Flexible Rocket:

MUSYNID

Contents

This design example considers the control of a rocket in the ascent phase. The pitch motion of the rocket is unstable and is stabilized using thrust vectoring. One issue is that modern rockets have lower frequency elastic modes and hence these flexible modes have a significant effect on the dynamics. In this example, a variety of robust control methods are used to design control laws for the rocket pitch dynamics, namely

- (LP) classical loopshaping
- (GM) Glover-McFarlane loopshaping (`cprop2cact`)
- (MS) Mixed-Sensitivity (`mixsyn`)
- (mu) DK-iteration, with a Mixed-Sensitivity performance criterion

Advantages and disadvantages of the approaches are highlighted.

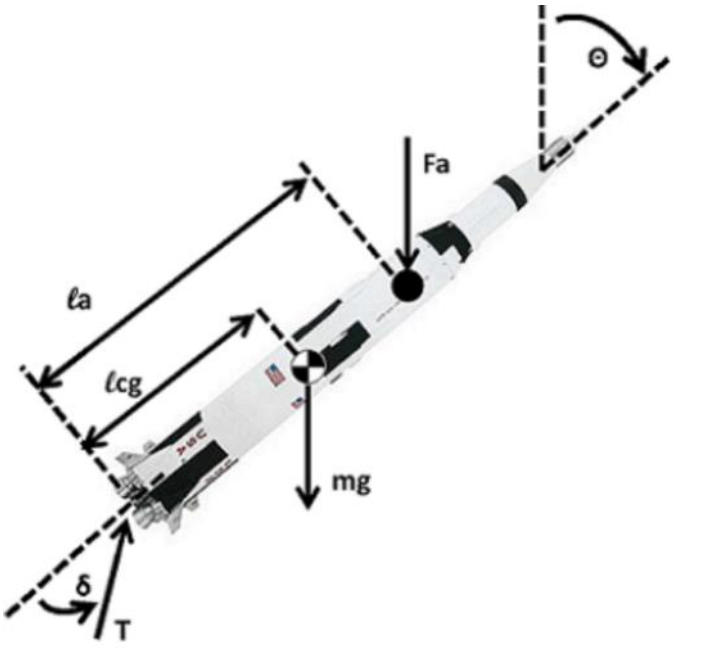


Figure: Flexible Rocket.

Reference: D.F. Enns, "Rocket stabilization as a structured singular value synthesis design example," IEEE Control Systems Magazine, June 1991, p. 67-73.

High Order Rocket Model

G10 is a 10-state model. It has 2 states associated with the rigid-body short-period motion. The remaining 8 states correspond to 4 elastic modes. This linearized model is obtained at the maximum dynamic pressure condition in the ascent. The input is a thrust vectoring control, u_tvc (units are 10^9*lbs), accomplished via gimbaled engines at the base of the rocket. The output is a pitch rate measurement, y_gyro (rad/sec), from a gyro at the rocket center.

```
load G10ss;
G10 = ss(A10,B10,C10,D10);
G10.InputName = {'u_tvc'};
G10.OutputName = {'y_gyro'};
```

The rocket has unstable short-period dynamics with a pole at s=+1.00. In addition, the lowest elastic mode has a natural frequency of 6.13 rad/sec. The control is difficult due to the closeness of the unstable mode and first flex mode.

```
damp(G10)
```

```
       Eigenvalue          Damping     Frequency

  1.00e+00                -1.00e+00     1.00e+00
 -1.09e+00                 1.00e+00     1.09e+00
 -6.15e-02 + 6.13e+00i     1.00e-02     6.13e+00
 -6.15e-02 - 6.13e+00i     1.00e-02     6.13e+00
 -1.68e-01 + 1.68e+01i     1.00e-02     1.68e+01
 -1.68e-01 - 1.68e+01i     1.00e-02     1.68e+01
 -3.67e-01 + 3.67e+01i     9.99e-03     3.67e+01
 -3.67e-01 - 3.67e+01i     9.99e-03     3.67e+01
 -5.20e-01 + 5.21e+01i     9.99e-03     5.21e+01
 -5.20e-01 - 5.21e+01i     9.99e-03     5.21e+01

(Frequencies expressed in rad/seconds)
```

