CONTROL ROBUSTO LINEAL:
ideas básicas y caso de estudio

Antonio Sala

DISA – AI^2 Universitat Politècnica de València

Ciclo de Conferencias UNED Automática – 2022

UNED

Sobre mí [*Antonio Sala*]

Trabajo como CU en Univ. Polit. Valencia, Depto. Ing. Sistemas y Automática, también adscrito a instituto AI^2.

Me dedico a cosas bastante variadas:

1. control óptimo, predictivo, robusto, LPV/Takagi-Sugeno.
2. Control inteligente, aprendizaje, estadística multivariable, ...

3. Vídeos docentes de Automática ¿TE SUSCRIBES?
   - Canal YouTube A. Sala: Modelado, Identificación, Control
     https://www.youtube.com/channel/UCILAiYOqDUihxt-0norW76g
   - Colección de materiales asociados a los vídeos:
     http://personales.upv.es/asala/docenciaonline/

*Agradezco UNED (Sebastián Dormido, Fernando Morilla) invitación a esta charla.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Control Robusto: motivación

Los errores de modelado son omnipresentes: no-linealidad, dinámica no modelada, parámetros mal estimados, ... Es importante diseñar controladores con garantías de que lo van a tolerar adecuadamente (conseguir *prestaciones robustas*).

Objetivos:

- Introducir el problema a resolver y su perspectiva histórica.
- Comprender las ideas básicas que subyacen al control robusto lineal basado en $\mathcal{H}_\infty$ y pequeña ganancia escalado (μ-síntesis, $\approx$1990's), basándose en un caso de estudio aplicado a $G(s) = \frac{1}{s^2}$.

Sección 1

Preliminares

Objetivos del control (robusto... ¿o todas las técnicas?)

Los procesos tienen errores de modelado: no-linealidades y dinámica de alto orden (no modeladas o simplificadas), parámetros mal identificados o variantes en el tiempo...

El control debe, por orden de prioridad, garantizar:

1. Que "simulación" se parezca a "experimento" **ROBUSTEZ**
2. Que "simulación" tenga prestaciones "buenas" (si se cumple 1, ello implica que "experimento" será satisfactorio). **PRESTACIONES**
3. Que los modelos y las estrategia de control sean "simples" de entender, implementar, mantener **BAJA COMPLEJIDAD**

Consideraciones "cualitativas/prácticas" control robusto

1.- Limitaciones de amplitud de señales:

- No excites no-linealidades, no satures.

2.- Limitaciones de rapidez de señales:

- Si tu identificación era ante "escalón" (baja frec), no excites altas frecuencias.
- El modelo de tu proceso y tu actuador a alta frecuencia es BASURA.
- No cambies referencias demasiado rápido: en procesos paso-bajo, seguirlas requiere amplitudes y frecuencias altas en acc. de control.

3.- Si algo no funciona, compra más sensores y mejora ancho de banda de actuadores.

4.- Diseña procesos fáciles de controlar (poco acoplados, rápidos, repetibles, bien condicionados, . . .).

*Esto es lo que la gente "práctica" hace para que "las cosas funcionen", sin saber ni quién era Laplace.

Revisión histórica de enfoques en control robusto* *LINEAL

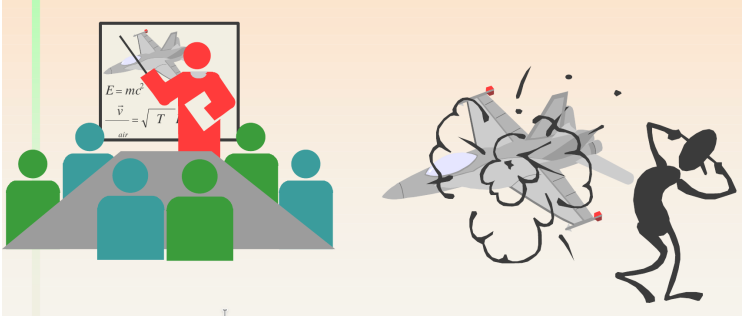
- Prueba y error... sintonizar un PID a mano que funcione, evitar ganancias altas en procesos estables (desde que se inventó la realimentación), reglas transp. 6.
- Márgenes de fase, ganancia, retardo en PIDs (1940-60), metodologías gráficas SISO en frecuencia (diag. Bode, Nyquist, Nichols). $1 + GK = 0$ robusto si $G(j\omega)K(j\omega)$ lejos de -1.
- Pasividad, Popov, Criterio del Círculo (1950-60) para incert. no lineal.
- Quantitative Feedback Theory (1960), añadir "plantillas" de posibles errores de modelado a lo anterior. Una buena propuesta para 1960 (resolución gráfica), pero no para 1990+ (sólo aplica a SISO y mejor optimizar con Matlab).
- Control óptimo $\mathcal{H}_\infty$ y μ-síntesis (1985-95).
  Incluye y generaliza todo lo anterior. [Objetivo de esta charla]
- Extensiones (Integral Quadratic Constraints, Sistemas Politópicos, Planif.Ganancia, retardo variante) a partir de 1995... [Doctorado]

