Prestaciones robustas, caso de estudio Matlab 5/(s+1)^2: mu síntesis

Este código ejecutó sin errores en Matlab **R2022a**

Presentaciones en vídeo:

http://personales.upv.es/asala/YT/V/cerp4mu.html , http://personales.upv.es/asala/YT/V/cerp5mu.html .

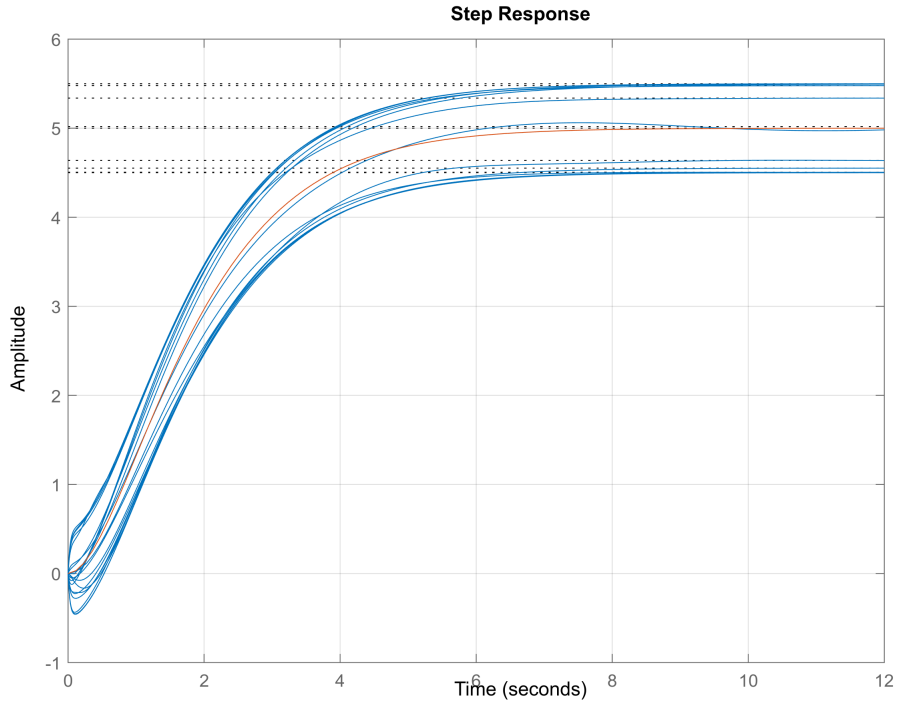
Objetivos: Comprender las ideas subyacentes a la teoría sobre prestaciones robustas y pequeña ganancia escalado con un ejemplo monovariable de segundo orden, ejecutado con mu-síntesis (búsqueda automática iterativa de multiplicadores) en vez de buscar "a mano" multiplicadores como en vídeos previos que sólo tenían interés didáctico/ilustrativo, pero **no** eran aconsejables como herramienta de diseño en un caso real.

