

Grado en Ingeniería Informática

# Relación de ejercicios

Informática Industrial – Tema 6

OSCAR JIMENEZ FERNANDEZ  
29-11-2018

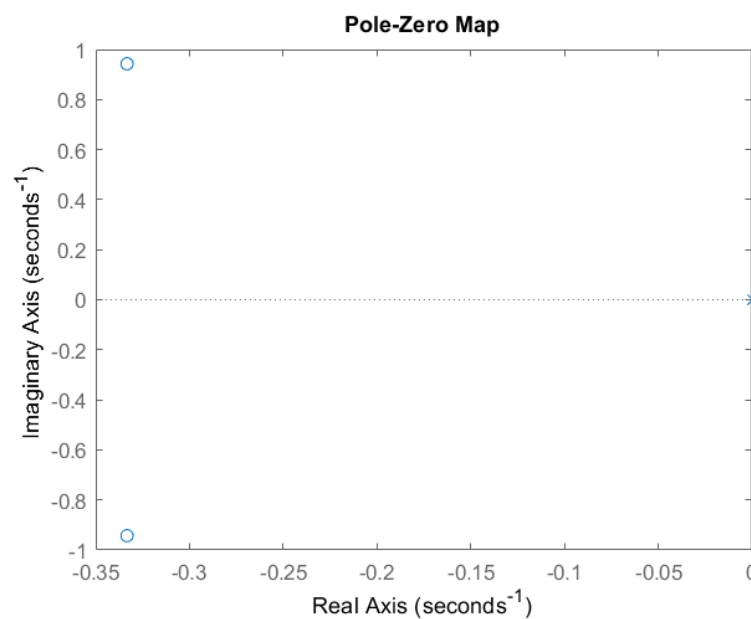
1. Dada la función de transferencia:  $H(s)=6s/(3s^2+2s+3)$ . a) Obtener y representar gráficamente los ceros y polos (roots) b) Obtener su representación en el espacio de estados. (Nota: usar tf2zp, pzmap, tf2ss). c) Obtener la respuesta a escalón unitario usando step.

A.-

```
n = [6 0];  
d = [3 2 3];  
zn = roots(n);  
pd = roots(d);  
pzmap(zn,pd);
```

ó





```
n = [6 0];  
d = [3 2 3];  
[zn,pd,k] = tf2zp(n,d);  
pzmap(zn,pd);
```

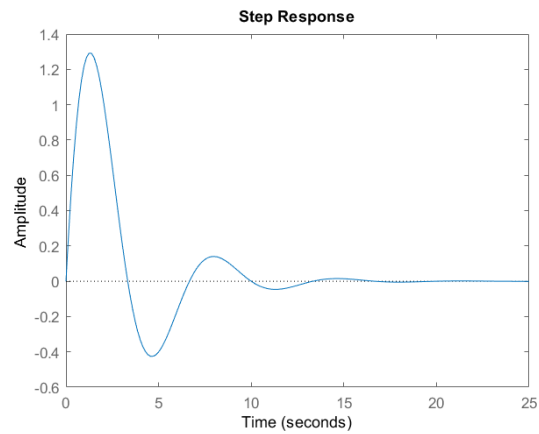


B.-

```
num = [6 0];  
den = [3 2 3];  
[A,B,C,D] = tf2ss(num,den);  
step(A,B,C,D,1);
```

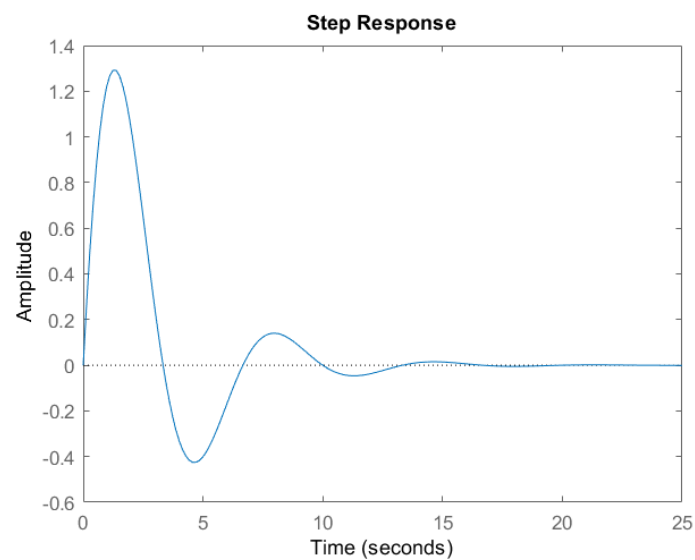
Los valores obtenidos para cada una de las variables son:

	A	$[-0.6667 \ -1; 1 \ 0]$
	B	$[1; 0]$
	C	$[2 \ 0]$
	D	$0$

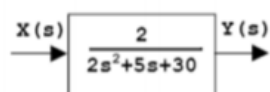


C.-

```
s = tf('s');
H = 6*s / (3*s^2 + 2*s + 3);
step(H);
```



2. a) Representar (con Matlab y también con Simulink) la respuesta a escalón unitario del siguiente sistema de 2º orden:

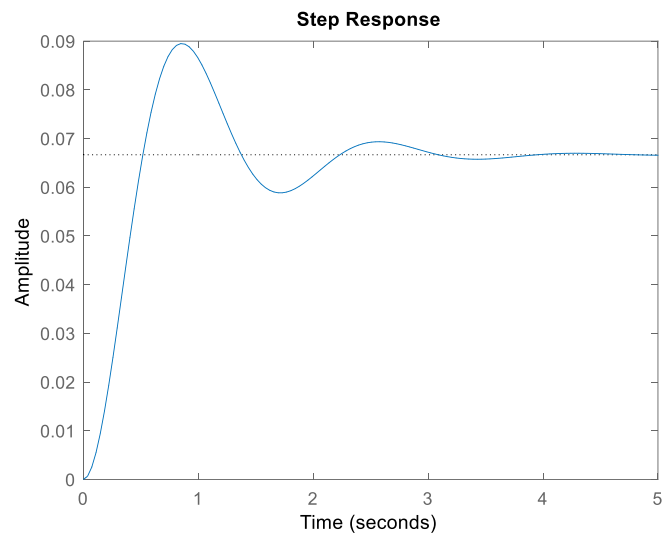


- b) Obtener su representación en el espacio de estados, representarla con Simulink y visualizar la respuesta a un escalón unitario.

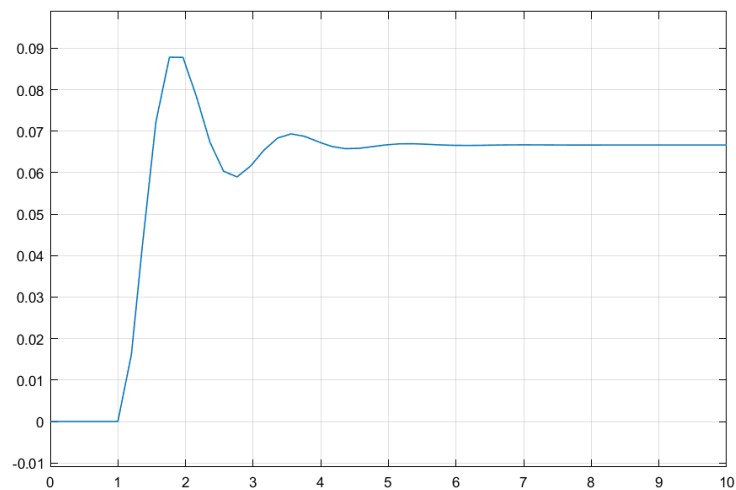
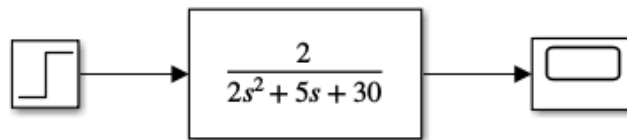
A.-

En Simulink tendríamos:

```
num = [2];  
den = [2 5 30];  
step(num,den);
```



La representación en Matlab sería:



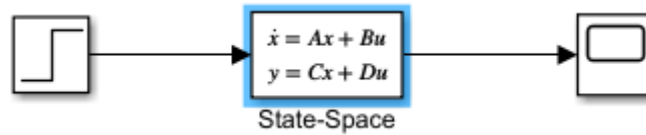
B.-

Obtenemos primeramente los valores de A, B, C, D con Matlab para después poder representarla en Simulink y visualizar el resultado.

```
num = [2];
den = [2 5 30];
[A,B,C,D] = tf2ss(num,den);
```

A	[-2.5000 -15;1 0]
B	[1;0]
C	[0 1]
D	0

Usamos los valores obtenidos para la representación en Simulink:



State Space

State-space model:  
 $\dot{x} = Ax + Bu$   
 $y = Cx + Du$

Parameters

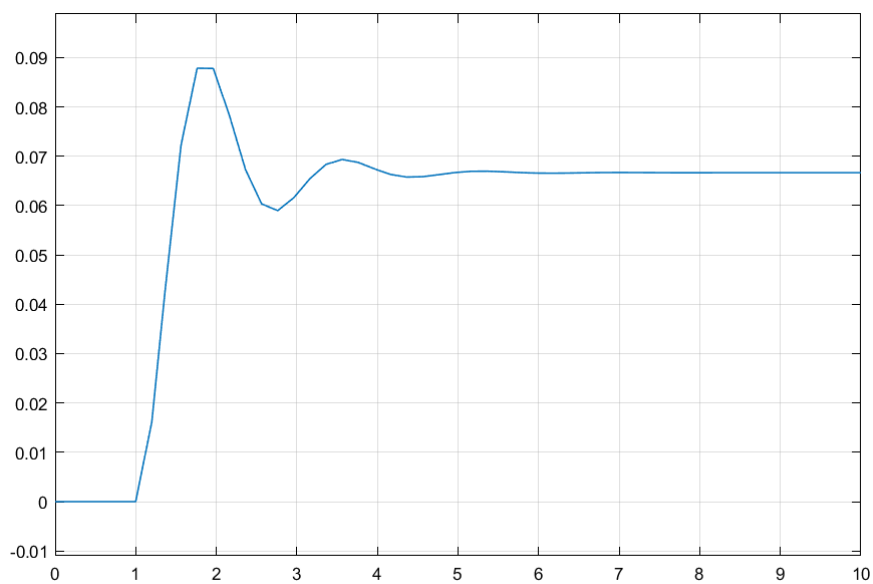
A:

B:

C:

D:

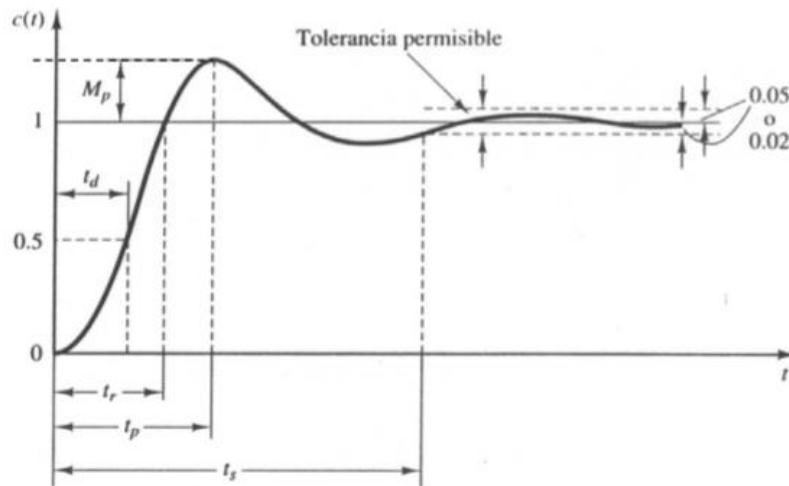
Obteniendo como era de esperar, la misma gráfica que en el apartado anterior:



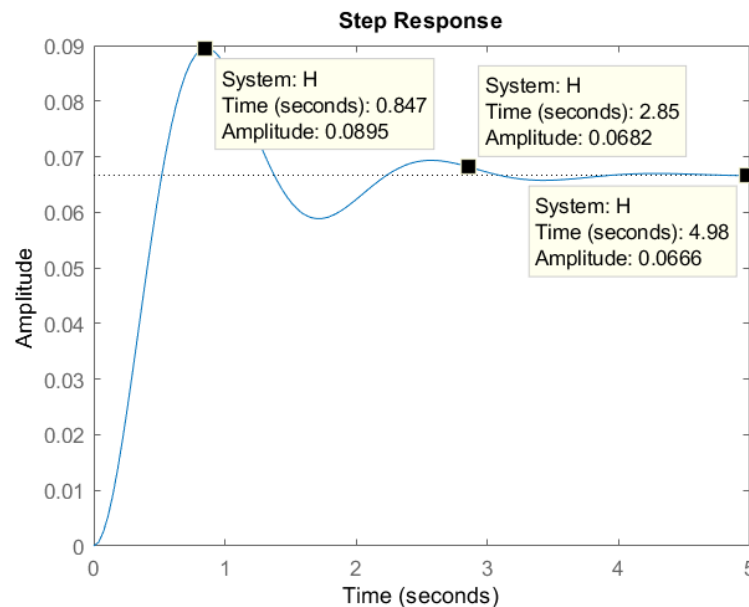
3. Sobre la respuesta gráfica del apartado a) del ejercicio anterior, se deberán tomar con precisión las siguientes medidas:

- Sobredisparo  $M_p(\%)$  (overshoot) en %.
- Tiempo de establecimiento  $t_s$  (settling time) (dentro de margen de  $\pm 5\%$  del valor final).

Nota: Comprobar que los resultados coinciden con los obtenidos con la función `stepinfo` de Matlab.



El valor de consigna se establece en torno a 0.0666, por lo que el 5% respecto al mismo representa un 0.00333, por lo que la tolerancia permisible estará entre 0.063327 y 0.06993



Observamos así, como el sobre-disparo se sitúa en torno a 0.0895 lo que correspondería a un 34,18%  $((0.0895-0.0667)*100/0.0667)$  y el tiempo de asentamiento es de 2.85s, con una amplitud dentro de los márgenes de tolerancia permisible del 5% con un valor de 0.0682.

Si comparamos ahora con los resultados proporcionados por la función 'stepinfo' tenemos:

```
>> stepinfo(H)

ans =

    struct with fields:

        RiseTime: 0.3500
        SettlingTime: 2.8803
        SettlingMin: 0.0589
        SettlingMax: 0.0895
        Overshoot: 34.2330
        Undershoot: 0
        Peak: 0.0895
        PeakTime: 0.8474
```

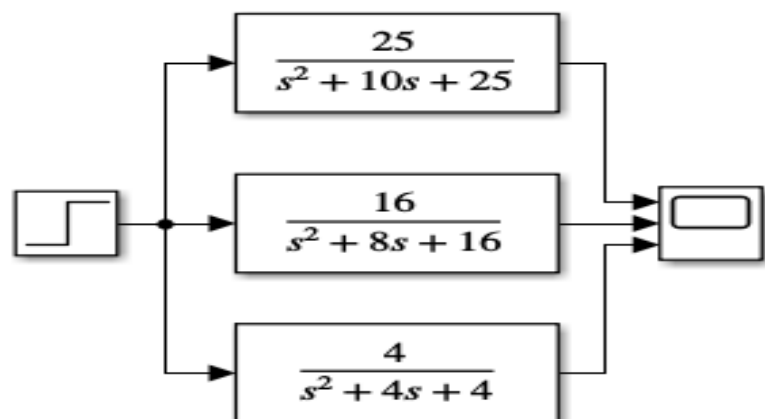
Podemos ver como las mediciones se aproximan prácticamente a los valores obtenidos con la función, que nos indica un valor de 2.88s para el tiempo de asentamiento y un 34.23% para el sobre-disparo.