Reduced-Order model via modal truncation

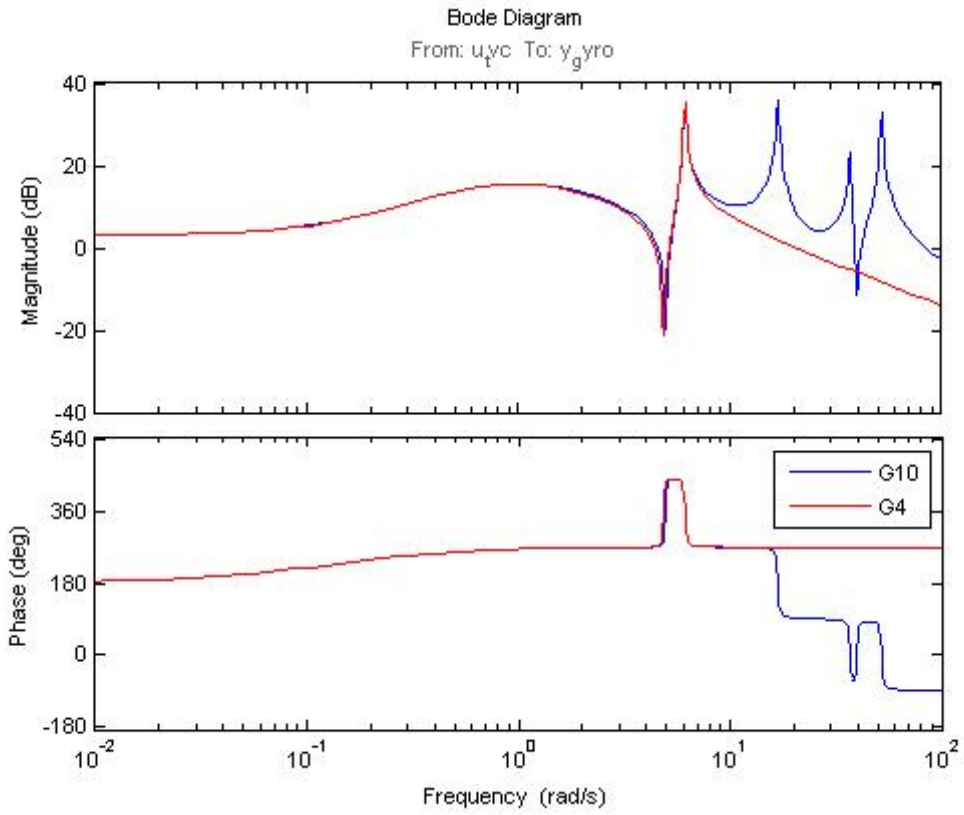
It is useful to create a reduced order (lower state) model for the control design. Specifically, a 4-state model G4 is obtained by truncating the three highest frequency flexible modes. G4 retains the rigid-body, short period dynamics and the lowest frequency flexible mode.

```
eidx = [2:4 7:9];
G4 = modred(G10,eidx);
```

The Bode plot below compares the high-order (G10) and the reduced-order (G4) models.

```
figure
Nw = 500;
wRange = {0.01 100};
```

```
opts = bodeoptions;
opts.PhaseMatching ='on';
opts.PhaseMatchingValue = 180;
bodeplot(G10,'b',G4,'r',wRange,opts);
legend('G10','G4');
```



Design 1 (LP): Classical Loopshaping

The first controller `Kcl` will be a classical loop-shaping design. The controller consists of a PI law (integralboost) for low-frequency tracking and a second-order Butterworth filter (rolloff) for high-frequency roll-off. The proportional and integral gains are chosen to:

- Place the gain crossover frequency of the loop transfer function at wgc=2rad/sec. This cross-over is roughly a factor of three below the first flexible mode.
- Place the corner frequency of the PI law at 1 rad/sec. In other words, the integral characteristics of the controller dominate below 1 rad/sec which is roughly a factor of 6 below the first flexible mode.

A second-order Butterworth filter with a 10 rad/sec corner frequency is included in the controller to ensure gain roll-off before the higher frequency flexible modes.

```
wBoost = 1;
wRO = 10;
kIB = integralboost(wBoost);
kRO = rolloff(wRO,2);
wgc = 2;
kGain = 1/getmag(G10*kIB*kRO,wgc);
kLP = kGain*kIB*kRO;
```

Assess Loopshaping Controller

The controller designed via loopshaping stabilizes the plant although the gain and phase margins could be improved via further tuning.

```
assessL(G10*kLP)
```

```
ans =

     GainMargin: [1.1630e-17 0.4536 1.9552 5.8208 266.5266 3.3281]
    GMFrequency: [0 0.3690 9.2906 36.9295 39.6691 52.1893]
    PhaseMargin: [45.3750 -160.9226 19.2928 -49.9377 153.7560]
    PMFrequency: [2.0000 5.6909 7.2164 16.0451 17.3939]
    DelayMargin: [0.3960 0.6105 0.0467 0.3373 0.1543]
    DMFrequency: [2.0000 5.6909 7.2164 16.0451 17.3939]
         Stable: 1
            bwS: 0.2635
            bwT: 16.8894
        bwRatio: 64.1067
          PeakS: 3.3531
          PeakT: 3.0112
```