Tabla de Contenidos

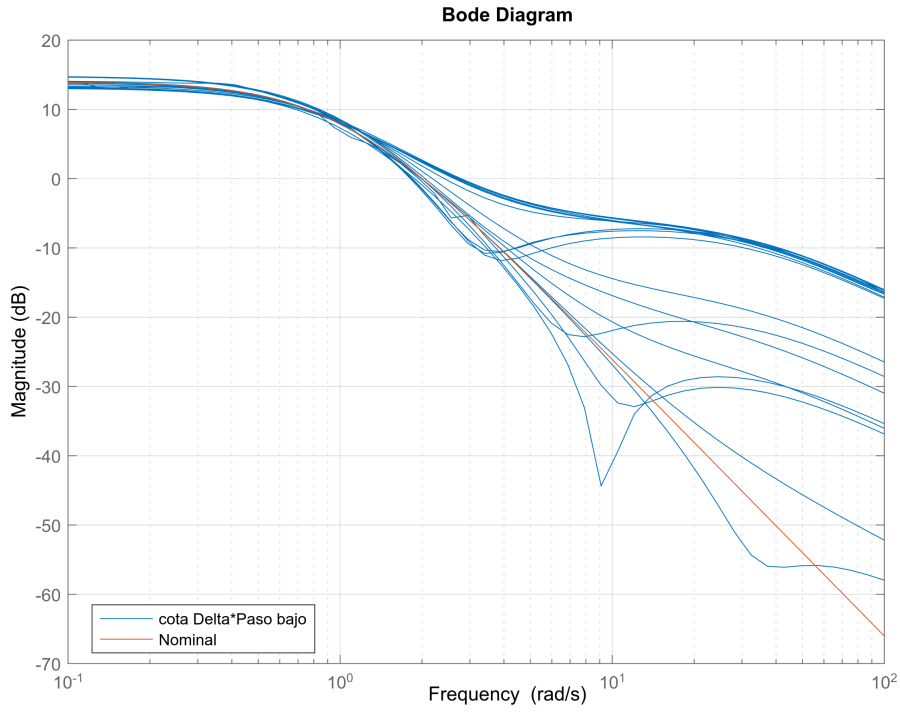
Modelo de planta a controlar 1
Cota deseada de prestaciones robustas (en frecuencia) 3
Diseño Mu-Síntesis para prestaciones robustas 3
Objetivo teórico 3
Planta Generalizada 2x2 para musyn 4
Planta Generalizada 3x3 a partir de 2x2+incertidumbre (por comparar, esto NO es necesario) 5
Planta generalizada ponderada para musyn 6
Mu-síntesis 7
Controladores de orden reducido 7
Reducción de orden del regulador 7
Optimización directa de un regulador tipo PID 9
Simulación prest. robustas en tiempo y frecuencia 10

Modelo de planta a controlar

```
s=tf('s');
G=5/(s+1)^2; %modelo NOMINAL

BoundDelta=0.5; %tamaño error modelado "Delta" aditivo no estructurado
Greal=G+BoundDelta*ultidyn("DeltaNormalized"); %modelo incierto para mu-síntesis

Greal_parasimular=G+BoundDelta*ultidyn("DeltaNormalized")*1/(0.03*s+1);
%añado "matar" Greal a partir de 33 rad/s. No usado en análisis ese
%paso-bajo final.

step(Greal_parasimular,G,12), grid on
```

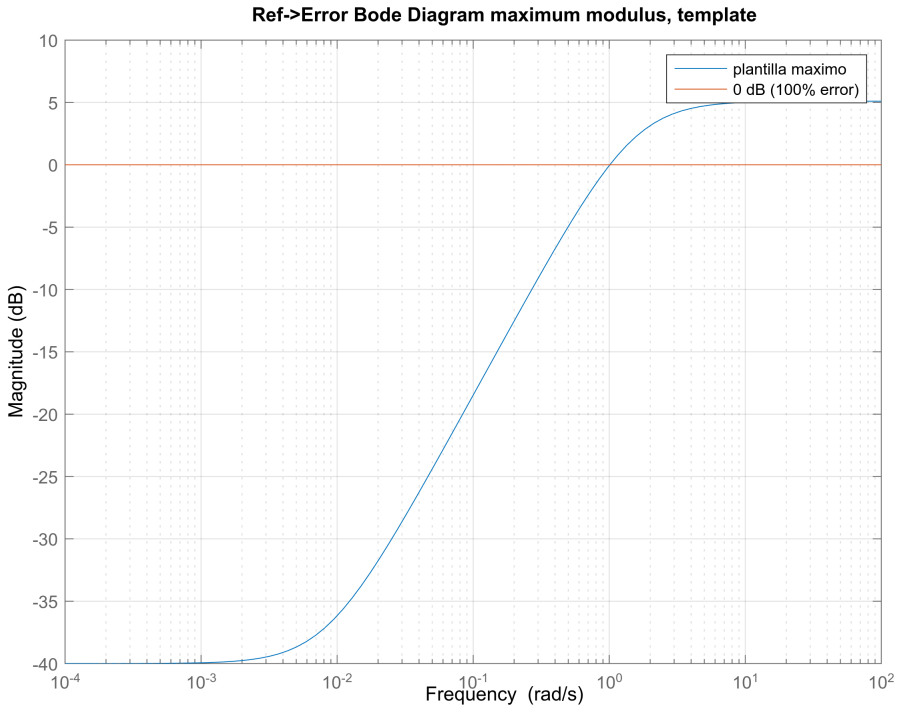



```
bodemag(Greal_parasimular,logspace(-1,2)), grid on, hold on
bodemag(G,logspace(-1,2)), hold off, legend("cota Delta*Paso bajo", "Nominal",Location=
```