Sección 2

Enfoque "cuantitativo": análisis de robustez

Objetivos del enfoque cuantitativo

Incluir en el análisis o diseño el **error de modelado**. Intentar **cuantificar** la "probabilidad", "confianza" o "margen" de que un diseño funcione "en la práctica".



Modelado de sistemas lineales inciertos

El proceso real se supondrá $G_{real}(s, \Delta)$ siendo Δ un vector de "incrementos" de parámetros inciertos o dinámica no modelada que toma valores en un cierto conjunto $\mathbf{\Delta}$, siendo su **valor nominal** de $\mathbf{0}$.

Ejemplo red RC: $G = 1/(RCs + 1)$, con valores nominales de $R = 100$, $C = 10^{-4}$ y tolerancias de 5% + dinámica de actuador (fuente alim.) no modelada. Lo modelamos como:

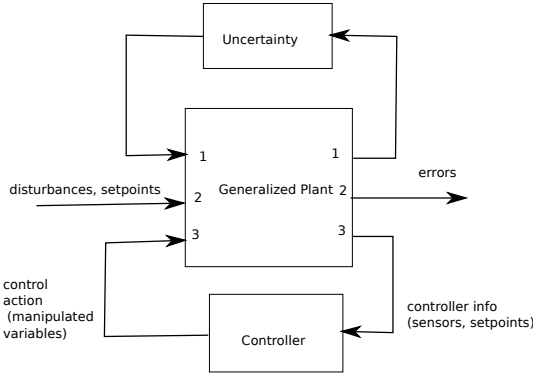
$$G_{real}(s, \underbrace{\delta_R, \delta_C, \delta_N}_{\Delta}) = \frac{1}{100(1+\delta_R) \cdot 10^{-4}(1+\delta_C) \cdot s + 1} \cdot (1 + \delta_N(s))$$

y acotaremos el "tamaño" de δ_R, δ_C, y δ_N (el último en función de la frecuencia).

$$G_{nominal} = G(s, 0, 0, 0) = \frac{1}{10^{-2}s+1}$$

El problema generalizado con incertidumbre

Planta generalizada 3×3 con incertidumbre:



*Con adecuados escalados y ponderaciones en frecuencia, incertidumbre $\|\Delta\|_\infty < 1$:
objetivo de control: que $\|errores\|_2 \leq \|\text{ent. exógenas}\|_2$ para todo Δ tal que $\|\Delta\|_\infty < 1$.

*No se puede optimizar directamente controlador: Δ es **desconocida**.

Sección 3

Ejemplo Robust Control Toolbox (MATLAB)

Doble integrador con dinámica de actuador y masa inciertas, fuerzas externas: diseño mu-síntesis

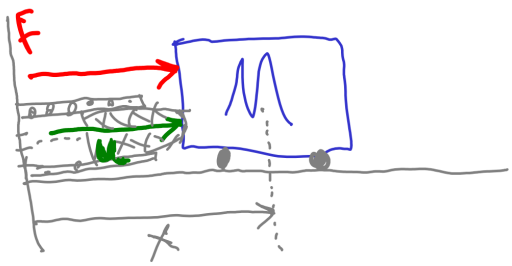
Este código ejecutó sin errores en Matlab **R2022a**

Objetivos: comprender el workflow de un diseño μ-síntesis en un caso monovariable de seguimiento de referencias y rechazo de perturbaciones a la entrada (combinado en un único problema), en un ejemplo sencillo (doble integrador).

