Control con Modelos de Entrada y Salida

Ingeniería Electrónica de Comunicaciones

Eva Besada Portas

Departamento de Arquitectura de Computadores y Automática.
Universidad Complutense de Madrid

Curso 2020-2021

PIDs: PID anti-windup I

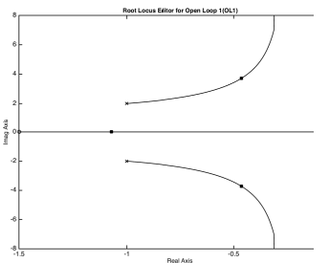
En los sistemas reales, la señales que el actuador puede aplicar sobre la planta se encuentran limitadas por un valor máximo y mínimo. Por lo tanto, éste no puede aplicar los valores de las señales obtenidas por el controlador que se encuentran fuera del rango válido, sustituyéndose en ese caso el valor de la señal de control máxima/mínimo por el del limite aplicable más cercano.

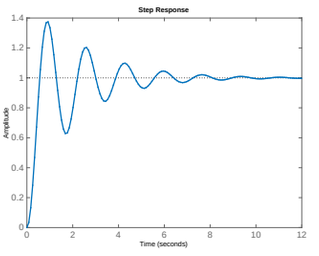
Este comportamiento realista, modelable mediante una saturación, hace que el sistema deje de comportarse como un sistema lineal y que la acción integral del PID degrade (retarde) la respuesta del sistema.

Ejemplo: $G_p(s) = \frac{10}{s^2+2s+5}$ controlador con un controlador PI y analizar el comportamiento del sistema cuando el actuador se encuentra limitado a valores entre ± 1

Sintonizar el PI, p.e. con el lugar de las raíces. Como $G_c(s) = K_p + \frac{K_i}{s} = K_p \frac{s+a}{s}$, colocar un polo en s=0 y un cero movible en -a.

$K_p = 1$, $K_i = 1{,}5$

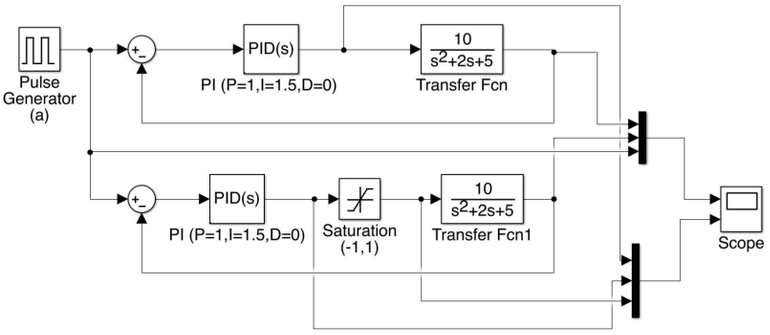




PIDs: PID anti-windup II

Ejemplo (cont.): $G_p(s) = \frac{10}{s^2+2s+5}$ y $G_c(s) = \frac{1+1,5s}{s}$. Actuador limitado al rango ± 1

Para ver como le afecta el limite al sistema, necesitamos usar Simulink, ya que la saturación es un elemento no lineal con el que no se puede trabajar con la toolbox de Control.



Modelamos dos veces el sistema (con y sin saturación), para representar sobre el mismo visor las señales de los dos casos.

r(t): una onda cuadrada entre [0,a], con un periodo de 30 segundos ($y_{SisIdeal}(t)$ tiene $t_s \approx 10s$)

PI se implementa con un bloque PI, haciendo $K_d = 0{,}0$, $K_p = 1{,}0$ y $K_i = 1{,}5$.

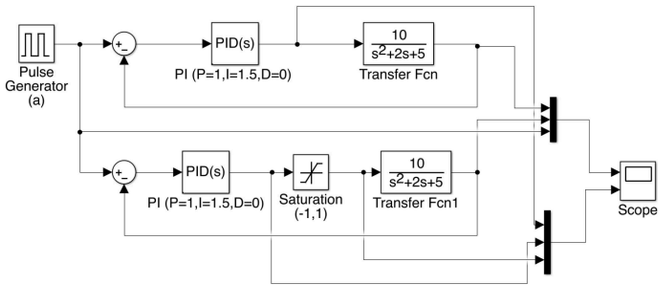
La saturación es un bloque no lineal. Fijamos los límites entre ± 1.