Design 2 (GM): Improve robustness margins with CPROP2CACT

Try `cprop2cact` to improve the margins, at some expense in gain

```
[Gamma,kGM] = cprop2cact(G10,kLP);
```

Gamma is relatively small, so the procedure should show some improvement.

```
Gamma
assessL(G10*kGM)
```

```
Gamma =

    2.5219


ans =

     GainMargin: [4.8309e-15 0.3994 2.8207 2.5985 4.0795 177.9848 3.7178]
    GMFrequency: [0 0.2298 5.7371 11.7635 36.7826 39.8544 51.8193]
    PhaseMargin: [1x7 double]
    PMFrequency: [2.2486 5.9989 6.5902 16.5167 16.7971 17.6818 19.0264]
    DelayMargin: [0.3657 0.6987 0.1507 0.3115 0.2465 0.2982 0.1545]
    DMFrequency: [2.2486 5.9989 6.5902 16.5167 16.7971 17.6818 19.0264]
         Stable: 1
            bwS: 0.3774
            bwT: 18.3498
        bwRatio: 48.6232
          PeakS: 1.6612
          PeakT: 1.6857
```

Design 3 (MS): H-infinity Mixed-Sensitivity Design

Next, the H-infinity mixed-sensitivity method will be used to design a control law. The design interconnection is shown in the figure below. The main performance objective is to minimize the weighted tracking error. The performance weight `W1 = Wperf` is chosen to specify a bound on the output-sensitivity function S, i.e. reference to tracking error. In particular, if the final control design satisfies `|Wperf*S|_infty` <1 then S< 1/|Wperf| at each frequency. The performance weight is chosen to achieve low-frequency tracking error less than 0.01 with S<1 up to frequencies of 0.2 rad/sec. Moreover, `Wperf` enforces a high frequency gain bound of S<2. This provides good classical margins by ensuring that the Nyquist plot of the loop `G*K` does not come close to -1.

The control weight `W2 = Wu` is chosen as a fourth-order high pass filter with a low frequency gain of 0.01, unity gain crossover at 5 rad/sec, and high frequency gain of 10^4. This penalizes the control effort at high frequencies. The fourth-order weight forces the controller to roll-off rapidly to avoid exciting the high frequency flexible modes.

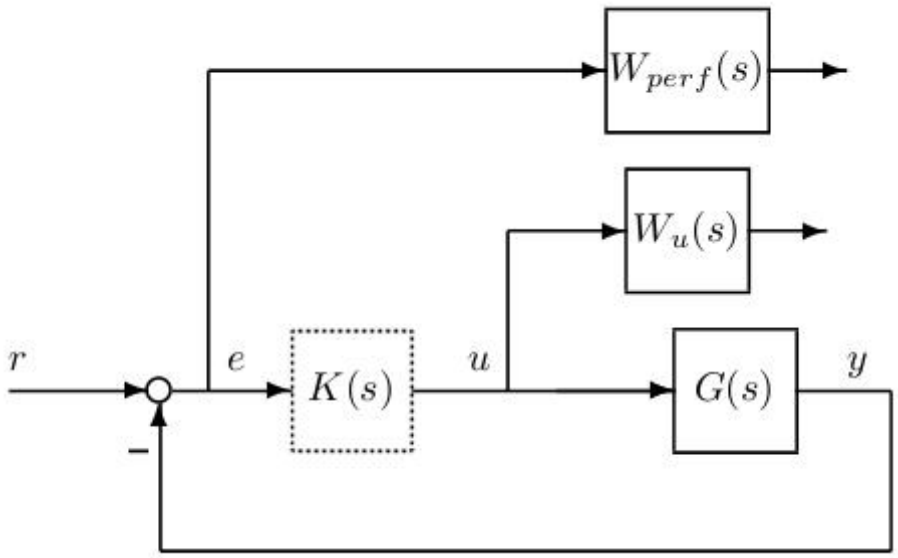
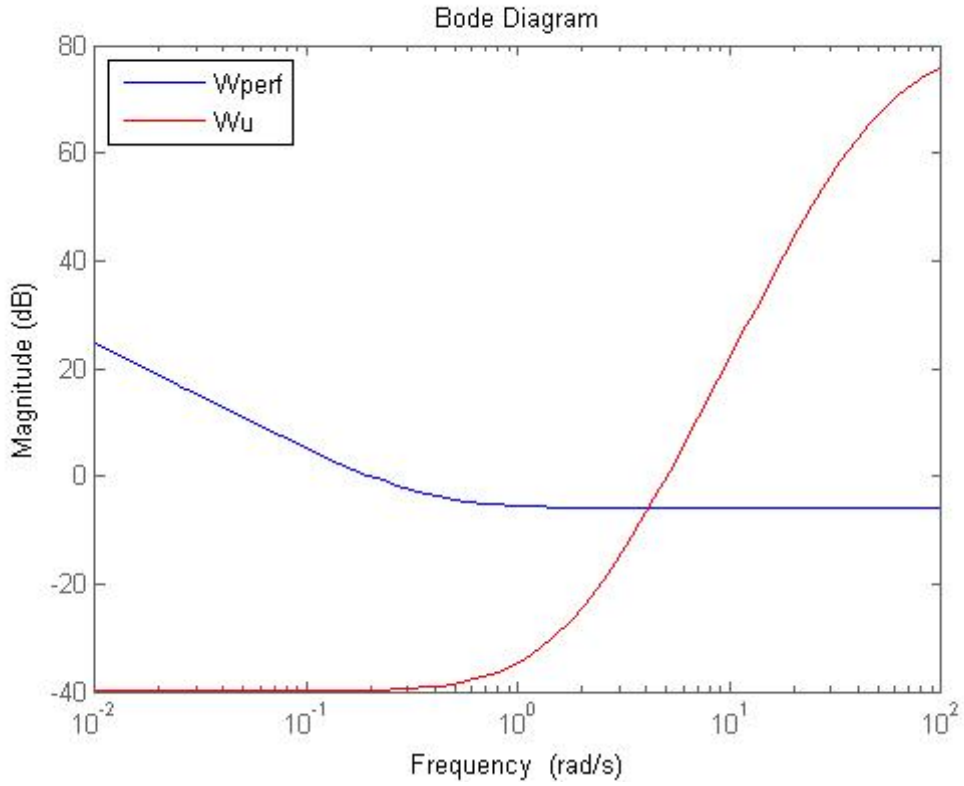


