Controladores PID-Análisis y Diseño

1 Introducción

Simulink® es una extensión de MATLAB® para la simulación de sistemas dinámicos. Al ser un entorno gráfico, resulta bastante sencillo de emplear.

1. Se iniciará el programa **MATLAB R2010a** como se muestra en la figura de abajo.

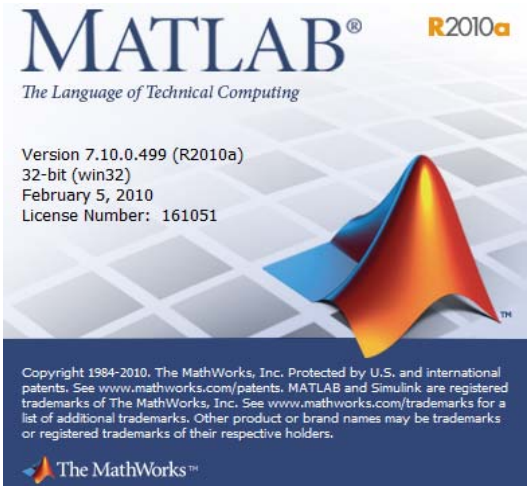


Figura 1. Pantalla de inicio Matlab

2. Después de haberse iniciado el programa, **MATLAB R2010a** muestra su pantalla de trabajo.

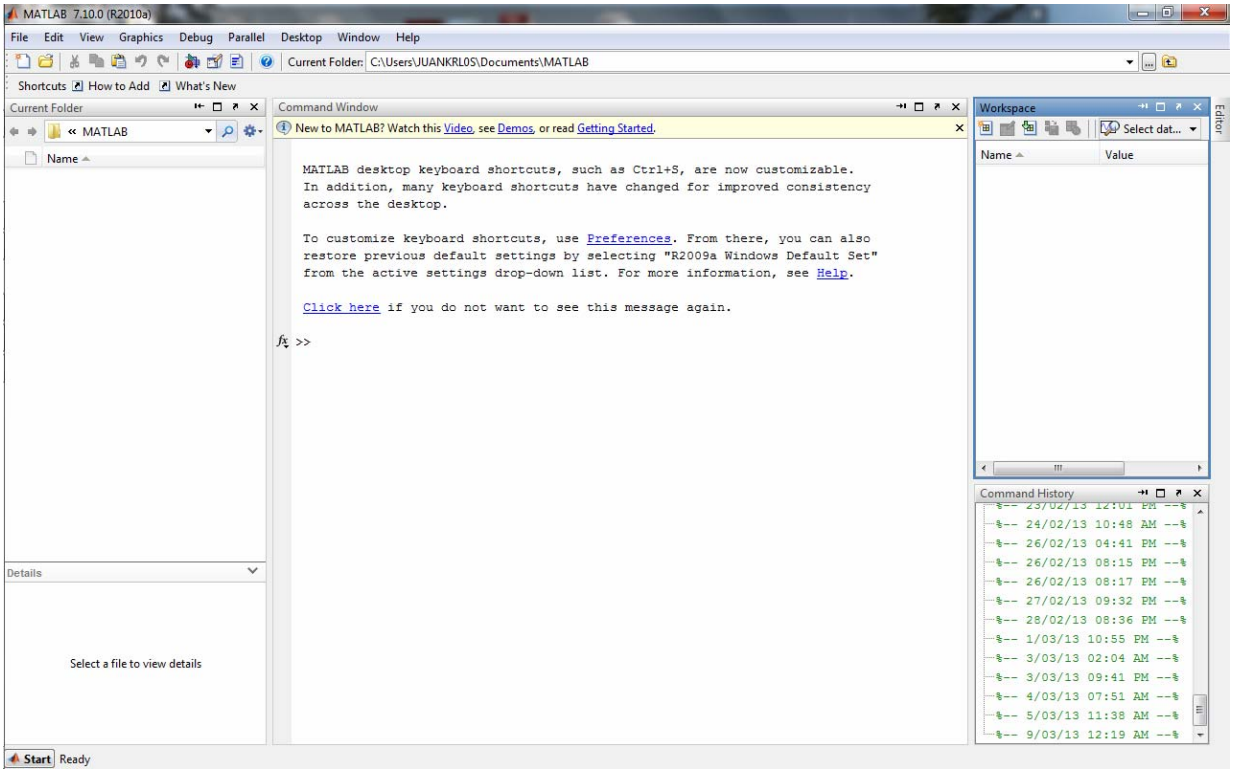


Figura 2. Pantalla de trabajo Matlab

3. Dirigirse con el ratón a la ventana de **Command Windows** y dar un clic con el botón izquierdo del ratón sobre ésta, escribir la palabra SImulink y al finalizar la escritura dar un **Enter** con el teclado, Simulink es la **toolbox de Matlab** que utilizara para simular el sistema con el controlador PID.

