Control con 2 grados de libertad (*vs* 1GL): enfoque $\mathscr{H}_\infty$, seguimiento referencias (proceso inestable)

Este código ejecutó correctamente en Matlab **R2022b**

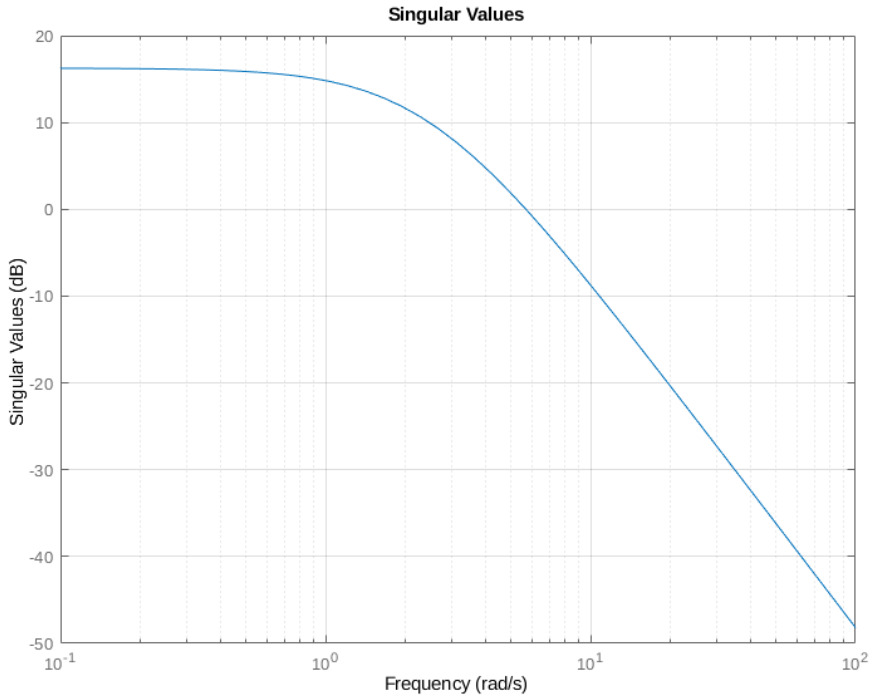
Presentación en vídeo en: http://personales.upv.es/asala/YT/V/hi2gls.html

Objetivo: plantear una planta generalizada (inestable) con control de dos grados de libertad para seguir referencias, y comparar el resultado del control $\mathscr{H}_\infty$ obtenido con el problema de 1 grado de libertad con los mismos objetivos de control.

Tabla de Contenidos

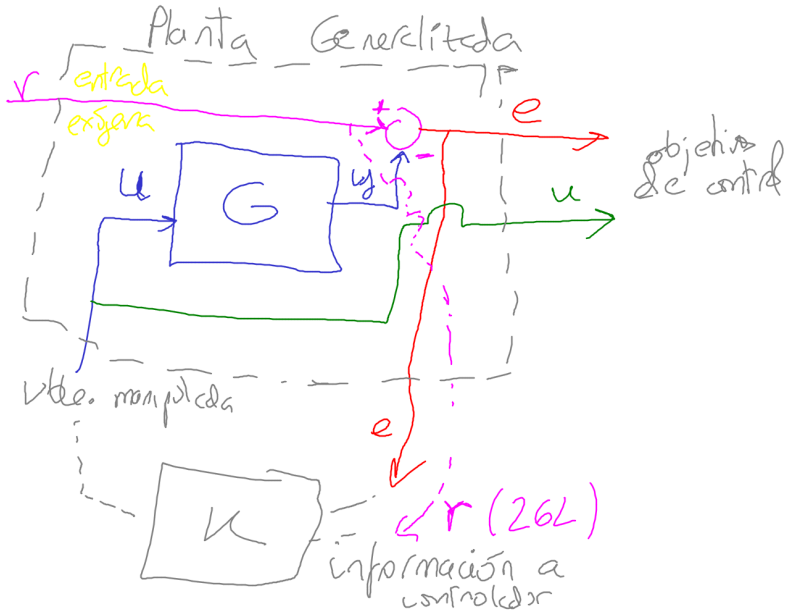
Modelo

```
s=tf('s');
G=6.5*(2*3/(s-2)/(s+3));
%G=6.5*(2*3/(s+2)/(s+3)); %Con esto no hay diferencia en prestaciones, ver nota final
sigma(G), grid on
```



Problema de control a resolver

Plantearemos un problema de seguimiento de referencias:



Se desea minimizar la norma (ponderada) de $e = r - Gu$, y de u, con los adecuados pesos indicando límites de saturación, especificaciones de control, etc.