Figure: Control Design Interconnection

```
% Performance Weight
DCgain = 100;
Crossover = 0.2;
HFgain = 1/2;
Wperf = makeweight(DCgain,Crossover,HFgain);

% Control penalty
DCgain = (0.01)^(1/4);
Crossover = 5;
HFgain = 10;
Wu = makeweight(DCgain,Crossover,HFgain);
Wu = Wu^4;

% Figure of weights for performance and control penalties
figure;
bodemag(Wperf,'b',Wu,'r',wRange);
legend('Wperf','Wu','location','NorthWest');
```



The mixed sensitivity control design can be performed with the `mixsyn` function. This function forms the system interconnection diagram shown above (with the weights) and solves for the H-infinity optimal controller using `hinfsyn`. The `mixsyn` function also allows a weight on the output-complementary sensitivity function to be specified. This weight is not used in the current design and is set to empty. The four-state reduced-order model `G4` is used in the design. The optimal gain satisfies `GAMms<1` which implies that the design objectives are satisfied.

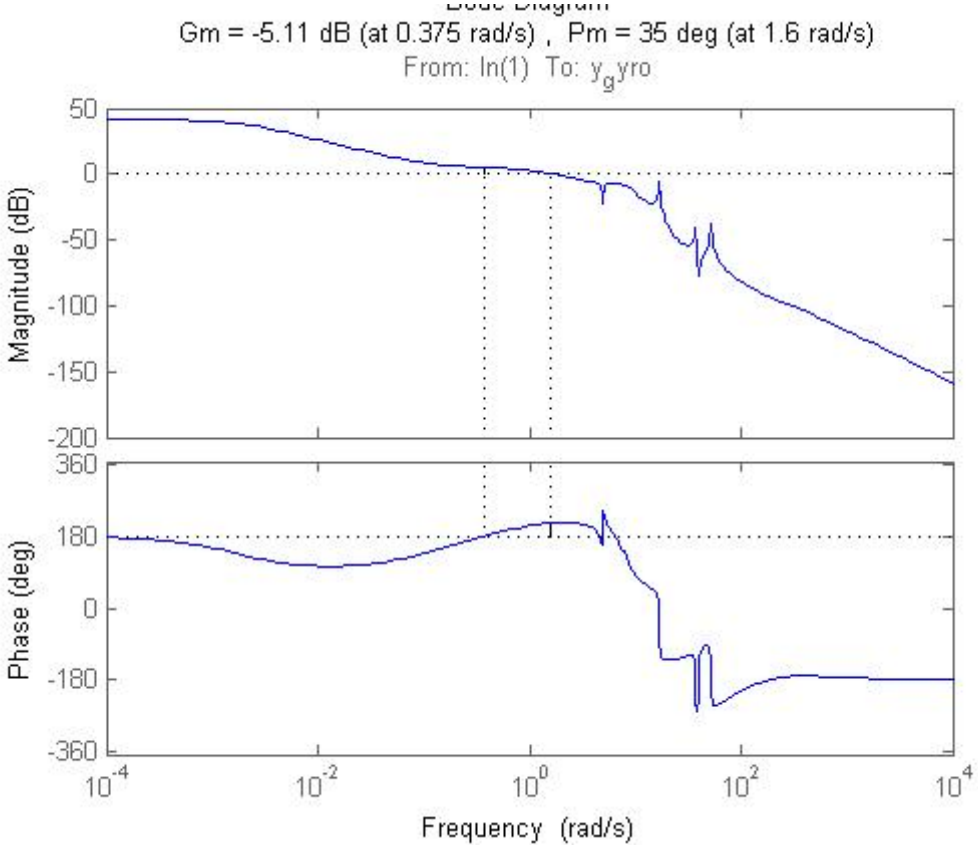
```
WT = [];
[kMS,clMS,gammaMS,infoMS] = mixsyn(G4,Wperf,Wu,WT);
gammaMS
```

```
gammaMS =

    0.8586
```