Cota deseada de prestaciones robustas (en frecuencia)

```
TargetBandwidth=1.01; %aprox. inversamente prop. tiempo de subida
templateErr=makeweight(0.01,TargetBandwidth,1.8);
bodemag(templateErr,tf(1)), grid on
title("Ref->Error Bode Diagram maximum modulus, template")
legend("plantilla maximo","0 dB (100% error)")
```

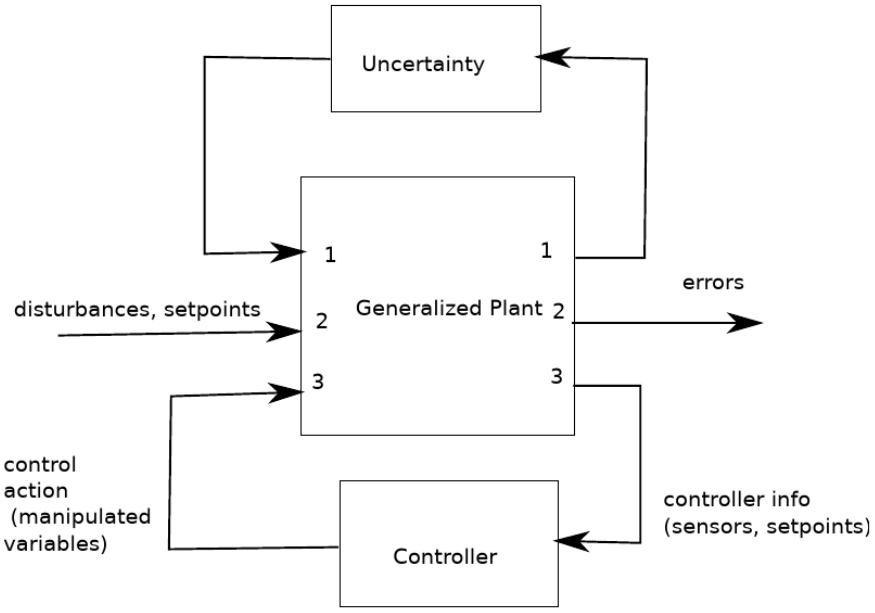


Nota: por simplicidad no limitamos la acción de control por tema "prestaciones" (saturación, etc.), aunque en aplicaciones sí sería recomendable. De todos modos, la robustez ante incertidumbre aditiva se consigue limitando "u", con lo que implícitamente sí estamos en cierto modo limitando "u".

Diseño Mu-Síntesis para prestaciones robustas

Objetivo teórico

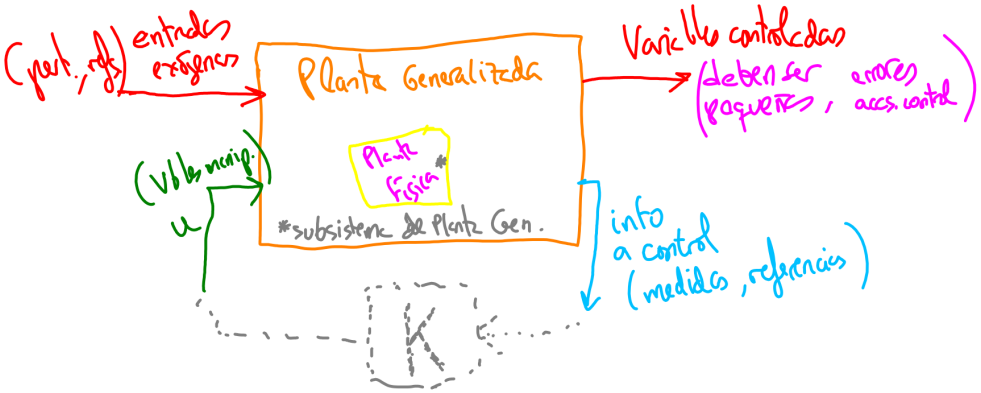
Trataríamos de construir una planta generalizada 3x3 con incertidumbre para aplicar (por ejemplo) teorema de pequeña ganancia escalado si quisiéramos ejecutar la búsqueda de multiplicadores por nuestra cuenta (como se ha hecho en otros vídeos anteriores, "manualmente").