La información al controlador será:

(a) el propio error e (1 grado de libertad $u = K(s)e$), o

(b) el par (e, r) en el caso de 2 grados de libertad, donde $u = K_{error}(s) \cdot e + K_{ref}(s) \cdot r$.

Planta Generalizada no ponderada que codifica el problema de control

```
PlantaGen=minreal(ss([1 -G;0 1;1 -G;1 0]));
PlantaGen.InputName={'r','u'};
PlantaGen.OutputName={'err','u','err_{med}','r_{copy}'};
size(PlantaGen)
```

```
State-space model with 4 outputs, 2 inputs, and 2 states.
```

```
PlantaGen1GL=PlantaGen(1:3,:);
PlantaGen1GL.InputName'
```

```
ans = 1×2 cell
'r'         'u'
```

```
PlantaGen1GL.OutputName'
```

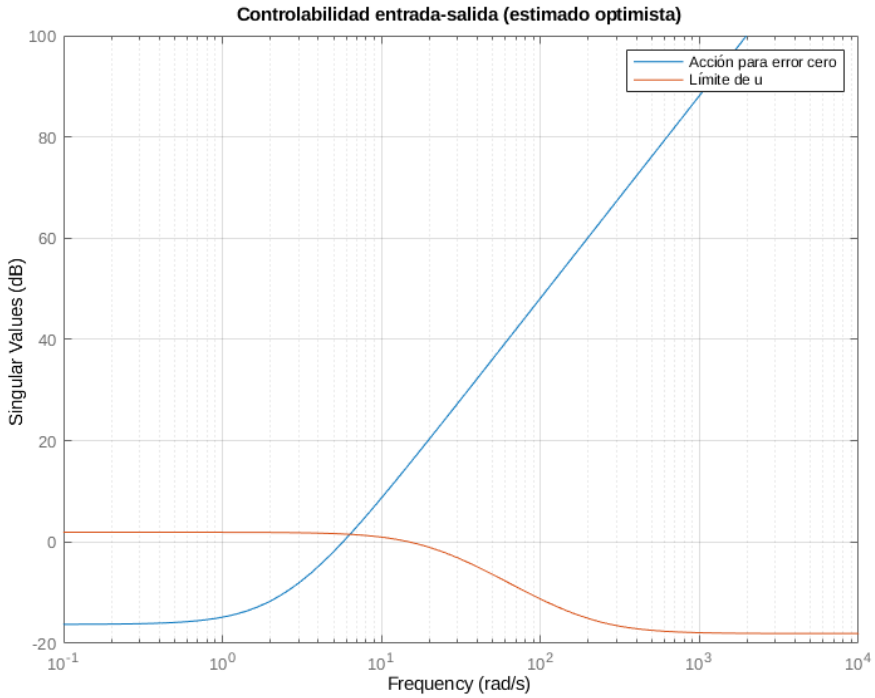
```
ans = 1×3 cell
'err'       'u'         'err_{med}'
```

```
size(PlantaGen1GL)
```

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

Planta Generalizada Ponderada y cálculo de control

```
Amplitud_ref=1; %exógenas
Amplitud_u=1.25; %sat. vbles. manipuladas
anchobanda_actuador=20;
Plantilla_u= ...
Amplitud_u*(1/(anchobanda_actuador*10)*s+1)/(1/anchobanda_actuador*s+1);
sigma(inv(G)*Amplitud_ref,Plantilla_u), grid on,title('Controlabilidad entrada-salida
legend("Acción para error cero","Límite de u")
```



El estimado optimista del tiempo de establecimiento podría ser $t_{e,best} \approx \pi/6 = 0.52$ segundos.

```
Wu=1/Plantilla_u; %los pesos de salidas generalizadas son inversa de límites
W_ref=Amplitud_ref;
PesoEntradaGeneralizada=blkdiag(W_ref,1);
```

Los objetivos de control intentaremos que sean "todo lo rápidos posibles"... cada diseño llegará a una rapidez (ancho de banda) diferente.