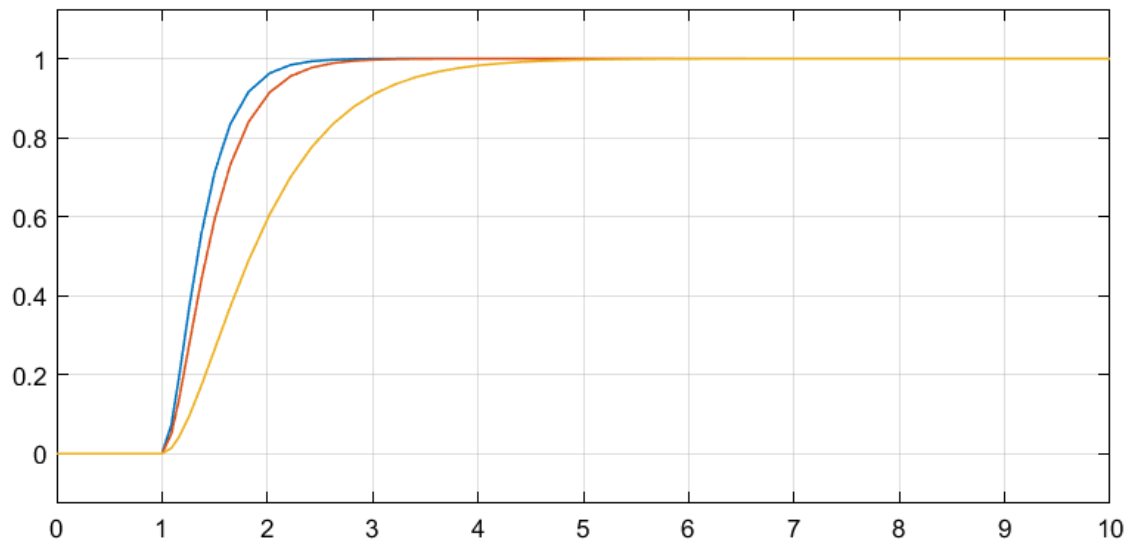
4. Obtener con Simulink la respuesta a un escalón y señal cuadrada de los distintos tipos (con ejemplos numéricos) de f.d.t. de 2º orden (subamortiguado, sobreamortiguado, con amortiguamiento crítico, etc...

Amortiguamiento Crítico:

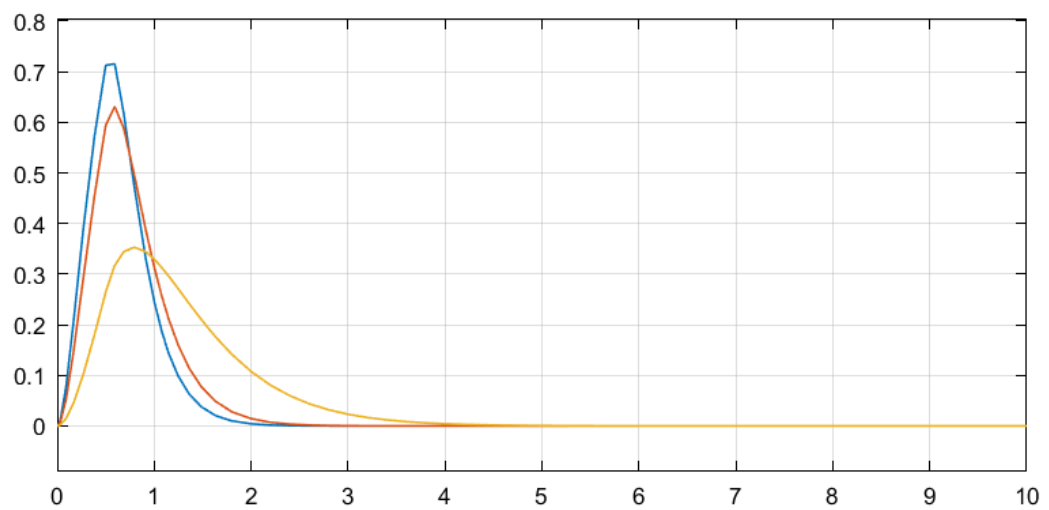
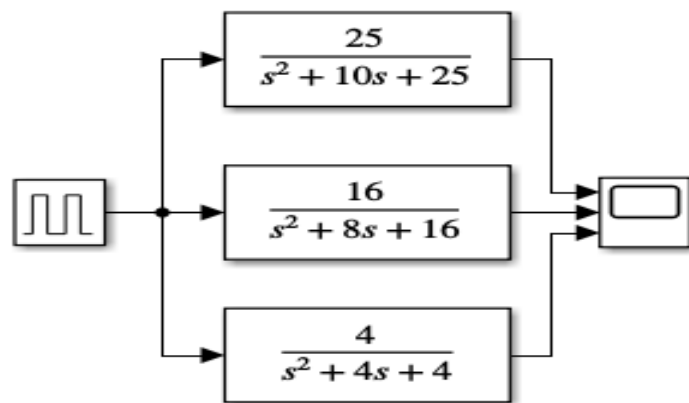
Para el caso de amortiguamiento crítico, debemos considerar  $\xi = 1$ , y mostraremos diferentes valores de  $\omega_n = (2 \text{ (amarilla)}, 4 \text{ (roja)}, 5 \text{ (azul)})$ .

- Escalón:





- Señal Cuadrada:

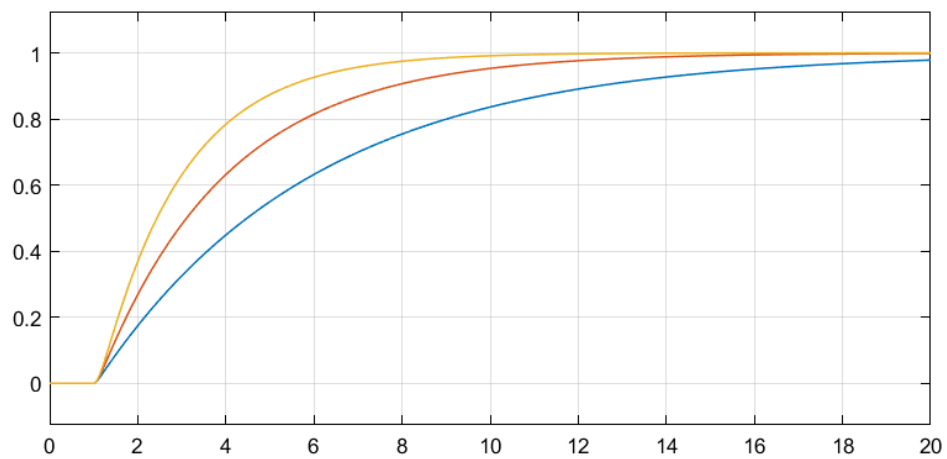
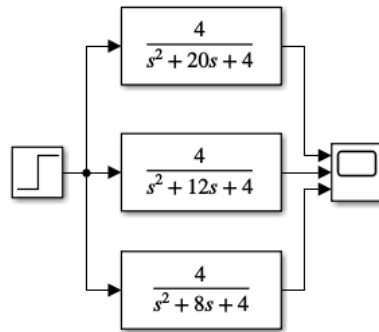




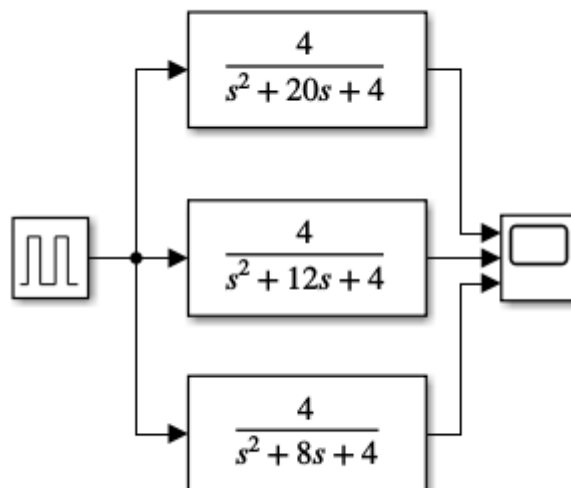
Sobreamortiguamiento:

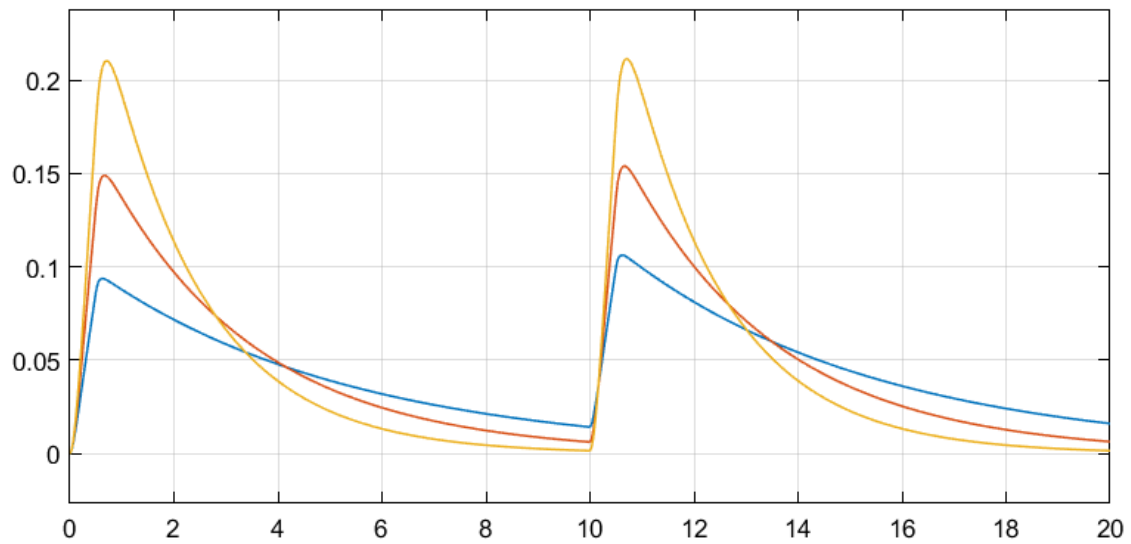
Para el caso de Sobreamortiguamiento, debemos considerar  $\xi > 1$ , mostrando los resultados para diferentes valores (2 (amarillo), 3 (rojo), 5 (azul)) y un valor de  $\omega_n = 2$ .

- Escalón:



- Señal Cuadrada:

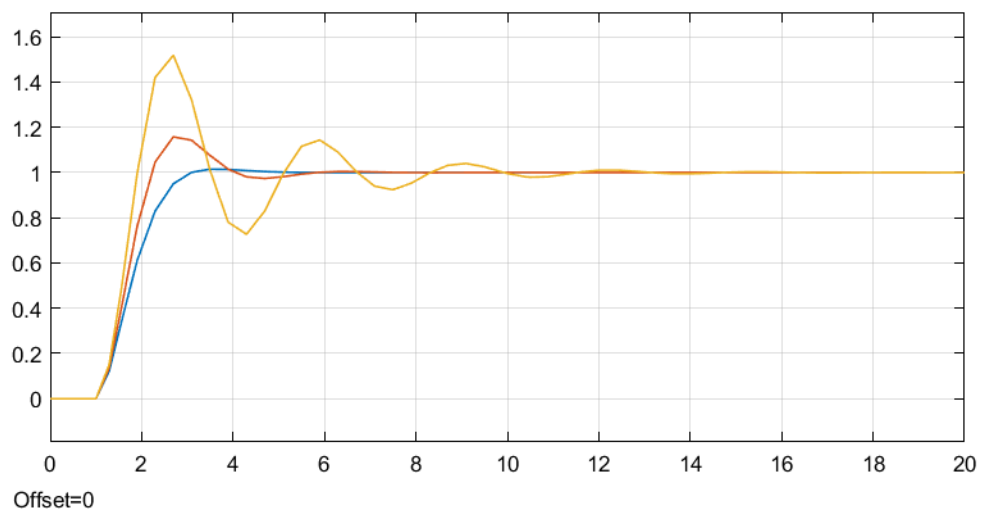
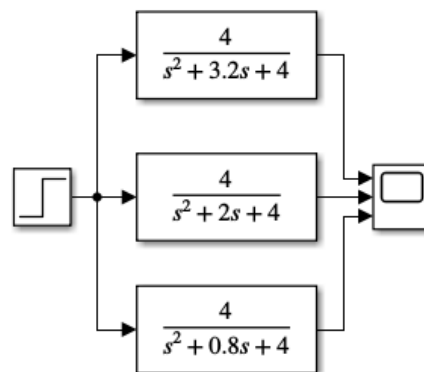




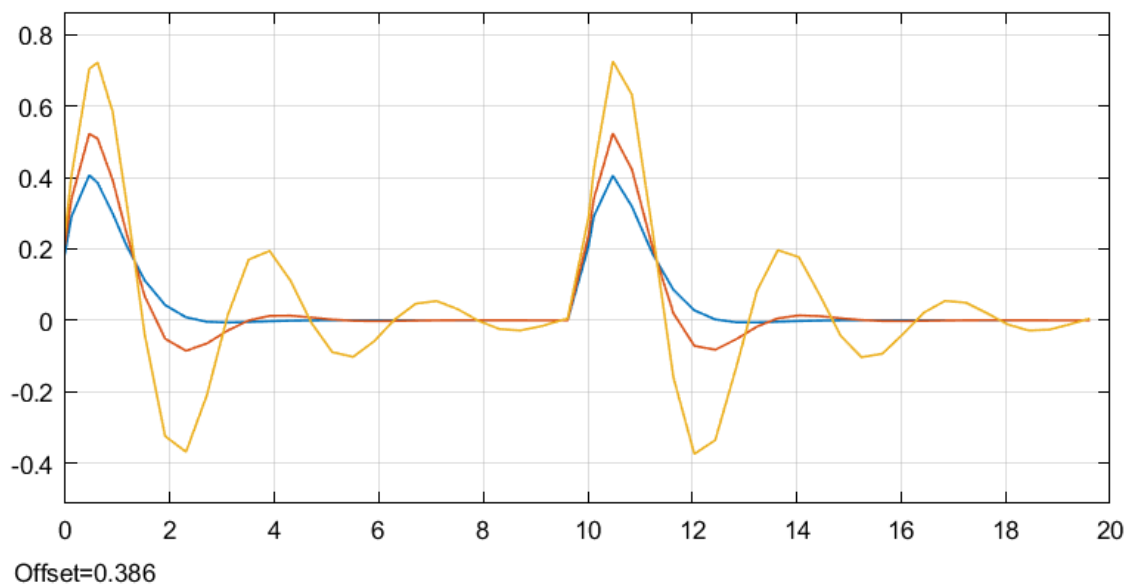
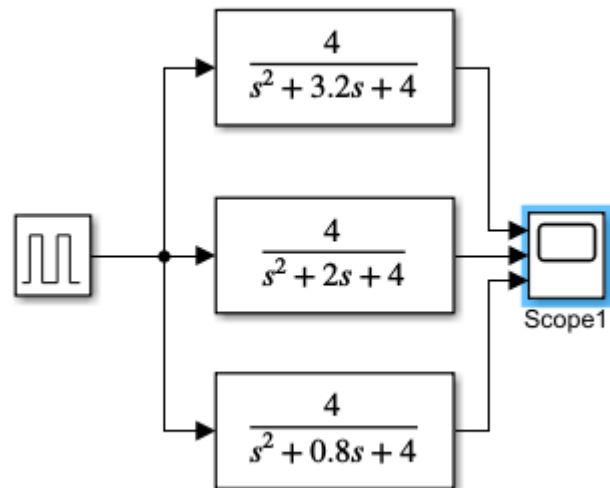
Sub-amortiguamiento:

Para el caso de Sub-amortiguamiento, debemos considerar  $0 < \xi < 1$ , mostrando los resultados para diferentes valores (0.2 (amarillo), 0.5 (rojo), 0.8 (azul)) y un valor de  $\omega_n = 2$ .