Si nosotros, no obstante, construimos con un modelo incierto sólo la parte 2x2 que se refiere a los grupos 2 y 3, el Robust Control Toolbox de Matlab se encarga de extraer la incertidumbre y plantear (internamente) la planta 3x3. [Cuando digo 2x2 o 3x3 me refiero a "grupos" de señales, cada grupo tiene unas dimensiones diferentes dependiendo de cada problema en concreto a resolver].

*Si queremos ver "explícitamente" la planta 3x3 que utiliza **`musyn`**, lo podemos hacer con el comando **`lftdata`** (ver abajo), pero **no** es necesario ese uso explícito en la mayoría de los casos sencillos (ya se encarga musyn de hacerlo), y planteamos la planta generalizada con la incertidumbre "dentro".

Planta Generalizada 2x2 para musyn



No es recomendable dos "occurrences" si físicamente sólo hay "una". Mejor escribimos `[err;err]=[1 -Greal;1 -Greal]*[ref;u]` sin multiincidencia de Greal:

```
PlantaGenMu=minreal([[1;1] [-1;-1]*Greal] )%if we execute "ss" then uncertainty is dest
```

```
PlantaGenMu =

  Uncertain continuous-time state-space model with 2 outputs, 2 inputs, 2 states.
  The model uncertainty consists of the following blocks:
```

```
    DeltaNormalized: Uncertain 1x1 LTI, peak gain = 1, 1 occurrences

Type "PlantaGenMu.NominalValue" to see the nominal value, "get(PlantaGenMu)" to see all properties, and "P
```

Planta Generalizada 3x3 a partir de 2x2+incertidumbre (por comparar, esto NO es necesario)

Extraemos la incertidumbre con `lftdata` como sigue (realmente NO es necesario hacerlo, ya lo hace `musyn` internamente, transparente a nosotros):

```
[M,Delta]=lftdata(PlantaGenMu);
size(M)
```

```
State-space model with 3 outputs, 3 inputs, and 2 states.
```

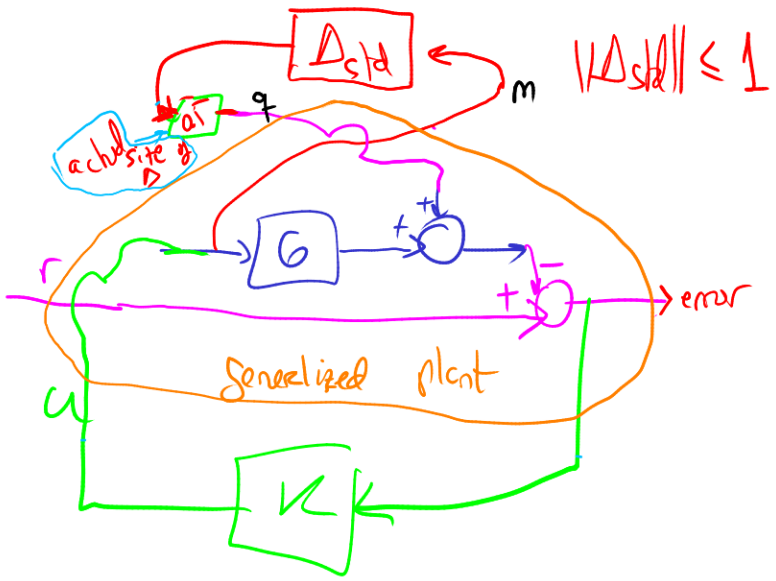
```
Delta
```

```
Delta =

  Uncertain continuous-time state-space model with 1 outputs, 1 inputs, 0 states.
  The model uncertainty consists of the following blocks:
    DeltaNormalized: Uncertain 1x1 LTI, peak gain = 1, 1 occurrences

Type "Delta.NominalValue" to see the nominal value, "get(Delta)" to see all properties, and "Delta.Uncerta
```

```
zpk(M)
```

```
ans =

  From input 1 to output...
   1:  0

   2:  -0.5

   3:  -0.5

  From input 2 to output...
   1:  0

   2:  1

   3:  1

  From input 3 to output...
   1:  1

         -5
   2:  -------
       (s+1)^2

         -5
   3:  -------
       (s+1)^2

Continuous-time zero/pole/gain model.
```