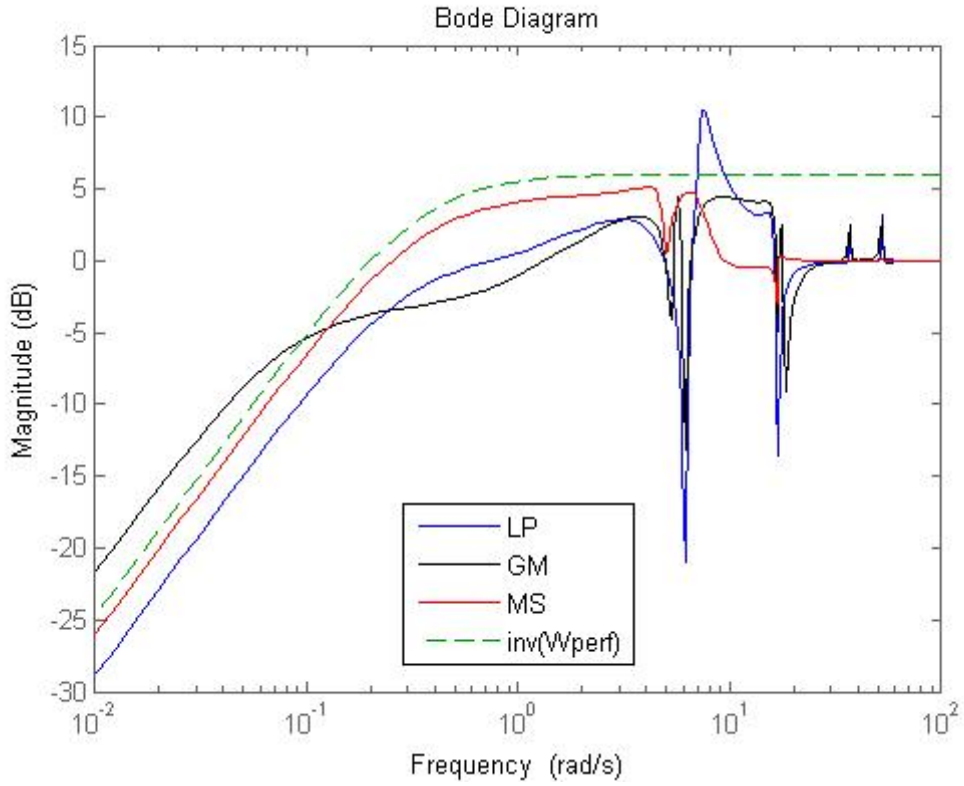
The controller designed using the mixed sensitivity approach has improved gain and phase margins as compared to the loopshaping design. Again, the margins could be improved by further tuning.

```
figure
margin(G10*kMS)
```