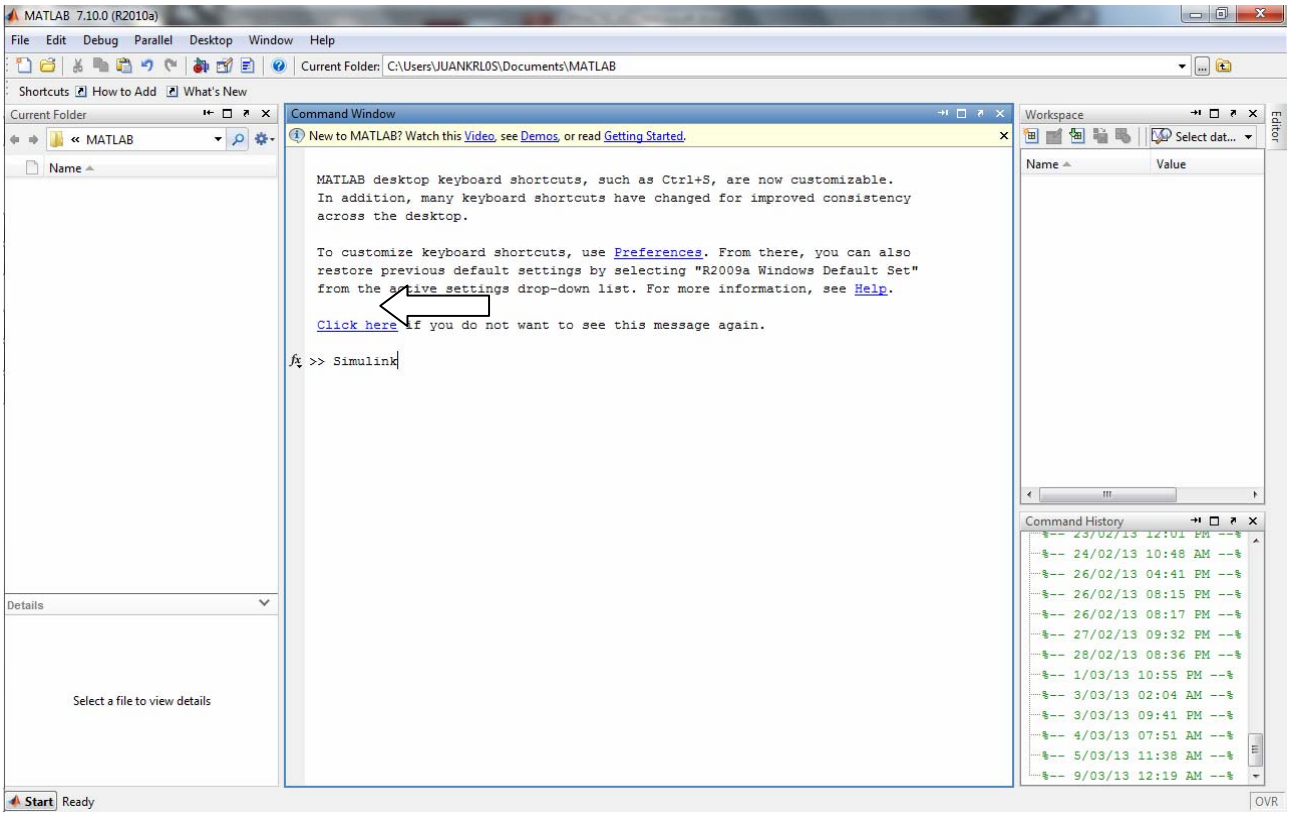


Figura 3. Ingreso a simulink

4. Saldrá por pantalla una ventana gráfica, como la de la Fig. 4, que contiene todas las librerías que el entorno de "Simulink" bajo Matlab soporta:

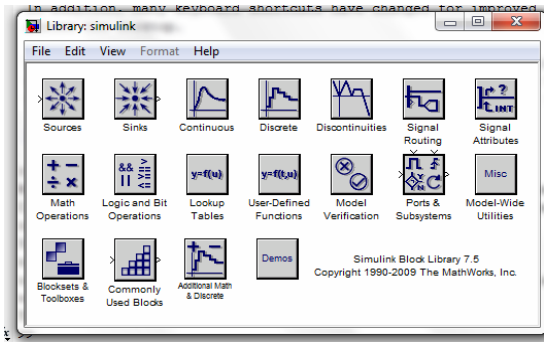


Figura 4 Librería simulink

5. Elegir un nuevo fichero donde se guardara el modelo: seleccionar en el menú File - New - Model. Tendrá la situación de la Fig. 5

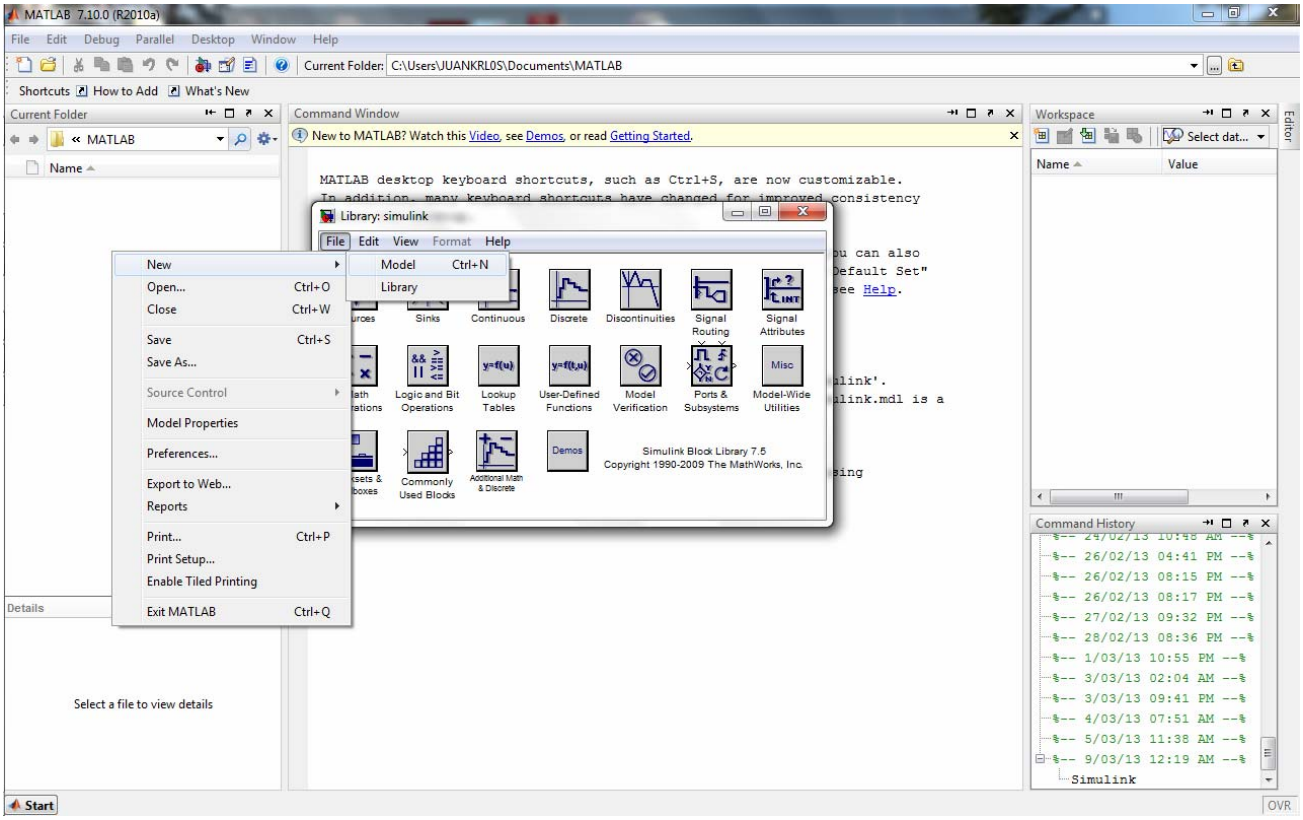


Figura 5 Nuevo modelo simulink

6. Para describir la ecuación del PID se necesitara además los bloques integrador 'Integrator' y derivador 'derivative'. Entonces se está en disposición de describir la ecuación (1.2) utilizando bloques.

