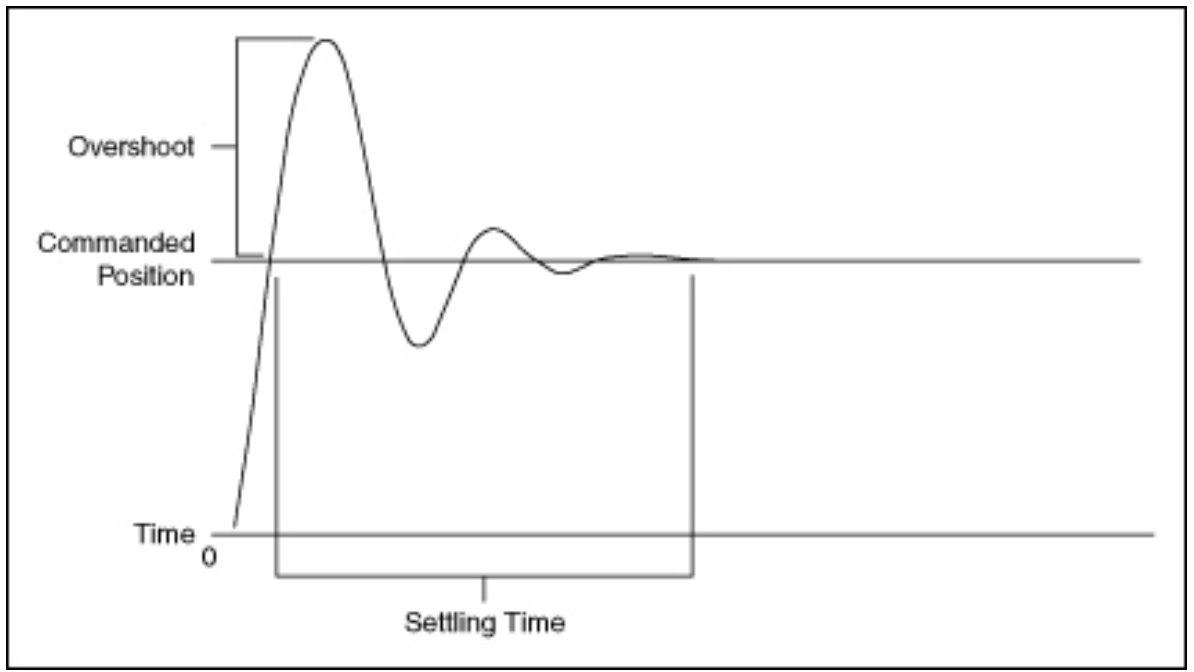
Objetivos de control

- **Error en estado estacionario** $e_{ss} = |y_{ss} - y_d|$



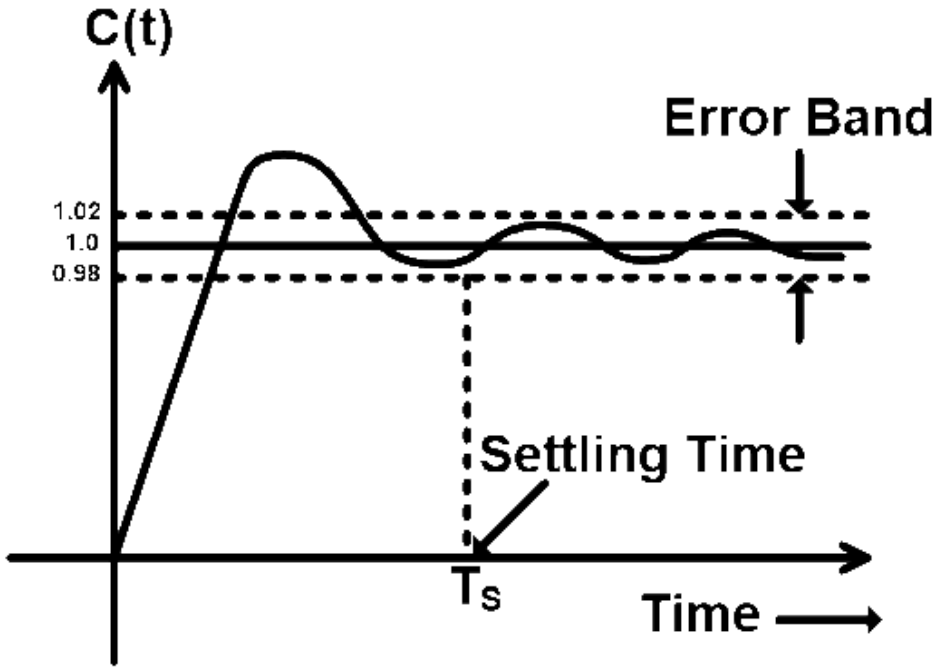
Desempeño del controlador

- **Sobrepaso máximo:** $M_p = \frac{|y_{max} - y_{ss}|}{|y_{ss}|} \cdot 100\,\%$