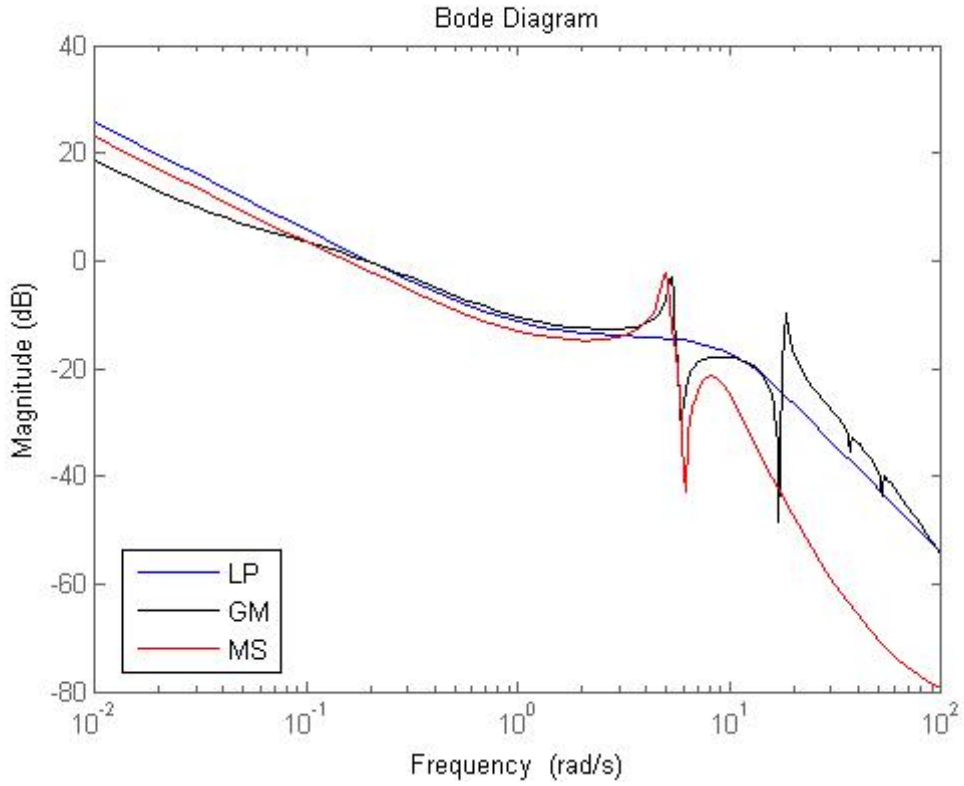
The next figure compares the closed-loop sensitivity transfer function for both control designs. The sensitivity functions are calculated using the full 10-state model. The figure also shows the inverse of the performance weight. `Sms < inv(Wperf)` at each frequency because the optimal gain `GAMms<1`. `Kms` achieves a lower peak sensitivity and slightly higher bandwidth as compared to the loopshaping design.

```
sLP = feedback(1,G10*kLP);
sGM = feedback(1,G10*kGM);
sMS = feedback(1,G10*kMS);
figure
bodemag(sLP,'b',sGM,'k',sMS,'r',inv(Wperf),'g--',wRange)
legend('LP','GM','MS','inv(Wperf)','Location','Best');
```



The next figure compares the frequency responses of the controllers All controllers have similar low frequency behavior although `kMS` does not have a pure integrator. `kMS` has a much sharper roll-off above the first flex mode as compared to the others. This effect is due to the choice of `Wu` and hence `kMS` is less likely to excite the high frequency flexible modes. It is also noted that `kMS` inverts the first flex mode (note the sharp zero near j6). This cancellation is fine as long as the frequency of the first flexible mode is known. However, this may cause issues if there is uncertainty in the first flexible mode.

```
figure;
bodemag(kLP,'b',kGM,'k',kMS,'r',wRange)
legend('LP','GM','MS','Location','Best');
```



Uncertainty Modeling

Variations in the frequency of the first flexible mode can be modeled using parametric uncertainty. Let wn1 denote the natural frequency of the first flexible mode. wn1^2 appears in the (3,1) entry of the state matrix of the four-state, reduced order model. A UREAL object is used to account for 6% uncertainty in wn1^2 which roughly corresponds to 3% uncertainty in wn1.

```
[A4,B4,C4,D4] = ssdata(G4);
```

Cast A4 as a `umat`

```
A4unc = umat(A4);
```

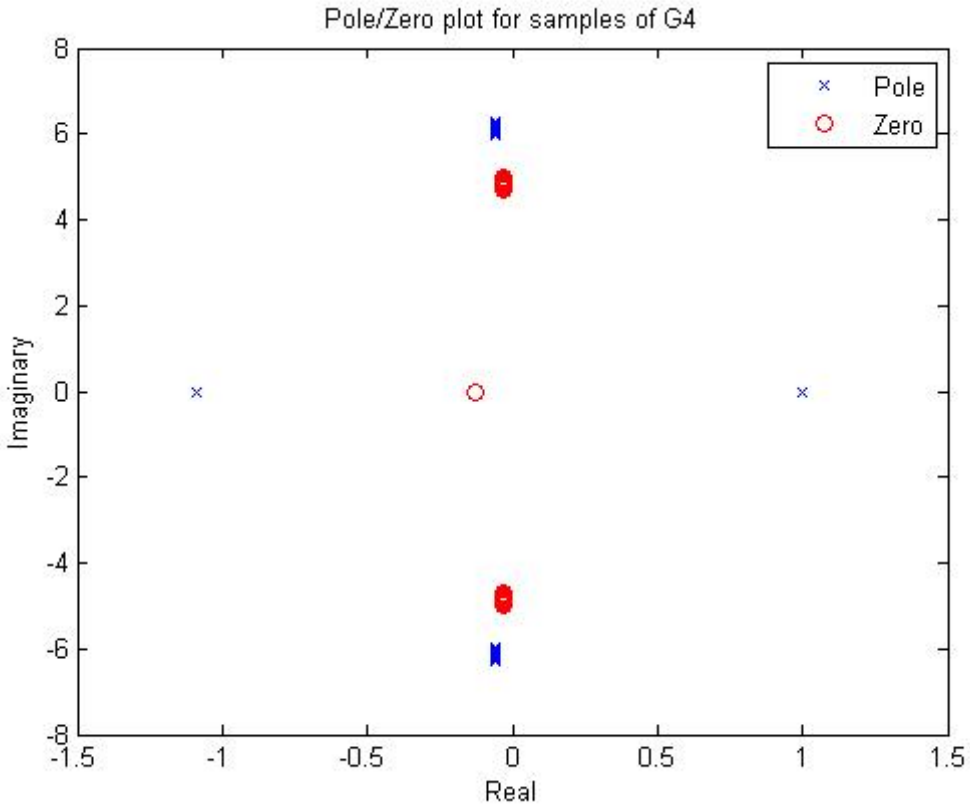
Replace (3,1) element with a `ureal` of appropriate nominal value and 6% uncertainty.

```
A4unc(3,1) = ureal('wn1sq',A4(3,1),'Percentage',[-6 6]);
G4unc = ss(A4unc,B4,C4,D4);
```