Tabla de Contenidos

Diagrama conceptual del proceso a controlar

Masa (incierta) a ser movida a una cierta posición con actuador incierto y perturbaciones de fuerza (perturbación a la entrada, en la "jerga" de control de procesos) a rechazar.



Construcción modelo planta física incierto

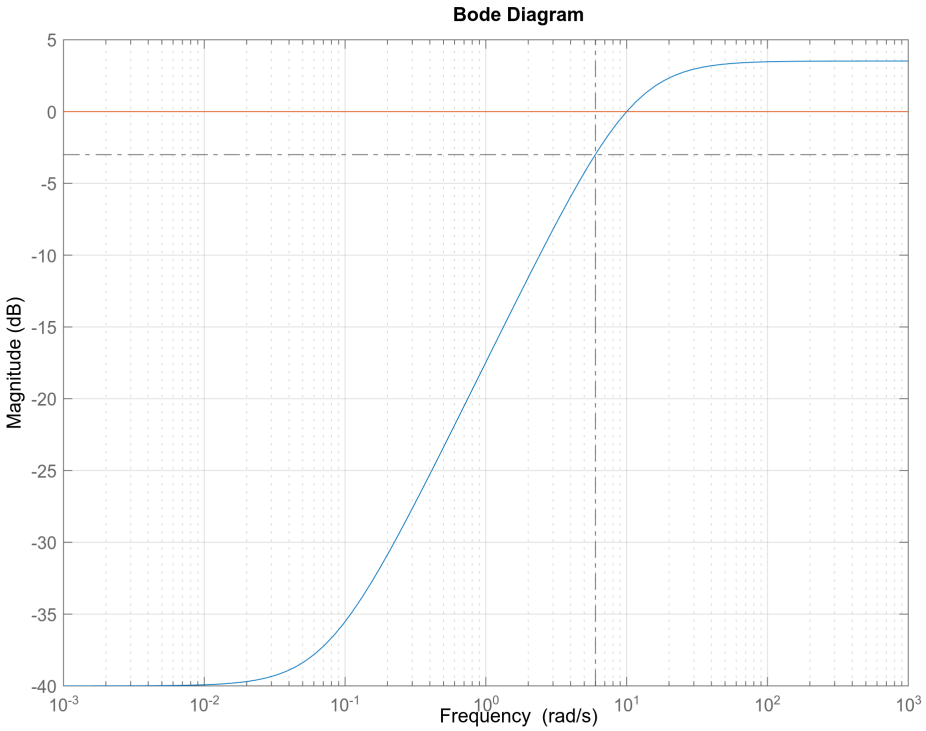
- Modelo incertidumbre en la masa:

```
Mnominal=1;

W_unc_in=tf(0.05);
Delta2=ureal("Delta_percent_masa",0); %número al azar entre -1 y +1
Masa=Mnominal*(1+W_unc_in*Delta2); %Masa incierta

s=tf('s');
G=1/Masa*1/s^2
```

```
G =

  Uncertain continuous-time state-space model with 1 outputs, 1 inputs, 2 states.
  The model uncertainty consists of the following blocks:
    Delta_percent_masa: Uncertain real, nominal = 0, variability = [-1,1], 1 occurrences

Type "G.NominalValue" to see the nominal value, "get(G)" to see all properties, and "G.Uncertainty" to int
```

- Peso incierto actuador electromecánico:

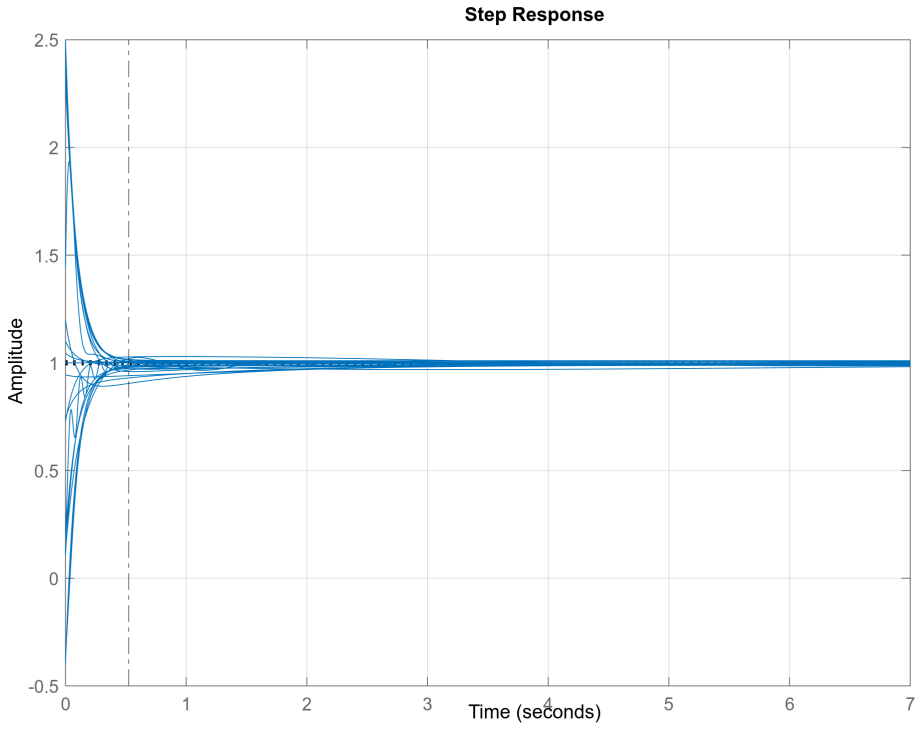
```
W_unc_act=makeweight(0.01,10,1.5);
bodemag(W_unc_act,tf(1)), grid on, xline(6,'-.'),yline(-3,'-.')
```




```
W_unc_act.InputName={'u'};
W_unc_act.OutputName={'w1'};
```

Construyamos modelo incierto con el toolbox del actuador:

```
actuador_nominal=1;
Delta1=ultidyn("Delta_actuador");
actuador = actuador_nominal*(1+W_unc_act*Delta1);
step(actuador,7), grid on, xline(pi/6,'-.') %relacion ancho banda con tiempo subida o e
```



- Modelo del sistema físico sujeto a la suma de fuerzas perturbación+manipulada:

```
sys=G*[1 actuador];
sys.InputName={'f','u'};sys.OutputName={'pos'};
sys
```

```
sys =

  Uncertain continuous-time state-space model with 1 outputs, 2 inputs, 3 states.
  The model uncertainty consists of the following blocks:
    Delta_actuador: Uncertain 1x1 LTI, peak gain = 1, 1 occurrences
    Delta_percent_masa: Uncertain real, nominal = 0, variability = [-1,1], 1 occurrences

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

```
tf(minreal(sys.NominalValue))
```

```
1 state removed.

ans =

  From input "f" to output "pos":
   1
  ---
  s^2

  From input "u" to output "pos":
   1
  ---
```

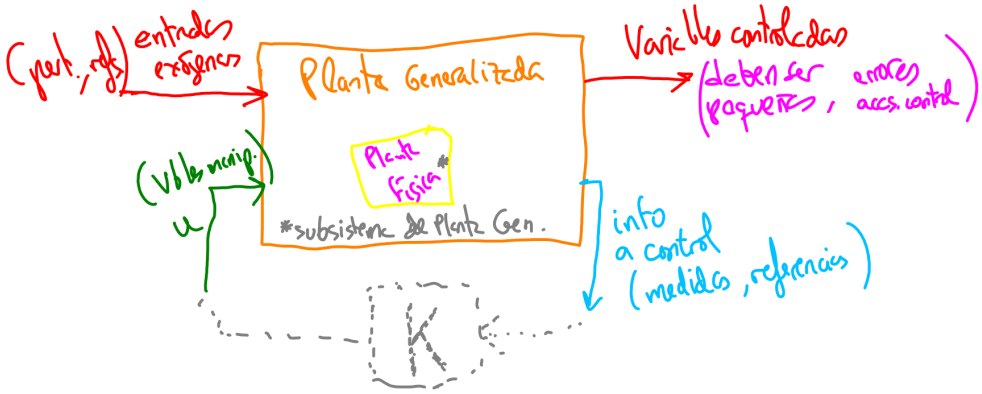
```
   s^2