Desempeño del controlador

- **Tiempo de asentamiento**



Sistemas lineales de primer orden

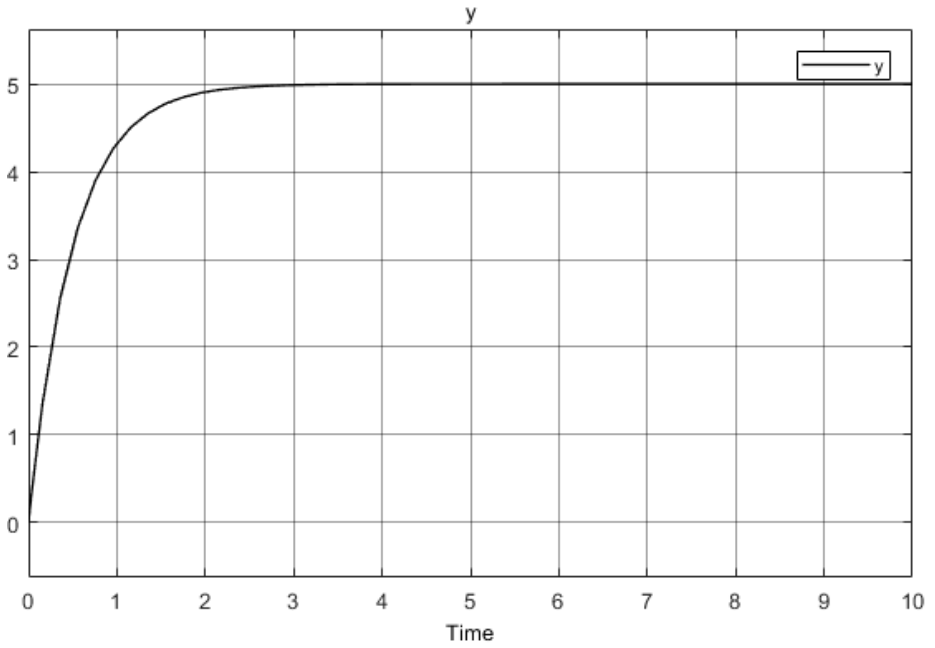
- **Función de transferencia:**

$$Y(s)=\frac{b}{s+a}U(s)$$

- **Dominio del tiempo:**

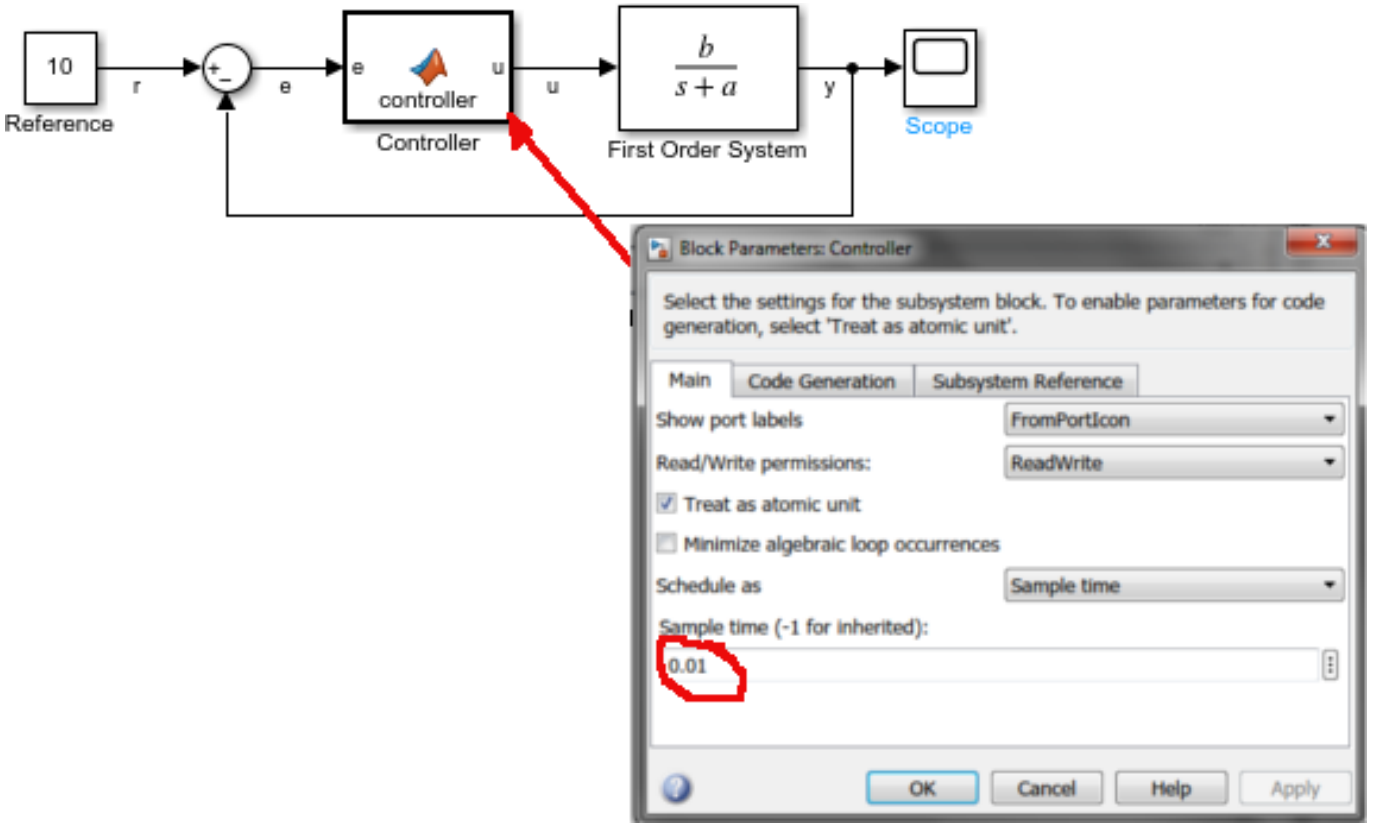
$$\dot{x}(t)=-a\,x(t)+b\,u(t)$$

- **Ejemplos: crecimiento biológico, motores eléctricos, procesos químicos, hidráulica, control de crucero, etc.**



- Constante de tiempo: **1/a**
- Ganancia en estado estacionario: **b/a**

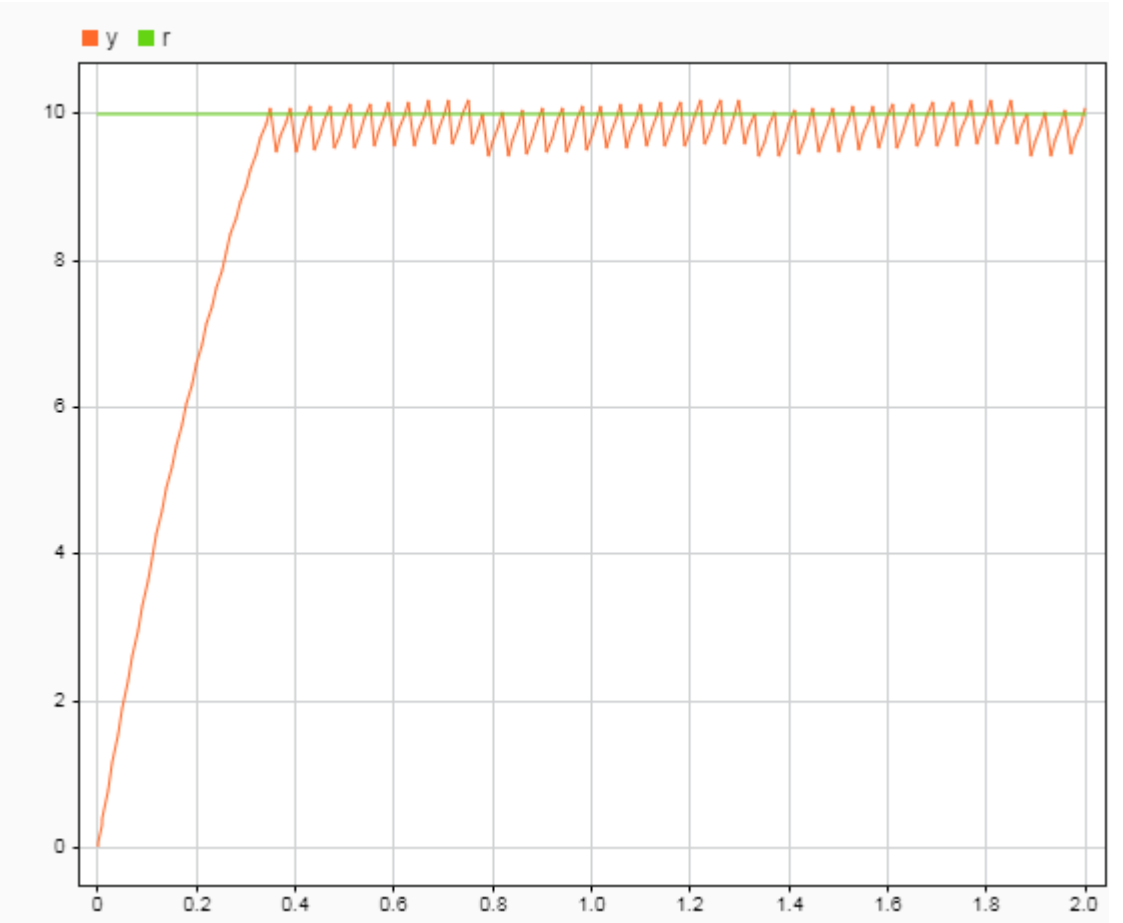
Simulación en Matlab/Simulink



Controlador ON/OFF

firstOrder ▸ Controller

```
1   function u = controller(e)
2
3   if(e>0)
4       u=4;
5   elseif (e<0)
6       u=-4;
7   else
8       u=0;
9   end
10
```



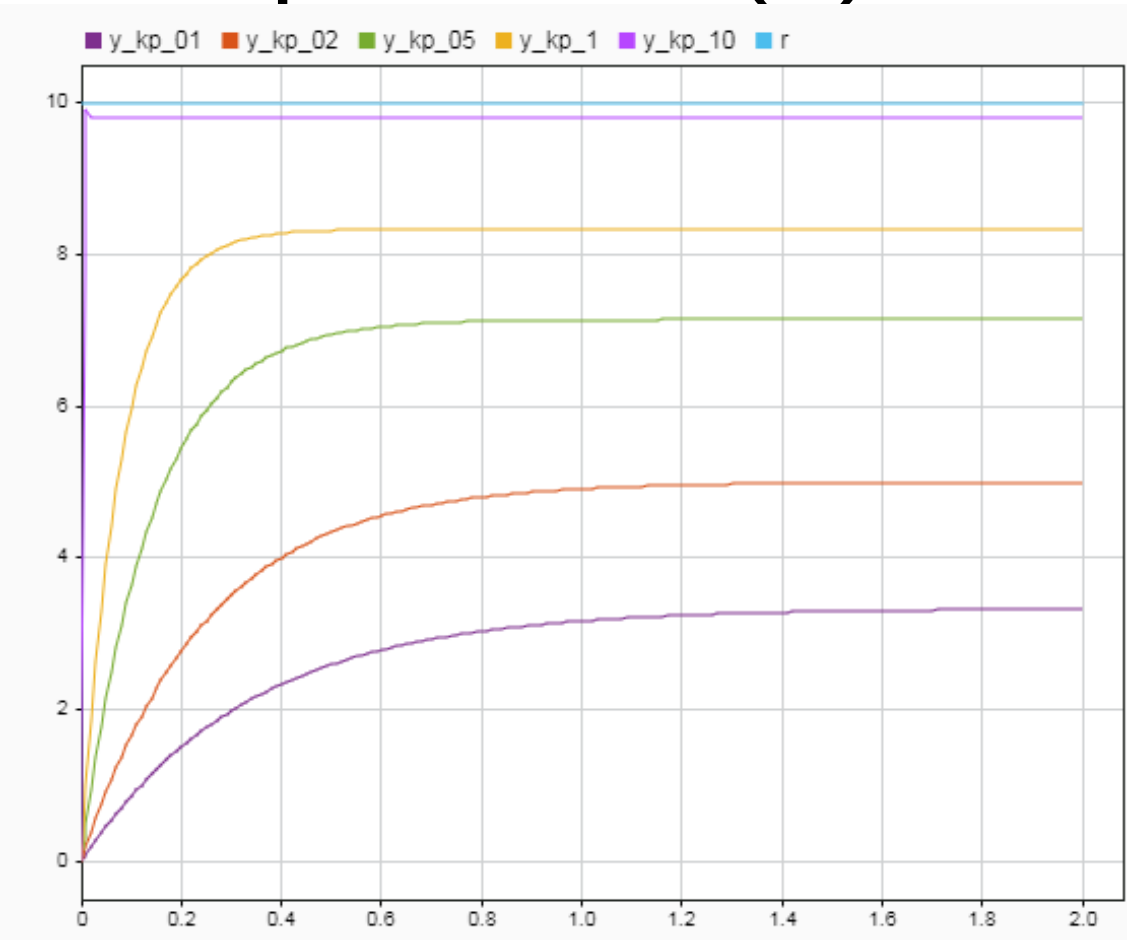
Controlador Proporcional (P)

