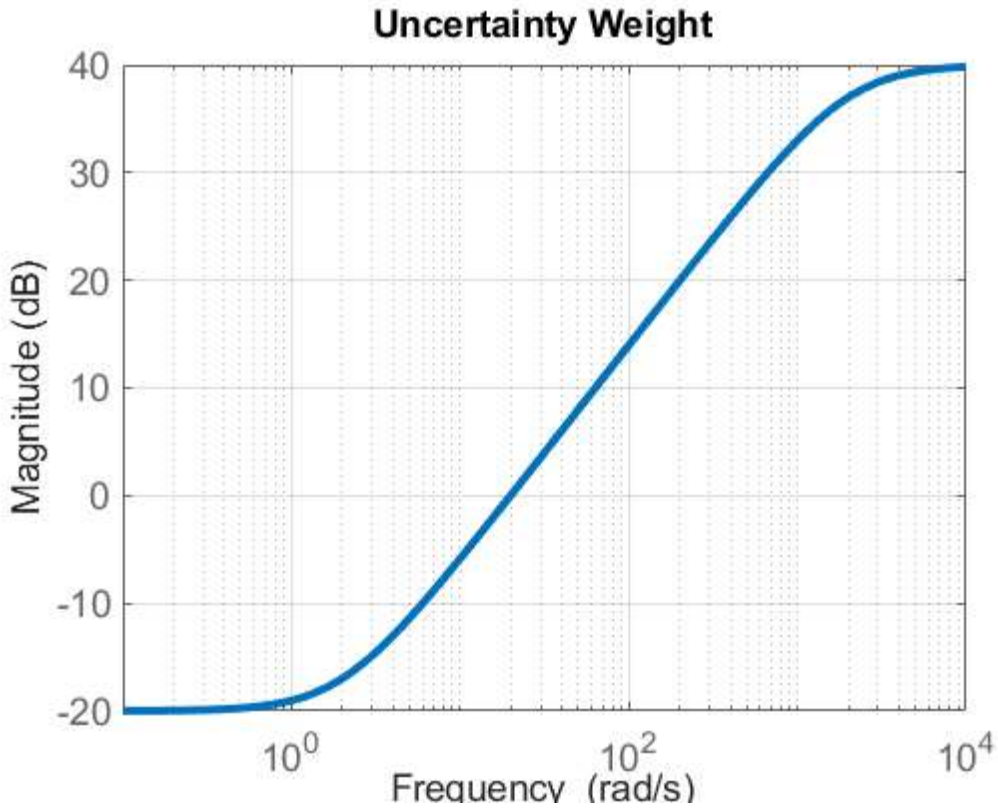
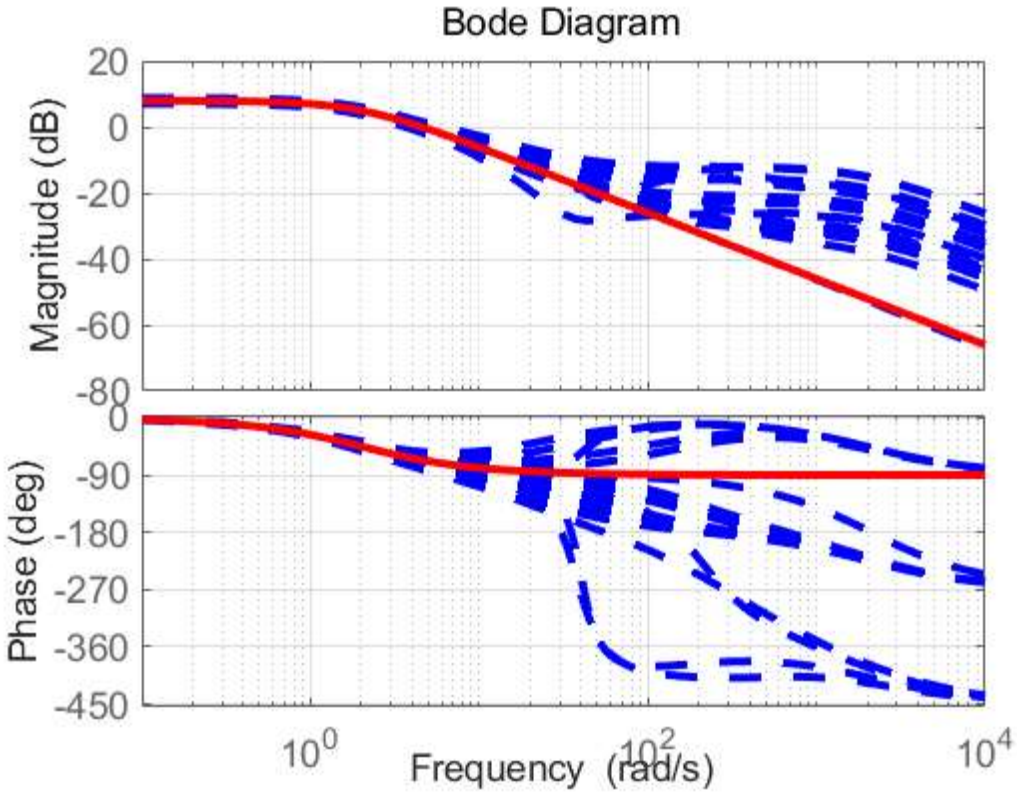
SISO Small Gain Example