Coincide con lo que nosotros habíamos propuesto al buscar "a mano" el multiplicador en otros vídeos (bueno, incorporando además el peso de la incertidumbre). En efecto




```
PlantGenOtherVideo=minreal(ss([0 0 1;-1 1 -G;-1 1 -G])); %NOT NEEDED when using musyn
zpk(PlantGenOtherVideo*blkdiag(BoundDelta,1,1)) %We add input weight with size of uncer
```

```
ans =

  From input 1 to output...
   1:  0

   2:  -0.5

   3:  -0.5

  From input 2 to output...
   1:  0

   2:  1

   3:  1

  From input 3 to output...
   1:  1

         -5
   2:  -------
       (s+1)^2

         -5
   3:  -------
       (s+1)^2

Continuous-time zero/pole/gain model.
```

Planta generalizada ponderada para musyn

```
Win=blkdiag(1,1);
Wout=minreal(blkdiag(1/templateErr,1));
```

```
1 state removed.
```

```
PlantaGenPond=Wout*PlantaGenMu*Win %la incertidumbre está dentro de PlantaGenMu
```

```
PlantaGenPond =

  Uncertain continuous-time state-space model with 2 outputs, 2 inputs, 3 states.
  The model uncertainty consists of the following blocks:
    DeltaNormalized: Uncertain 1x1 LTI, peak gain = 1, 1 occurrences

Type "PlantaGenPond.NominalValue" to see the nominal value, "get(PlantaGenPond)" to see all properties, an
```

Mu-síntesis

Minimizar la norma infinito con búsqueda de multiplicadores lo haríamos con:

```
[Kmu,GAM]=musyn(PlantaGenPond,1,1); GAM %no gastamos ni "M" ni "Delta" del lftdata, ya
```

```
D-K ITERATION SUMMARY:
------------------------------------------------------------------
                     Robust performance               Fit order
------------------------------------------------------------------
  Iter         K Step       Peak MU       D Fit             D
    1           1.021        1.021        1.025             4
    2           1.001        1.001        1.005             8
    3          0.9986       0.9986       0.9987             8
    4          0.9981       0.9981       0.9984             8

Best achieved robust performance: 0.998
GAM = 0.9981
```

*****El multiplicador para un ultidyn SISO es un $S(s)$ dinámico, que mejora las prestaciones que se podrían probar respecto a un multiplicador constante visto en otros vídeos preliminares. Bueno, se le denomina "$D(s)$" en la salida de `musyn`, cuando presenta `D fit, Fit Order.`

Sería una opción "teóricamente mejor" si la incertidumbre es lineal e invariante en el tiempo...
pero observad cómo el multiplicador dinámico sube el orden del regulador resultante. A cambio, no aseguraría robustez ante no-linealidad con distorsión armónica o variante en el tiempo.

```
size(Kmu) %lti uncertainty permite pesos en frecuencia, por eso sube el orden.
```

```
State-space model with 1 outputs, 1 inputs, and 11 states.
```

Controladores de orden reducido

Tenemos varias opciones:

- reducir el orden regulador obtenido por musyn,
- Indicar a musyn mediante ciertas opciones que el orden del escalado "D" no sea tan elevado
- Directamente optimizar algo de orden menor (como un PID)

Vamos a ver la primera y tercera.

Reducción de orden del regulador

```
hsvd(Kmu) %muchos valores singulares son muy pequeños, podemos reducir con (esperemos)
```

```
ans = 11×1
   10.9823
```

```
    0.4618
    0.2640
    0.0316
    0.0006
    0.0005
    0.0003
    0.0001
    0.0001
    0.0000
     ⋮
```

```
Kmu_reduced=balred(Kmu,3);
wcgain(lft(PlantaGenPond,Kmu_reduced))
```

```
ans = struct with fields:
           LowerBound: 0.9984
           UpperBound: 1.0005
    CriticalFrequency: 0.9415
```

```
zpk(Kmu_reduced)
```

```
ans =

  -0.063784 (s-9031) (s+1.182) (s+0.895)
  --------------------------------------
     (s+665.3) (s+4.892) (s+0.008398)

Continuous-time zero/pole/gain model.
```