- Escalón:

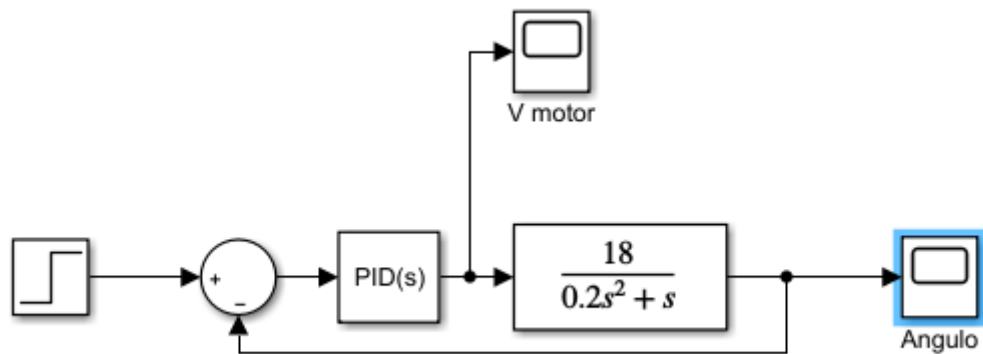


- Señal Cuadrada:

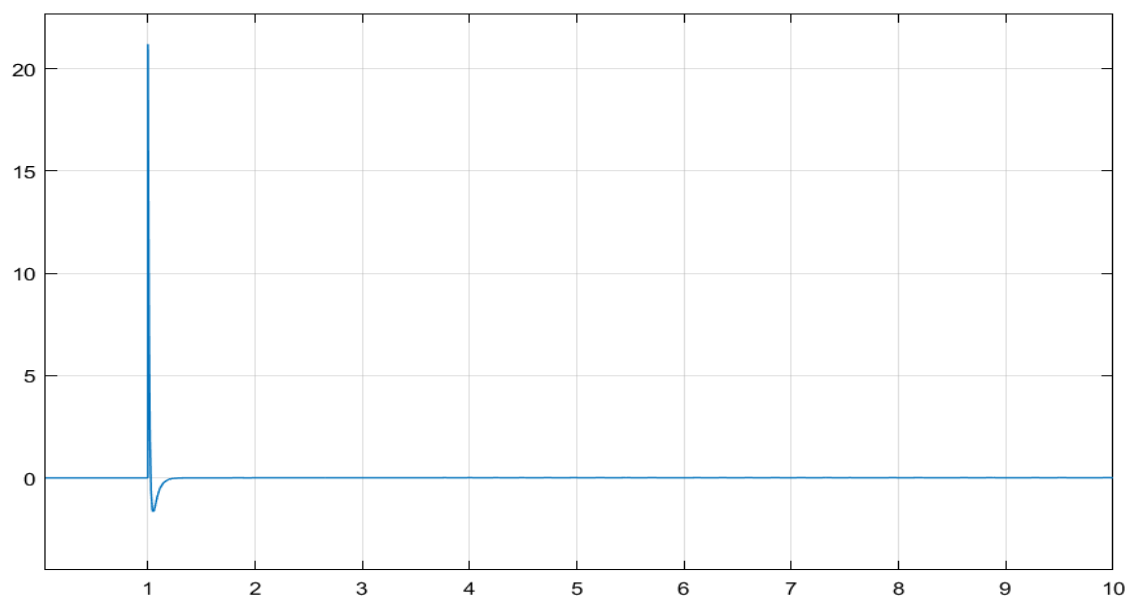
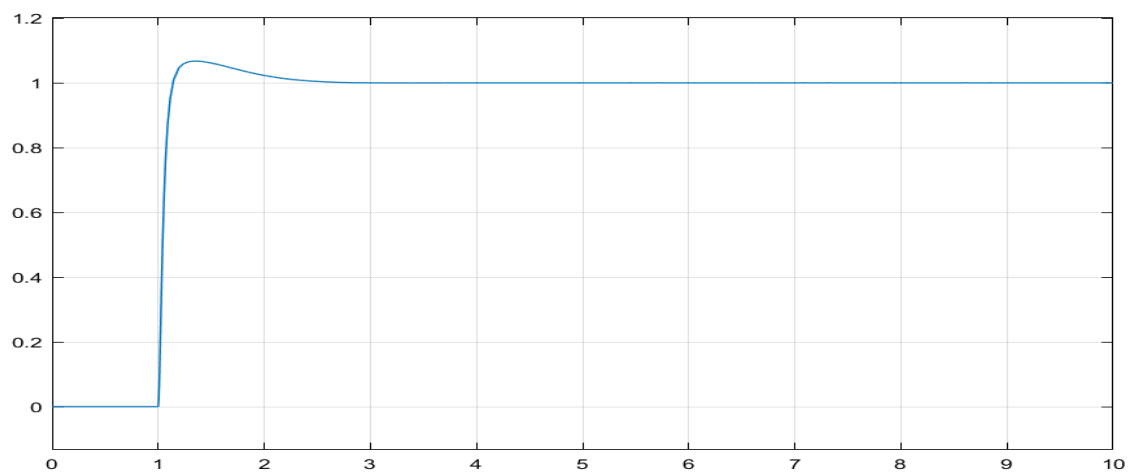


5. Un servomotor tiene la f.d.t:  $G(s)=18/(0.2s^2+s)$ . Realizar con Simulink un sistema de control de posición angular con control PID (módulo de Simulink). Obtener las distintas respuestas a un escalón unitario probando con diferentes valores de las constantes  $K_p$ ,  $K_i$  y  $K_d$  del controlador PID. Hacer el ajuste con pidtune.

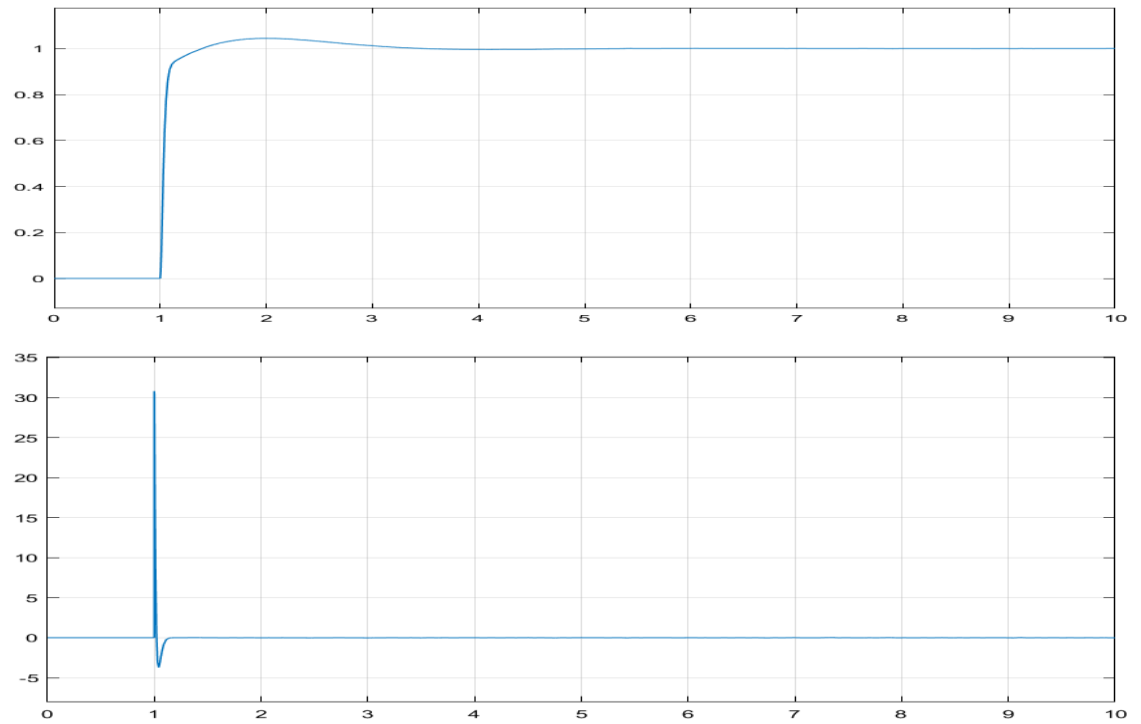
Partimos del siguiente controlador PID para el control del servo-motor de acorde a la función de transferencia dada:



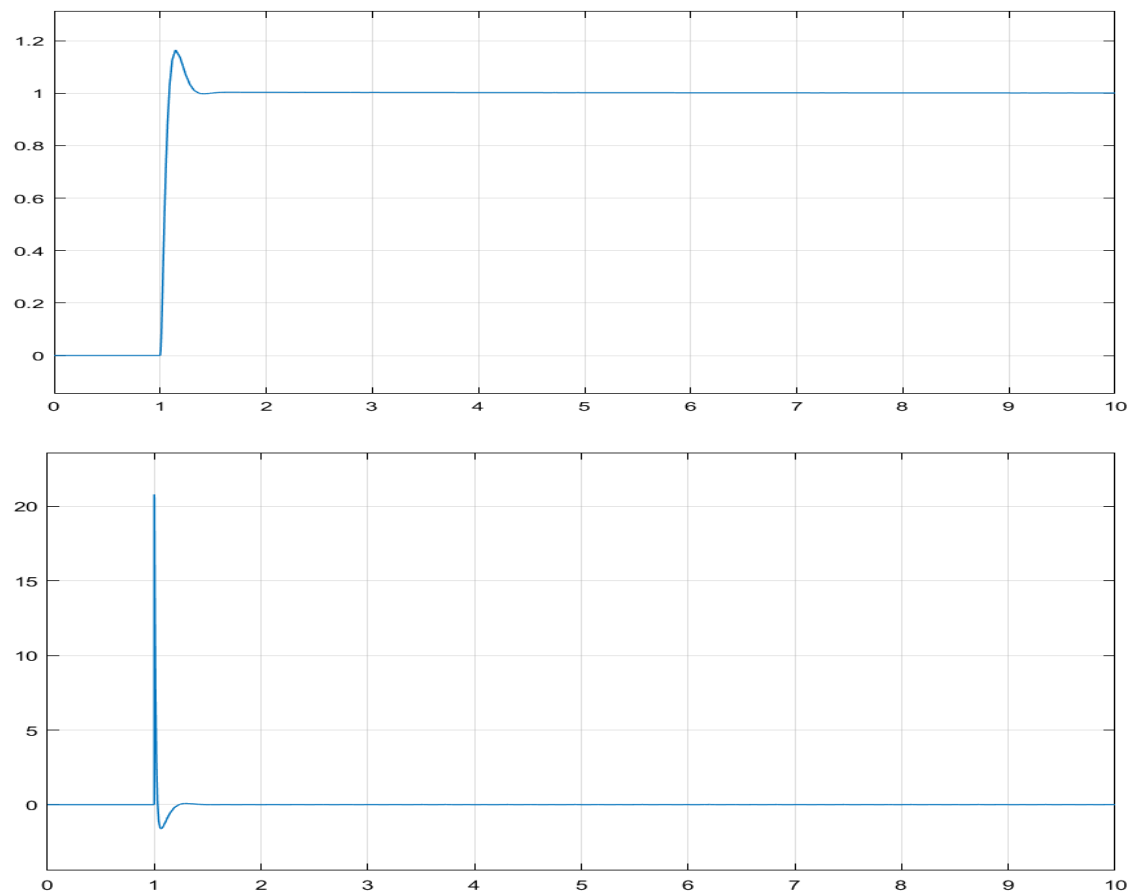
Para los valores de  $k_p = 1.2$ ,  $k_d = 0.2$ ,  $k_i = 2$ , obtenemos la siguiente respuesta para el ángulo y el voltaje del motor:



Para los valores de  $k_p = 0.8$ ,  $k_d = 0.3$ ,  $k_i = 1.2$ , obtenemos la siguiente respuesta para el ángulo y el voltaje del motor:



Para los valores de  $k_p = 2.8$ ,  $k_d = 0.18$ ,  $k_i = 0.5$ , obtenemos la siguiente respuesta para el ángulo y el voltaje del motor:



Ahora empleamos los valores de las constantes proporcionados por la función `pidtune` hallados en Matlab:

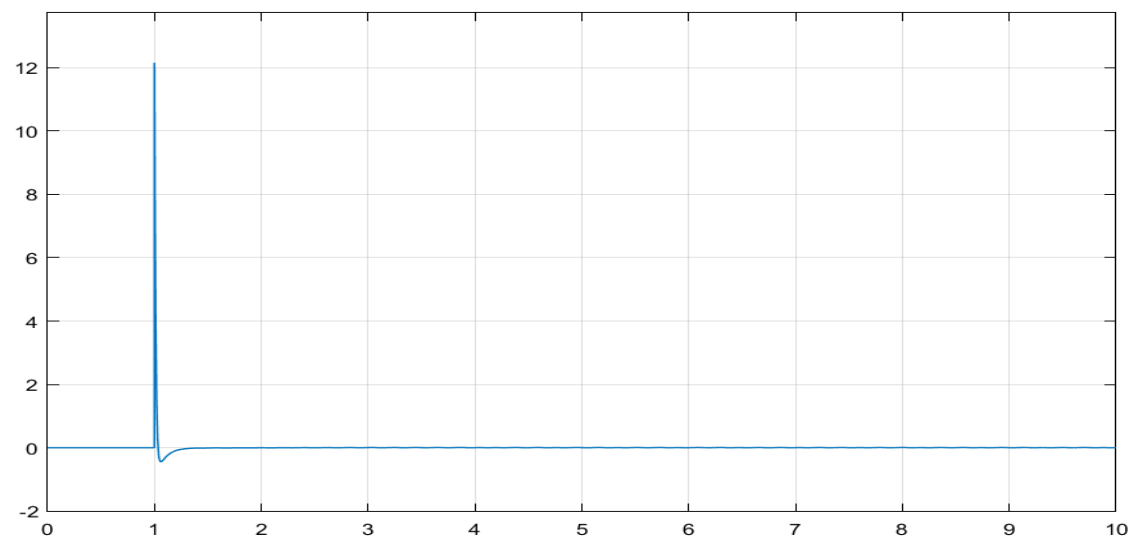
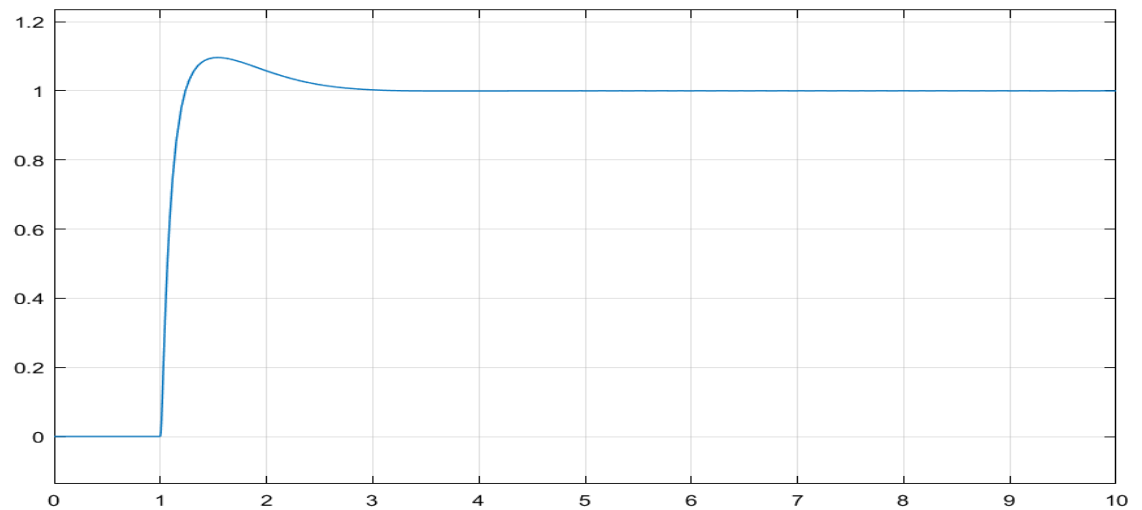
```
s = tf('s');  
sys = 18/(0.2*s^2 + s);  
fdt_PID = pidtune(sys, 'pid', 10);  
fdt_PID =
```