$$u = k_p e$$

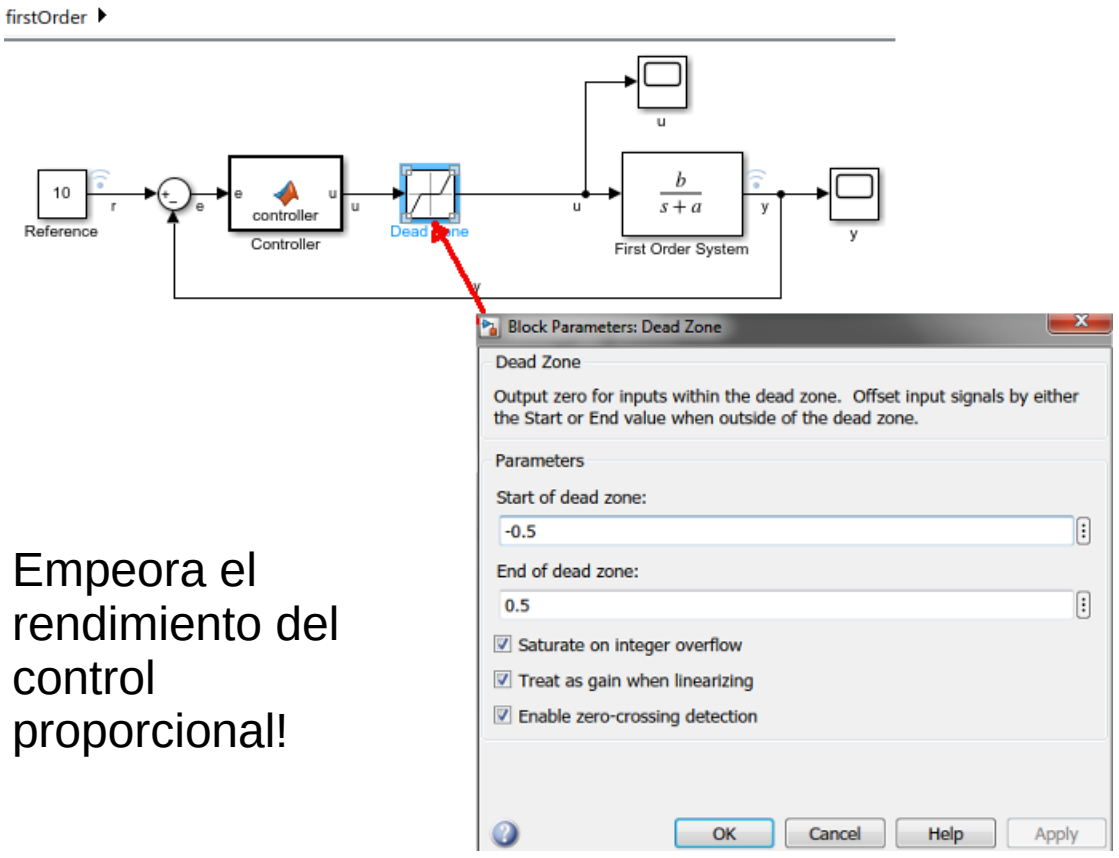
firstOrder ▸ Controller

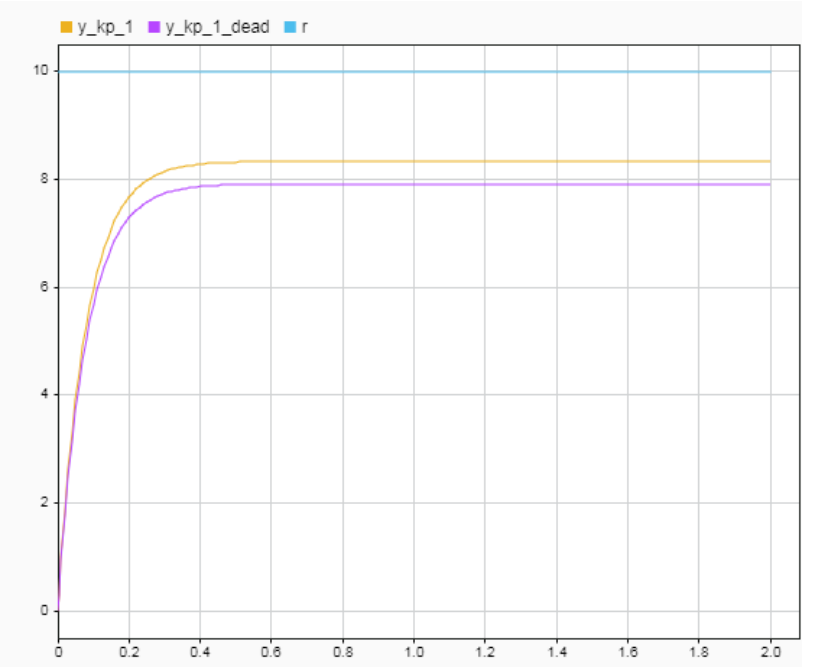
```
1   function u = controller(e)
2
3   kp = 1;
4   u = kp*e;
5
```

Solo puede llegar a error = 0 en estado estacionario con kp infinita!



Zona muerta





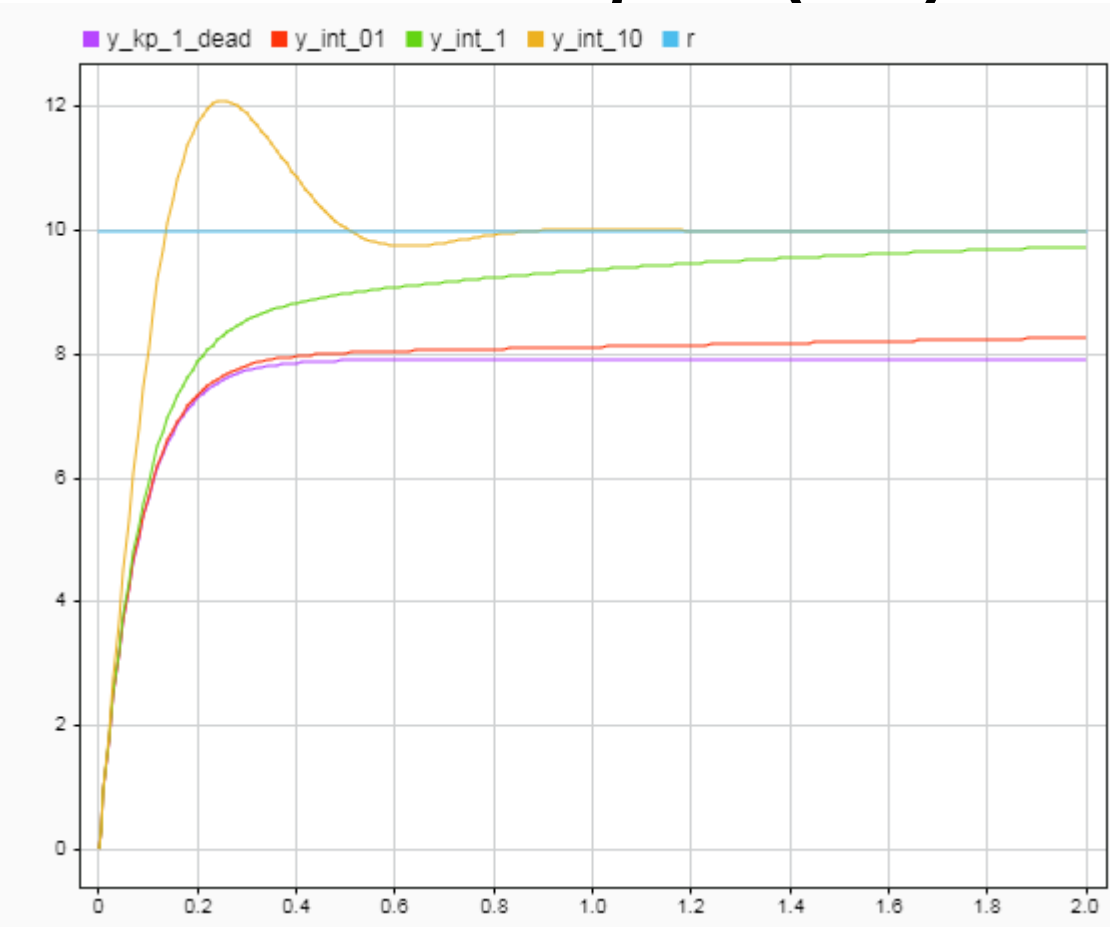
Empeora el rendimiento del control proporcional!