El diseño 2GL:

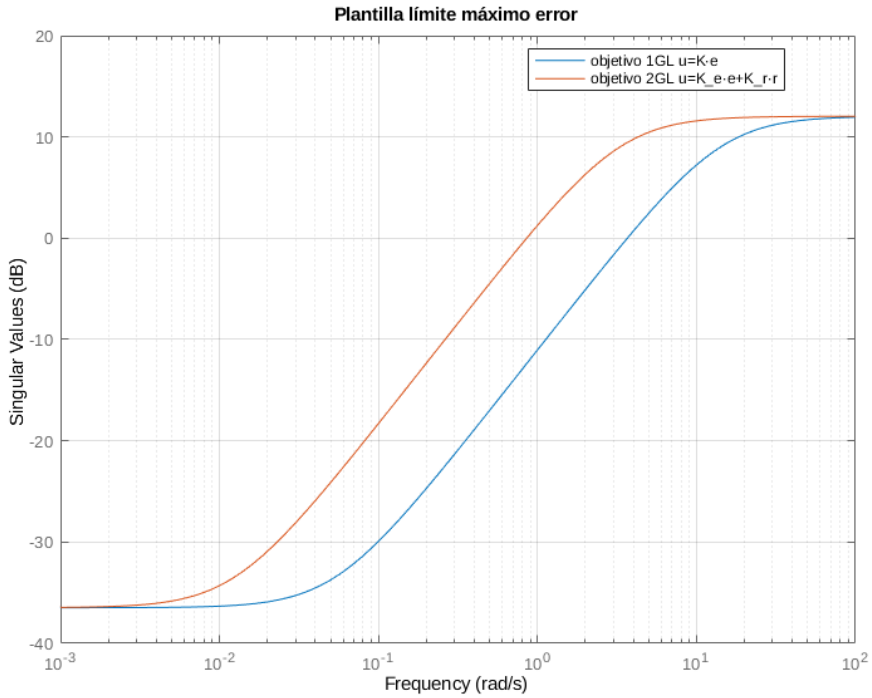
```
anchobandaerror=3.67;
Plantilla_err=makeweight(0.015,anchobandaerror,4);
Werr=1/Plantilla_err; %los pesos de salidas generalizadas son inversa de límites
```

El diseño 1GL:

```
anchobandaerror1GL=0.854;
Plantilla_err1GL=makeweight(0.015,anchobandaerror1GL,4);
Werr1GL=1/Plantilla_err1GL; %los pesos de salidas generalizadas son inversa de límites
```

Dibujemos las plantillas de error:

```
sigma(Plantilla_err,Plantilla_err1GL), grid on, title('Plantilla límite máximo error')
legend("objetivo 1GL u=K·e","objetivo 2GL u=K_e·e+K_r·r","Location","best")
```




```
PesoSalida2GL=blkdiag(Werr,Wu,1,1);
PlantaGenPonderada=minreal(ss(PesoSalida2GL*PlantaGen*PesoEntradaGeneralizada));
```

```
1 state removed.
```

```
PesoSalida1GL=blkdiag(Werr1GL,Wu,1);
PlantaGenPonderada1GL=minreal(ss(PesoSalida1GL*PlantaGen1GL*PesoEntradaGeneralizada));
```

```
1 state removed.
```

```
[K,CL,GAM,~]=hinfsyn(PlantaGenPonderada,2,1,hinfsynOptions('RelTol',1e-3));
GAM
```