El Scope tiene dos gráficas:

Superior	$y_{SisIdeal}(t)$	$y_{SisReal}(t)$	$r(t)$
Inferior	$u_{SisIdeal}(t)$	$u_{SisReal}(t)$	$a_{SisReal}(t)$

PIDs: PID anti-windup III

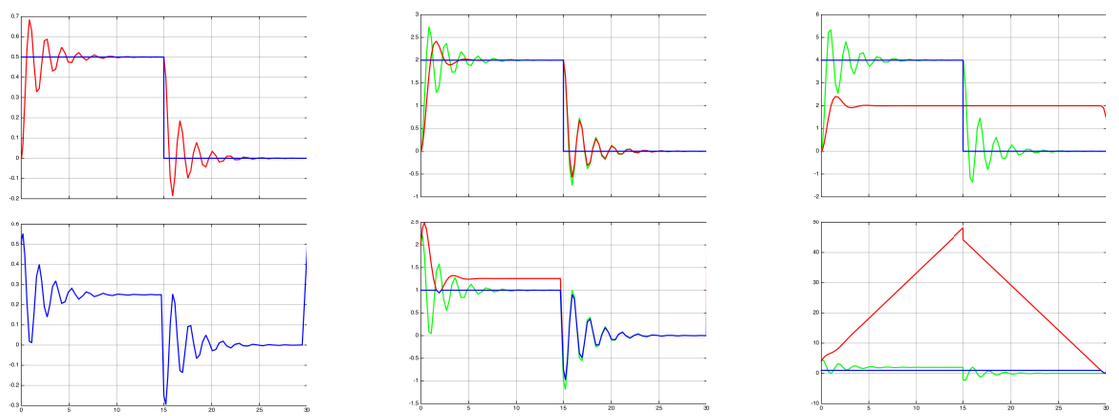
Ejemplo (cont.): $G_p(s) = \frac{10}{s^2+2s+5}$ y $G_c(s) = \frac{1+1,5s}{s}$. Actuador limitado al rango ± 1



	Superior	Inferior
Verde	$y_{SisIdeal}(t)$	$u_{SisIdeal}(t)$
Roja	$y_{SisReal}(t)$	$u_{SisReal}(t)$
Azul	$r(t)$	$a_{SisReal}r(t)$

Vamos a ver la respuesta del sistema para diferentes $r(t)$ ondas cuadradas entre [0,a].

$a = 0{,}5$ $a = 2$ $a = 4$



PIDs: PID anti-windup IV

Ejemplo (cont.): $G_p(s) = \frac{10}{s^2+2s+5}$ y $G_c(s) = \frac{1+1,5s}{s}$. Actuador limitado al rango ± 1

Los comportamientos del sistema real, en cada caso, son:

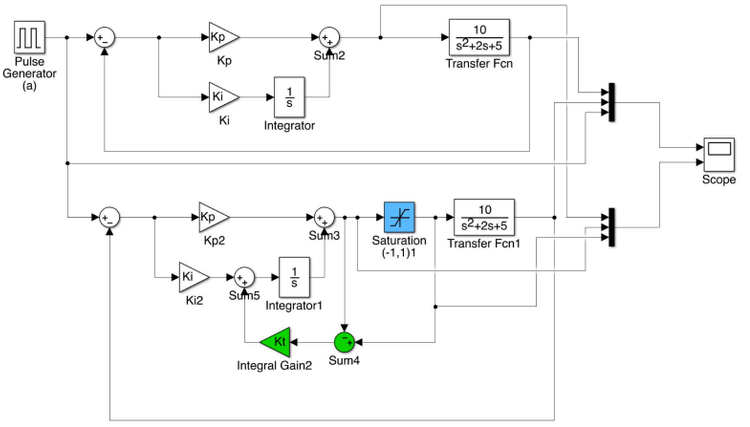
- $a = 0,5$: la señal de control nunca llega a los límites de la saturación. Por lo tanto, no hay diferencia entre la respuesta y la señal de control que hay entre los dos métodos.
- $a = 2,0$: la señal de control se encuentra saturada la mayor parte del tiempo. Sin embargo, la salida puede seguir a la entrada, ya que la ganancia propia de la planta (K=2) permite que la salida sea igual a la entrada (2) con el valor proporcionado por el controlador saturado. La señal de control nunca llega a tener el valor de 1 en el estacionario, ya que cuando el error pasa a ser 0, tenía un valor diferente. Pero como el actuador tiene el valor necesario (1), la discrepancia 1) no influye para el primer tramo de la referencia y 2) retarda la respuesta del sistema para el segundo tramo.
- $a = 4,0$: La señal de control se encuentra saturada siempre. Además, como para la acción máxima (1) la planta devuelve 2, existe un error constante 4-2. La acción integral acumula ese error de forma indefinida (el integrador se embala), hasta que se produce un cambio en la referencia. Aunque, el cambio en la referencia hace que el error cambie inmediatamente de signo, el valor acumulado en la acción integral no desaparece durante bastante tiempo. Por lo tanto, el actuador no modificará su valor de 1 hasta que el integrador se desembale, degradandose la respuesta temporal del sistema.

  En este último caso se produce el embale (wind-up, crecimiento indefinido) de la acción integral. Para evitarlo, hay que modificar el PID de forma que tenga en cuenta la discrepancia entre la señal de control u(t) y su versión saturada.

PIDs: PID anti-windup V

Existen diferentes métodos para corregir el fenómeno de windup: saturación de la integral, congelación de la integral o seguimiento del punto de operación.

En el método de tracking, lo que se hace es añadir unos elementos corrector a la acción integral (marcados en verde en la figura) que modifican el valor de la señal integrada en función de la discrepancia que existe entre la señal de control u(t) y su versión saturada.



El funcionamiento se basa en:

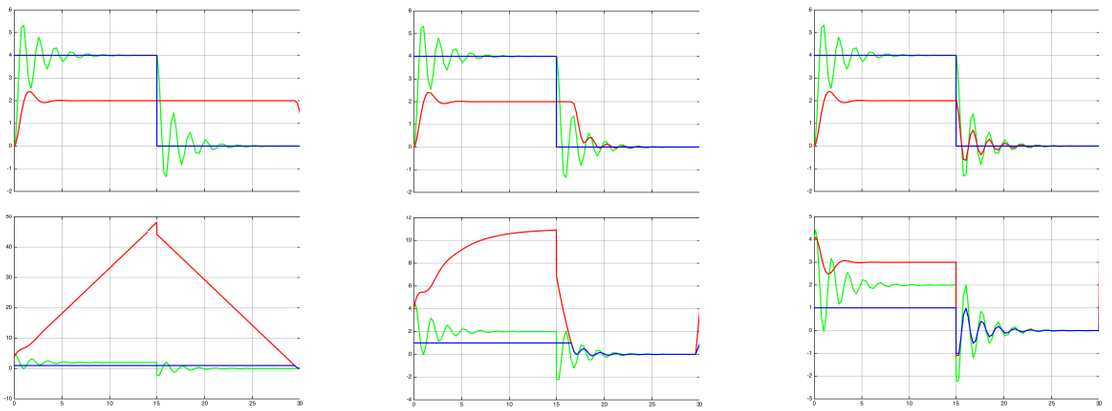
- Si la señal de control no se encuentra saturada, el término corrector vale 0 y por lo tanto, no afecta al funcionamiento del PID.
- Si la señal de control se encuentra saturada, la señal correctora toma un valor negativo que contrarresta el error constante del estacionario. En un momento dado se compensa el error con el término corrector, la entrada al integrador se anula, y la señal de control alcanza un valor constante.

PIDs: PID anti-windup VI

Además, la velocidad de respuesta depende del valor de la nueva ganancia K_t, que suele tomar un valor proporcional a K_i.