The pole/zero plot below shows the variation in first flex mode pole/zero locations due to the inclusion of the UREAL uncertainty in the model `G4unc`. Notice that the short period poles and zero are unaffected by the model uncertainty.

```
figure;
for i1=1:25,
    Gsamp = usample(G4unc);
    p = pole(Gsamp);
    z = zero(Gsamp);
    plot(real(p),imag(p),'bx',real(z),imag(z),'ro'); hold on;
```

```
end
hold off;
xlabel('Real');
ylabel('Imaginary');
title('Pole/Zero plot for samples of G4');
legend('Pole','Zero');
```



Design 4: Robust Mixed-Sensitivity Design with DKSYN

The third design uses the model `G4unc` and thus includes uncertainty in the location of the first flexible mode. The system interconnection is formed using `iconnect` and the mixed sensitivity control design (with plant uncertainty) is performed using `dksyn`. The resulting controller is denoted `kMU`. The optimal gain satisfies gammaMU<1 which implies that the design objectives are satisfied for all models in the uncertainty set.

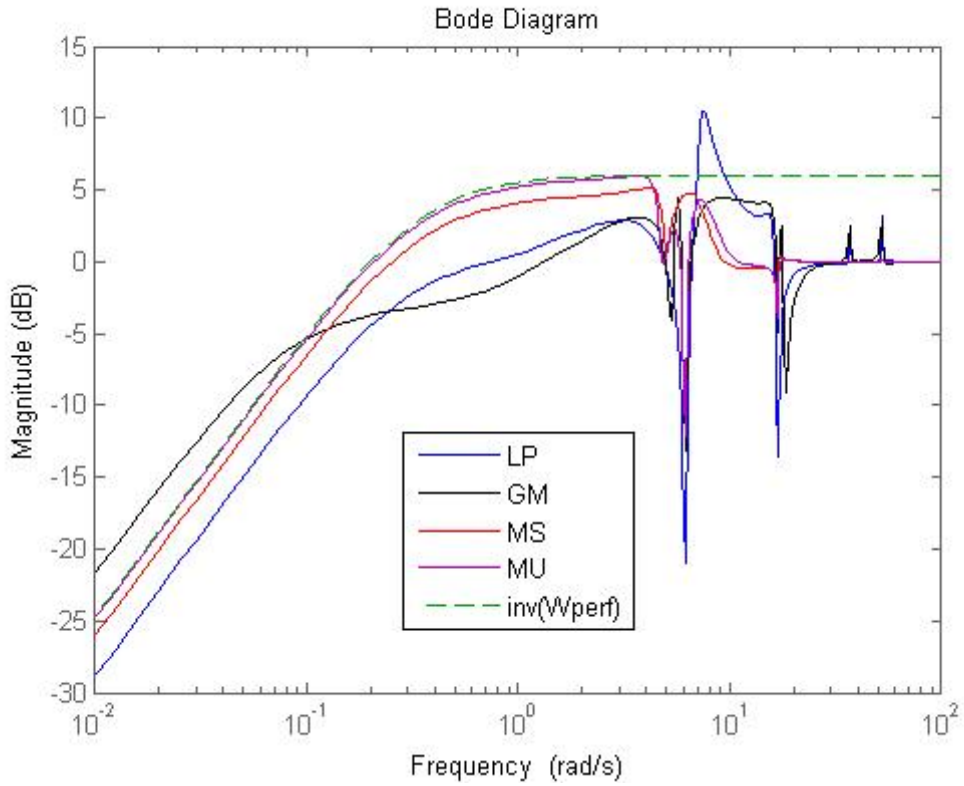
```
% Construct design interconnection with |iconnect|.  Any general interconnection
% function (eg., |sysic| or |connect|) would also suffice.
[ny,nu] = size(G4unc);
M = iconnect;
r = icsignal(ny);
u = icsignal(nu);
e = icsignal(ny);
M.Input = [r;u];
M.Output = [Wperf*e; Wu*u; e];
M.Equation{1} = equate(e,r-G4unc*u);
Munc = M.System;

% Design controller with |dksyn|
nmeas = 1;
ncon = 1;
```

```
opt = dkitopt;
%opt.DisplayWhileAutoIter = 'on';
opt.NumberOfAutoIterations = 4;
[kMU,clpMU,PerfMargin,infoMU] = dksyn(Munc,nmeas,ncon,opt);
```