Como hay un polo "rápido" vamos a quitarlo con `freqsep`, independientemente de su efecto `hsvd`, a ver qué podemos obtener:

```
[Kslow,Kfast]=freqsep(Kmu_reduced,150);
zpk(Kslow)
```

```
ans =

  -0.063784 (s-0.3137) (s+43.86)
  ------------------------------
      (s+0.008398) (s+4.892)

Continuous-time zero/pole/gain model.
```

```
zpk(Kfast)
```

```
ans =

   621.13
  ---------
  (s+665.3)

Continuous-time zero/pole/gain model.
```

```
Kmu_reduced2=Kslow+dcgain(Kfast);
wcgain(lft(PlantaGenPond,Kmu_reduced2)) %seguimos teniendo prestaciones robustas
```

```
ans = struct with fields:
```

```
           LowerBound: 0.9965
           UpperBound: 0.9983
    CriticalFrequency: 0.7940
```

```
zpk(Kmu_reduced2) %it's a PID+'noise filter' controller
```

```
ans =

  0.86976 (s+0.915) (s+1.151)
  ---------------------------
    (s+0.008398) (s+4.892)

Continuous-time zero/pole/gain model.
```

El controlador resultante se parece mucho a un PID con filtro de ruido... ello sugiere la optimización directa de un PID, que vemos a continuación.

Optimización directa de un regulador tipo PID

```
PID=tunablePID('PIDtest','PID');
ClosedLoopWithPID=lft(PlantaGenPond,PID);tic
[TunedCL,GAM]=musyn(ClosedLoopWithPID); GAM,toc
```

```
D-K ITERATION SUMMARY:
-----------------------------------------------------------------
                      Robust performance               Fit order
-----------------------------------------------------------------
  Iter          K Step       Peak MU       D Fit             D
    1           1.022        1.022        1.024             4
    2           1.005        1.005        1.008             8
    3           1.002        1.001        1.002             8
    4           1.001        1.001        1.001             8

Best achieved robust performance: 1
GAM = 1.0009
Elapsed time is 1.900283 seconds.
```

Pues no hemos perdido prácticamente nada respecto al regulador "state space" genérico. Bueno, igual habría que reducir un poquito el ancho de banda para estar "debajo" de 1, pero un 1 por mil no va a importar lo más mínimo en la práctica (la tolerancia interna de los solvers h-infinito por defecto ya es del 1 por cien)...

```
TunedPID=pid(TunedCL.Blocks.PIDtest)
```

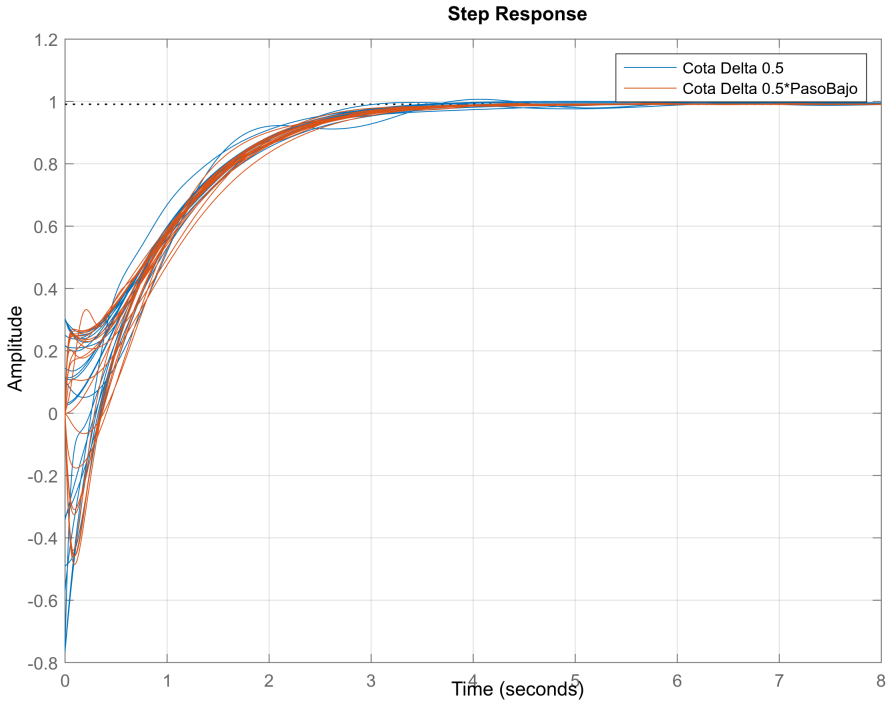
```
TunedPID =

             1            s
  Kp + Ki * --- + Kd * --------
             s          Tf*s+1

  with Kp = 0.332, Ki = 0.189, Kd = 0.11, Tf = 0.206

Name: PIDtest
Continuous-time PIDF controller in parallel form.
```

```
zpk(TunedPID) %se parece al Kmu_reduced2 de arriba, tolerancias aparte consiguen básica
```

