PRÁCTICA 2

Controladores PID

Análisis y Diseño

Álvaro Baena Nuevo 52237

A404

1. Introducción

Lo primero que se propone en la realización de esta práctica es usar el entorno gráfico simulink. Para ello iniciamos Matlab y escribimos en la ventana de command Windows la palabra "simulink" y damos a enter. Esto nos servirá para simular el sistema con el controlador PID.

Para describir la ecuación del PID se necesitara además los bloques integrador 'Integrator' y derivador 'derivative'

Para implementar el controlador PID necesitamos los bloques integrador, derivador y proporcional. Así conseguimos describir la ecuación del PID con bloques.

$$G_{PID} = \left(K_p + \frac{K_i}{S} + K_d S) \right)$$

Ejercicio : **cada grupo elige un sistema de 1er o 2º orden con valores distintos de ganancia estática, constante de tiempo, parámetro de amortiguamiento, frecuencia natural no amortiguado y se repite el diseño del PID siguiendo los pasos anteriores utilizando los métodos de diseño de PID (teoría de control, apuntes de clase)**

Mi propuesta es el control de tensión de un generador. Este sistema consta del controlador PID, de un amplificador, de un inductor, del generador y de una fase de realimentación por medio de un sensor.

La FDT del amplificador será:

$$Ga(s) = \frac{10}{0.1s + 1}$$

La FDT del inductor:

$$Gi(s) = \frac{1}{0.4s + 1}$$

La FDT del generador:

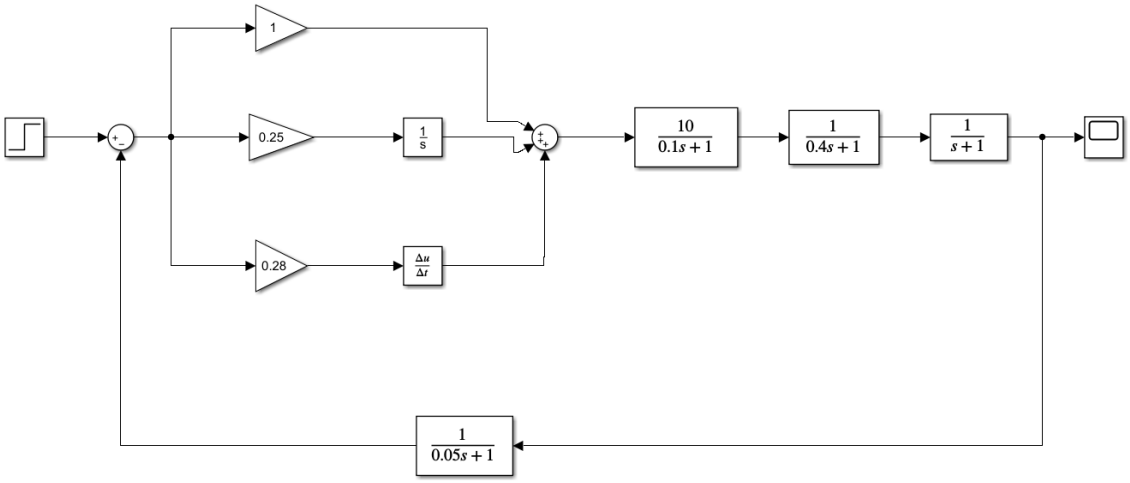
$$Gg(s) = \frac{1}{s + 1}$$

La FDT del sensor:

$$Gs(s) = \frac{1}{0.05s + 1}$$

Nuestro PID tendrá las siguientes constantes:

- Kp=1
- Ki=0.25
- Kd=0.28



Primero analizamos el sistema sin tener en cuenta el PID. La G(s) del sistema será:

$$G(s) = \frac{250}{(s+10)(s+2.5)(s+1)}$$

Y la H(s):

$$H(s) = \frac{20}{(s+20)}$$

Por tanto la función de transferencia del sistema realimentado será:

$$M(s) = \frac{G(s)}{1+G(s)H(s)} = \frac{250}{s4+33.5s3+307.5s2+775s+5500}$$

Los polos de esta ecuación característica serán:

S1=-0.2±4.48j

S2=-16.55±0.47j

Los dos polos conjugados de S2 se descartan al estar muy alejados.

Teorema del valor final:

Lim $M_{s->0}(s)=\frac{1}{22}$

El sistema reducido equivalente será:

S1=-0.2±4.48j=-σ ± Wdj=-ξWn ± $Wn(\xi^2-1)^{\wedge}\frac{1}{2}$

$Wn(\xi^2-1)^{\wedge}\frac{1}{2}=4.48$

$Wn^2=\frac{4.48^{\wedge}2}{0.98}=20.11$

$$Meq(s) = \frac{K}{s^2+2*0.045\sqrt{20.11}+20.11} = \frac{K}{s2+0.4s+20.11}$$

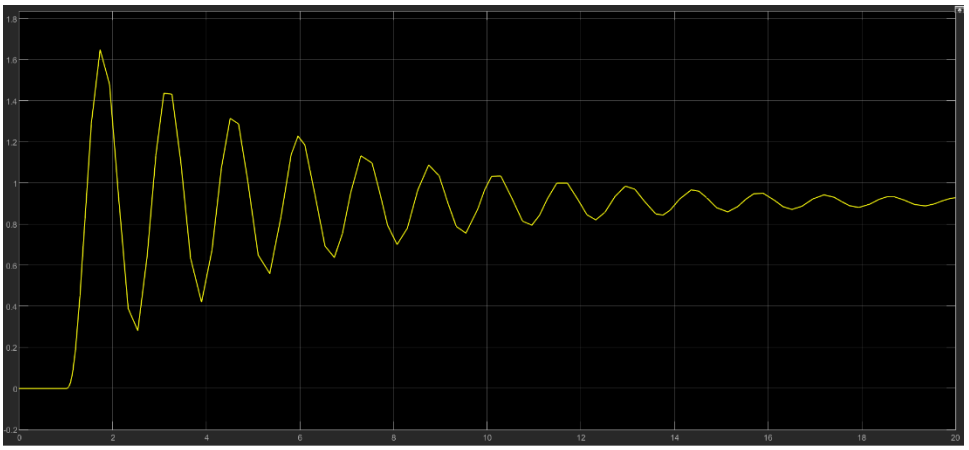
Lim $Meq_{s->0}(s)=\frac{K}{20.11}$

$\frac{1}{22}=\frac{K}{20.11}$

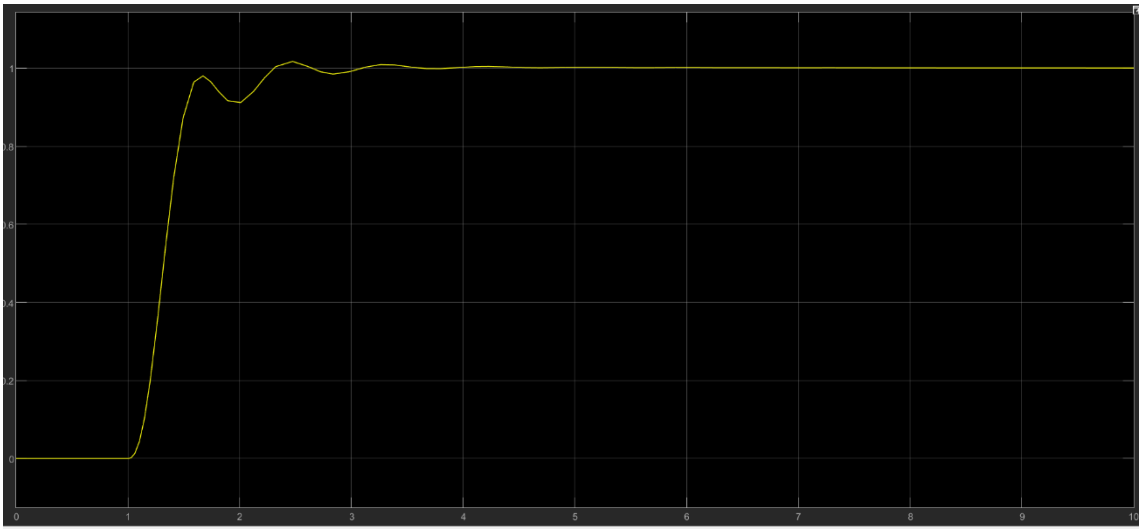
K=0.9

Meq(s)= $\frac{0.9}{s2+0.4s+20.11}$

Como podemos observar en la gráfica el sistema sin PID tiende a 0.9



Introduciendo el PID mejoramos la ganancia estática a 1 como se puede ver en la siguiente gráfica:



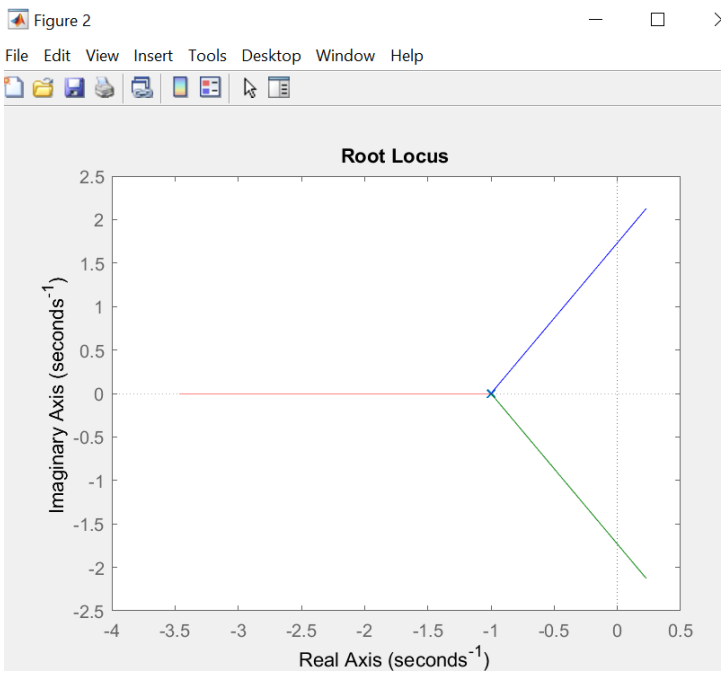
2. Métodos de diseño del PID

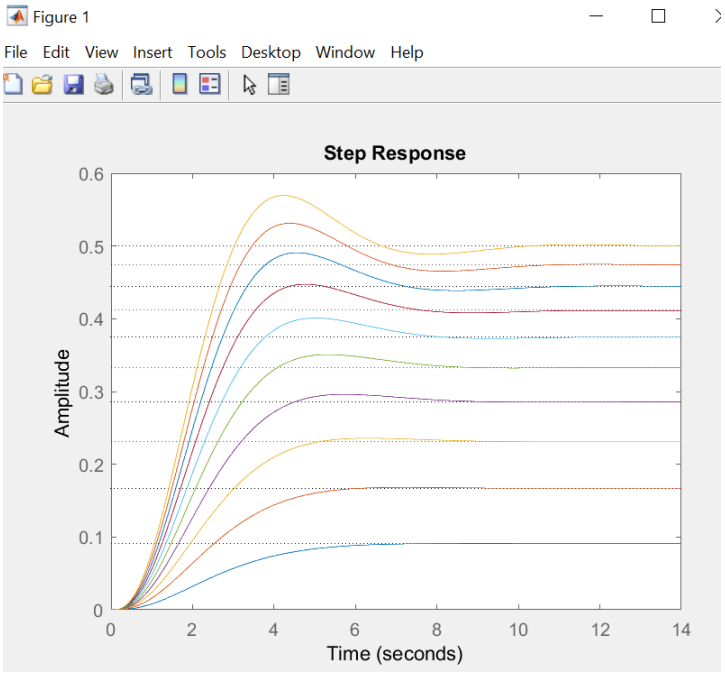
2.1. Controladores PID

Ejemplo 1: dada una FDT de un sistema de 3º orden G(s) = 1/(s + 1)3. Si se elige un controlador P, la respuesta del sistema en lazo cerrado para diferentes valores de Kp, se puede obtener utilizando el siguiente código MATLAB:

```
G=tf(1,[1,3,3,1]);
 for Kp=[0.1:0.1:1],
G_c=feedback(Kp*G,1);
step(G_c),
hold on;
end figure;
rlocus(G,[0,15])
```

Ejercicio : **Comentar los resultados obtenidos con respecto a la mejor respuesta en régimen permanente**





Reducir la acción proporcional, reducirá la velocidad de respuesta del sistema y aumentará su error permanente. Al aumentar la acción proporcional existe un punto de equilibrio en el que se consigue suficiente rapidez de respuesta del sistema y reducción del error, sin que el sistema sea demasiado inestable.

En los gráficos anteriores se puede observar el efecto de aumentar progresivamente la acción proporcional en un control de posición.