Control Proporcional Integral (PI)

$$u = k_p e + k_i \int e\,dt$$

firstOrder ▸ Controller

```
1  function u = controller(e)
2
3  Ts = 0.01;
4  persistent ei
5  if(isempty(ei)) ei = 0; end
6  kp = 1; ki = 0.1;
7  u = kp*e + ki*ei;
8  ei = ei + e*Ts;
9
```



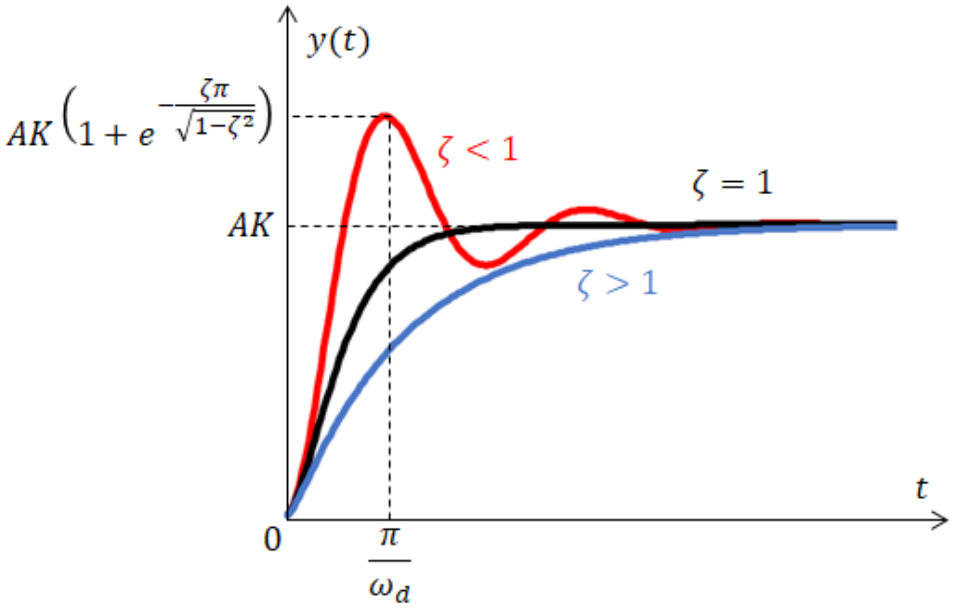
Sistema canónico de segundo orden

- **Función de transferencia:**

$$Y(s)=\frac{n(s)}{s^2+2\xi\omega_n s+\omega_n^2}R(s)$$

- **Sistema 1er. Orden con PI en lazo cerrado:**

$$Y(s)=\frac{b(k_p s+k_i)}{s^2+(k_p b+a)s+k_i b}$$



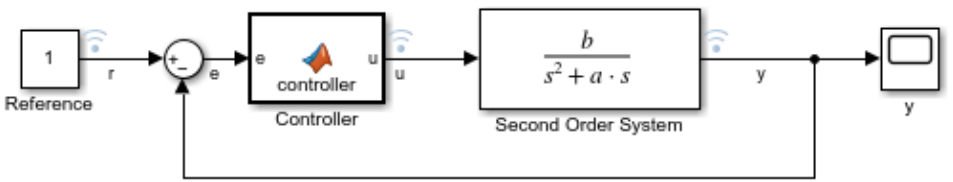
Sistemas lineales de segundo orden

- **Función de transferencia:**

$$Y(s)=\frac{b}{s(s+a)}U(s)$$

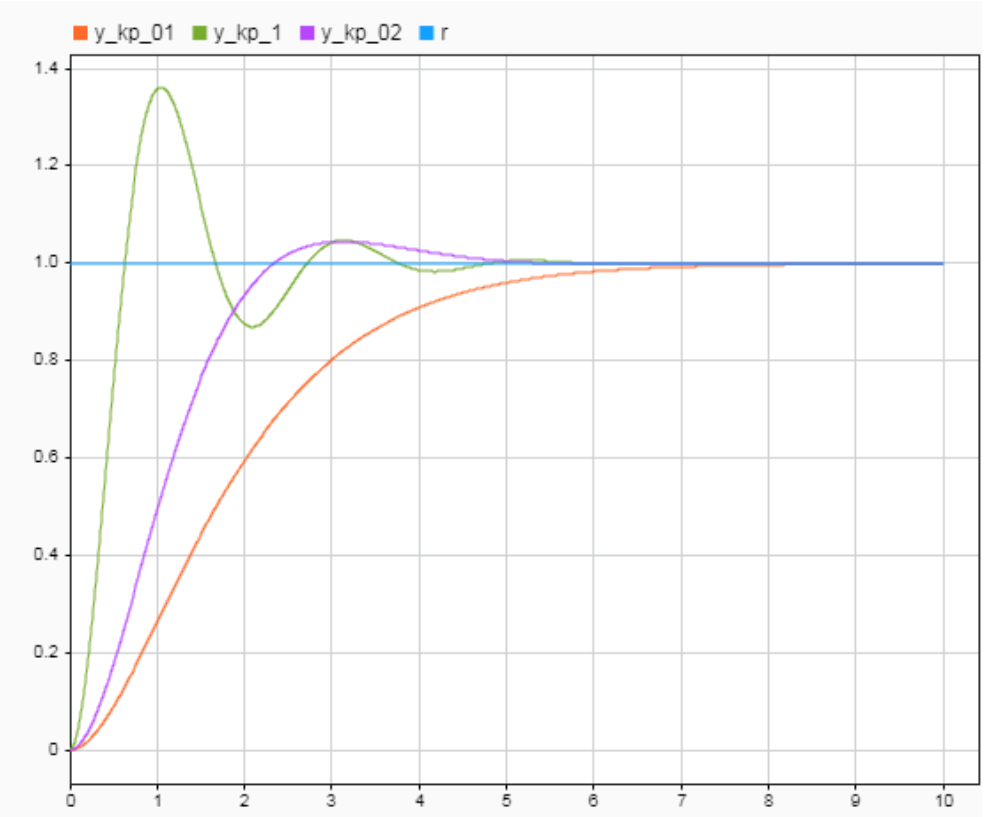
- **Dominio del tiempo:**

$$\dot{x}_1(t)=x_2(t)$$
$$\dot{x}_2(t)=-a\,x_2(t)+b\,u(t)$$



- **Ejemplo: control de posición de robots**

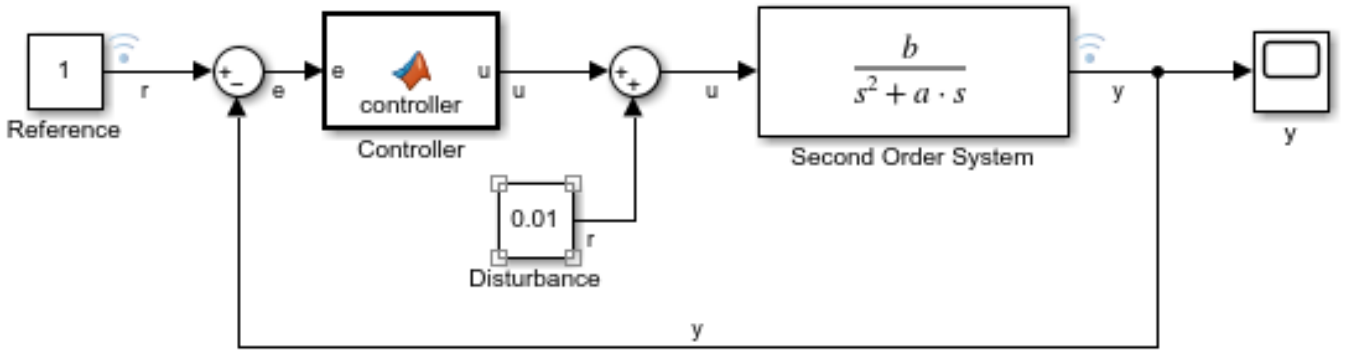
Control Proporcional (P)



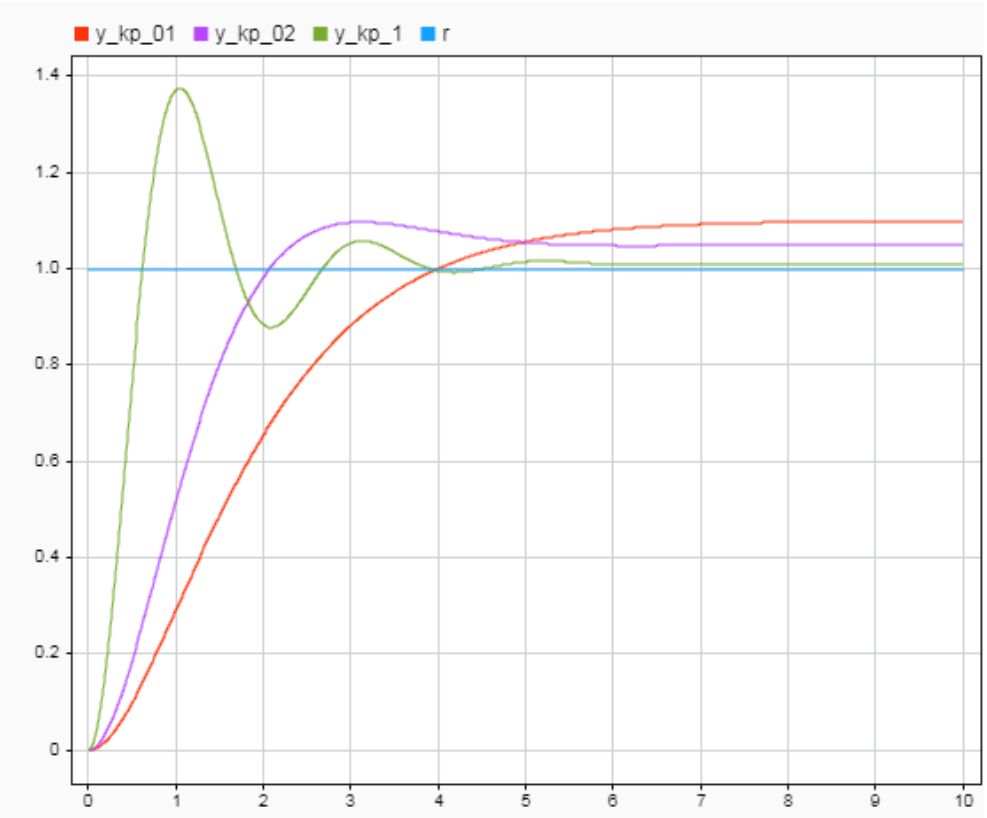
- Llega siempre a la referencia, para cualquier valor de kp!!!