Continuous-time transfer function.
```

```
%bodemag(sys,sys.NominalValue,logspace(-1,2,100)), grid on,legend("Realizaciones azar",
```

Construcción de planta generalizada para prestaciones robustas

Planta generalizada NO ponderada

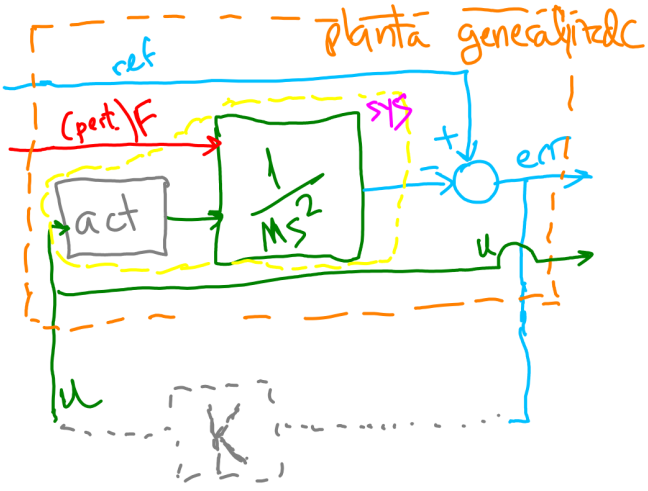
La planta "física" que hemos construido en la variable "`sys`" debe integrarse en un diagrama de bloques que defina el problema a resolver.



El problema es un combinado de rechazo de perturbación y seguimiento de referencia.

Las salidas de planta generalizada serán:

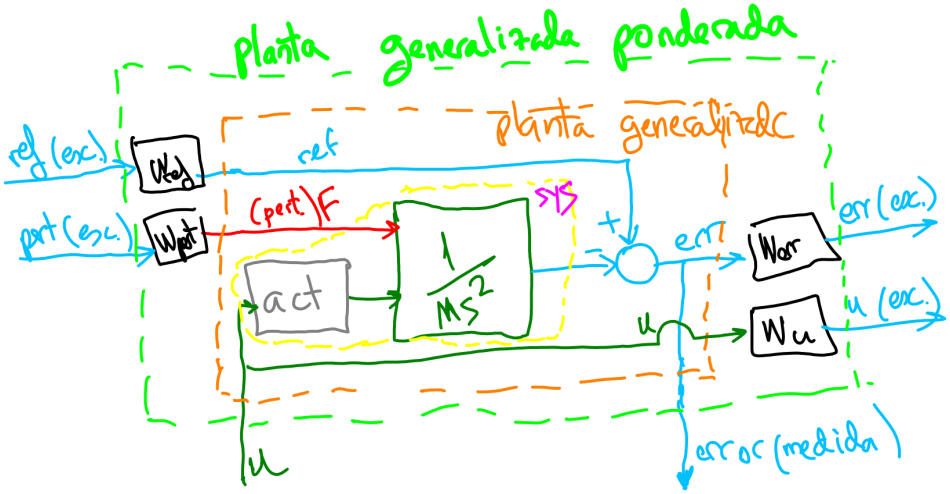
- error ante referencia, acción de control [prestaciones, grupo 1] y
- medida (error) disponible para control 1GL (grupo 2).