Contents

Plant Data

```
% Nominal Plant
G = tf(5,[1 2]);

% Uncertainty Weight
AU = 0.1;
MU = 100;
wU = 2;
WU = tf([1 wU],[1/MU wU/AU]);

figure(1)
bodemag(WU, {0.1, 1e4});
title('Uncertainty Weight')
grid on;
if exist('garyfyFigure','file'), garyfyFigure; end

% Uncertain system
% ULTIDYN creates a Matlab uncertain LTI system object.
Delta = ultidyn('Delta',[1 1]);
Gtil = G*(1+WU*Delta);

% Random samples of uncertainty
rng(0);   % Set seed for repeatable results
bopt = bodeoptions;
bopt.PhaseMatching = 'on';

figure(2)
bodeplot(Gtil,'b--',G,'r',{0.1, 1e4}, bopt);
grid on;
if exist('garyfyFigure','file'), garyfyFigure; end
```



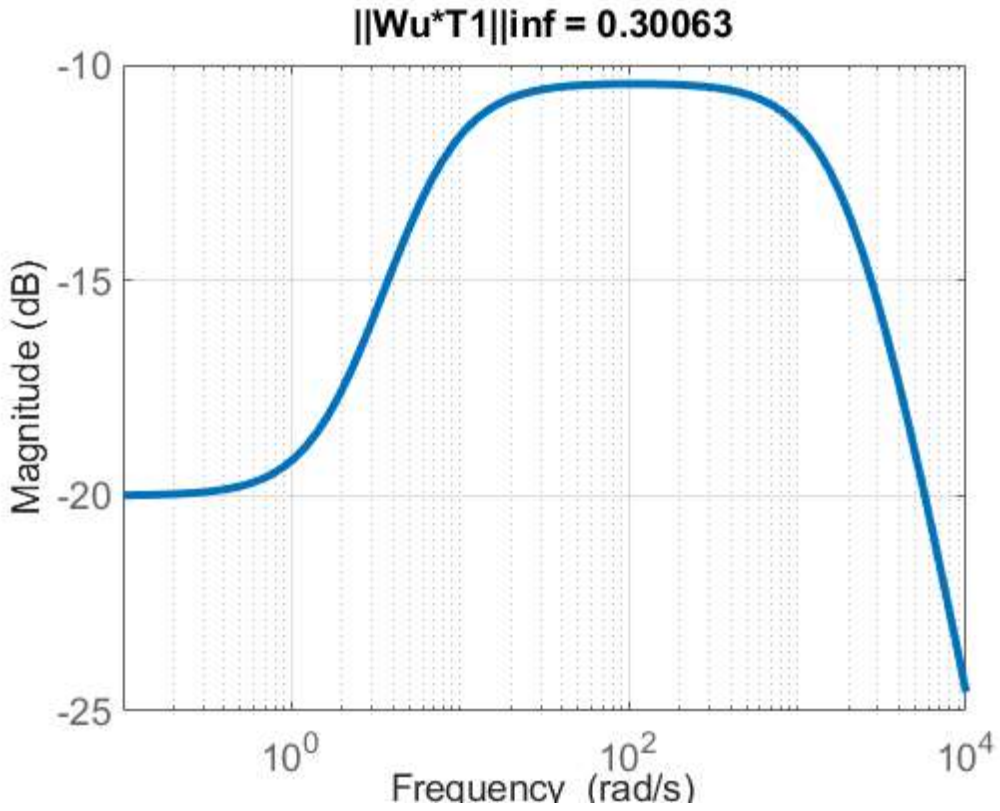


PI Controllers: Slow response / low loop bandwidth

```
% Desired loop bandwidth, rad/s
w1 = 6;
```

```