… Siempre y cuando no existan perturbaciones o algunos efectos no lineales (e.g. fricción seca)

Control Proporcional bajo perturbaciones no desvanecientes

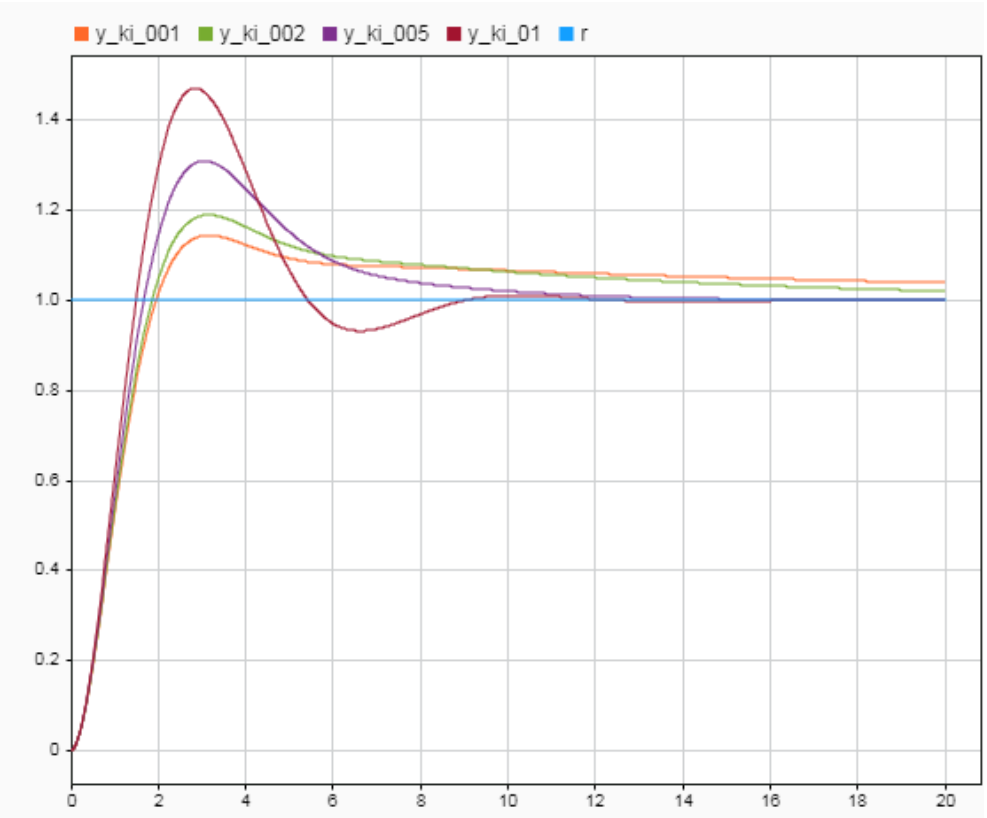


Control P con perturbación



- Existe siempre error en estado estacionario a menos que kp sea infinita
- Elegir kp grande empeora el sobrepaso máximo

Control PI con perturbación



- Se eligió kp = 0.2
- Elegir ki grande mejora el tiempo de asentamiento pero empeora el sobrepaso máximo!

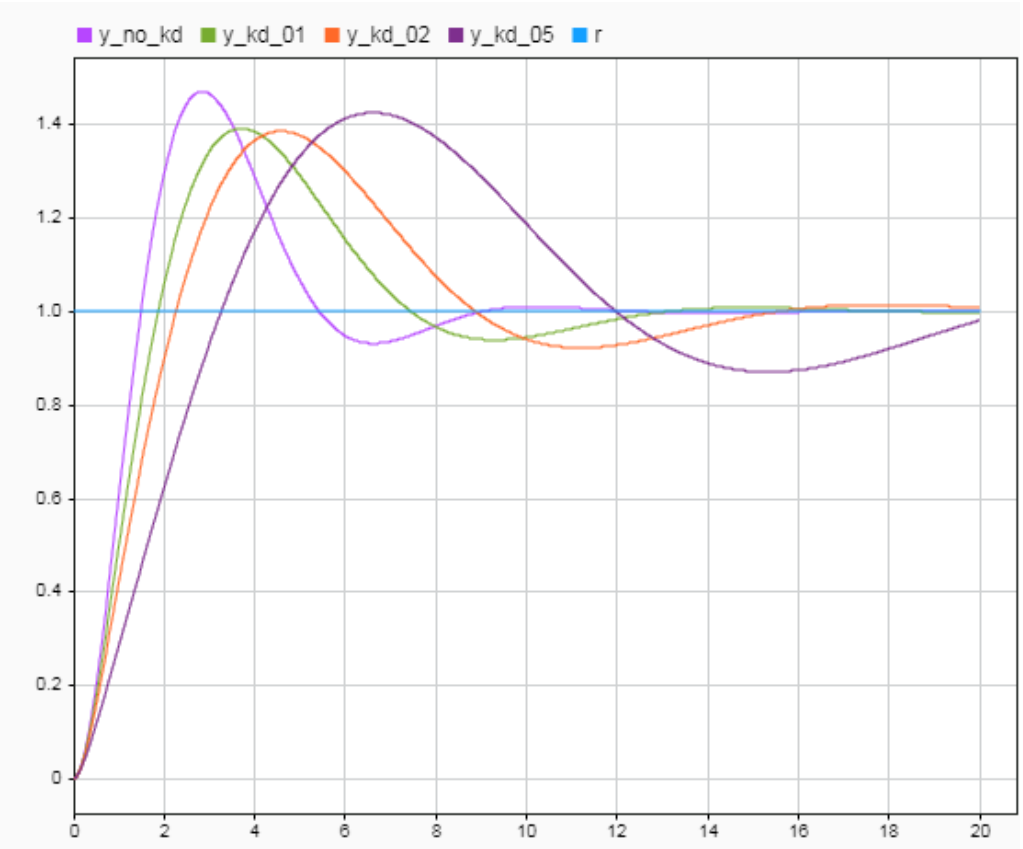
Control Proporcional Integral Derivativo (PID)

$$u = k_p e + k_i \int e\,dt + k_d \dot{e}$$

```
1   function u = controller(e)
2
3   Ts = 0.01;
4   persistent ei e_1
5   if(isempty(ei)) ei = 0; end
6   if(isempty(e_1)) e_1 = e; end
7   kp = 0.2; ki = 0.1; kd = 0.5;
8   ep = (e - e_1)/Ts;
9   u = kp*e + ki*ei + kd*ep;
10  ei = ei + e*Ts;
11  e_1 = e;
```

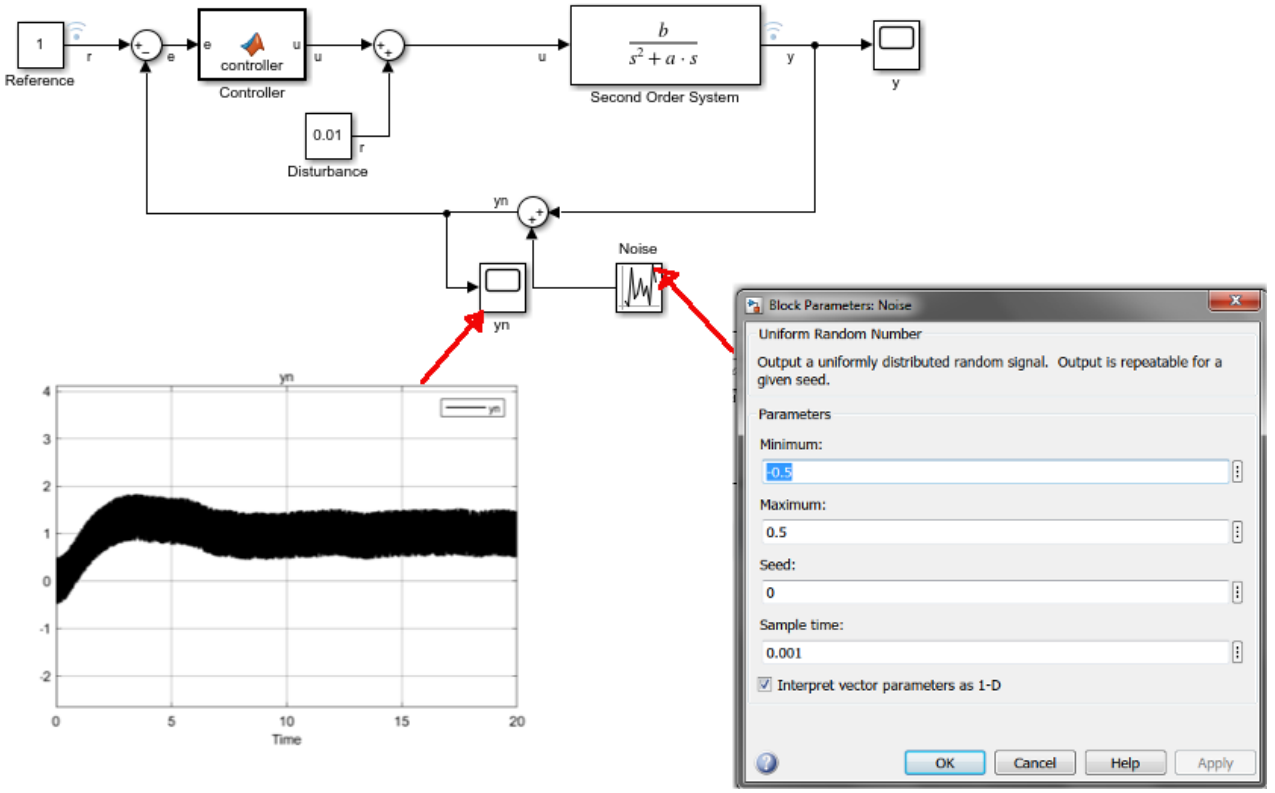
- En este código se utiliza un diferenciador numérico de primer orden (Euler)