Ejemplo (cont.): $G_p(s) = \frac{10}{s^2+2s+5}$ y $G_c(s) = \frac{1+1{,}5s}{s}$. Actuador limitado al rango ± 1

$a = 4{,}0$; $K_t = 0{,}0$ | $a = 4{,}0$; $K_t = K_i/5$ | $a = 4{,}0$; $K_t = K_i$



K_t elevados:

- Reducen el tiempo de reseteo
- Pueden ser problemáticos en los casos en los que haya ruido y se tenga un PID con acción derivativa.

PIDs: PID anti-windup - resumen

El fenómeno del windup de un controlador que tiene un integrador puro aparece cuando:

- El rango de funcionamiento del actuador se encuentra limitado, y por lo tanto, en vez de aplicar la señal de control sobre la planta, se usa la permitida por los limites del actuador.
- La señal de referencia, con los limites del actuador, no puede ser alcanzada, a pesar de que el controlador tenga un término integral que, en ausencia de los límites, anularía el error real del sistema. Un error no nulo fijo hace que el integrador acumule valores continuamente, hasta que la señal de referencia cambie. El cambio de señal de referencia no puede ser seguido de forma inmediata por la planta: antes hay que descontar los valores acumulados innecesariamente por el controlador.

El problema se corrige de diversas formas. La alternativa vista, el seguimiento (tracking) del punto de operación:

- Es la más eficiente
- Modifica la entrada al integrador en función de la discrepancia que existe entre la señal control y la señal usada por el actuador.
- Tiene un parámetro de diseño, K_t que 1) permite regular el tiempo de respuesta del PID anti-windup y 2) habitualmente se sintoniza con un valor proporcional a la K_i.

Esquema

Diseño óptimo - I

La sintonización de los parámetros de los controladores del tema (redes, PIDs, o cualquier control de entrada-salida (basado en función de transferencia)) también se puede realizar:

- Eligiendo las características del sistema que hay que optimizar o restringir.
- Utilizar un algoritmo de optimización que ajuste automáticamente los parámetros del controlador en función de los parámetros elegidos.

Como parámetros se minimizan habitualmente:

- El error verdadero ($e(t) = r(t) - y(t)$) del sistema a lo largo del tiempo:
  - $\int_0^{t_{max}} |e(\tau)| d\tau$: penaliza por igual todos los errores.
  - $\int_0^{t_{max}} e^2(\tau) d\tau$: penaliza más los errores de mayor magnitud.
  - $\int_0^{t_{max}} |e(\tau)| t^n d\tau$: penaliza más los errores finales a los iniciales.
  - $\int_0^{t_{max}} e^2(\tau) t^n d\tau$: penaliza más los errores grandes y los finales.
- La señal de control $u(\tau)$ a lo largo del tiempo (4 variantes similares a las anteriores, sustituyendo la $u(\tau)$ por la $e(\tau)$).
- Los tiempos (subida, pico, asentamiento).
- La sobre-elongación.

Diseño óptimo - II

Como método de optimización, vamos a usar las funciones:

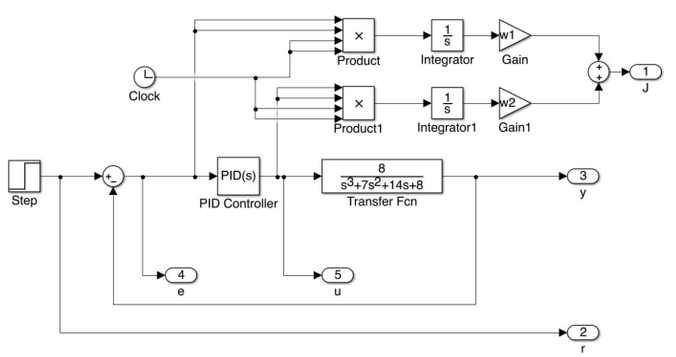
- `fminsearch` o `fminunc` para optimizar la combinación lineal de los parámetros elegidos siempre y cuando no haya restricciones:
  - `x=fminsearch(fun,x0)` utiliza la función `fun` para calcular la función que se desea optimizar y los parámetros iniciales del controlador almacenados en `x0` . Devuelve los parámetros de la mejor solución encontrada en `x`.
- `fmincon` para optimizar la combinación lineal de los parámetros elegidos siempre y cuando haya restricciones sobre otros parámetros.
  - `x=fminscon(fun,x0,[],[],[],[],minX,maxX,funcons)` utiliza la función `fun` para calcular la función que se desea optimizar, la función `funcons` para calcular cuanto falta para que se cumplan las restricciones y los parámetros iniciales del controlador almacenados en `x0` . Devuelve los parámetros de la mejor solución encontrada en `x`.
- Simulink para simular el comportamiento del sistema ante diferentes valores de los parámetros y calcular los valores de `fun` y `funcons` .
- IMPORTANTE: Estos métodos se quedan capturados en óptimos locales (el resultado depende del `x0` original).