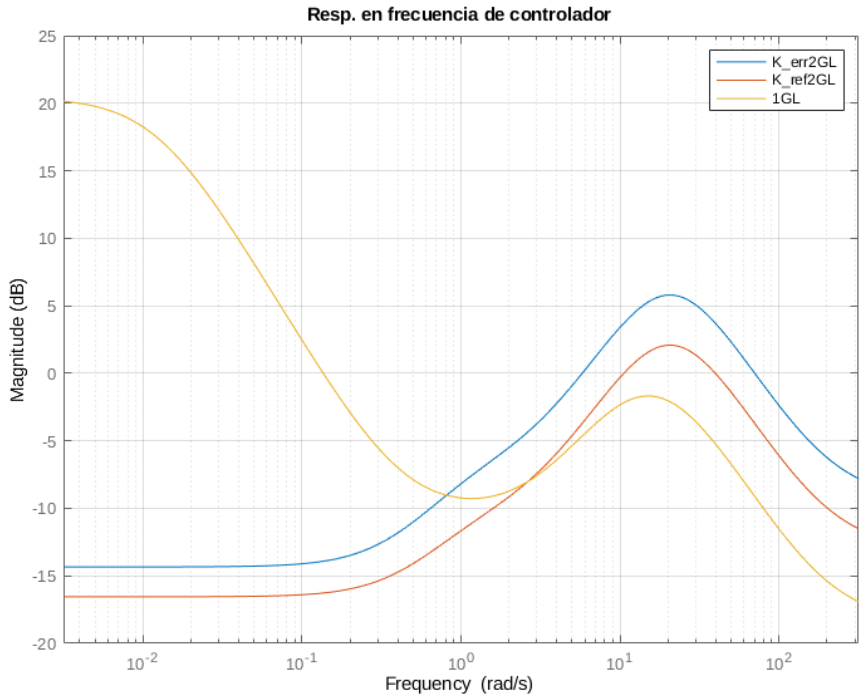
```
GAM = 0.9991
```

```
[K1GL,CL1GL,GAM1GL,~]=hinfsyn(PlantaGenPonderada1GL,1,1,hinfsynOptions('RelTol',1e-3));
GAM1GL
```

```
GAM1GL = 0.9997
```

Evaluación de la solución obtenida

```
bodemag(K(:,1),K(:,2),K1GL,logspace(-2.5,2.5,150)), grid on, legend('K_err2GL','K_ref2G
title("Resp. en frecuencia de controlador")
```




```
size(K)
```

```
State-space model with 1 outputs, 2 inputs, and 4 states.
```

```
size(K1GL)
```

```
State-space model with 1 outputs, 1 inputs, and 4 states.
```

```
CL_NoPesos=minreal(lft(PlantaGen,K));
```

```
2 states removed.
```

```
CL_NoPesos1GL=minreal(lft(PlantaGen1GL,K1GL));
```

```
1 state removed.
```

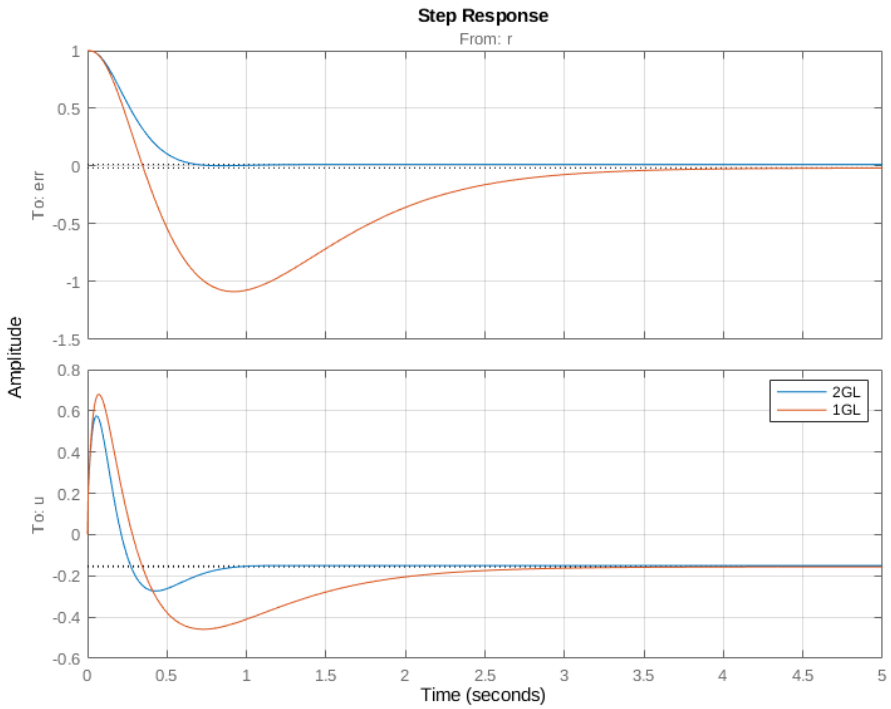
```
CL_NoPesos.InputName'
```

```
ans = 1×1 cell array
    {'r'}
```

```
CL_NoPesos.OutputName'
```

```
ans = 1×2 cell
'err'       'u'
```