Control PID



- En este código se utiliza un diferenciador numérico de primer orden (Euler)

Control PID con ruido



Filtro derivativo

- **Función de transferencia:**

$$Y(s)=\frac{\lambda}{s+\lambda}U(s)$$

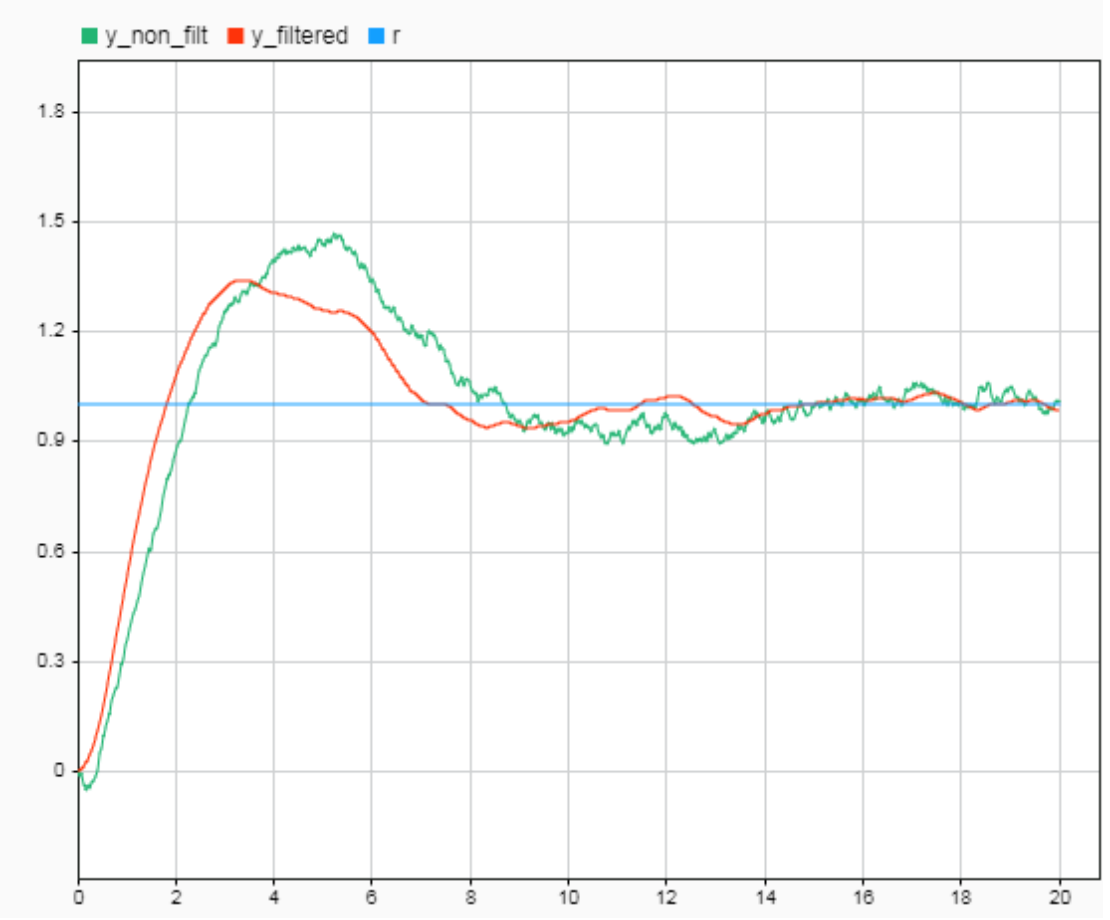
- **Dominio del tiempo:**

$$\frac{dy(t)}{dt}=\lambda(u(t)-y(t))$$

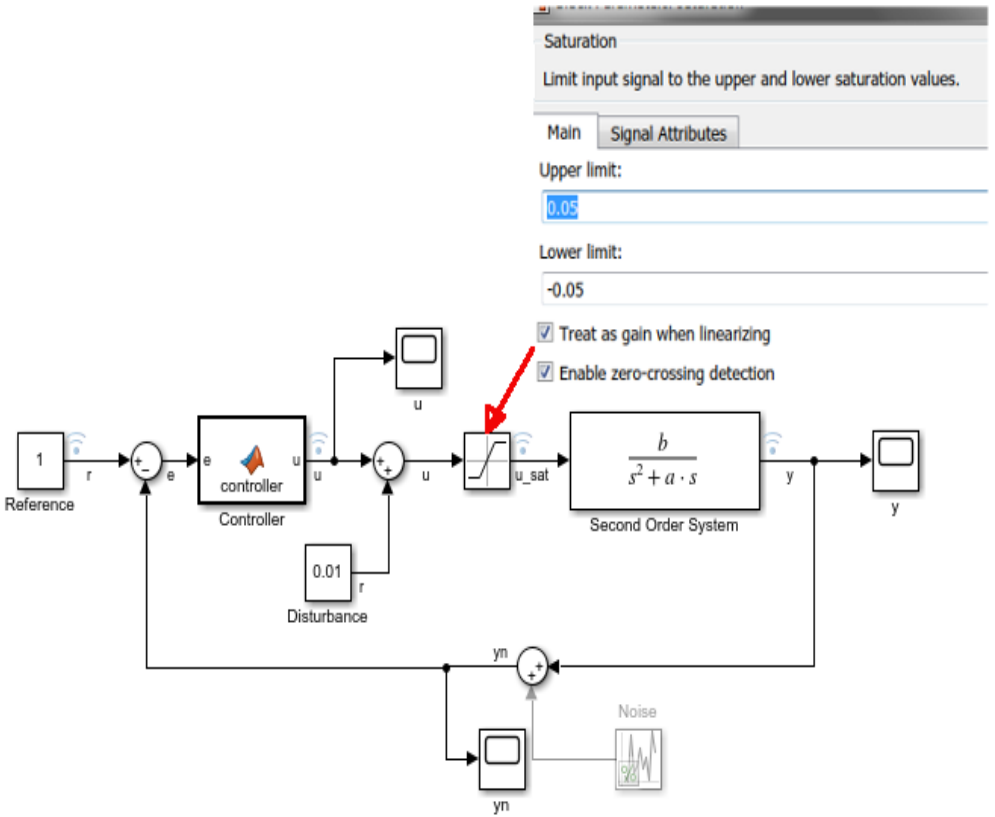
```
1   function u = controller(e)
2
3   Ts = 0.01;
4   persistent ei ef
5   if(isempty(ei)); ei = 0; end
6   if(isempty(ef)); ef = e; end
7   kp = 0.2; ki = 0.1; kd = 0.2;
8   lamfilt = 2;
9   efp = lamfilt*(e-ef);
10  u = kp*ef + ki*ei + kd*efp;
11  ei = ei + e*Ts;
12  ef = ef + efp*Ts;
```