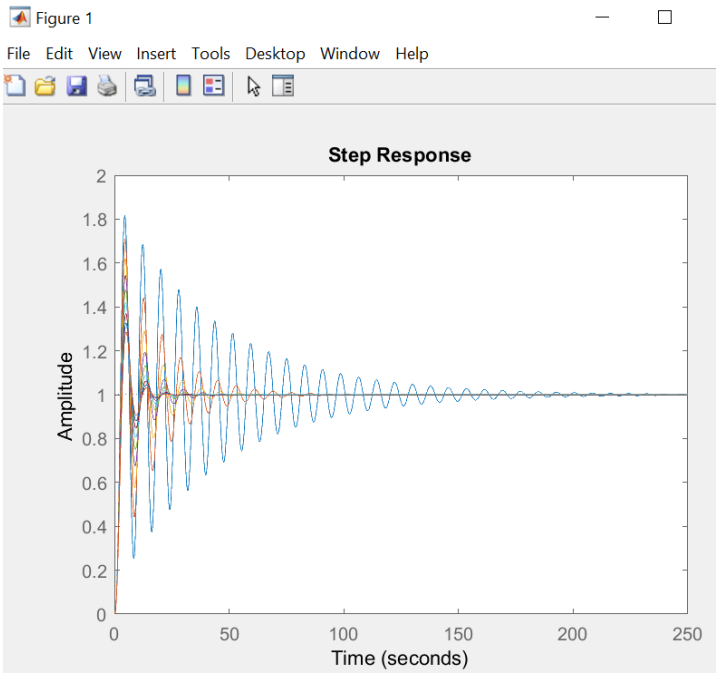
- Con una acción proporcional pequeña kp=0.1, el sistema es lento ya que tarda 6.6 segundos en alcanzar la posición deseada y su error es de 0. 9.
- Con una acción proporcional kp=0.2, el sistema es más rápido que en el caso anterior ya que tarda 5.45 segundos en alcanzar la posición deseada pero su error es de 0.833.
- Con una acción proporcional kp=0.3, el sistema no es más rápido que en el caso anterior, tardando 6.76 segundos en alcanzar la posición deseada pero su error es de 0.769.
- Con una acción proporcional kp=0.4, el sistema tarda 7.26 segundos en alcanzar la posición deseada pero su error es de 0.714.
- Con una acción proporcional kp=0.5, el sistema tarda 7.81 segundos en alcanzar la posición deseada con un error es de 0.667.
- Con una acción proporcional kp=0.6, el sistema tarda 6.91 segundos en alcanzar la posición deseada con un error es de 0.625.
- Con una acción proporcional kp=0.7, el sistema tarda 6.85 segundos en alcanzar la posición deseada con un error es de 0.58.
- Con una acción proporcional kp=0.8, el sistema tarda 6.61 segundos en alcanzar la posición deseada con un error es de 0.55.
- Con una acción proporcional kp=0.9, el sistema tarda 6.4 segundos en alcanzar la posición deseada con un error es de 0.526.
- Con una acción proporcional kp=1, el sistema tarda 8.31 segundos en alcanzar la posición deseada con un error es de 0.5.

A partir de k=1 se consigue disminuir todavía más el error permanente, pero la velocidad de respuesta no aumenta porque el sistema se vuelve tan inestable que la posición tarda mucho en establecerse en su estado final. Por lo tanto, la mejor opción es con Kp=0.9 donde presenta una sobreoscilación del 11.8% y un error permanente de 0.526. Si se desea mejorar esta respuesta hay que incorporar otro tipo de control.

Ejemplo 2

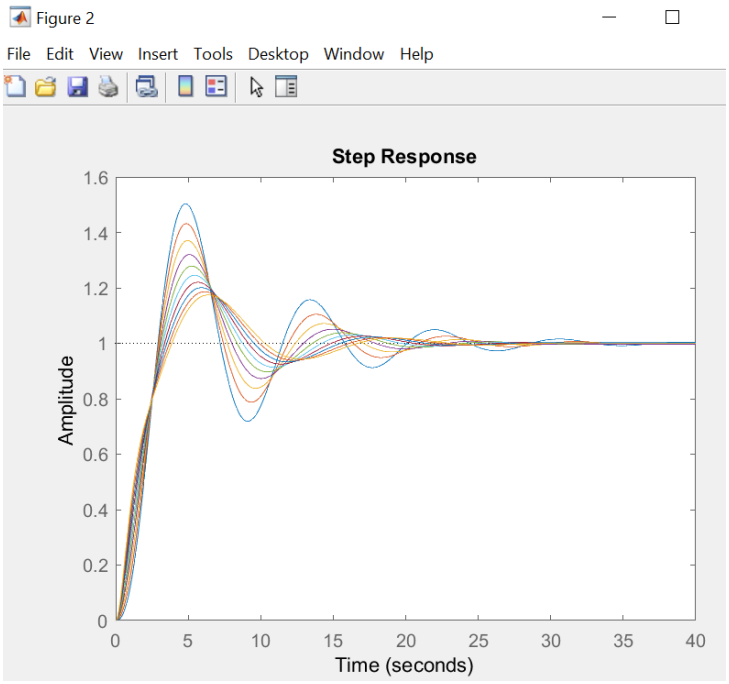
a)Si se fija Kp=1 y se aplica un controlador PI, la respuesta del sistema en lazo cerrado para diferentes valores de Ti se puede obtener utilizando el siguiente código MATLAB:

```
Kp=1; s=tf('s');
for Ti=[0.7:0.1:1.5]
Gc=Kp*(1+1/Ti/s);
G_c=feedback(G*Gc,1);
step(G_c),
hold on;
end figure
```



Como el error con la acción proporcional es demasiado grande se añade la acción integral variando su constante ki. Ahora el error máximo que se puede cometer es 0.8

a)Si se fija Kp=1 y Ti=1 y se aplica un controlador PID, la respuesta del sistema en lazo cerrado para diferentes valores de Td se puede obtener utilizando el siguiente código MATLAB:




```
Kp=1;

Ti=1;

s=tf('s');

for Td=[0.1:0.2:2]

Gc=Kp*(1+1/Ti/s+Td*s);

G_c=feedback(G*Gc,1);

step(G_c),

hold on;

end Figure
```

Ejercicio : Comentar los resultados obtenidos con respecto a la mejor respuesta en régimen transitorio

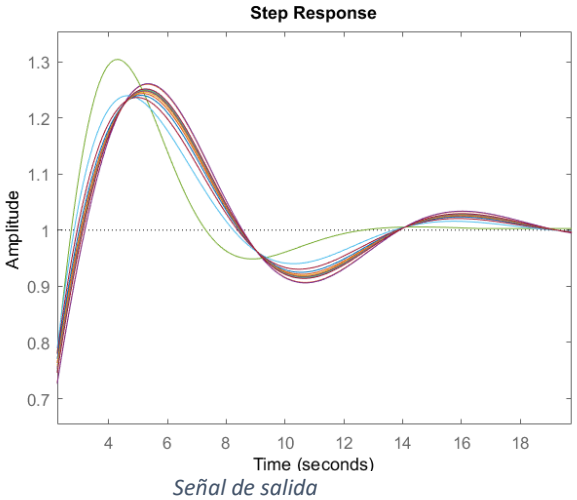
Si el sistema es demasiado inestable, se aumentará poco a poco la constante derivativa Kd para conseguir de nuevo estabilidad en la respuesta.