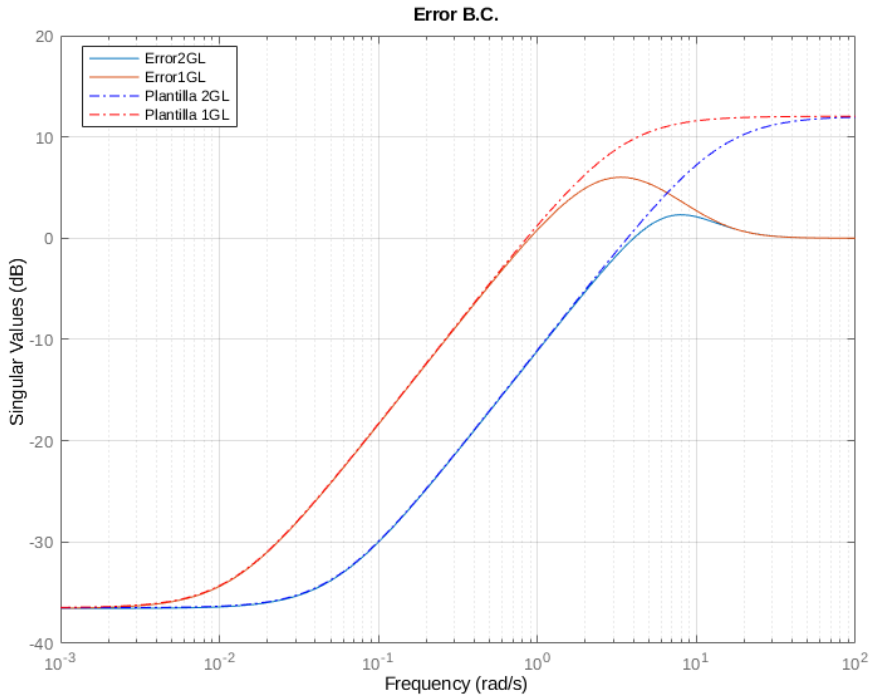
```
step(CL_NoPesos,CL_NoPesos1GL), grid on,legend('2GL','1GL')
```



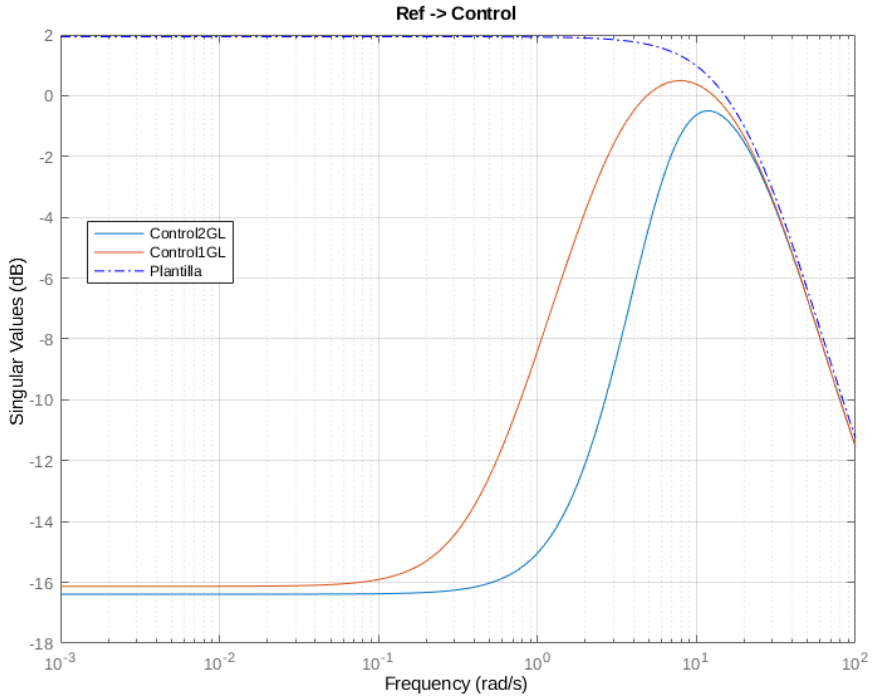
- El diseño 2GL se acerca a nuestro estimado de $t_{e,best}$, consiguiendo unos 0.6 segundos.