```
PlantaGen=[ [1;0;1] [-1;0;-1]*sys+[0 0;0 1;0 0] ];
%también lo podríamos haber hecho con el comando connect, por ejemplo.
PlantaGen.InputName={'ref','pert','u'};
PlantaGen.OutputName={'err','u','err_m'};
```

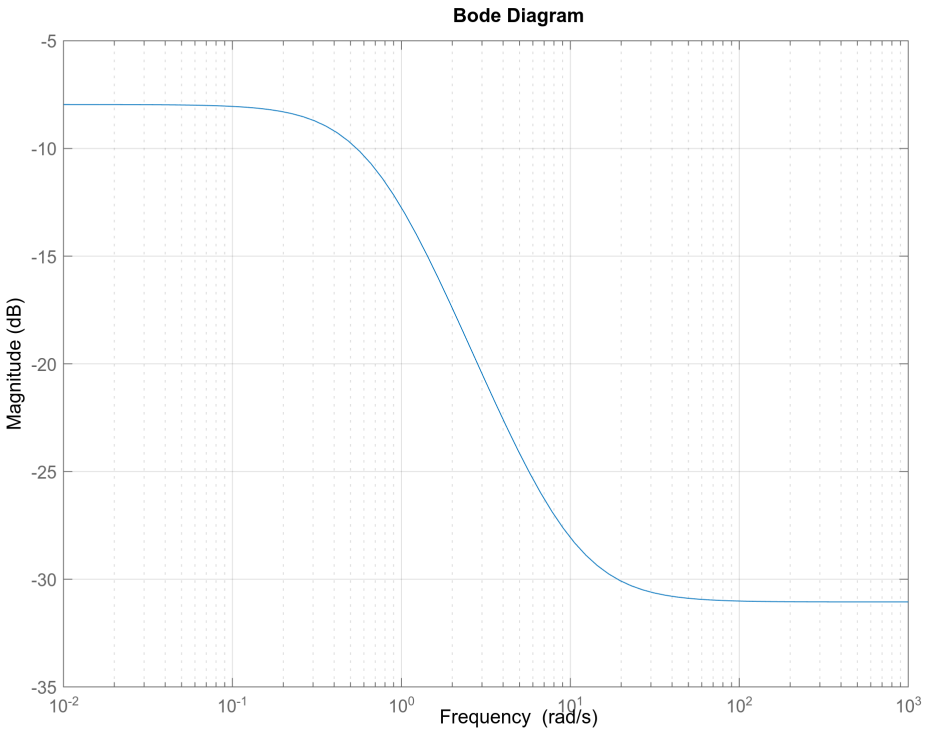
Planta generalizada ponderada

Las entradas y salidas (excepto las que van al controlador, que son unidades "físicas") deben escalarse para que las entradas "escaladas" tengan tamaño unidad, y que las salidas "escaladas" tengan tamaño igual o menor a la unidad cuando las especificaciones deseadas se cumplan.

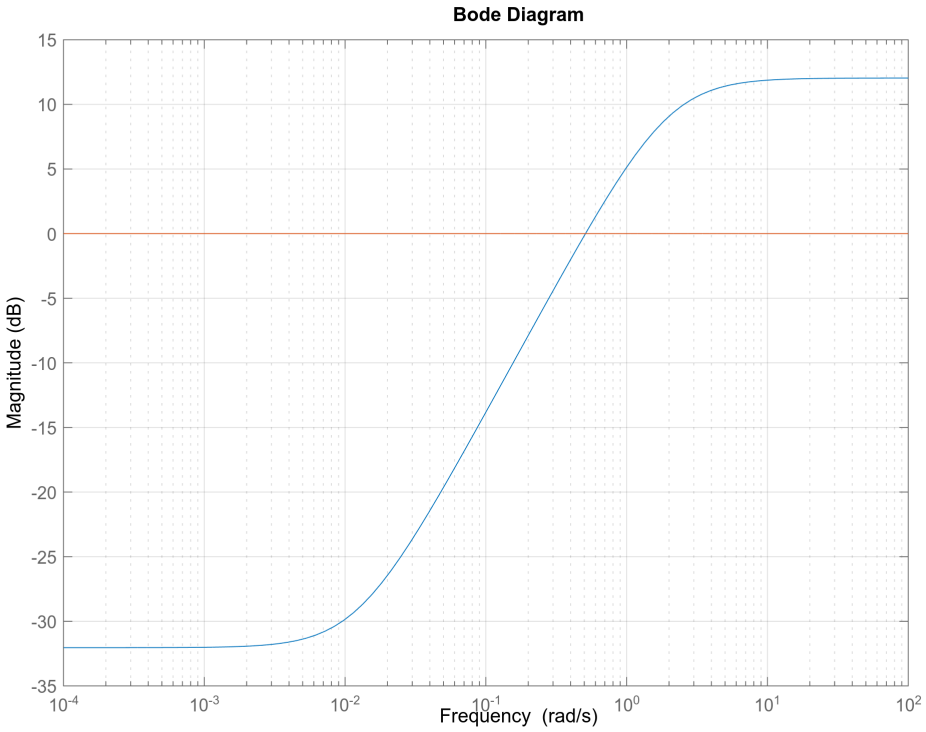


Ponderaciones para prestaciones:

```
W_ref=1; %así el filtro del error tiene unidades de "error de posición"
W_pert=(0.4+0.04*s)/(s/0.7+1); %peso entrada "externa"
bodemag(W_pert), grid on
```



```
anchobandaobjetivo=0.51; %lo optimizaremos por bisección
plantilla_err=@(anchobanda) makeweight(0.025,anchobanda,4);
bodemag(plantilla_err(anchobandaobjetivo), tf(1)), grid on
```