$$K_p + K_i * \frac{1}{s} + K_d * s$$

with  $K_p = 0.654$ ,  $K_i = 0.93$ ,  $K_d = 0.115$

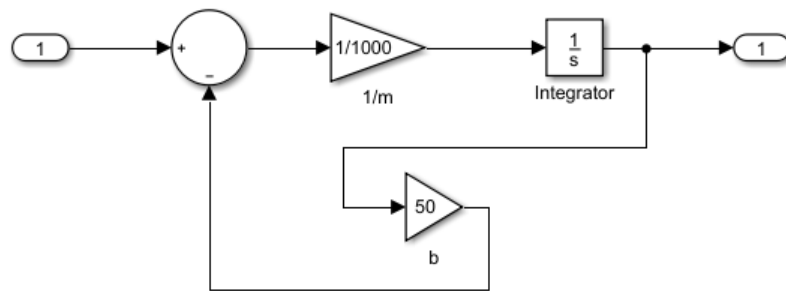
Continuous-time PID controller in parallel form.

Si introducimos dichos valores en el controlador obtenemos los siguientes resultados:

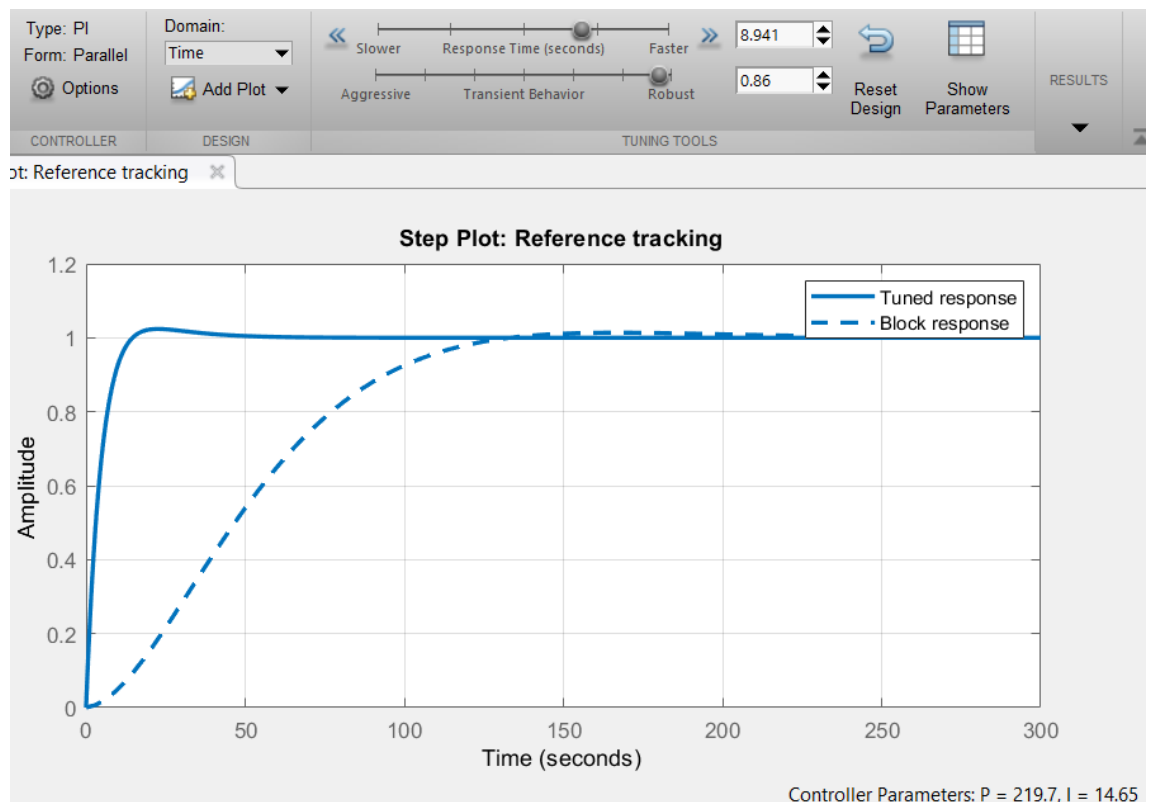


6. Simular la respuesta con Simulink de un sistema de control de velocidad de cruce de un automóvil (con controlador PI). Realizar el ajuste automático.

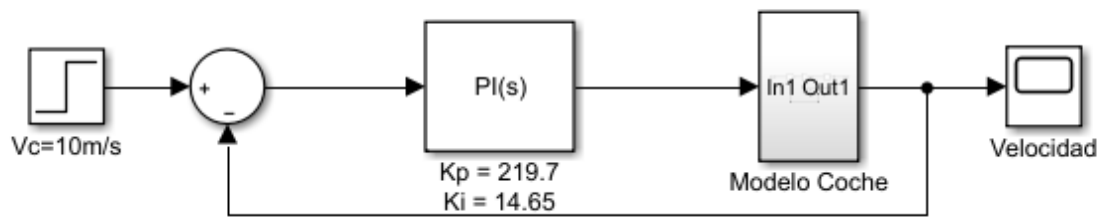
En primer lugar, creamos un subsistema para la simulación del modelo de velocidad del coche, teniendo en cuenta un valor para la constante de la masa del coche de 1000kg, y un valor para la constante de fricción de 50 N\*sec/m.



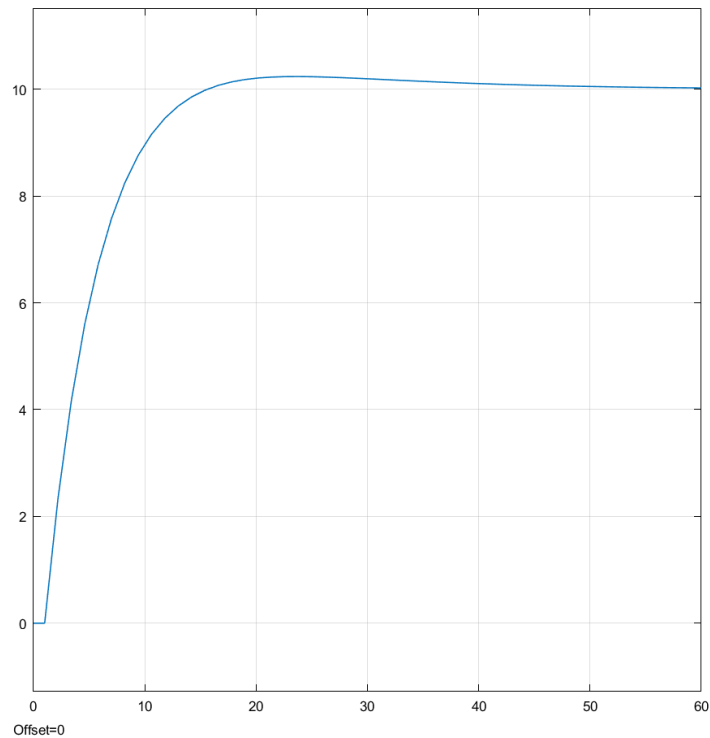
Empleamos un controlador PI con el valor de consigna de 10m/s. Los valores de las constantes  $K_p$  y  $K_i$  los calcularemos a partir del ajuste automático realizado por 'tune'.



Usando así un valor de  $K_p = 219.7$  y  $K_i = 14.65$ .