- El control 1GL, debido a la inestabilidad del sistema (que complica el problema de control), acaba teniendo un tiempo de establecimiento de 3 segundos.

```
sigma(CL_NoPesos(1,:),CL_NoPesos1GL(1,:),Plantilla_err,'-.',Plantilla_err1GL,'-.r',logs
```




```
sigma(CL_NoPesos(2,:),CL_NoPesos1GL(2,:),Plantilla_u,'-.',logspace(-3,2,150)), grid on,
```

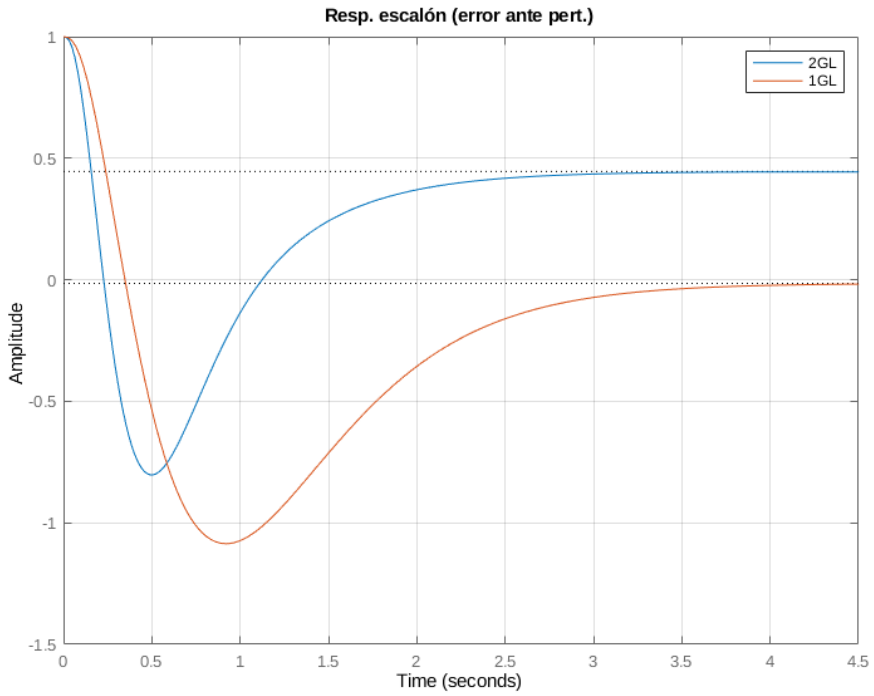


NOTA: optimalidad ante "referencia" del 2GL no implica que se comporte mejor ante "perturbaciones" o ante "errores de modelado".