$$G_{PID} = \left(K_p + \frac{K_i}{S} + K_d S \right) \qquad (1.2)$$

7. Unirlos los bloques de forma adecuada para describir dicha ecuación (1.2). Hacer el esquema como describe la Fig. 6.

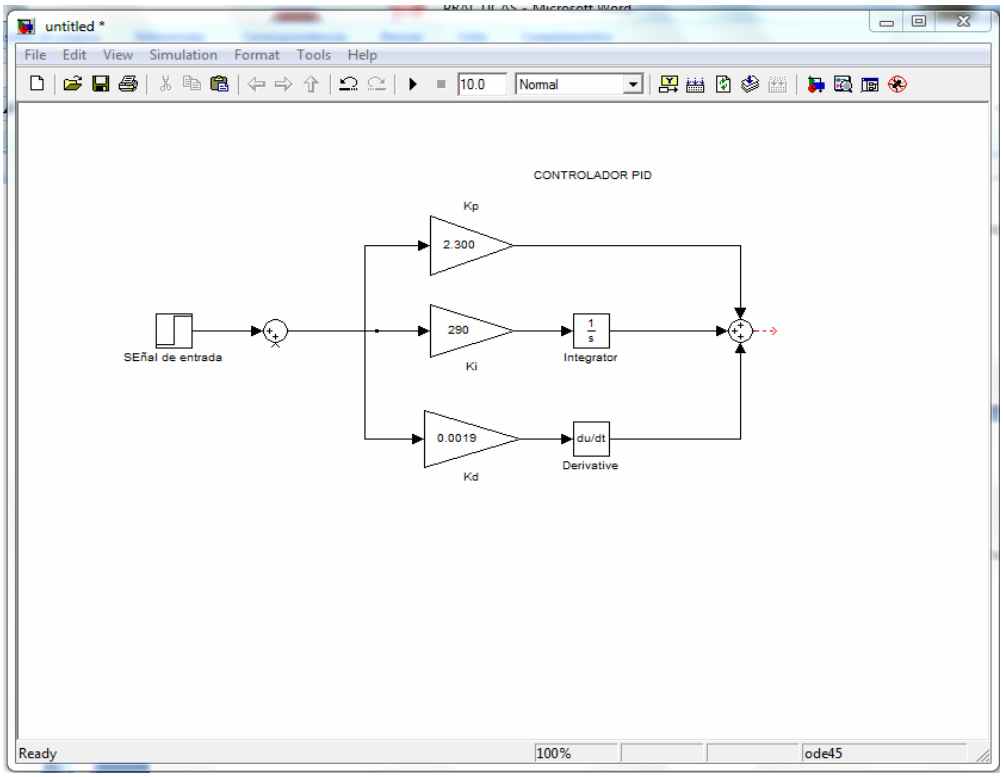


Figura 6 Diagrama del PID

8. Como puede verse en la Fig. 6, se ha editado el valor de algunos de los bloques. Daremos valores concretos a las constantes.

Kp=2.300

Ki=290

Kd=0.0019

9. Ahora se puede construir el modelo de un motor utilizando los bloques que se aprendieron anteriormente. De acuerdo a la figura 7 editar este bloque con los valores de cada constate del motor dados por la tabla 1, haciendo doble ‘click’. El modelo quedará como se muestra en la Fig. 8

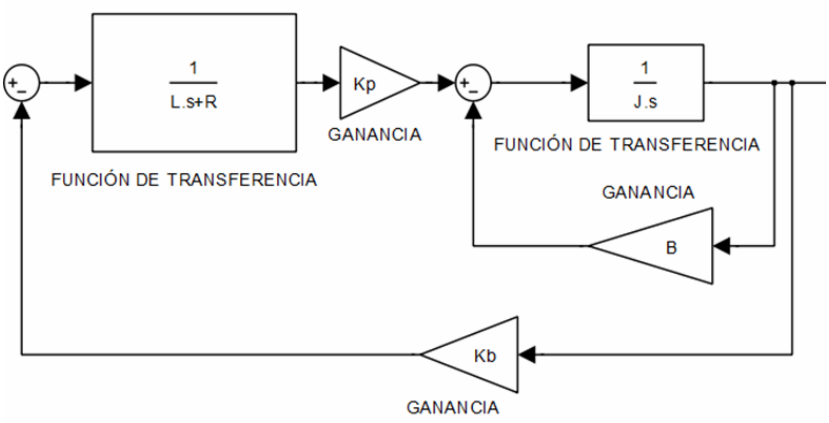


Figura 7 Diagrama con constantes del motor

Tabla 1 Parámetros del servomotor.

UNIMOTOR EZ					
R (ph - ph)	**L (ph - Ph)**	**K_P**	**K_b**	**J_m**	**B_m**
3.5 Ω	0.0119 H	0.93 $\frac{Nm}{Amp.}$	0.544 V.seg/rad	$1.5e^{-4} Kg.m^2$	0.0015 $\frac{N.ms}{rad.}$

Donde $u(t)$ es la tensión eléctrica aplicada al motor, R es la resistencia eléctrica, L representa la inductancia, $i(t)$ es la corriente eléctrica, $u_m(t)$ es la fuerza contraelectromotriz , K_b es la constante de la fuerza contraelectromotriz, $\omega(t)$ es velocidad angular del motor, $p_m(t)$ es par del motor, K_p es la constante del par, J es el momento de inercia del rotor y B es la constante de fricción viscosa.

Dado que el sistema propuesto es lineal, es posible aplicar la transformada de Laplace para obtener una función de transferencia.