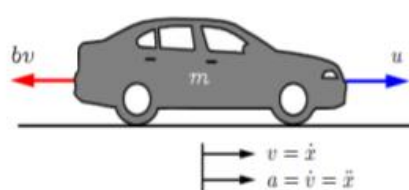


Obteniendo la siguiente respuesta:



Atendiendo a los valores dados, el coche tardaría alrededor de 50 segundos en estabilizar su velocidad en torno a 10 m/s.

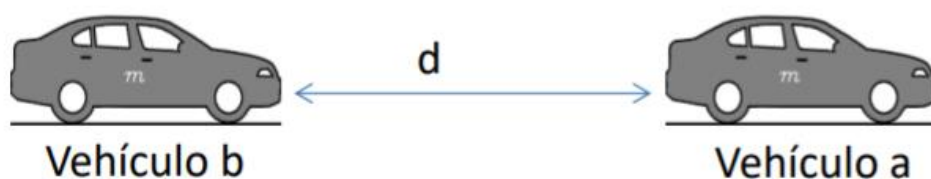
7. El modelo lineal simplificado de un vehículo de masa  $m$ , sometido a fricción (constante  $b$ ), y con un motor que aplica una fuerza  $F$  viene determinado por:



$$F - fr = ma$$

$$fr = bv$$

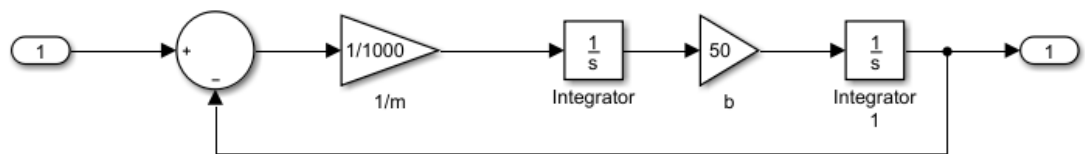
$$F - \left(b \frac{dx}{dt}\right) = m \frac{d^2x}{dt^2}$$



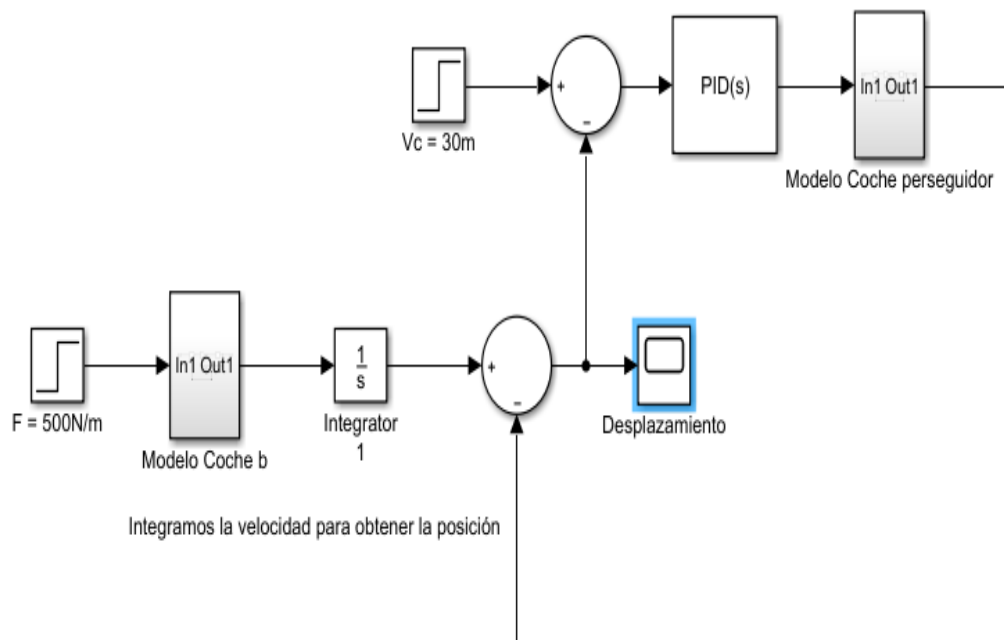


Realizar mediante Simulink un sistema de control PID de seguimiento de vehículos, que controle la fuerza del motor del coche perseguidor, de forma que se consiga mantener una distancia de seguridad (por ejemplo,  $d=30$  metros) en el tiempo menor posible. El vehículo perseguidor conoce en cada momento la posición  $x$  y la velocidad  $v$  del vehículo perseguido. El vehículo perseguido se mueve inicialmente acelerando desde la posición de parado hasta  $v=10$  m/s a partir de una fuerza  $F=500$  N.

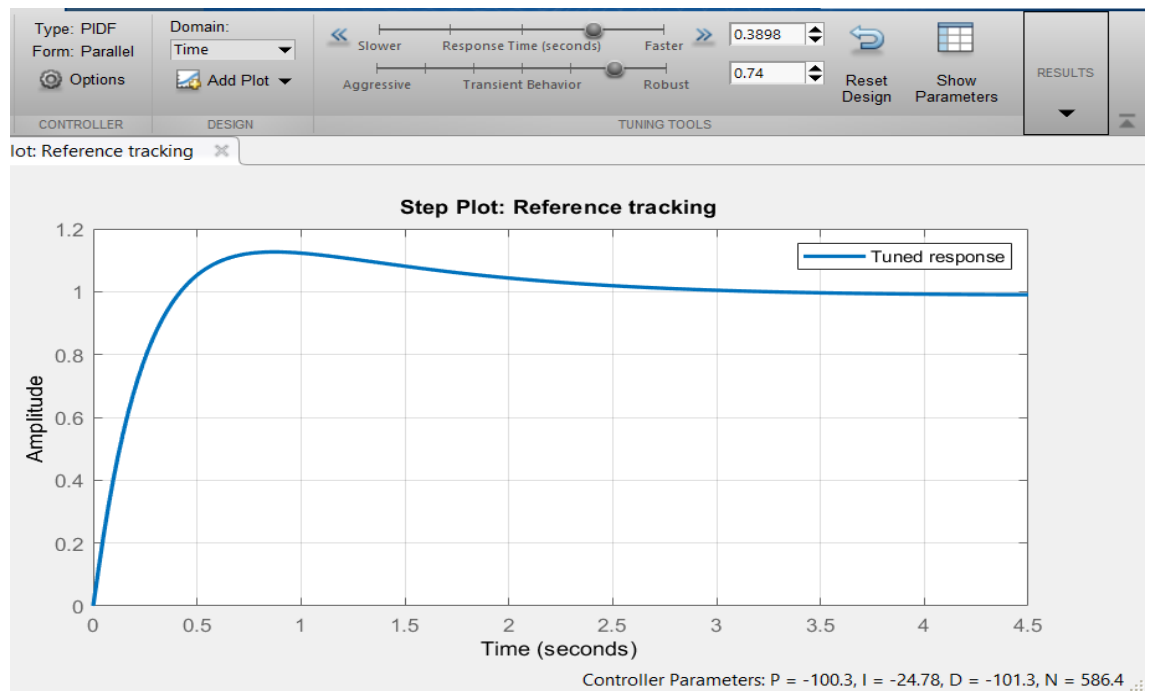
Para la realización de este ejercicio hemos reutilizado el modelo del coche generado para el ejercicio anterior, con la modificación en el del coche perseguidor, donde hemos añadido un integrador para obtener la posición a regular con el PID.



Lo que hacemos es obtener la posición de cada uno de los vehículos integrando la velocidad obtenida como salida de los modelos y esta es restada para hallar la diferencia, que es nuestra variable a medir, obteniendo así la separación entre los vehículos, e implementamos un PID con un valor de consigna establecido en 30m como se indica en el enunciado:



El ajuste del PID lo hemos realizado mediante la opción tune, atendiendo a los siguientes valores:



Como podemos apreciar en esta gráfica que nos muestra la distancia de separación, al principio, el coche a inicia su trayecto, por lo que la distancia de separación se incrementa, alcanzando un  $M_p$  de aproximadamente 34m, estabilizándose en 30 m tras 4 segundos.