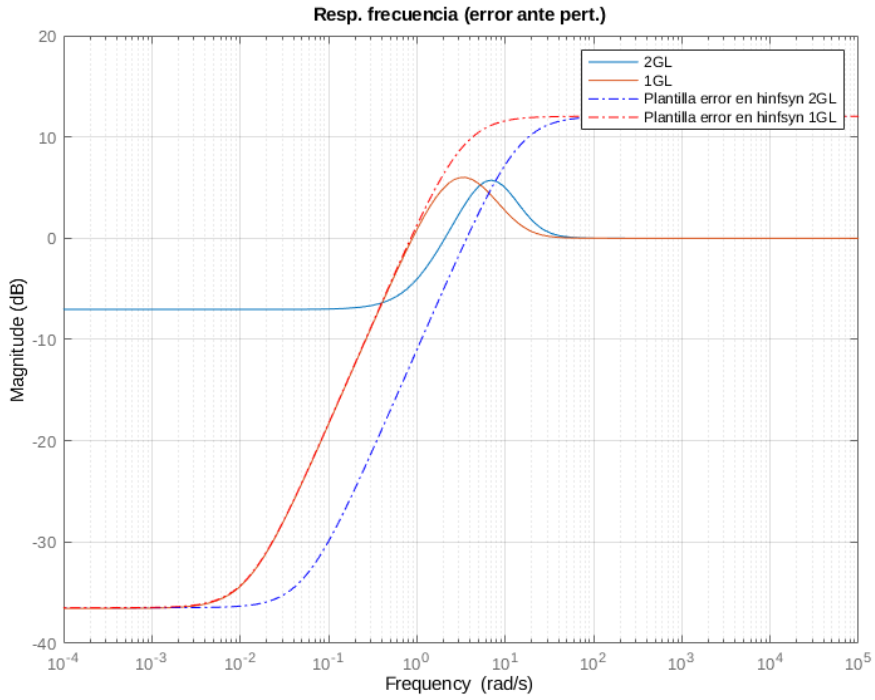
Respuesta ante perturbaciones a la salida

Ante perturbaciones (no referencias), las respuestas de error son:

```
L1=G*K1GL;L2=G*K(:,1);
error_pert1=feedback(1,L1); %esto es 1/(1+G*K1GL)
error_pert2=feedback(1,L2); %esto es 1/(1+G*K_error)
step(error_pert2,error_pert1), grid on, legend("2GL","1GL"), title("Resp. escalón (erro
```




```
bodemag(error_pert2,error_pert1,Plantilla_err,'-.',Plantilla_err1GL,'-.r'), grid on, le
```



En control 1GL respuesta ante referencia es igual que respuesta ante perturbación a la salida. En control 2GL no son iguales, y eso puede ser explotado para "optimizar mejor" la respuesta ante referencia.

En resumen, si existen perturbaciones significativas, deberán ser incorporadas a una planta generalizada con más entradas exógenas (con los adecuados pesos/escalados) para que el optimizador tenga constancia de ellas y de sus amplitudes.

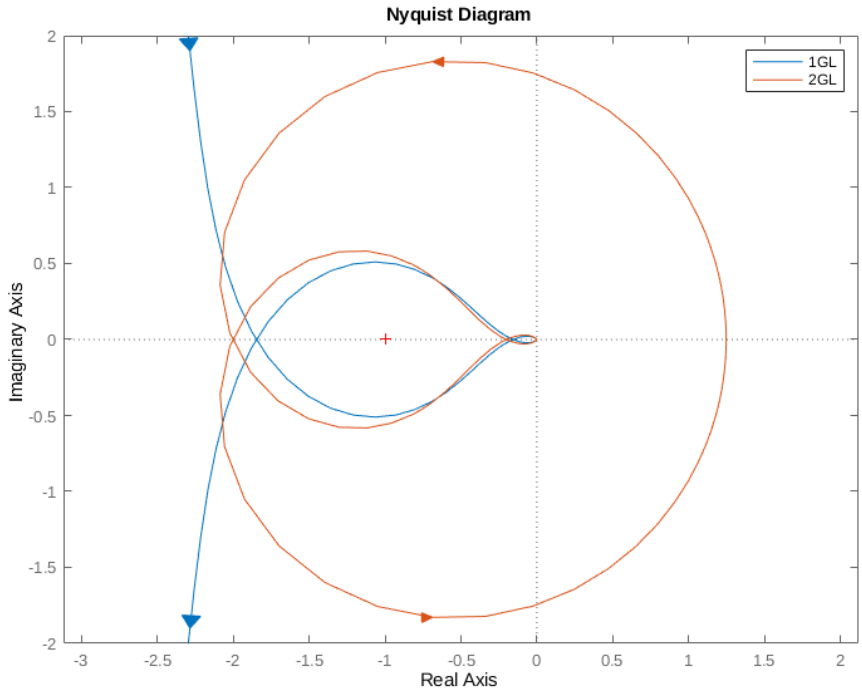
Márgenes de robustez

Maximizar prestaciones ante una cierta entrada no tiene por qué resultar en ser más tolerante a error de modelado. Pero está bien comprobarlo sobre todo si lo queremos "implementar en la práctica": analizar robustez antes de implementar siempre es aconsejable.