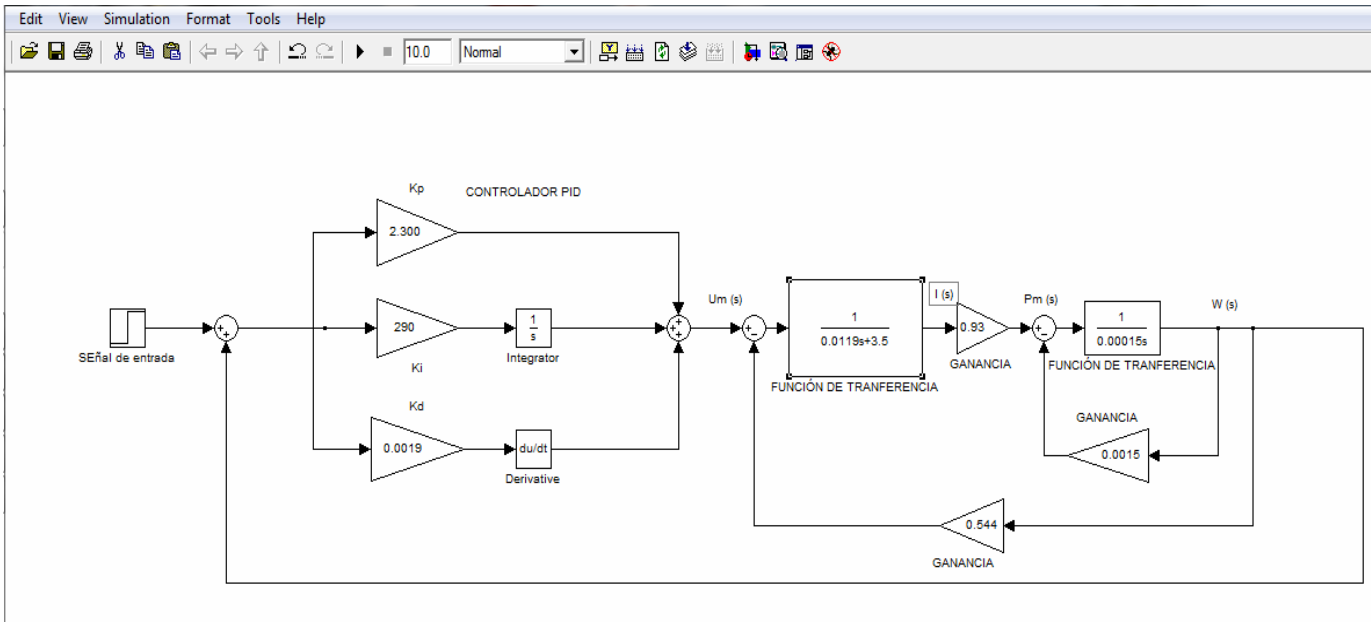


Figura 8 Controlador PID y motor

10. Para poder ver los resultados hay que poner un bloque que nos muestre la velocidad del eje del motor frente al tiempo. Seleccionamos de la librería ‘Sinks’ el bloque ‘Scope’.

11. Para definir el tiempo de simulación acceder al menu de la ventana del modelo en la parte de arriba en Simulation – Configuration Parameters. La situación será como la mostrada en la Fig. 9

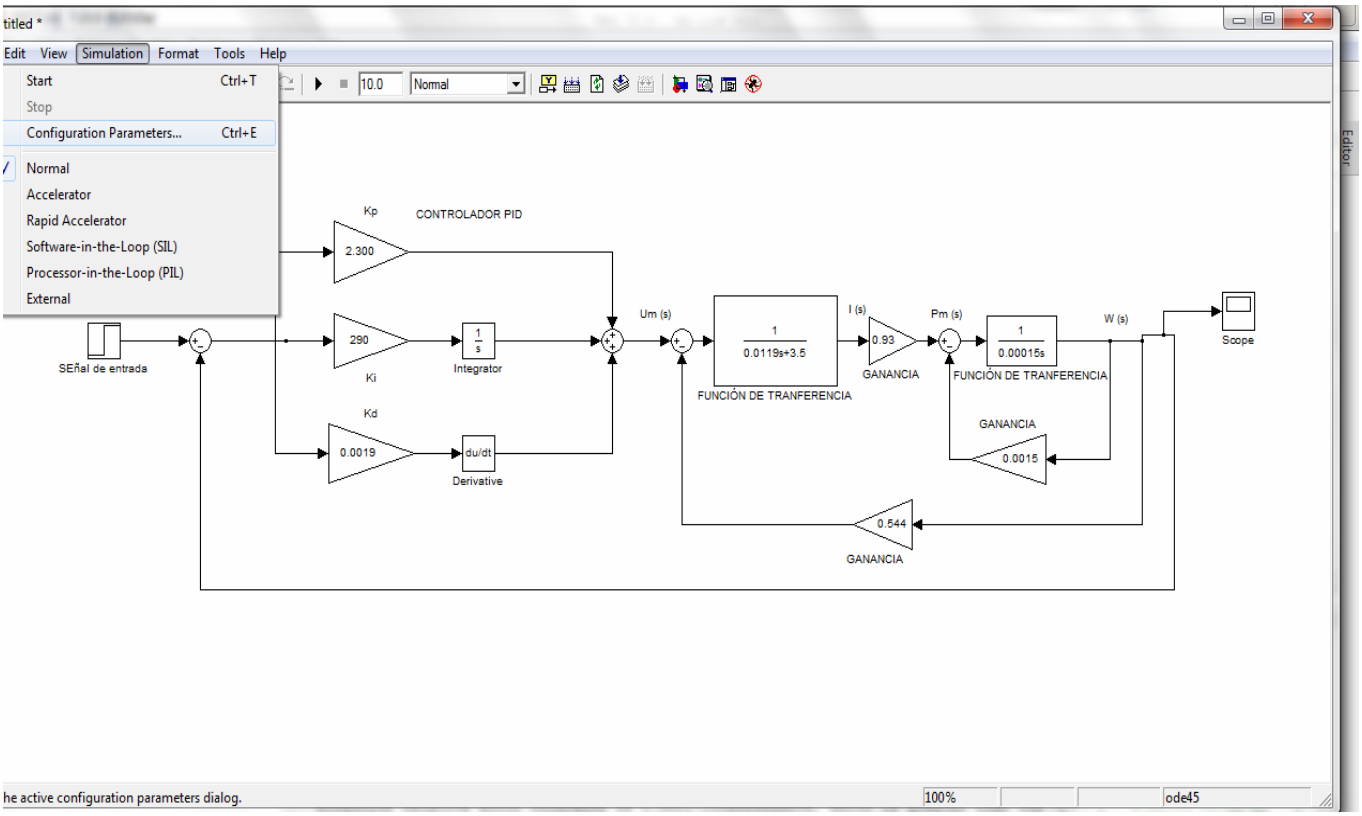


Figura 9 Configuración de parámetros de simulación

12. Se abrirá una ventana (la situación será como la mostrada en la Fig. 10 donde es posible definir entre otros parámetros el tiempo de simulación, el método de resolución y el paso fijo o variable. Dejar los dos últimos como están y poner el tiempo de simulación a 10 segundos y dar clic en ok.