```
W_err= @(bw) minreal(1/plantilla_err(bw));
plantilla_u=5; %límite de saturación de actuador
W_u=1/plantilla_u;
```

Una vez definidos los pesos para cada problema en concreto, este código construye la planta generalizada ponderada (la hacemos función de un parámetro de ancho de banda a probar):

```
W_inputs=blkdiag(W_ref,W_pert);
W_outputs=@(bw) blkdiag(W_err(bw),W_u);
PlantaGenPond=@(bw) blkdiag(W_outputs(bw),1)*PlantaGen*blkdiag(W_inputs,1);
```

Cálculo del control robusto por método mu-síntesis

Ya podemos diseñar el controlador. Con incertidumbre en la planta, lo hacemos con μ-síntesis (mixed-mu para ser menos conservativo con bloques ureal).

```
PGP_musyn=PlantaGenPond(anchobandaobjetivo);
```

```
1 state removed.
```

Regulador state-space genérico

```
tic
```

```
[Kmu,CLperf]=musyn(PGP_musyn,1,1,musynOptions(MixedMu='on')); toc
```

```
DG-K ITERATION SUMMARY:
-----------------------------------------------------------------------
                       Robust performance                 Fit order
-----------------------------------------------------------------------
  Iter         K Step       Peak MU       DG Fit            D      G
    1           2.293        2.266        2.277            2      2
    2           1.035        1.022        1.033           16      4
    3           1.008            1        1.011           18      4
    4           1.007            1        1.007           18      4
    5           1.008        1.001        1.011           18      4

Best achieved robust performance: 1

Elapsed time is 5.083272 seconds.
```

```
CLperf
```

```
CLperf = 1.0000
```

La solución óptima ante incertidumbre estructurada puede tener orden muy alto...

```
size(Kmu)
```

```
State-space model with 1 outputs, 1 inputs, and 27 states.
```

Luego lo reduciremos, mientras podamos mantener prestaciones.

Regulador PID

```
anchobandaobjPID=0.48;
PGP_pid=PlantaGenPond(anchobandaobjPID);
```

```
1 state removed.
```

```
Regu=tunablePID("elPID",'PID');
TunableCL=lft(PGP_pid,Regu);
tic,
[TunedCL,CLperf]=musyn(TunableCL,musynOptions(MixedMu='on'));toc
```

```
DG-K ITERATION SUMMARY:
-----------------------------------------------------------------------
                       Robust performance                 Fit order
-----------------------------------------------------------------------
  Iter         K Step       Peak MU       DG Fit            D      G
    1           2.271        2.187        2.209            8      2
    2           1.022        1.018        1.029           16      4
    3           1.008       0.9974        1.008           16      4
    4           1.008       0.9971        1.008           16      4
    5           1.008       0.9968        1.008           16      4

Best achieved robust performance: 0.997

Elapsed time is 5.124756 seconds.
```

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

```
TunedPID =

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

  with Kp = 0.629, Ki = 0.188, Kd = 1.06, Tf = 0.288

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

Musyn: Reducción de orden

```
K2=balred(Kmu,4);
wcgain(lft(PGP_musyn,K2)) %prestaciones robustas con orden reducido? Bajamos orden mien
```