efp es la derivada filtrada del error y ef es el error filtrado

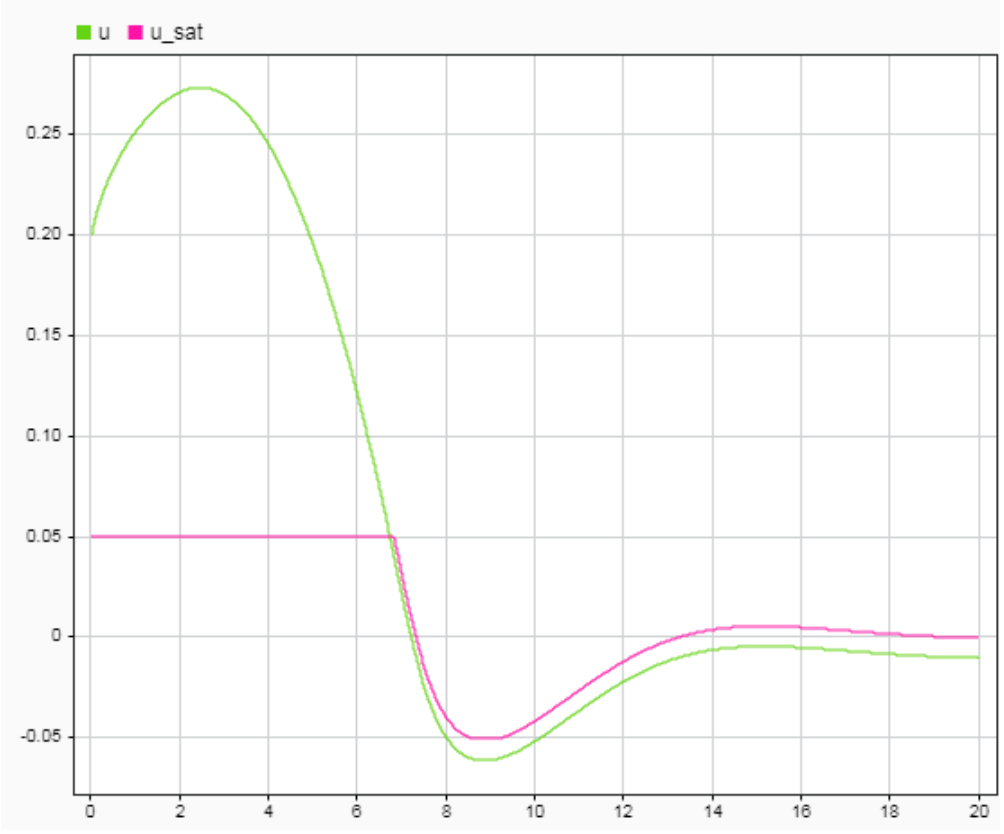
Control PID con filtro



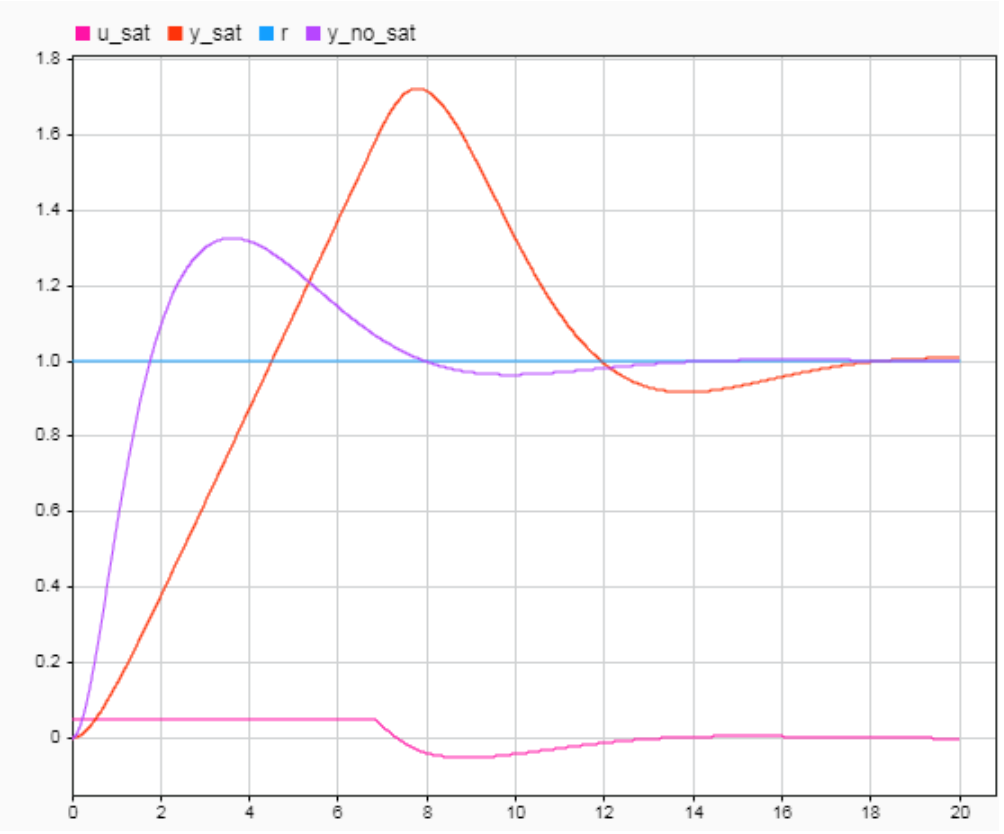
Saturación en actuadores



Saturación en actuadores

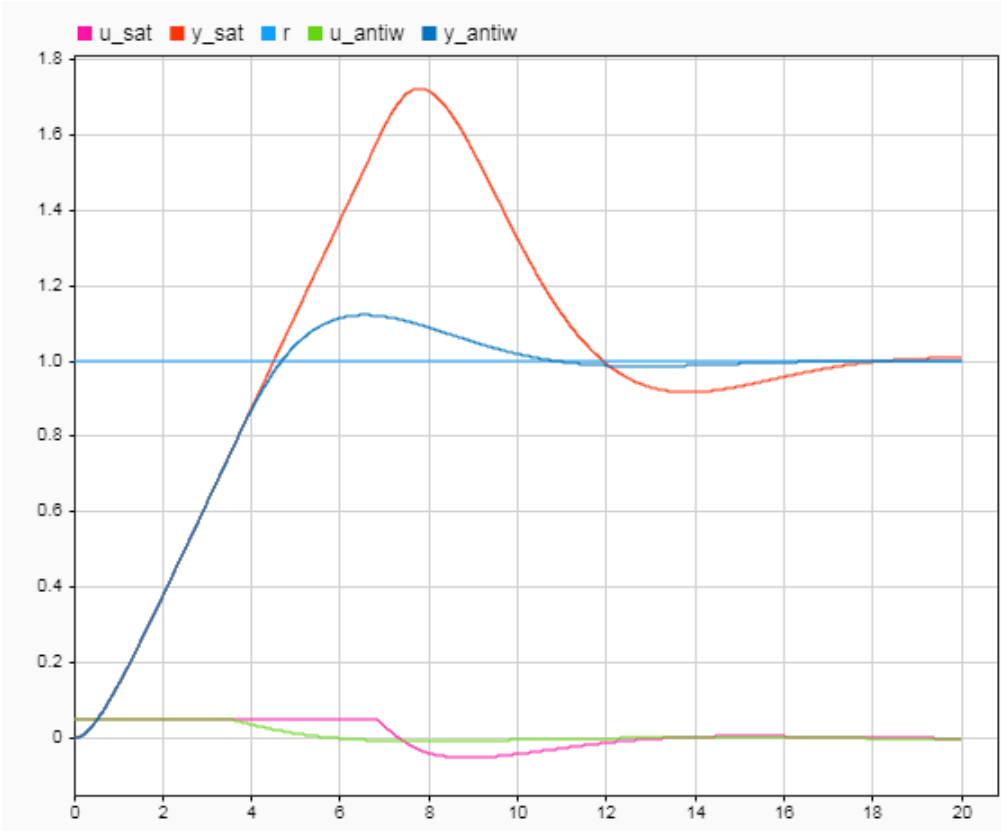


Saturación en actuadores



- A este problema se le conoce como **windup**
- Existen varios métodos para combatirlos **(anti-windups)**

Anti-windup de clamping

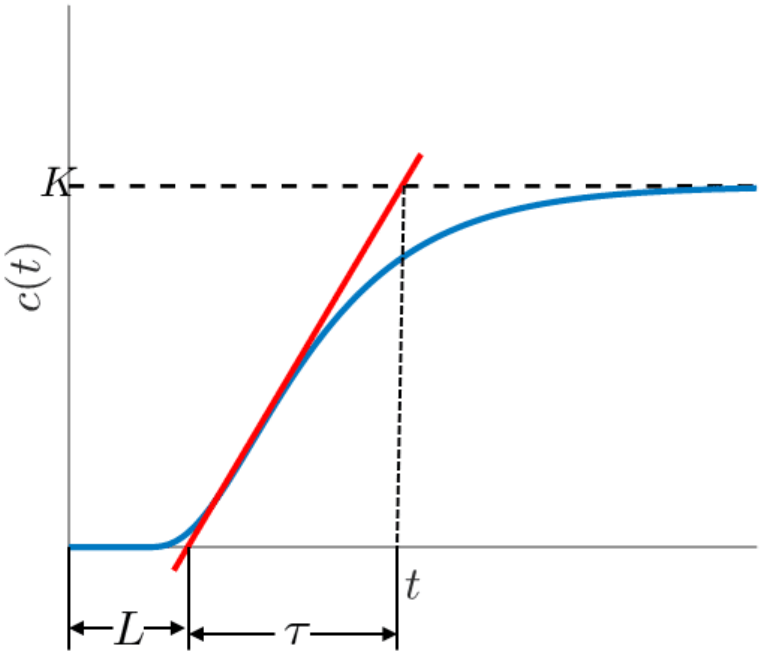


secondOrder ▸ Controller

```
1  function u = controller(e)
2
3  Ts = 0.01;
4  persistent ei ef
5  if(isempty(ei)); ei = 0; end
6  if(isempty(ef)); ef = e; end
7  kp = 0.2; ki = 0.1; kd = 0.2;
8  lamfilt = 2;
9  efp = lamfilt*(e-ef);
10 u = kp*ef + ki*ei + kd*efp;
11 ei = ei + e*Ts;
12 ef = ef + efp*Ts;
13
14 umax = 0.05;
15 if(abs(u)>umax&&e*u>0)
16     ei = 0; % antiwindup
17 end
```