% Design PI Controller via loopshaping
wL = w1;
kg = 1/abs(evalfr(G,1j*wL));
bb = sqrt(10);
Kb = tf([bb wL],[1 0])/sqrt(bb^2+1);
K1 = kg*Kb;

% Check nominal stability
L1 = G*K1;
T1 = feedback(G*K1,1);
isstable(T1)

% Classical margins: Phase margin is relatively large
AM1 = allmargin(L1)

% Check robust stability condition
% Norm is < 1 so the feedback system is robustly stable
n1 = norm(WU*T1,inf);

figure(3)
bodemag(WU*T1,{0.1, 1e4});
title(['||Wu*T1||inf = ' num2str(n1)]);
grid on;
if exist('garyfyFigure','file'), garyfyFigure; end
```

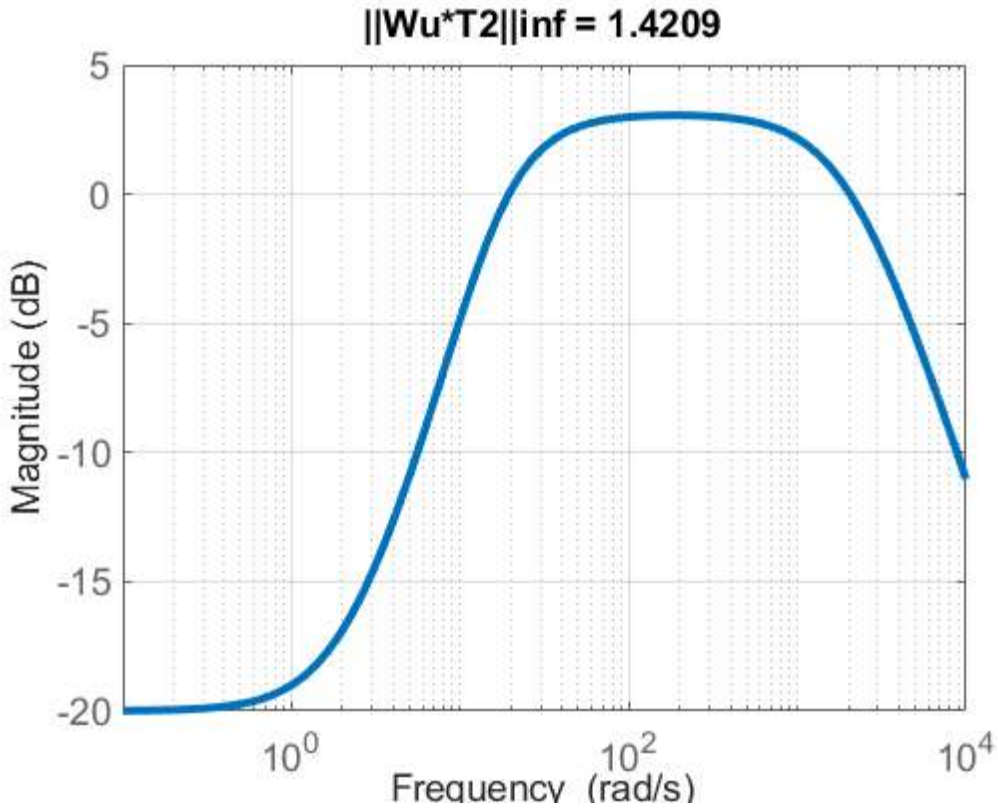
```
ans =
  logical
   1
AM1 =
  struct with fields:

     GainMargin: [1×0 double]
    GMFrequency: [1×0 double]
    PhaseMargin: 90.8866
    PMFrequency: 6.0000
    DelayMargin: 0.2644
    DMFrequency: 6.0000
         Stable: 1
```