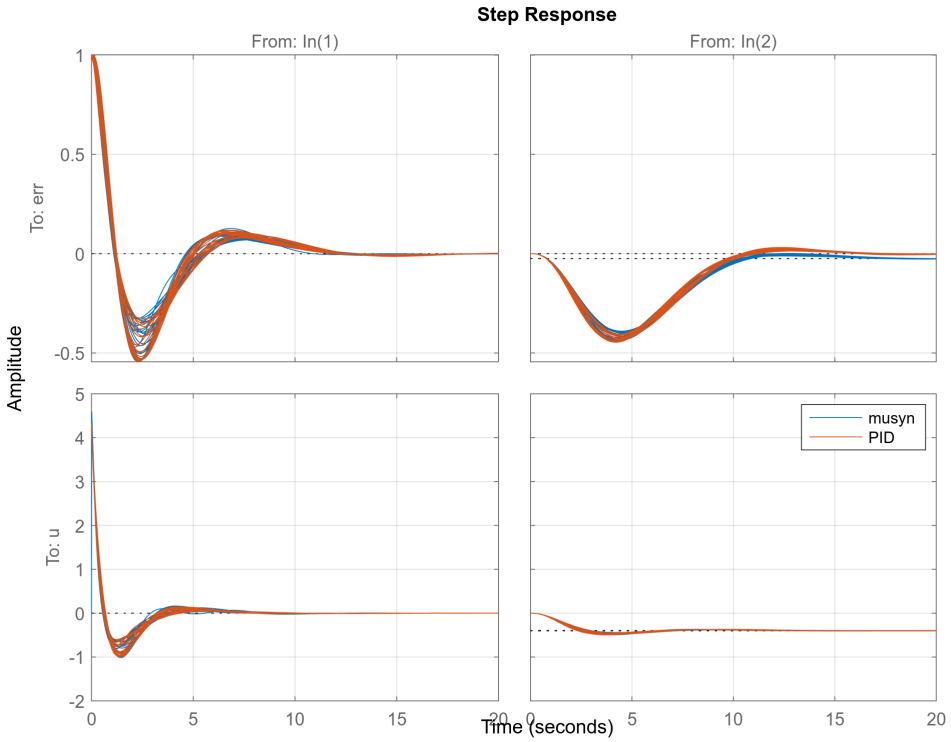
```
ans = struct with fields:
           LowerBound: 1.0061
           UpperBound: 1.0082
    CriticalFrequency: 0.7456
```

```
CL=lft(PlantaGen,K2);
%bodemag(CL*W_inputs,[plantilla_err(anchobandaobjetivo); plantilla_u]*[1 1],logspace(-2
%legend("resultado","plantilla",Location="best")
```

Respuesta temporal en bucle cerrado

Esta es la respuesta conseguida con musyn + red. orden, cuando las entradas tienen la amplitud dada por el "peso". Superponemos el óptimo μ-síntesis y PID, para ver la diferencia.

```
CLPID=lft(PlantaGen,TunedPID);
step(CL*W_inputs,CLPID*W_inputs,20), grid on %resp. temporal... realmente no hemos dise
legend("musyn","PID")
```



Se parecen bastante.

Sección 4

Conclusiones Finales

Conclusiones (teoría)

- Encontrar la cota de error de Δ (*en frecuencia, incluso*) es difícil (probabilístico) en identificación experimental con ruido.
- Esto era state-of-the-art en 1995. Muchos de los métodos en Robust Control Toolbox de Matlab.
  - μ-síntesis (tma. pequ. gan. escalado, multiplicadores) no es convexo: mínimos locales posibles. $\mathcal{H}_\infty$ sí es convexo (LMI/Riccatti): sol. única y eficiente (bueno, no exactamente, con ciertas tolerancias).
- La teoría actual (2000−...) ha avanzado a LPV (con tasas de variación de parámetros), gain-scheduling, consideración específica del retardo (Krasovskii) y sampled-data, IQC, basadas en LMIs y también puede considerar la parte "conocida" como sistema polinomial (Sum of Squares).

Conclusiones (práctica)

Aplicaciones las hay en el ámbito "académico" (papers), en aeroespacial (ESA), en audio (YY-filter)... pero no tantas implantadas en "process control":

- El control predictivo (MPC) es el estandard si no son PIDs... en procesos estables, robust stability se soluciona ponderando más la acción de control en el índice.
- Incluir saturación, fricción seca, da resultados bastante conservadores... pero saturación en actuadores (o hard constraints en estados) no es problema para MPC.
  MPC: hard constraints en tiempo; H-infinito: hard constraints en frecuencia
- La interfaz poco clara con la identificación estadística es un problema: difícil estimar cotas de error basado en datos experimentales.
- Elevado orden de reguladores (excepto optim. fuerza bruta PIDs o `hinfstruct`).
- Poco clara la tolerancia a fallos "abruptos" (cf. multilazo con buena RGA, si uno se avería los otros siguen controlando su parte):
  - robustez *vs.* detección/reconfiguración/adaptación/gain scheduling ante fallos?.

...THE END...

Para quien esté interesado en todo esto:

- Zhou & Doyle, *Essentials of Robust Control*
- Vídeos de los capítulos 18 y 24–...–29 de mi colección en
  http://personales.upv.es/asala/docenciaonline/Cursos/Apuntes.html
- Documentación y ejemplos de la Robust Control Toolbox de Matlab.