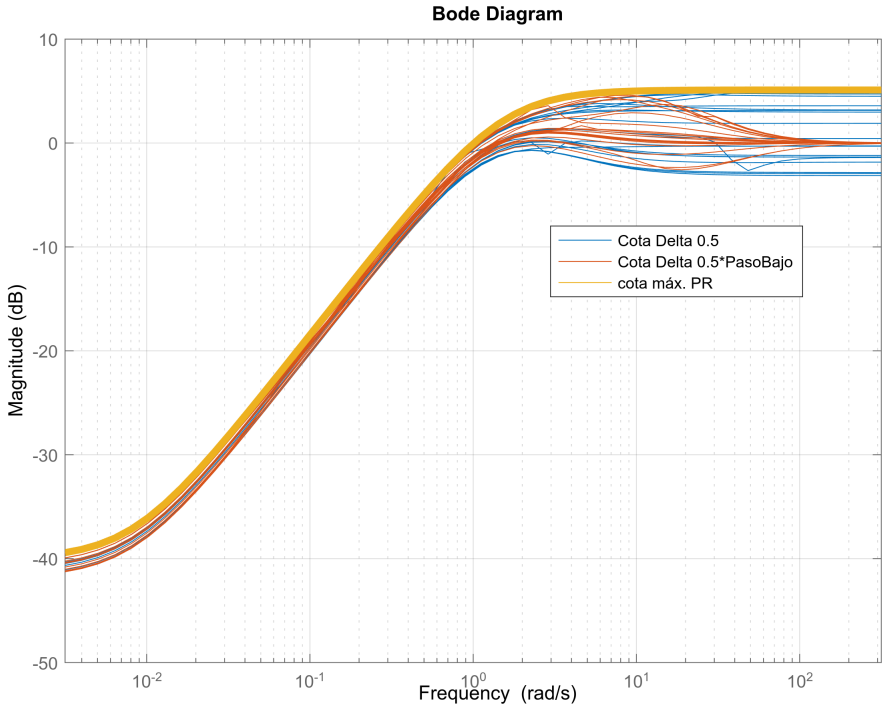
```
ans =

  0.86741 (s+1.19) (s+0.8884)
  ---------------------------
          s (s+4.863)

Name: PIDtest
Continuous-time zero/pole/gain model.
```

Simulación prest. robustas en tiempo y frecuencia

```
Kfinal=Kmu_reduced2;
%Kfinal=TunedPID;
step(feedback(Greal*Kfinal,1),feedback(Greal_parasimular*Kfinal,1),8), grid on
legend("Cota Delta 0.5","Cota Delta 0.5*PasoBajo")
```