Podemos ver que el PID incrementa la velocidad y la estabilidad de la respuesta gracias a la adición de la acción derivativa. También podemos observar que independientemente del valor de kd que tengamos para el PID se produce menos sobreoscilación.

2.2. Efecto de la patada en la consigna

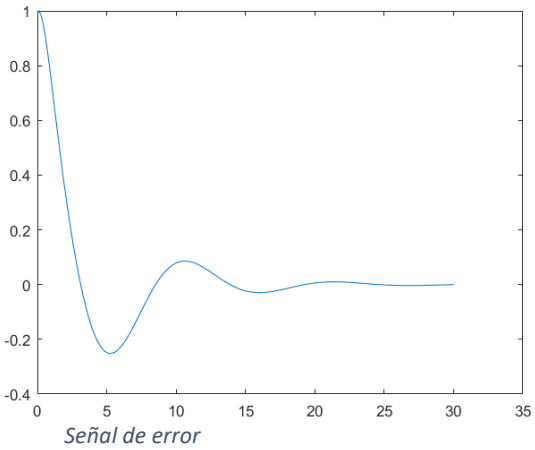
Ejemplo 2: Dado un sistema cuya FDT G(s) = 1/(s + 1)3. Los parámetros del PID son Kp = 1, Ti = 1, and Td = 1 simular el sistema con Matlab variando N. dibujar la respuesta transitoria y dibujar la respuesta del error (ver fig. 12).

```
Td=1; Gc=Kp*(1+1/Ti/s+Td*s);

step(feedback(G*Gc,1)),

hold on

for N=[100,1000,10000,1:10]

Gc=Kp*(1+1/Ti/s+Td*s/(1+Td*s/N));
```



Señal de salida

```
G_c=feedback(G*Gc,1);
step(G_c)
end figure;
[y,t]=step(G_c);
err=1-y;
plot(t,err)
```



Señal de error

Ejercicio : Determinar el mejor valor de N

Se puede ver que cuanto mayor es el valor de N más lento es el sistema y también gana en estabilidad

N=2

$$D = -\frac{KT_d s}{1+(T_d/N)s}\cdot y$$

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

Ejercicio: Cada grupo elige una FDT distinta de un sistema de control. Se diseña un controlador PID. Este paso debe prepararse por cada grupo antes de la práctica. El monitor comprueba la solución teórica al comenzar la práctica. Se utiliza el código Matlab para simular el sistema controlado. Comentar los resultados.

Mi FDT elegida es:

$$G(s) = \frac{1}{s(s+1)(s+5)}$$

El sistema realimentado será:

$$\frac{C(s)}{R(s)} = \frac{Kp}{s(s+1)(s+5)+Kp}$$

Ecuación característica: $s^3+6s^2+5s+kp=0$

Para estudiar su estabilidad utilizamos el criterio de Routh:

S3	1	5
S2	6	kp
S1	$\frac{30-kp}{6}$	
S0	kp	

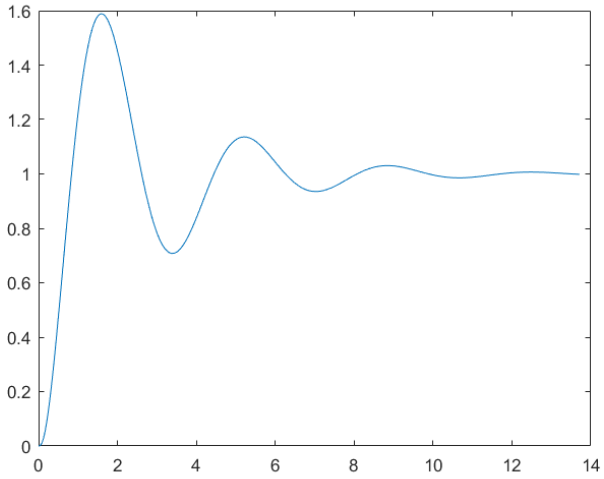
Por lo tanto la Kcrítica es 30. Para encontrar la frecuencia de la oscilación sostenida hacemos s=jw.

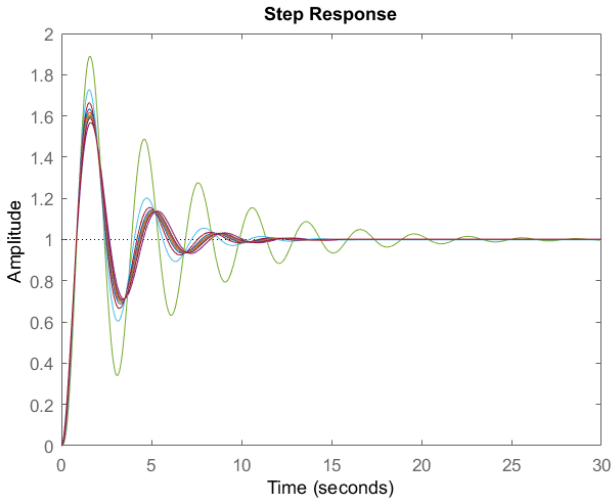
$jw^3+6jw^2+5jw+30=0$ $\quad$ $w=\sqrt{5}$ $\quad$ $Per=\frac{2\pi}{w}=\frac{2\pi}{\sqrt{5}}=2.809$