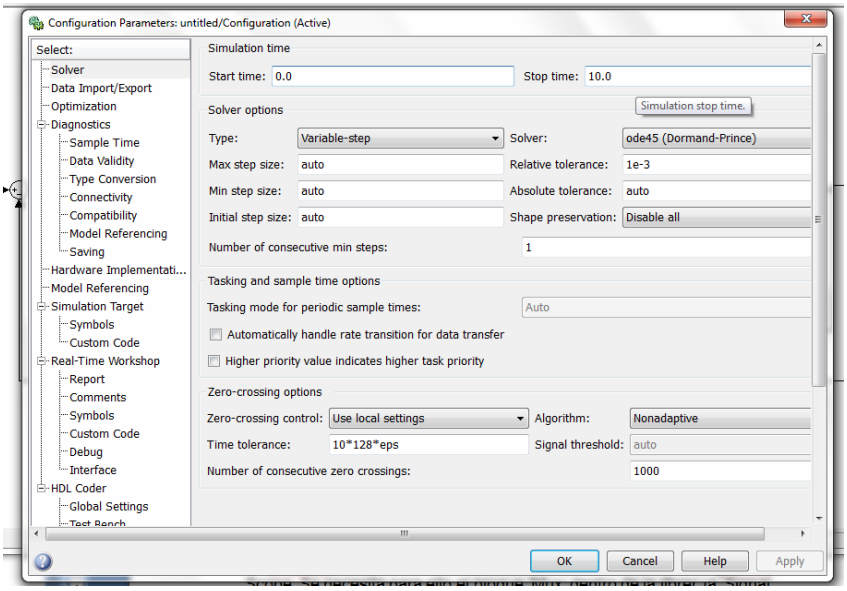


Figura 10 tiempo de muestreo

13. Una vez acabada la simulación tendremos el resultado que puede verse en la Fig. 11.

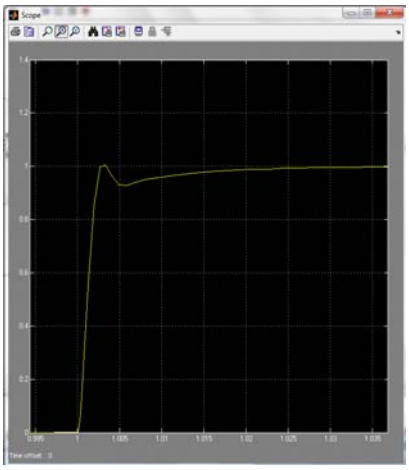


Figura 11 Respuesta de velocidad

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)

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

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
```

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

2.2 Efecto de la patada en la consigna

Si la entrada es escalón, debido al termino D, u(t) contendrá un impulso. En un PID real en lugar de Tds se emplea $\frac{T_d s}{1+\gamma T_d s}$ con $\gamma = 0{,}1$ y u(t) no contendrá impulso sino una función de forma de un pulso estrecho. Tal fenómeno se denomina patada en el punto de consigna.

1. amplificación de ruido

sea la señal ruido: $r = a \cdot sen(\omega t)$

tras la acción D: $y_D = aKT_d\omega \cdot \cos(\omega t)$

se obtiene una señal dependiente de la frecuencia; si esta es elevada, la salida y_D aumenta considerablemente.

Solución: incorporar un filtro de primer orden en la acción D (filtro),

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

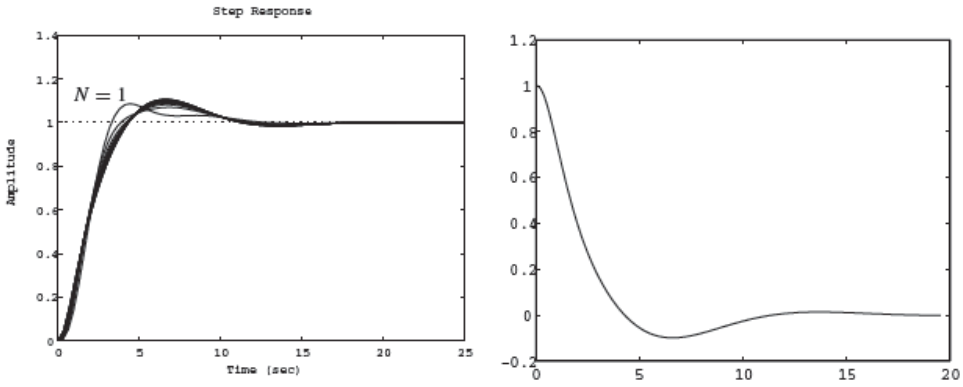
$$U(s) = K_p\left(1 + \frac{1}{T_i s} + \frac{sT_d}{1+s\frac{T_d}{N}}\right) E(s).$$

- a bajas frec. D ≈ -KTd*s* (efecto D normal)
- a altas frec. D ≈ -KN (sólo ganancia y esta limitada a KN). Esto significa que el ruido con alta frecuencia no se puede amplificarse más que KN. N=8 ~20)

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



(a) output signal (b) error signal

Fig. 12

Ejercicio : Determinar el mejor valor de N

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.

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

Se puede observar un gran salto en t=0 en la señal del error (fig. 12). Esto quiere decir que no es aconsejable tener el efecto derivada del PID en lazo directo.

Por tanto, en la práctica, se prefiere añadir el termino derivada en el lazo de realimentación lo que se denomina PI-D. (ver Fig 13).

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s}\right),$$

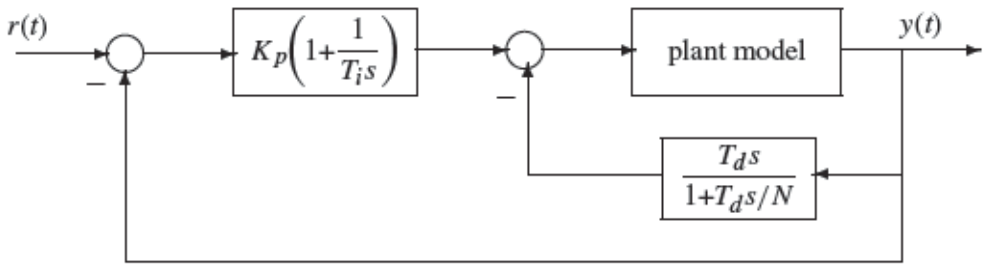


Fig 13

$$H(s) = \frac{(1 + K_p/N)T_i T_d s^2 + K_p(T_i + T_d/N) + K_p}{K_p(T_i s + 1)(T_d s/N + 1)}.$$

Ejemplo 3.

Diseñar un PI-D para el mismo sistema del ejemplo anterior con *G(s)* = 1*/(s* + 1*)*3, utilizando la siguiente MATLAB código:

```
>> G=tf(1,[1,3,3,1]); Ti=1; Td=1; Kp=1; 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)
```

La respuesta del lazo cerrado con el PID y PI-D se ven en la Fig 14. la respuesta con PI-D es mas lenta con mas oscilaciones que la del PID

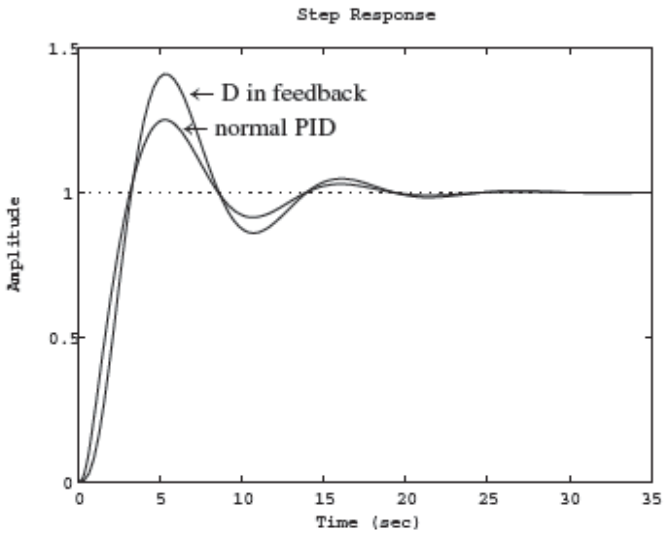


Fig 14

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.

2.4 Métodos Empíricos de diseño:

Ziegler-Nichols (ZN) respuesta al escalón

Ziegler-Nichols (ZN) respuesta en frecuencia

2.4.1 Ziegler-Nichols (ZN) respuesta al escalón

- ZN – respuesta al escalón: (no necesario modelo previo)

    a) Se basan en el ensayo en cadena abierta ante entrada escalón.

    b) Determinación del retraso L y constante de tiempo T

    c) Obtener parámetros de la tabla

La formula de ZN se basa en un modelo de un system de 1er orden con retraso (first-order plus dead time (FOPDT) cuya respuesta se muestra en la figura 15.

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

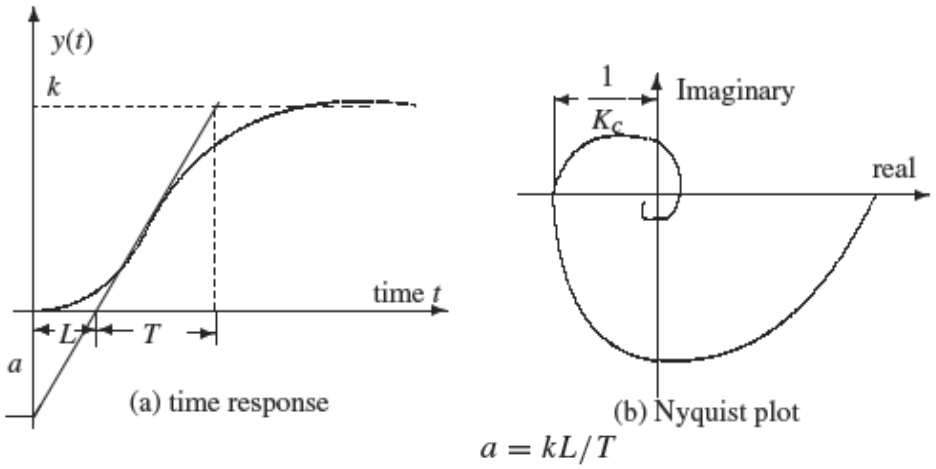


Figura 15. *Respuestas del modelo FOPDT.*

La siguiente table de ZN ofrece los valores optimos de los parametros del PID

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	$1.2/a$	$2L$	$L/2$	$0.6K_c$	$0.5T_c$	$0.12T_c$

La siguiente función MATLAB zn() se usa para diseñar controladores PI/PID aplicando las formulas de Ziegler–Nichols:

```
function [Gc,Kp,Ti,Td,H]=zn(key,vars)
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
 [Gc,H]=escribirpid(Kp,Ti,Td,N,key);
```

La siguiente función escribir pid () se usa para devolver los parámetros del controlador PID

```
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
```

se utiliza la siguiente formula de MATLAB para ejecutar el código anterior

[*Gc*,*Kp*,*Ti*,*Td*]=zn(key,vars),

donde key determina el tipo del controlador; con key = 1 para el controlador P, key = 2 para el controlador PI, y key = 3 para el controlador PID. vars = [*K, L, T,N*]

Ejemplo 4.

Dado una planta de 4º orden:
$G(s) = 10/$
$(s + 1)(s + 2)(s + 3)(s + 4)$

Con el siguiente código Matlab:

```
>> s=tf('s'); G=10/(s+1)/(s+2)/(s+3)/(s+4);
step(G); k=dcgain(G)
```

la respuesta en lazo directo se ve en la figura 16-a. Con un error en régimen permanente de 0.4167. Los parámetros del FOPDT son $k = 0.2941$, $L = 0.76$, y $T = 2.72 - 0.76 = 1.96$. Aplicando el siguiente código MATLAB, se diseñan los controladores P, PI, y PID

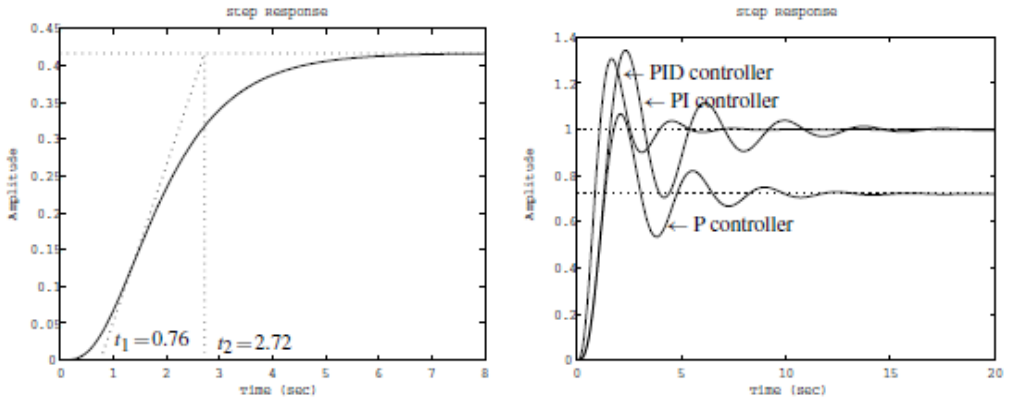
```
>> L=0.76; T=2.72-L; [Gc1,Kp1]=zn(1,[k,L,T,10])
[Gc2,Kp2,Ti2]=zn(2,[k,L,T,10])
[Gc3,Kp3,Ti3,Td3]=zn(3,[k,L,T,10])
```

Los controladores P, PI, y PID controladores son,

$$G_p(s) = 6.1895,\ G_{PI}(s) = 5.57\left(1 + \frac{1}{2.5308s}\right),\ G_{PID}(s) = 7.4274\left(1 + \frac{1}{1.52s} + 0.38s\right).$$

Las respuestas en lazo cerrado se obtienen utilizando el siguiente código MATLAB y se muestran en la figura (16-b)

```
>> G_c1=feedback(G*Gc1,1); G_c2=feedback(G*Gc2,1);
G_c3=feedback(G*Gc3,1); step(G_c1,G_c2,G_c3);
```



(a) open-loop step response (b) closed-loop step response

figura 16

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.

2.4.2 Ziegler-Nichols (ZN)-respuesta en frecuencia

Ejemplo 5.

La ganancia y frecuencia de cruce en -180º del sistema anterior se puede ver en la figura 15-b

```
>> G=tf(10, [1,10,35,50,24]);
nyquist(G); axis([-0.2,0.6,-0.4,0.4])
[Kc,pp,wg,wp]=margin(G); [Kc,wg], Tc=2*pi/wg;
[Gc1,Kp1]=zn(1,[Kc,Tc,10]); Kp1
[Gc2,Kp2,Ti2]=zn(2,[Kc,Tc,10]); [Kp2,Ti2]
[Gc3,Kp3,Ti3,Td3]=zn(3,[Kc,Tc,10]); [Kp3,Ti3,Td3]
```

El margen de ganancia y su frecuencia de cruce son 12.6, and 2.2361 rad/sec respectivamente. Los controladores son

$$G_{\mathrm{p}}(s)=6.3, \quad G_{\mathrm{PI}}(s)=5.04\left(1+\frac{1}{2.2479s}\right), \quad G_{\mathrm{PID}}(s)=7.56\left(1+\frac{1}{1.405s}+0.3372s\right)$$

El diagrama de nyquist se muestra en la figura Fig. 17-a. Con estos controladores, la respuesta en lazo cerrado (fig 17-b) se puede obtener como sigue:

```
>> G_c1=feedback(G*Gc1,1); G_c2=feedback(G*Gc2,1);
G_c3=feedback(G*Gc3,1); step(G_c1,G_c2,G_c3);
```

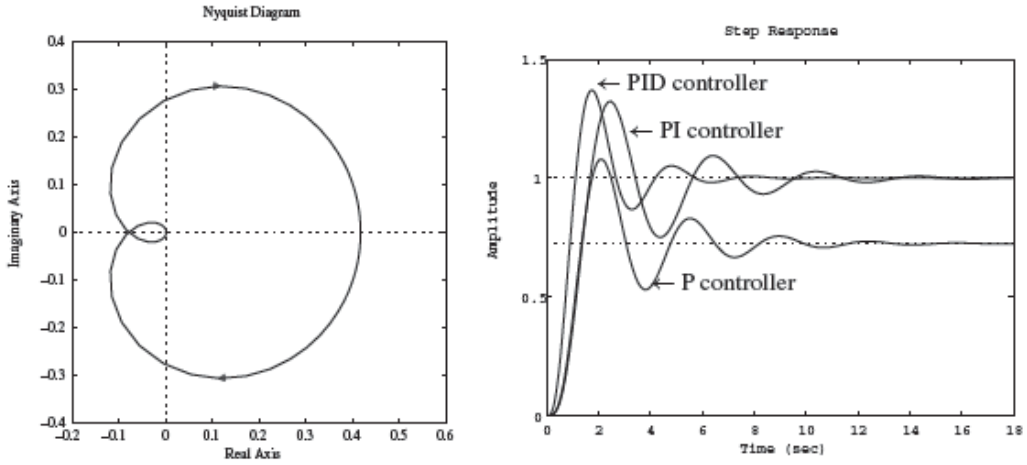


Figure 17.

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.

2.4.3 Chien–Hrones–Reswick

Los métodos de ZN proporcionan buena respuesta ante cambios de carga.

- Criterio de diseño: consiguen una razón de decrecimiento de 1/4, lo que da un $\zeta = 0.22$ (muy subamortiguado).

Ante cambios de consigna la salida puede no responder de forma óptima, generalmente de pobre amortiguación.

Si no cumple los requisitos, REAJUSTAR.

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 T_d y aumentando el valor de T_i con respecto al método de ZN.

El código Matlab para implementar CHR es el siguiente:

```
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);
```

Las siguientes tablas muestran los valores de los parametros del PID utilizando CHR

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	$0.5L$	$0.95/a$	$1.4T$	$0.47L$

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	$0.6/a$	$4L$		$0.7/a$	$2.3L$	
PID	$0.95/a$	$2.4L$	$0.42L$	$1.2/a$	$2L$	$0.42L$

La funcion chreswickpid () es:

[*Gc*,*Kp*,*Ti*,*Td*]= chreswickpid (key,typ,vars)

key = 1, 2, 3 para P, PI, and PID respectivamente. La variable typ denota el tipo del criterio utilizado. typ = 1 se usa para el control ante una señal de referencia y cualquier otro valor se usa para reducir perturbaciones.
vars = [*k,L, T,N,Os*] con *Os* = 0 denota no pico de oscilaciones. Cualquier otro valor denota 20% pico de oscilaciones.

Ejemplo 6.

Repetir el ejemplo 5. aplicando el controlador CHR