Diseño óptimo - III

Ejemplo: Dada la planta $G_p(s) = \frac{8}{(s+1)(s+2)(s+4)}$ determinar los parámetros del PID que minimizan el error y la señal de control.

Para resolver el problema, podemos tomar como punto de partida los valores de las constantes determinadas por el método de Zieglers-Nichols.

Necesitamos tener un modelo que nos permita simular el comportamiento del sistema para unos valores del PID, y obtener los valores de las funciones de error y control cuadrático que queremos optimizar.



Para minimizar el error y la señal de control simultáneamente, vamos a optimizar su combinación lineal:

$w_1 \int_0^{t_{max}} e^2(\tau)d\tau + w_2 \int_0^{t_{max}} u^2(\tau)d\tau$.

Los valores de los pesos w_1 y w_2 reflejan la importancia de cada objetivo y sus magnitudes.

Diseño óptimo - IV

Ejemplo (cont.): Dada la planta $G_p(s) = \frac{8}{(s+1)(s+2)(s+4)}$ determinar los parámetros del PID que minimizan el error y la señal de control.

Después de construir el modelo Simulink, hay que escribir una función de Matlab que lance su simulación cada vez que la función de optimización (`fminsearch` o `fminunc`) tenga que evaluar un PID diferente.

```
function [Jend,simval]=Optim1Fun(x,w1,w2)

Kp=x(1);
Ki=x(2);
Kd=x(3);
simopt=simset('SrcWorkspace','current');
[t,x,J,r,y,e,u]=sim('Optim1',40,simopt);
Jend=J(end);
simval=[t,r,y,e,u,J];
```

La función de Matlab:

- Recibe los valores del PID en `x` , y los pesos `w1` y `w2` (para poder optimizar con diferentes combinaciones lineales de los objetivos).
- Pone en las variables que usa el modelo Simulink para configurar el PID los valores de `x` .
- Indica que los parámetros de la simulación hay que cogerlos de la función actual (por defecto elegiría los que se definen desde la linea de comandos de Matlab)
- Lanza la simulación.
- Devuelve el valor de la J final y la evolución de diferentes variables (para poder dibujarlas nosotros después).

Diseño óptimo - V

Ejemplo (cont.): Dada la planta $G_p(s) = \frac{8}{(s+1)(s+2)(s+4)}$ determinar los parámetros del PID que minimizan el error y la señal de control.

Finalmente, se escribe el programa de Matlab que utiliza los métodos de optimización y muestra los resultados.

```
%Optimización
%Pesos
w1=1;w2=0.1;
%Kp,Ki,Kd
PID0=[6.6472,8.4908, 1.3010];
[J0,simval0]=Optim1Fun(PID0,w1,w2);
opts=optimset('Display','iter');
PID=fminsearch(@(PID)Optim1Fun(PID,w1,w2),PID0,opts);
[J,simval]=Optim1Fun(PID,w1,w2);
%Representación gráfica
subplot(3,1,1);hold on;title('r(t), y(t)');
plot(simval0(:,1),simval0(:,2),'b')
plot(simval0(:,1),simval0(:,3),'r')
plot(simval(:,1),simval(:,3),'g')
legend('r(t)','y_{PID0}(t)','y_{PIDOpt}(t)');
subplot(3,1,2);hold on;title('e(t)');
plot(simval0(:,1),simval0(:,4),'r');
plot(simval(:,1),simval(:,4),'g');
subplot(3,1,3);hold on;title('u(t)');
plot(simval0(:,1),simval0(:,5),'r');
plot(simval(:,1),simval(:,5),'g');
```

El programa de Matlab:

- Inicializa los valores del PID (con los valores de Zieglers-Nichols)
- Simula con PID inicial (para comparar después)
- Optimizar con `fminsearch`
- Simula con el mejor PID obtenido.
- Representa graficamente los resultados.