Sintonización

- **Ziegler-Nichols** 1$^{er.}$ Orden:



Controlador	K_p	τ_i	τ_d
P	$\frac{\tau}{KL}$	∞	0
PI	$0.9\frac{\tau}{KL}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{\tau}{KL}$	$2L$	$0.5L$

Aplica para sistemas de primer orden con retardo L

Sintonización

- **Ziegler-Nichols** 2do. Orden:

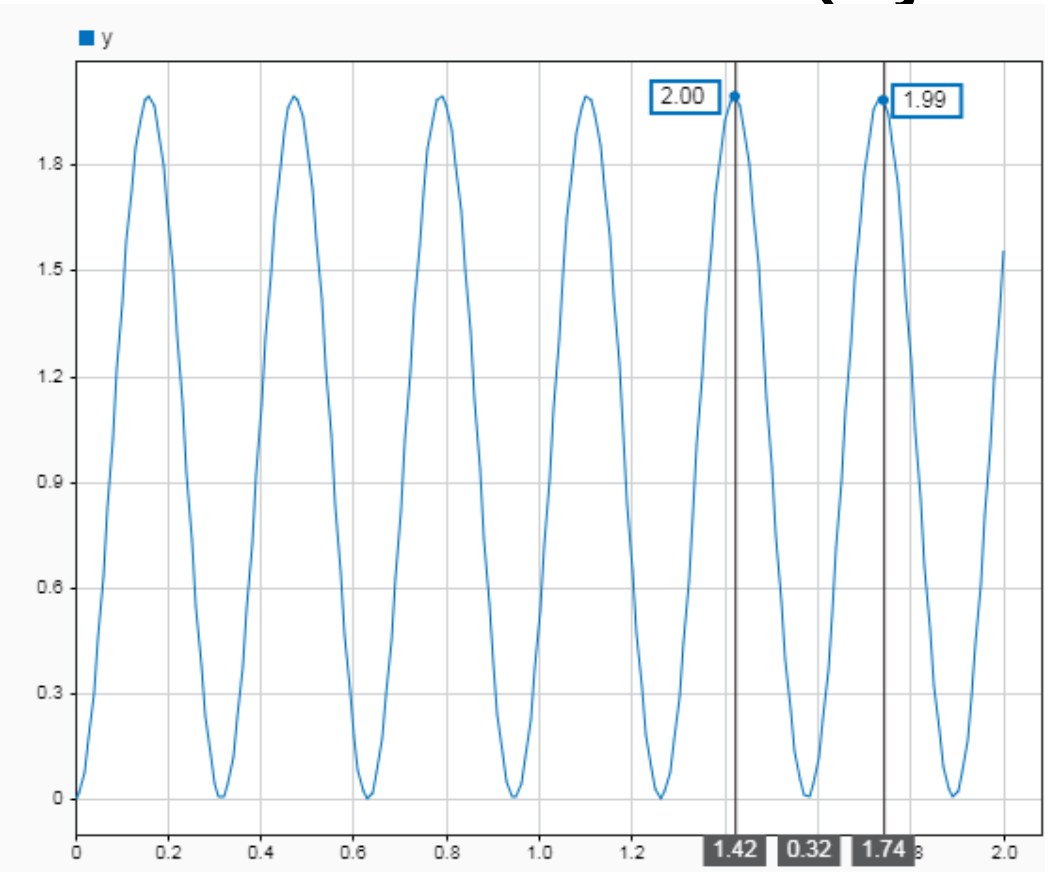
1) Se aplica sólo la parte P y se va aumentando Kp hasta que la respuesta sea oscilatoria

2) Una vez oscilando se guarda esa ganancia como Ku y se mide el Periodo de las oscilaciones como Tu

Control Type	K_p	T_i	T_d	K_i	K_d
P	$0.5K_u$	–	–	–	–
PI	$0.45K_u$	$0.8\bar{3}T_u$	–	$0.54K_u/T_u$	–
PD	$0.8K_u$	–	$0.125T_u$	–	$0.10K_uT_u$
classic PID[2]	$0.6K_u$	$0.5T_u$	$0.125T_u$	$1.2K_u/T_u$	$0.075K_uT_u$
Pessen Integral Rule[2]	$0.7K_u$	$0.4T_u$	$0.15T_u$	$1.75K_u/T_u$	$0.105K_uT_u$
some overshoot[2]	$0.3\bar{3}K_u$	$0.50T_u$	$0.3\bar{3}T_u$	$0.6\bar{6}K_u/T_u$	$0.1\bar{1}K_uT_u$
no overshoot[2]	$0.20K_u$	$0.50T_u$	$0.3\bar{3}T_u$	$0.40K_u/T_u$	$0.06\bar{6}K_uT_u$

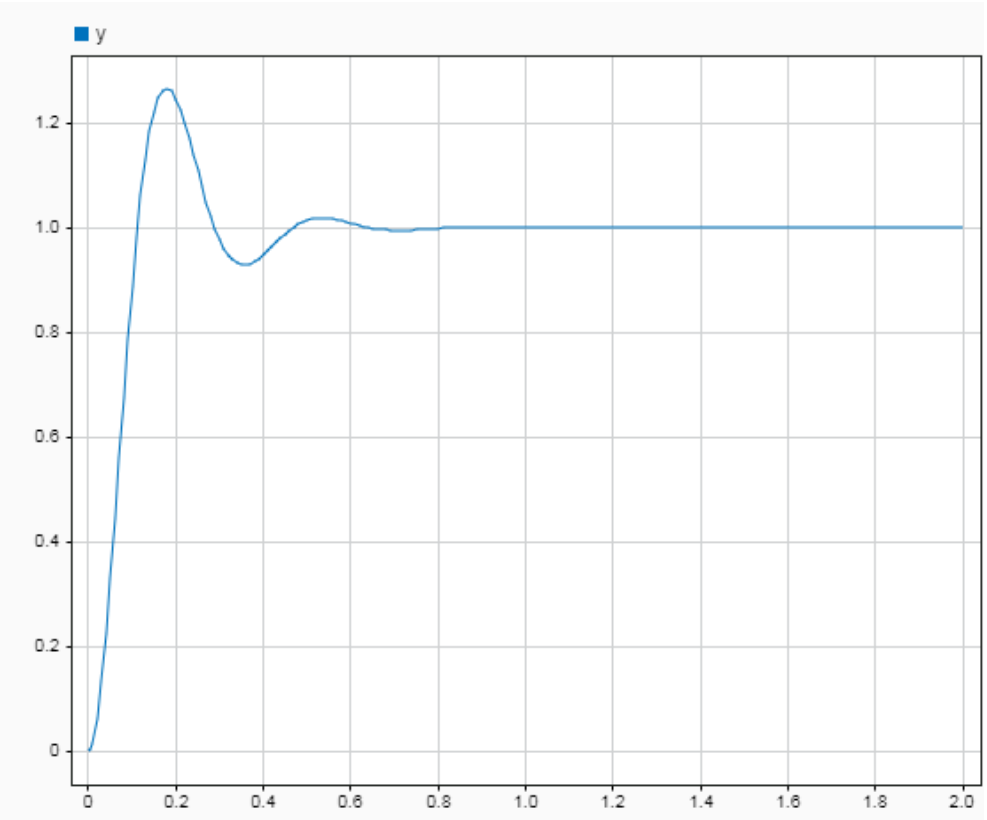
Aplica para sistemas de segundo orden que se puedan hacer oscilar con un controlador proporcional

Sintonización PD (ejemplo)



Ku = 40
Tu = 0.32

Sintonización PD (ejemplo)



Ku = 40
Tu = 0.32

PD:
Kp = 0.8 Ku = 32
Kd = 0.1 Ku*Tu = 1.28

En resumen...

- La ganancia **proporcional** disminuye el error pero aumenta el sobrepaso
- La ganancia **integral** disminuye el error en estado estacionario pero aumenta el sobrepaso, aún más si las entradas de control están saturadas
- La ganancia **derivativa** suaviza la respuesta pero si se aumenta se pueden amplificar los efectos del ruido
- Se puede utilizar un **filtro** de primer orden para disminuir el efecto del ruido aunque incorpora retraso en las señales
- Cuando hay saturación en las entradas de control se puede utilizar un **antiwindup** para mitigar los efectos causados por la acción integral
- Si se tiene un modelo del sistema se puede utilizar para sintonizar el controlador. Alternativamente se puede recurrir a las ganancias sugeridas por **Ziegler-Nichols**

Referencias

- Aström, K & Hägglund, T. (1995). PID Controllers: Theory, Design, and Tuning. 2nd Edition. Instrument Society of America. Disponible en: https://aiecp.files.wordpress.com/2012/07/1-0-1-k-j-astrom-pid-controllers-theory-design-and-tuning-2ed.pdf