kp=0.6kcr=18

Ti=0.5Per=1.405 Ki=$\frac{Kp}{Ti} = 12.81$

Td=0.125Per=0.35124 Kd=Kp*Td=6.32





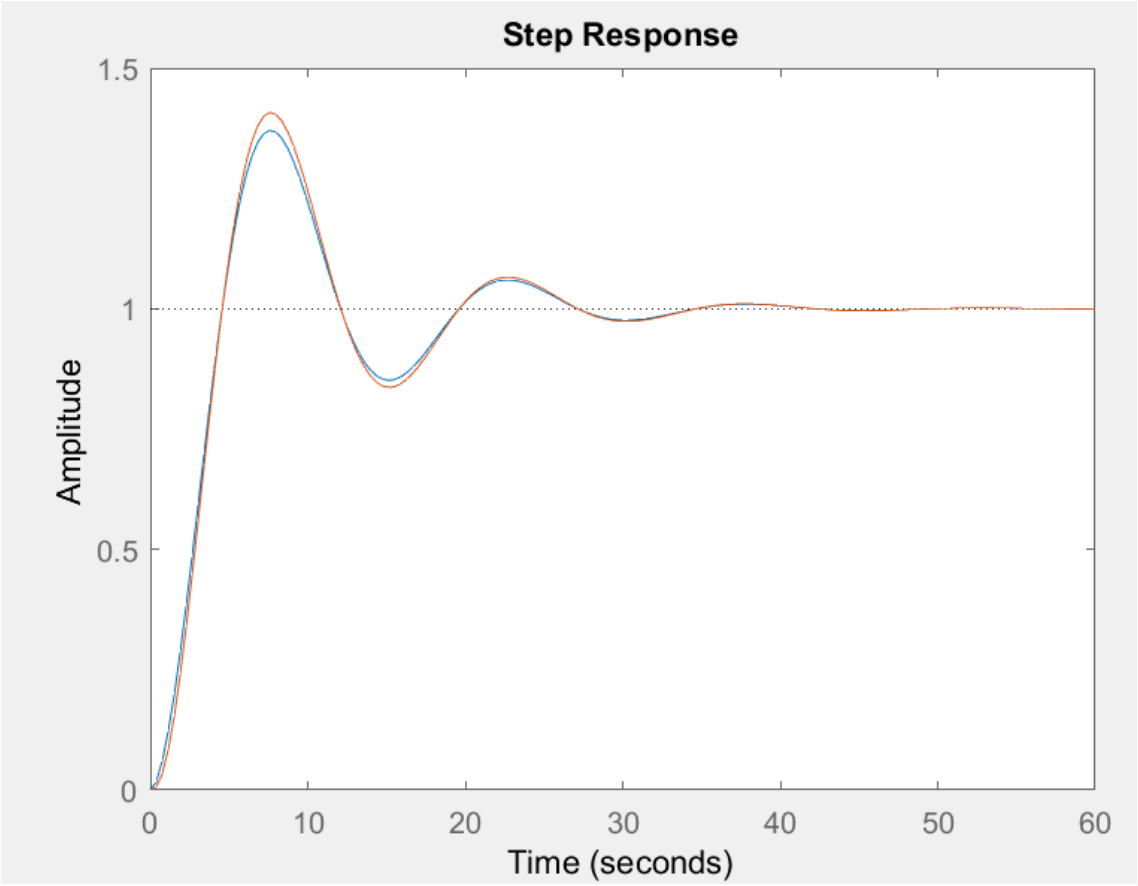
2.3. Controlador PID con la parte derivada en el lazo de realimentación

Ejercicio: Cada grupo elige una FDT distinta de un sistema de control. Se diseña un controlador PI-D. Este paso debe prepararse por cada grupo antes de la práctica. El monitor comprueba la solución teórica al comenzar la práctica. Se utiliza el código Matlab para simular el sistema controlado. Comentar los resultados.

Yo escojo la misma FDT que en el apartado anterior(G(s)=1/(s(s+1)(s+5))). El código en Matlab será:

```
G=tf(1,[1 6 5 1]);

Kp=1; Ti=1.405; Td=0.35124;

N=10;

s=tf('s');

Gc=Kp*(1+1/Ti/s+Td*s/(1+Td*s/N));

G_c=feedback(G*Gc,1);

Gc1=Kp*(1+1/s/Ti);

H=((1+Kp/N)*Ti*Td*s^2+Kp*(Ti+Td/N)*s+Kp)/(Kp*(Ti*s+1)*(Td/N*s+1));

G_c1=feedback(G*Gc1,H);

step(G_c,G_c1)
```

Las gráfica resultante es:

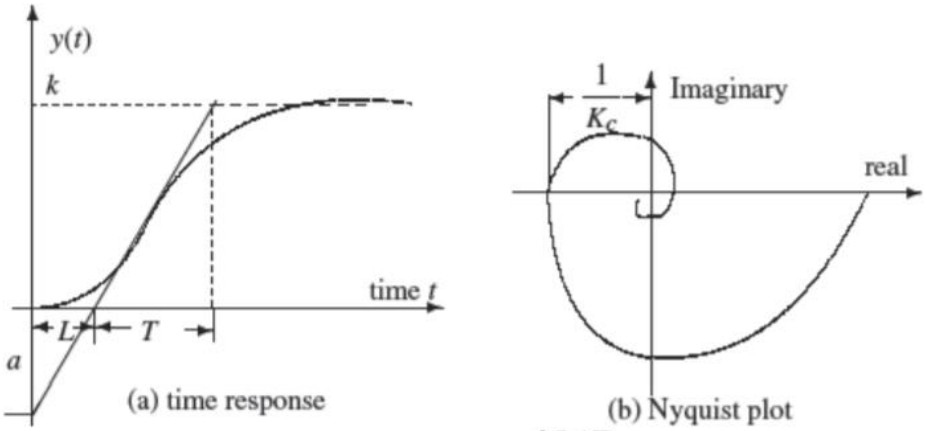


2.4 Métodos Empíricos de diseño

2.4.1. Ziegler-Nichols (ZN) respuesta al escalón

La formula de ZN se basa en un modelo de un sistema de primer orden con retraso (first-order plus dead time (FOPDT))

$$G(s) = \frac{k}{1+sT}\,e^{-sL}.$$



$$a = kL/T$$

Ejercicio: Cada grupo elige una FDT distinta de un sistema de control. Se diseña un controlador PID utilizando ZN. Este paso debe prepararse por cada grupo antes de la práctica. El monitor comprueba la solución teórica al comenzar la práctica. Se utiliza el código Matlab para simular el sistema controlado. Comentar los resultados.