```
>> s=tf('s'); G=10/((s+1)*(s+2)*(s+3)*(s+4));
        [Gc1,Kp,Ti,Td]=zn(3,[k,L,T,N])
         [Gc2,Kp,Ti,Td]= chreswickpid (3,1,[k,L,T,N,0])
        [Gc2,Kp,Ti,Td]= chreswickpid (3,1,[k,L,T,N,20])
        [Gc2,Kp,Ti,Td]= chreswickpid (3,2,[k,L,T,N,0])
```

Los 4 controladores PID son,

$$G_1(s) = 8.4219\left(1+\frac{1}{1.5764s}+0.3941s\right),\ G_2(s) = 4.2110\left(1+\frac{1}{2.3049s}+0.3941s\right),$$

$$G_3(s) = 6.6674\left(1+\frac{1}{3.2268s}+0.3704s\right),\ G_4(s) = 6.6674\left(1+\frac{1}{1.8917s}+0.3310s\right).$$

Las respuestas en lazo cerrado ante entrada escalón se obtienen utilizando el siguiente código MATLAB

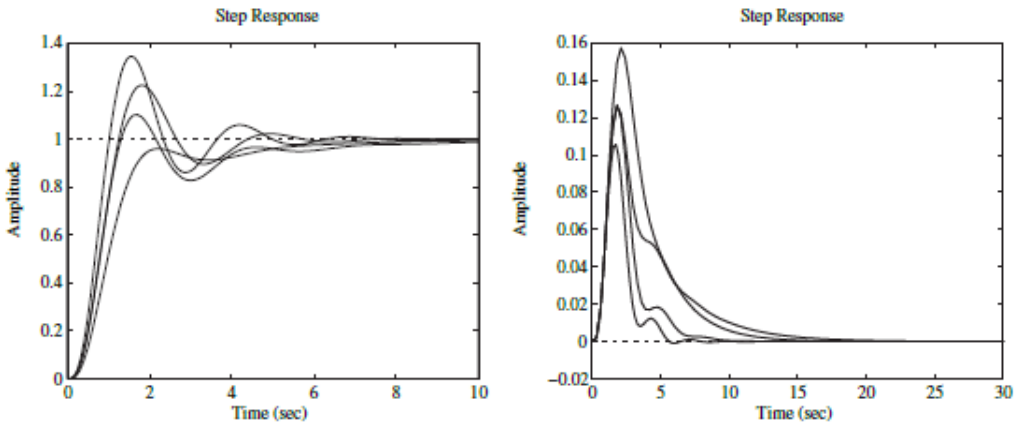
```
>> step(feedback(G*Gc1,1),feedback(G*Gc2,1),feedback(G*Gc3,1),...
feedback(G*Gc4,1),10)
```

como se puede ver en la figura 18-a, la respuesta con 0% de pico de sobre oscilación ofrece unos resultados satisfactorios.

Las respuestas en lazo cerrado ante perturbaciones se obtienen utilizando el siguiente código MATLAB y se muestran en la figura 18-b

```
>> step(feedback(G,Gc1),feedback(G,Gc2),feedback(G,Gc3),...
feedback(G,Gc4),30)
```

como se puede apreciar que con el CHR las perturbaciones se reducen tangiblemente en comparacion con ZN



(a) set-point step response (b) disturbance step response

Figura 18. Respuesta en lazo cerrado del controlador CHR

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.

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 K_i.
- A diferencia de ZN, este método toma en cuenta L y T (τ en la tabla) para ajustar los tiempos de la acción I y D.

$a = kL/T$ and $\tau = L/(L + T)$.

La function MATLAB cohenpid() se usa para diseñar el PID como sigue:

```
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 es

[*Gc*,*Kp*,*Ti*,*Td*,*H*]=cohenpid(key,vars) , donde vars es vars = [*k,L, T,N*].

La siguiente tabla muestra los valores de los parámetros del PID utilizando 2.4.4 Cohen–Coon Tuning Algorithm

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	$\frac{1.35}{a}\left(1+\frac{0.18\tau}{1-\tau}\right)$	$\frac{2.5-2\tau}{1-0.39\tau}L$	$\frac{0.37-0.37\tau}{1-0.81\tau}L$

Ejemplo 7.

Repetir la FDT del ejemplo anterior

```
>> G=tf(10,[1,10,35,50,24]);
[Gc1,Kp1]=cohenpid(1,[k,L,T,10])
[Gc2,Kp2,Ti2]=cohenpid(2,[k,L,T,10])
[Gc3,Kp3,Ti3,Td3]=cohenpid(5,[k,L,T,10])
[Gc4,Kp4,Ti4,Td4]=cohenpid(3,[k,L,T,10])
```

Y los controlador son: $G1(s) = 7.8583$, $G2(s) = 8.3036\,(1 + 1/1.5305s)$,
$G3(s) = 9.0895(1 + 0.1805s)$, $G4(s) = 10.0579\,(1 + 1/1.7419s + 0.2738s)$.

La respuesta del lazo cerrado se obtiene como:

```
>> G_c1=feedback(G*Gc1,1); G_c2=feedback(G*Gc2,1);
G_c3=feedback(G*Gc3,1); G_c4=feedback(G*Gc4,1);
step(G_c1,G_c2,G_c3,G_c4,10)
```

las respuestas en lazo cerrado se muestran en Fig. 19.

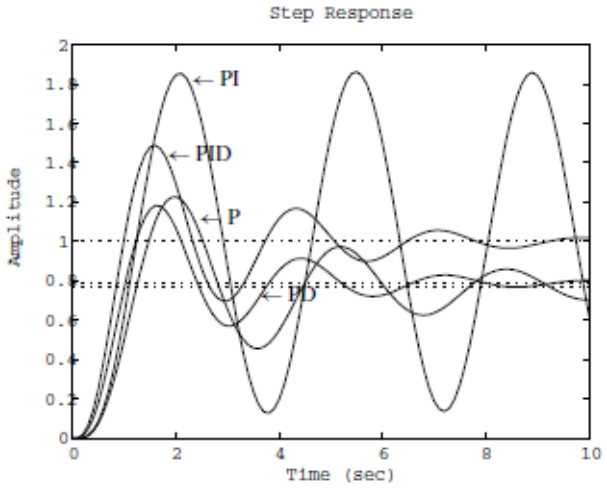


Figura 19. Respuesta en lazo cerrado del controlador Cohen–Coon

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.