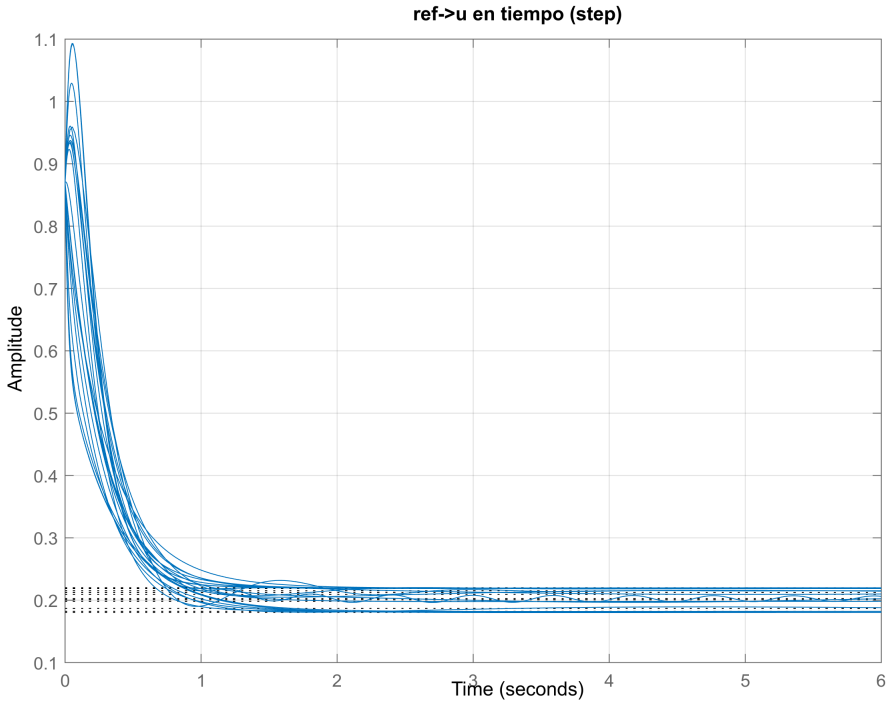



```
bodemag(feedback(1,Greal*Kfinal),feedback(1,Greal_parasimular*Kfinal),templateErr,logsp
h_line=findobj(gcf,'type','line');
h_line(46).LineWidth=4; %trial and error to get the right line to make thicker
legend("Cota Delta 0.5","Cota Delta 0.5*PasoBajo","cota máx. PR",Location="best") %erro
```

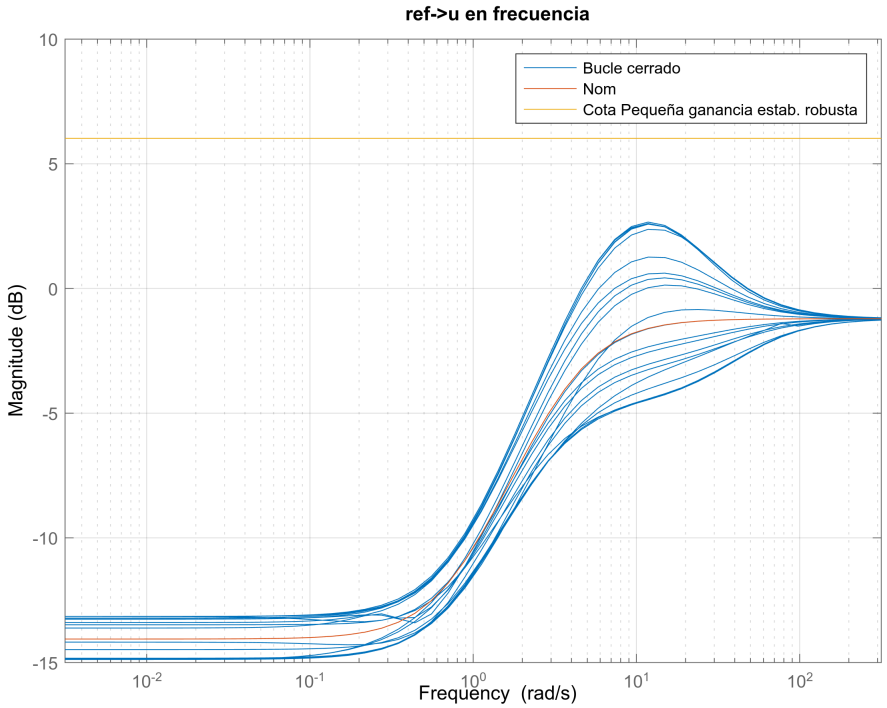


● ¿Acción de control? Todo control "final" debe tener una acción de control "razonable". No lo hemos limitado teóricamente, pero es importante, por lo que vamos a ver cuál es:

```
step(feedback(Kfinal,Greal_parasimular),6), grid on
title("ref->u en tiempo (step)")
```




```
Fu=feedback(Kfinal,Greal_parasimular);
bodemag(Fu,logspace(-2.5,2.5)), hold on
bodemag(Fu.NominalValue,tf(1/BoundDelta),logspace(-2.5,2.5)), hold off
grid on, title("ref->u en frecuencia"),legend("Bucle cerrado","Nom","Cota Pequeña ganar
```



Obviamente, está por debajo de la cota de estabilidad robusta (pequeña ganancia clásico, no escalado) porque asegurar prestaciones robustas es más exigente que asegurar sólo estabilidad robusta.