Para este ejercicio parto de mi FDT:G(s)=1/(s(s+1)(s+5) y sus constantes:

Kp=18

Ti=1.405 Ki=$\frac{Kp}{Ti} = 12.81$

Td=0.35124 Kd=Kp*Td=6.32

Podemos calcular los parámetros de ZN a partir de la siguiente tabla:

Controller	from step response			from frequency response		
type	K_p	T_i	T_d	K_p	T_i	T_d
P	$1/a$			$0.5K_c$		
PI	$0.9/a$	$3L$		$0.4K_c$	$0.8T_c$	
PID	$1.2/a$	$2L$	$L/2$	$0.6K_c$	$0.5T_c$	$0.12T_c$

Ti=2L L=0.7025

Td=L/2 a=0.067

0.5Tc=Ti Tc=2.81

0.12Tc=Td

<u>El código a introducir será el siguiente:</u>

Primero vamos a hacer que Matlab nos devuelva los parámetros de ZN que previamente se han calculado teóricamente. A parte representará la respuesta del sistema al escalón.

```
s=tf('s');G=tf(1,[1 6 5 1]);
```

```
k=dcgain(G)
L=0.7025; T=2.81;
[Gc3,Kp3,Ti3,Td3]=zn(3,[k,L,T,10])
G_c3=feedback(G*Gc3,1); step(G,G_c3);
```

- La función zn() será:

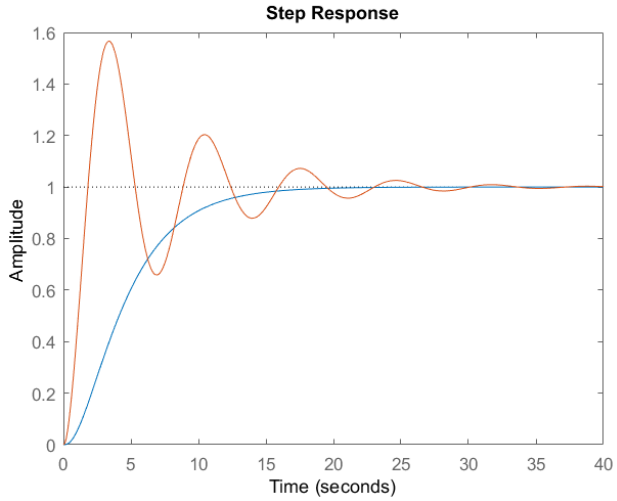
```
function [Gc,Kp,Ti,Td,H]=zn(key,vars)
Kp=[]; Ti=[];
Td=[]; H=1;
if length(vars)==4,
K=vars(1);
L=vars(2);
T=vars(3);
N=vars(4);
a=K*L/T;
if key==1, Kp=1/a;
elseif key==2, Kp=0.9/a;
Ti=3.33*L;
elseif key==3 || key==4, Kp=1.2/a;
Ti=2*L;
Td=L/2;
end
elseif length(vars)==3,
K=vars(1);
Tc=vars(2);
N=vars(3);
if key==1, Kp=0.5*K;
elseif key==2, Kp=0.4*K; Ti=0.8*Tc;
elseif key==3 || key==4, Kp=0.6*K; Ti=0.5*Tc; Td=0.12*Tc;
end
elseif length(vars)==5,
K=vars(1);
Tc=vars(2);
```

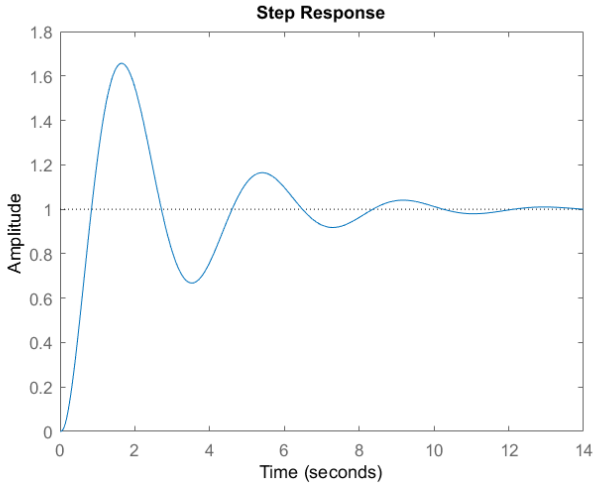
```
rb=vars(3);
N=vars(5);
pb=pi*vars(4)/180;
Kp=K*rb*cos(pb);
if key==2, Ti=-Tc/(2*pi*tan(pb));
elseif key==3 || key==4, Ti=Tc*(1+sin(pb))/(pi*cos(pb)); Td=Ti/4;
end
end
```

- La función escribirpid():

```
[Gc,H]=escribirpid(Kp,Ti,Td,N,key);
switch key
case 1, Gc=Kp;
case 2, Gc=tf(Kp*[Ti,1],[Ti,0]);
H=1;
case 3, nn=[Kp*Ti*Td*(N+1)/N,Kp*(Ti+Td/N),Kp];
dd=Ti*[Td/N,1,0];
Gc=tf(nn,dd); H=1;
case 4, d0=sqrt(Ti*(Ti-4*Td));
Ti0=Ti;
Kp=0.5*(Ti+d0)*Kp/Ti;
Ti=0.5*(Ti+d0);
Td=Ti0-Ti;
Gc=tf(Kp*[Ti,1],[Ti,0]);
nH=[(1+Kp/N)*Ti*Td, Kp*(Ti+Td/N), Kp];
H=tf(nH,Kp*conv([Ti,1],[Td/N,1]));
case 5, Gc=tf(Kp*[Td*(N+1)/N,1],[Td/N,1]);
H=1;
end
```

Esta función escribirpid () se usa para devolver los parámetros del controlador PID

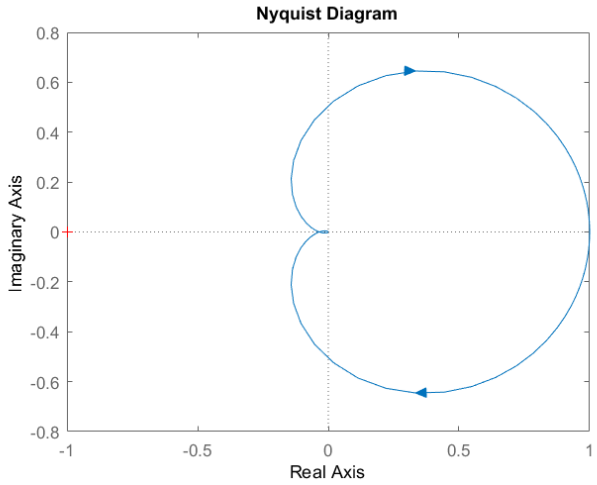




2.4.2. Ziegler-Nichols (ZN)-respuesta en frecuencia

Para representar la respuesta en frecuencia hay que poner el siguiente código:

```
G=tf(1, [1 6 5 1]);
nyquist(G); axis([-0.2,0.6,-0.4,0.4])
[Kc,pp,wg,wp]=margin(G);
[Kc,wg],
Tc=2*pi/wg;
[Gc3,Kp3,Ti3,Td3]=zn(3,[Kc,Tc,10]);
 [Kp3,Ti3,Td3];
G_c3=feedback(G*Gc3,1); step(G_c3)
```



2.4.3 Chien–Hrones–Reswick

El método de Ziegler-Nichols a veces presenta problemas ante cambios de consigna ya que la salida puede no responder de forma óptima, generalmente de pobre amortiguación.