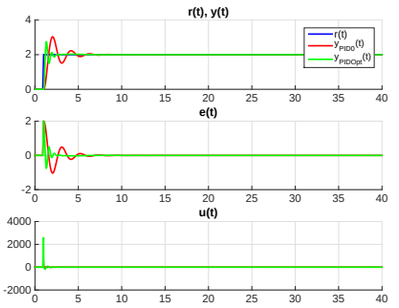
Diseño óptimo - VI

Ejemplo (cont.): Dada la planta $G_p(s) = \frac{8}{(s+1)(s+2)(s+4)}$ determinar los parámetros del PID que minimizan el error y la señal de control.

Vemos los resultados para diferentes combinaciones de los pesos, comparando los resultados contra los del

PID Original (Zieglers-Nichols): $K_p = 6{,}65$; $K_i = 8{,}5$ y $K_d = 1{,}3$.

$w_1 = 1$; $w_2 = 0{,}00$

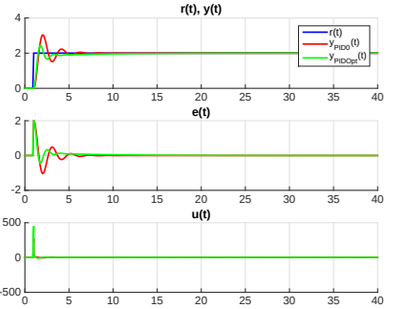


$J_{PID0} = 5{,}78$; $J_{opt} = 0{,}71$

$K_p = 6{,}65$; $K_i = 8{,}49$;

$K_d = 1{,}3$

$w_1 = 1$; $w_2 = 0{,}01$

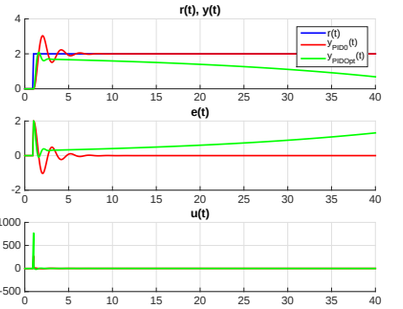


$J_{PID0} = 825{,}4$; $J_{opt} = 825{,}43$

$K_p = 8{,}5$; $K_i = 1{,}36$;

$K_d = 2{,}18$

$w_1 = 1$; $w_2 = 1$



$J_{PID0} = 8{,}4 \cdot 10^4$; $J_{opt} = 4{,}6 \cdot 10^4$

$K_p = 6{,}04$; $K_i = -0{,}28$;

$K_d = 3{,}81$

Si se parten de puntos iniciales diferentes, se pueden llegar a puntos finales diferentes.
Utilizar `fmincon` para poner limites a los parámetros del controlador.

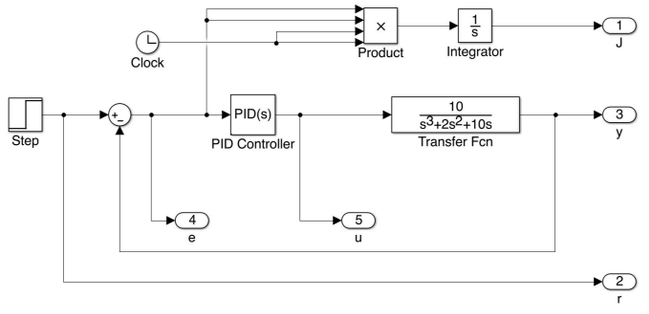
Diseño óptimo - VII

Ejemplo: Dada la planta $G_p(s) = \frac{10}{(s^3+2s^2+10s)}$ determinar los parámetros del PID que 1) minimizan el error y 2) aseguran que $t_s < 7$.