Aunque se puede generalizar la metodología $\mathscr{H}_\infty$ a control robusto, aquí no ha sido el caso y se trata sólo de "control óptimo", no "robusto", lo único con cierta relación con robustez que garantizamos es "no saturar".

Por ejemplo, analizando los márgenes clásicos (ganancia, fase, disco) basados en Nyquist, tenemos:

```
nyquist(L1,L2), legend("1GL","2GL")
axis([-2.5 1.5 -2. 2.]), axis equal
```




```
allmargin(L1) %1GL
```

```
ans = struct with fields:
      GainMargin: [0.0147 0.5409 6.3469]
    GMFrequency: [0 1.0846 14.0405]
    PhaseMargin: 28.9633
    PMFrequency: 3.2458
    DelayMargin: 0.1557
    DMFrequency: 3.2458
         Stable: 1
```

```
allmargin(L2) %2GL
```

```
ans = struct with fields:
      GainMargin: [0.4985 4.9468]
    GMFrequency: [2.0619 19.1957]
    PhaseMargin: 30.8373
    PMFrequency: 5.6416
    DelayMargin: 0.0954
    DMFrequency: 5.6416
         Stable: 1
```

```
diskmargin(L1) %tolerancia a multiplicación/división por "disk margin"
```

```
ans = struct with fields:
           GainMargin: [0.6153 1.6252]
          PhaseMargin: [-26.7926 26.7926]
           DiskMargin: 0.4763
           LowerBound: 0.4763
           UpperBound: 0.4763
            Frequency: 2.1908
    WorstPerturbation: [1×1 ss]
```

```
diskmargin(L2) %tolerancia a multiplicación/división por "disk margin"
```

```
ans = struct with fields:
           GainMargin: [0.5748 1.7399]
          PhaseMargin: [-30.2233 30.2233]
           DiskMargin: 0.5401
           LowerBound: 0.5401
           UpperBound: 0.5401
            Frequency: 4.5751
    WorstPerturbation: [1×1 ss]
```

Al no estar en los criterios de diseño, en principio no está "garantizado" que la robustez de 1GL sea mayor o menor que 2GL (también las prestaciones son diferentes).

Conclusiones

Los diseños 2GL propuestos para el procesos inestable en consideración consiguen:

- mejores prestaciones (4 veces más ancho de banda de bucle cerrado) ante cambio de referencia
- peores prestaciones ante perturbaciones a la salida
- similar robustez (tolerancia a error de modelado en términos de márgenes de fase, ganancia, disco... bueno, un poco peor el 2GL en margen de retardo) que el caso 1GL (control basado en error).

Básicamente, el optimizador optimiza "lo que le indique la planta generalizada ponderada" y el "óptimo para una cosa" no tiene por qué serlo "para otra cosa". Problemas de perturbaciones, o de perturbaciones y referencias simultáneamente, o que incorporen error de modelado significativo, deberán ser abordados con modificaciones a la planta generalizada y a sus pesos.

Casos estables?

NOTA: si el proceso fuera "estable" y "fase mínima", las prestaciones de seguimiento de referencia serían IDÉNTICAS para 1GL, 2GL (o incluso para bucle abierto, **`PlantaGenBA=PlantaGen([1 2 4],:)`**). Por eso se ha puesto el ejemplo "inestable". Si no hay "perturbaciones entre controlador y planta", en caso estable y fase mínima se puede "cancelar" lo que se quiera, y se puede conseguir cualquier función de referencia arbitraria estable M(con igual o más grado relativo que G) entre referencia y salida tanto en bucle abierto $u = Qr$, con $Q = G^{-1}M$ como en bucle cerrado (inspirado en control por modelo interno IMC, $u = Ke$, $K = Q/(1 - GQ)$) como en cualquier "mezcla" de ambos. Este ejemplo no tendría sentido con una planta estable y fase mínima sin perturbaciones... Incluso, con ruido de medida en sensor, lo óptimo sería "bucle abierto" (que no tiene ruido).