El metodo de Chien–Hrones–Reswick PID se caracteriza por ser:

• Variante del método de ZN.

• Criterio de diseño:

– Rápida respuesta con 20% de sobreoscilación.

– Rápida respuesta con 0% de sobreoscilación.

• Ajuste de parámetros distintos para cambios de carga o cambios de SP.

• Para cambios de carga: uso de parámetros a y L.

• Para cambios de SP: uso de parámetros a, L y T.

• Consigue un 0% de sobreoscilación disminuyendo el valor de K y Td y aumentando

el valor de Ti con respecto al método de ZN.

Las siguientes tablas muestran los parámetros del PID utilizando este método.

Controller	with 0% overshoot			with 20% overshoot		
type	K_p	T_i	T_d	K_p	T_i	T_d
P	$0.3/a$			$0.7/a$		
PI	$0.35/a$	$1.2T$		$0.6/a$	T	
PID	$0.6/a$	T	$0.5L$	$0.95/a$	$1.4T$	$0.47L$

Ejercicio: Cada grupo elige una FDT distinta de un sistema de control. Se diseña un controlador PID utilizando CHR. Este paso debe prepararse por cada grupo antes de la práctica. El monitor comprueba la solución teórica al comenzar la práctica. Se utiliza el código Matlab para simular el sistema controlado. Comentar los resultados.

Se tiene la FDT anterior G(s)=1/(s(s+1)(s+5) y los parámetros de ZN: L=0.7025, a=0.067 y Tc=2.81

A partir de los parámetros de ZN se puede calcular los parámetros del PID utilizando el método de CHR. De forma teórica será:

- Con 0% de sobreoscilación:

Kp=0.6/a Kp=9

Ti=T Ti=2.81

Td=0.5L Td=0.35124

- Con 20% de sobreoscilación:

Kp=0.95/a Kp=14.2

Ti=1.4T Ti=3.93

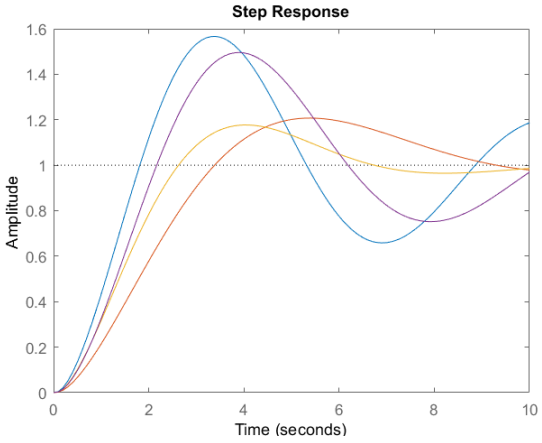
Td=0.47L Td=0.33

Aunque de igual forma estos nuevos parámetros los calcula Matlab. El código es el siguiente:

PID CHR 0% sobreoscilacion

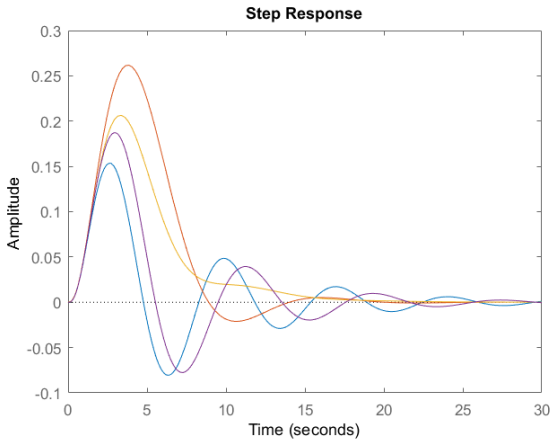
```
s=tf('s');G=tf(1,[1 6 5 1]);
k=dcgain(G)
L=0.7025; T=2.81; N=10;
```

```
[Gc1,Kp,Ti,Td]=zn(3,[k,L,T,N])
[Gc2,Kp2,Ti2,Td2]= chreswickpid (3,1,[k,L,T,N,0])
%señal referencia 0% sobreoscilacion
[Gc3,Kp3,Ti3,Td3]= chreswickpid (3,1,[k,L,T,N,20])
%señal referencia 20% sobreoscilacion
[Gc4,Kp4,Ti4,Td4]= chreswickpid (3,2,[k,L,T,N,0])
%reduccion perturbaciones 0% sobreoscilacion

step(feedback(G*Gc1,1),
feedback(G*Gc2,1),
feedback(G*Gc3,1),
feedback(G*Gc4,1),10);
%resp. lazo cerrado al escalon
```



Respuesta lazo cerrado al escalón

```
step(feedback(G,Gc1),
feedback(G,Gc2),
feedback(G,Gc3),
feedback(G,Gc4),30);
%resp. lazo cerrado ante perturbaciones
```



Respuesta lazo cerrado ante perturbaciones

- La función chreswickpid () será :

Chien-Hrones-Reswick

```
function [Gc,Kp,Ti,Td,H]=chreswickpid(key,tt,vars)
K=vars(1);
L=vars(2);
T=vars(3);
N=vars(4);
a=K*L/T;
Ti=[];
Td=[];
ovshoot=vars(5);
if tt==1,
```

```
TT=T;
else TT=L;
 tt=2; end
if ovshoot==0,
KK=[0.3,0.35,1.2,0.6,1,0.5; 0.3,0.6,4,0.95,2.4,0.42];
else
KK=[0.7,0.6,1,0.95,1.4,0.47; 0.7,0.7,2.3,1.2,2,0.42];
end
switch key
case 1, Kp=KK(tt,1)/a;
case 2, Kp=KK(tt,2)/a; Ti=KK(tt,3)*TT;
case {3,4}, Kp=KK(tt,4)/a; Ti=KK(tt,5)*TT; Td=KK(tt,6)*L;
end
[Gc,H]=escribirpid(Kp,Ti,Td,N,key);
```