Para resolver el problema, podemos tomar como punto de partida los valores de las constantes determinadas por el método de Zieglers-Nichols.

Necesitamos tener:

- Un modelo que nos permita simular el comportamiento del sistema para unos valores del PID, y obtener el valor de la función de error que queremos optimizar
- La función de Matlab que permite que `fmincon` lance la simulación para obtener el valor del error para cada propuesta de valores del PID.




```
function [Jend,simval]=Optim2Fun(x)

Kp=x(1);
Ki=x(2);
Kd=x(3);
simopt=simset('SrcWorkspace','current');
[t,x,J,r,y,e,u]=sim('Optim2',40,simopt);
Jend=J(end);
simval=[t,r,y,e,u,J];
```

Diseño óptimo - VIII

Ejemplo (cont.): Dada la planta $G_p(s) = \frac{10}{(s^3+2s^2+10s)}$ determinar los parámetros del PID que 1) minimizan el error y 2) aseguran que $t_s < 7$.

Además, para que `fmincon` pueda comprobar si se cumple la restricción $t_s < 7$ para cada propuesta de valores de PID, se necesita una segunda función de Matlab que:

- Recibe los valores del PID, las pone en las variables correspondientes, y lanza la simulación.
- Calcula el valor de t_s buscando el instante de tiempo a partir del que la señal está comprendida entre el $\pm 2\,\%$ del valor final.

```
function [C,Ceq]=Optim2Con(x,a)
Kp=x(1);Ki=x(2);Kd=x(3);
simopt=simset('SrcWorkspace','current');
[t,x,J,r,y,e,u]=sim('Optim2',40,simopt);
%Cálculo del t_s
ysup=1.02*y(end);
yinf=0.98*y(end);
%Indices donde se está en rango
k=find(y<ysup & y>yinf);
%Si nunca se está en rango
if isempty(k)
    ts=t(end);
else
%Jay k no consecutivo fuera de rango?
l=find(diff(k)>1);
if isempty(l)  %Son consecutivos
    ts=t(k(1));  %ts está en primer k
else
%ts en el k siguiente al ultimo fuera
    ts=t(k(l(end)+1));
end
end
C=ts-a;  %Vector con Restricciones<0
Ceq=[];  %Vector con Restricciones==0
```

- Devuelve el valor de las restricciones respecto al 0: $t_s < a \rightarrow t_s - A < 0$.

Diseño óptimo - IX

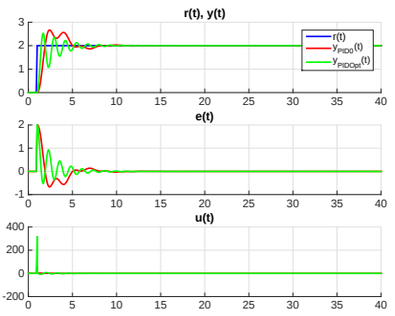
Ejemplo (cont.): Dada la planta $G_p(s) = \frac{10}{(s^3+2s^2+10s)}$ determinar los parámetros del PID que 1) minimizan el error y 2) aseguran que $t_s < 7$.

Finalmente, se escribe el programa de Matlab que utiliza los métodos de optimización y muestra los resultados.

```
%Optimización
a=7;
%Kp,Ki,Kd
PID0=[1.2,1.2079, 0.298];
[J0,simval0]=Optim2Fun(PID0);
C0=Optim2Con(PID0,a);
opts=optimset('Display','iter');
PID=fmincon(@(PID)Optim2Fun(PID),PID0,[],[],[],[],...
   [0,0,0],[inf,inf,inf],@(PID)Optim2Con(PID,a),opts)
[J,simval]=Optim2Fun(PID);
C=Optim2Con(PID0,a)
%Representación gráfica
subplot(3,1,1);hold on;title('r(t), y(t)');
h=plot(simval0(:,1),simval0(:,2),'b');
h=plot(simval0(:,1),simval0(:,3),'r');
h=plot(simval(:,1),simval(:,3),'g');
legend('r(t)','y_{PID0}(t)','y_{PIDOpt}(t)');
subplot(3,1,2);hold on;title('e(t)');
h=plot(simval0(:,1),simval0(:,4),'r');
h=plot(simval(:,1),simval(:,4),'g');
subplot(3,1,3);hold on;title('u(t)');
h=plot(simval0(:,1),simval0(:,5),'r');
h=plot(simval(:,1),simval(:,5),'g');
```

El programa de Matlab sigue los pasos utilizados en el ejemplo anterior, utilizándose en este caso la función de optimización `fmincon` .