PI Controllers: Slow response / low loop bandwidth

```
% Desired loop bandwidth, rad/s
w2 = 30;

% Design PI Controller via loopshaping
wL = w2;
kg = 1/abs(evalfr(G,1j*wL));
bb = sqrt(10);
Kb = tf([bb wL],[1 0])/sqrt(bb^2+1);
K2 = kg*Kb;

% Check nominal stability
L2 = G*K2;
T2 = feedback(G*K2,1);
isstable(T2)

% Classical margins: Phase margin is relatively large
AM2 = allmargin(L2)

% Check robust stability condition
% Norm is > 1 so the feedback system is *not* robustly stable.
% w2 is the frequency where WU*T2 achieves its peak gain.
[n2,w2] = norm(WU*T2,inf);

figure(4)
bodemag(WU*T2,{0.1, 1e4});
title(['||Wu*T2||inf = ' num2str(n2)]);
grid on;
if exist('garyfyFigure','file'), garyfyFigure; end
```

```
ans =
  logical
   1
AM2 =
  struct with fields:

     GainMargin: [1×0 double]
    GMFrequency: [1×0 double]
    PhaseMargin: 76.2657
    PMFrequency: 30.0000
    DelayMargin: 0.0444
    DMFrequency: 30.0000
         Stable: 1
```