- La función escribirpid () será :

```
function [Gc,H]=escribirpid(Kp,Ti,Td,N,key)
switch key
case 1, Gc=Kp;
case 2, Gc=tf(Kp*[Ti,1],[Ti,0]); H=1;
case 3, nn=[Kp*Ti*Td*(N+1)/N,Kp*(Ti+Td/N),Kp];
dd=Ti*[Td/N,1,0]; Gc=tf(nn,dd); H=1;
case 4, d0=sqrt(Ti*(Ti-4*Td)); Ti0=Ti; Kp=0.5*(Ti+d0)*Kp/Ti;
Ti=0.5*(Ti+d0); Td=Ti0-Ti; Gc=tf(Kp*[Ti,1],[Ti,0]);
nH=[(1+Kp/N)*Ti*Td, Kp*(Ti+Td/N), Kp];
H=tf(nH,Kp*conv([Ti,1],[Td/N,1]));
case 5, Gc=tf(Kp*[Td*(N+1)/N,1],[Td/N,1]); H=1;
end
```

2.4.4 Cohen–Coon

Método empírico para sistemas con retardos:

• Criterio de diseño: buen comportamiento ante variaciones de carga.

Razón decrecimiento d = 0.25

• Emplea parámetros a, L, T para ajuste del controlador.

• En controladores P, PD se consiguen elevadas K.

• En controladores PI, PID se consiguen elevadas Ki.

• A diferencia de ZN, este método toma en cuenta L y T (en la tabla) para ajustar

los tiempos de la acción I y D.

a = kL/T and τ = L/(L + T).

La siguiente tabla muestra los valores de los parámetros del PID utilizando Cohen–Coon.

Controller	K_p	T_i	T_d
P	$\frac{1}{a}\left(1+\frac{0.35\tau}{1-\tau}\right)$		
PI	$\frac{0.9}{a}\left(1+\frac{0.92\tau}{1-\tau}\right)$	$\frac{3.3-3\tau}{1+1.2\tau}L$	
PD	$\frac{1.24}{a}\left(1+\frac{0.13\tau}{1-\tau}\right)$		$\frac{0.27-0.36\tau}{1-0.87\tau}L$
PID	$\frac{1.35}{a}\left(1+\frac{0.18\tau}{1-\tau}\right)$	$\frac{2.5-2\tau}{1-0.39\tau}L$	$\frac{0.37-0.37\tau}{1-0.81\tau}L$

Ejercicio: Cada grupo elige una FDT distinta de un sistema de control. Se diseña un controlador PID utilizando Cohen–Coon. Este paso debe prepararse por cada grupo antes de la práctica. El monitor comprueba la solución teórica al comenzar la práctica. Se utiliza el código Matlab para simular el sistema controlado. Comentar los resultados.

Se tiene la FDT anterior G(s)=1/(s(s+1)(s+5) y los parámetros de ZN: L=0.7025,k=0.27 y Tc=2.81

Se hallan el parámetro a, la constante de tiempo τ y los parámetos del PID.

a=kL/T=0.067 τ=L/(L+T)=0.2

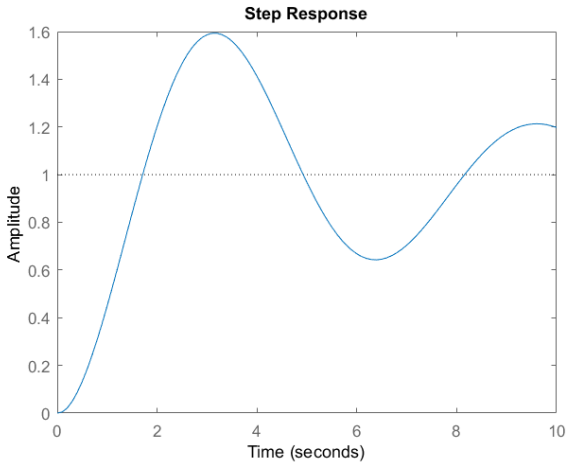
Kp=$\frac{1.35}{a}\left(1+\frac{0.18\tau}{1-\tau}\right)=21$

Ti=$\frac{2.5-2\tau}{1-0.39\tau}L=1.6$

Td=$\frac{0.37-0.37\tau}{1-0.81\tau}=0.35$

<u>El código principal en Matlab será</u>:

```
s=tf('s');G=tf(1,[1 6 5 1]);
k=dcgain(G);
L=0.7025;
T=2.81;
[Gc,Kp,Ti,Td]=cohenpid(3,[k,L,T,10])
G_c=feedback(G*Gc,1);
step(G_c,10) %resp. lazo cerrado
```



- La función Cohenpid(). Con esta función diseñamos el PID:

```
function [Gc,Kp,Ti,Td,H]=cohenpid(key,vars)
K=vars(1);
L=vars(2);
T=vars(3);
N=vars(4);
a=K*L/T;
 tau=L/(L+T);
Ti=[];
Td=[];
switch key
case 1,Kp=(1+0.35*tau/(1-tau))/a;
case 2,
Kp=0.9*(1+0.92*tau/(1-tau))/a;
Ti=(3.3-3*tau)*L/(1+1.2*tau);
case {3,4}, Kp=1.35*(1+0.18*tau/(1-tau))/a;
Ti=(2.5-2*tau)*L/(1-0.39*tau);
Td=0.37*(1-tau)*L/(1-0.81*tau);
case 5
Kp=1.24*(1+0.13*tau/(1-tau))/a;
Td=(0.27-0.36*tau)*L/(1-0.87*tau);
end
[Gc,H]=escribirpid(Kp,Ti,Td,N,key);
```

- La función escribirpid():

```
function [Gc,H]=escribirpid(Kp,Ti,Td,N,key)
switch key
case 1, Gc=Kp;
case 2, Gc=tf(Kp*[Ti,1],[Ti,0]);
H=1;
case 3, nn=[Kp*Ti*Td*(N+1)/N,Kp*(Ti+Td/N),Kp];
dd=Ti*[Td/N,1,0];
```

```
Gc=tf(nn,dd);
H=1;
case 4, d0=sqrt(Ti*(Ti-4*Td));
Ti0=Ti;
Kp=0.5*(Ti+d0)*Kp/Ti;
Ti=0.5*(Ti+d0);
Td=Ti0-Ti;
Gc=tf(Kp*[Ti,1],[Ti,0]);
nH=[(1+Kp/N)*Ti*Td, Kp*(Ti+Td/N), Kp];
H=tf(nH,Kp*conv([Ti,1],[Td/N,1]));
case 5, Gc=tf(Kp*[Td*(N+1)/N,1],[Td/N,1]);
H=1;
end
```