El resultado con Matlab 2014b:



$C_{PID0} = 0{,}77$; $C_{opt} = 0{,}53$
$J_{PID0} = 10{,}42$; $J_{opt} = 5{,}36$

Diseño óptimo - X

Ejemplo (cont.): Dada la planta $G_p(s) = \frac{10}{(s^3+2s^2+10s)}$ determinar los parámetros del PID que 1) minimizan el error y 2) aseguran que $t_s < 7$.

Como el algoritmo se queda capturado en un óptimo local, modificamos el programa, para lanzar varias optimizaciones en torno al valor del PID obtenido con Zieglers-Nichols.

```
%Optimización
a=7;
%Variables para almacenar
PID0Store=[];PIDStore=[];JStore=[];CStore=[];
%Bucle para tener varias optimizaciones
for k=1:20
 PID0=[1.2,1.2079, 0.298]-0*5+rand(1,3);
 PID=fmincon(@(PID)Optim2Fun(PID),PID0,[],[],[],[],...
   [0,0,0],[inf,inf,inf],@(PID)Optim2Con(PID,a));
 %Almaceno
 PID0Store(end+1,:)=PID0,PIDStore(end+1,:)=PID;
 JStore(end+1,:)=Optim2Fun(PID);
 CStore(end+1,:)=Optim2Con(PID,a);
end
%Entre las que cumplen elijo la mejor
i=find(CStore<=0);[mv,j]=min(JStore(i));
PID= PIDStore(i(j),:); %PID  elegida
[J,simval]=Optim2Fun(PID);C=Optim2Con(PID,a);
PID0=[1.2,1.2079, 0.298];  %PID de Zieglers-Nichols.
[J0,simval0]=Optim2Fun(PID0);C0=Optim2Con(PID0,a);
%Codigo que muestra resultados
```

El programa de Matlab:

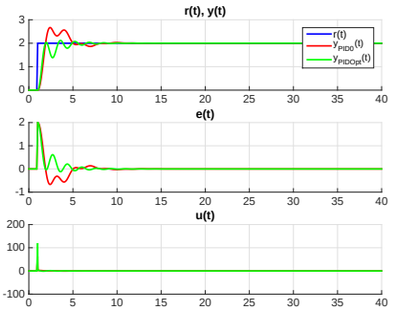
- Genera aleatoriamente y en torno a los valores propuestos por el método de Zieglers-Nichols los puntos de inicialización del algoritmo de optimización.
- Lanza varias optimizaciones sucesivas dentro de un bucle.
- Finalmente elige la que tiene menor error entre aquellas que cumplen las restricciones impuestas.

Diseño óptimo - XI

Ejemplo (cont.): Dada la planta $G_p(s) = \frac{10}{(s^3+2s^2+10s)}$ determinar los parámetros del PID que 1) minimizan el error y 2) aseguran que $t_s < 7$.

Los resultados obtenidos con el programa no siempre son los mismos, ya que dependen de los puntos iniciales que se prueben, y estos son generados aleatoriamente (en torno a los propuestos por Zieglers-Nichols).

En una ejecución del programa sobre Matlab, se han obtenido 3 soluciones factibles (que cumplen las restricciones) sobre los 20 intentos del bucle. La mejor (con mínimo valor de error) de dicha ejecución es:



	*PID*0	OPT
C	0.77	-0.65
J	10.42	4.34
K_p	1.2	1.3564
K_i	1.21	0.0002;
K_d	0.3	=0.5866

Los métodos vistos sirven para optimizar cualquier tipo de controlador (o problema de optimización no lineal). Aun así, a veces no consiguen encontrar solución, bien porque las restricciones impuestas no son posibles o porque el algoritmo se queda capturado en una zona no factible.