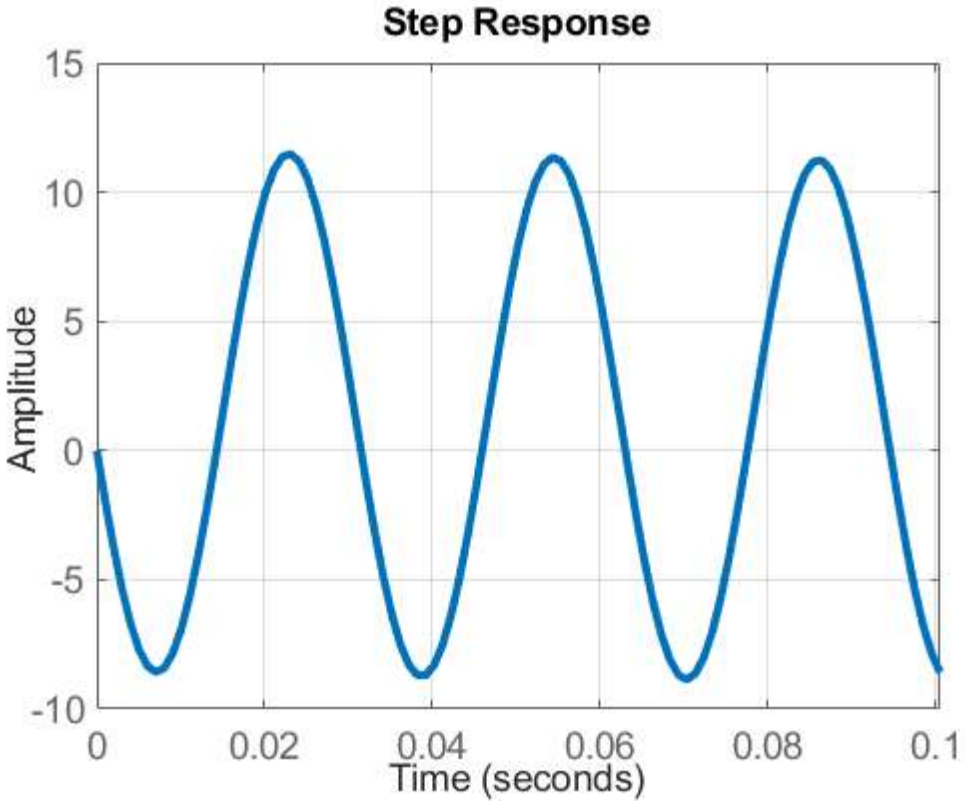


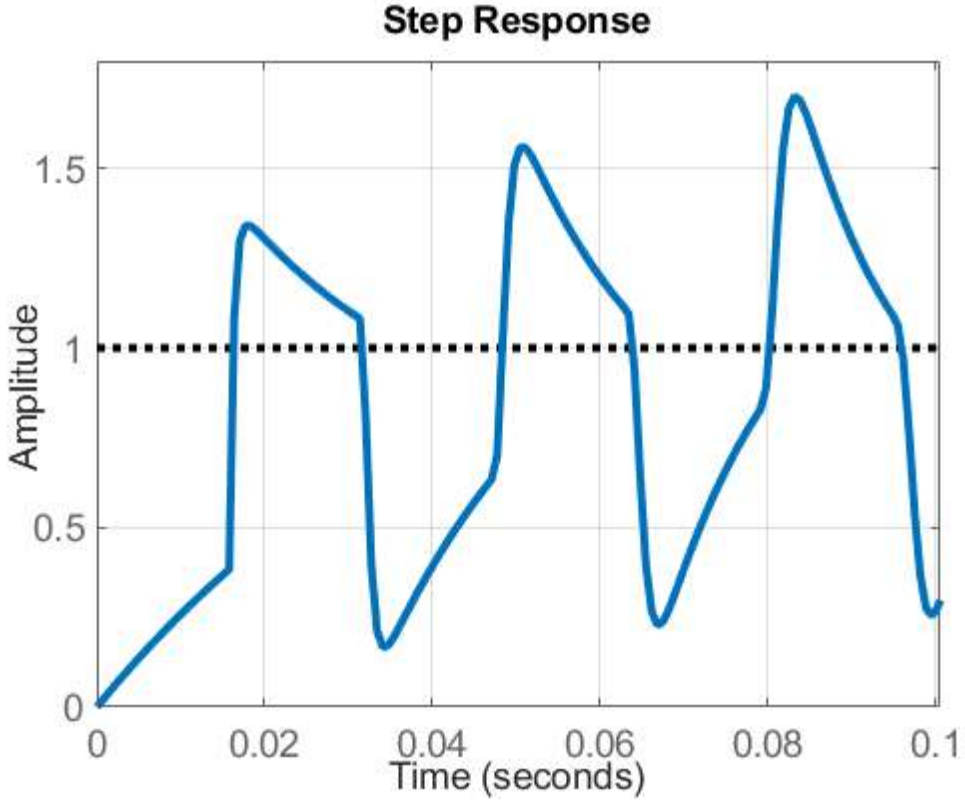
Construct Destabilizing Uncertainty

```
% Complex value that causes uncertainty
delta2 = 1/evalfr(-WU*T2,1j*w2);

% A) Destabilizing uncertainty using first-order fit
% Verify instability by checking closed-loop poles and via a step response
% The closed-loop will have poles on the imaginary axis at the frequency
% w2.  The step response will oscillate at this destabilizing frequency.
DeltaA = cnum2sys(delta2,w2);
GA = G*(1+WU*DeltaA);
TA = feedback(GA*K2,1);
damp(TA)

figure(5)
Tfinal = 20/w2;  % 20 cycles at the destabilizing frequency
```

```
step(TA,Tfinal)
grid on;
if exist('garyfyFigure','file'), garyfyFigure; end

% B) Destabilizing uncertainty using gain + delay
% Verify instability by checking the step response. We can't check poles
% due to the time delay. The step response will oscillate at the
% destabilizing frequency w2.
DeltaB = ss( abs(delta2) );
DeltaB.InputDelay = -angle(delta2)/w2;
GB = G*(1+WU*DeltaB);
TB = feedback(GB*K2,1);

figure(6)
step(TB,Tfinal)
grid on;
if exist('garyfyFigure','file'), garyfyFigure; end
```

Pole	Damping	Frequency (rad/seconds)	Time Constant (seconds)
-3.24e-01	1.00e+00	3.24e-01	3.09e+00
-1.31e+01	1.00e+00	1.31e+01	7.65e-02
6.57e-13 + 1.99e+02i	-3.30e-15	1.99e+02	-1.52e+12
6.57e-13 - 1.99e+02i	-3.30e-15	1.99e+02	-1.52e+12





Published with MATLAB® R2024a