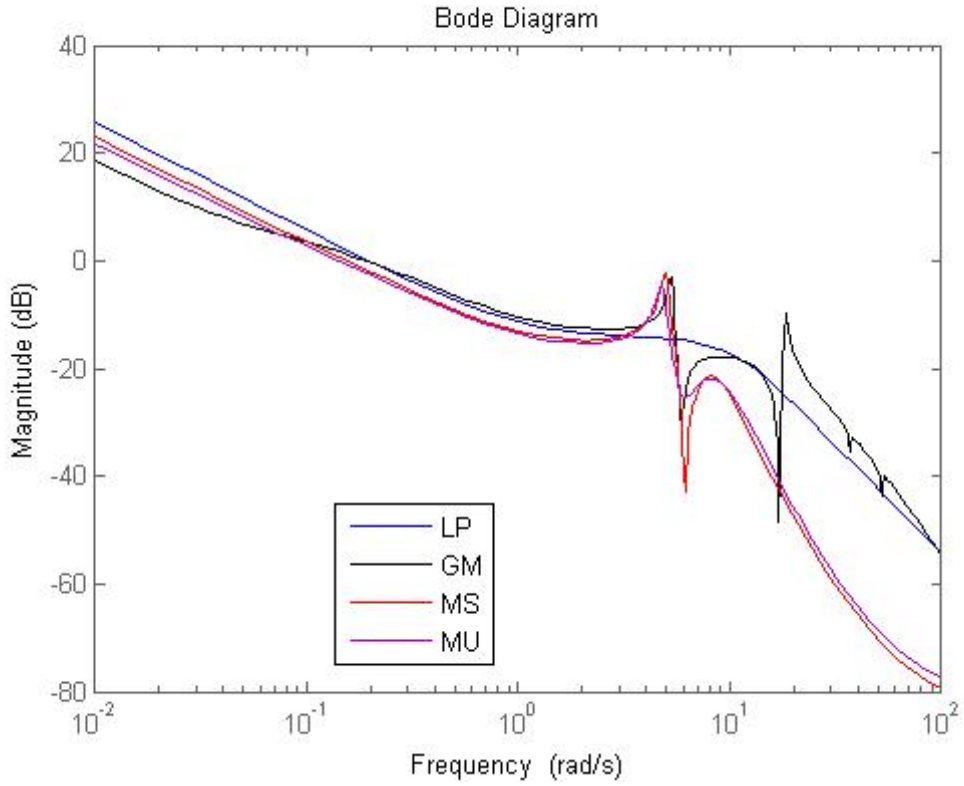
The next figure compares the closed-loop sensitivity transfer function for all control designs. The sensitivity functions are calculated using the full 10-state model, `G10`. The figure also shows the inverse of the performance weight. `sMU` lies slightly above `sMS` at most frequencies. As we will see, the controller `kMU` has sacrificed some tracking performance on the nominal system in order to achieve better robustness with respect to the uncertainty in the first flex mode frequency.

```
sMU = feedback(1,G10*kMU);
figure
bodemag(sLP,'b',sGM,'k',sMS,'r',sMU,'m',inv(Wperf),'g--',wRange)
legend('LP','GM','MS','MU','inv(Wperf)','Location','Best');
```



The next figure compares the frequency responses of the three controllers. The mixed sensitivity designs `Kms` and `Kdk` are similar except that `Kdk` does not invert the first flexible mode. Instead the phase of `Kdk` ensures stabilization of this mode. This behavior is due to the inclusion of the real parameter uncertainty in the design model `G4unc` which accounts for variations in the natural frequency of the first flex mode.

```
figure;
bodemag(kLP,'b',kGM,'k',kMS,'r',kMU,'m',wRange)
legend('LP','GM','MS','MU','Location','Best');
```

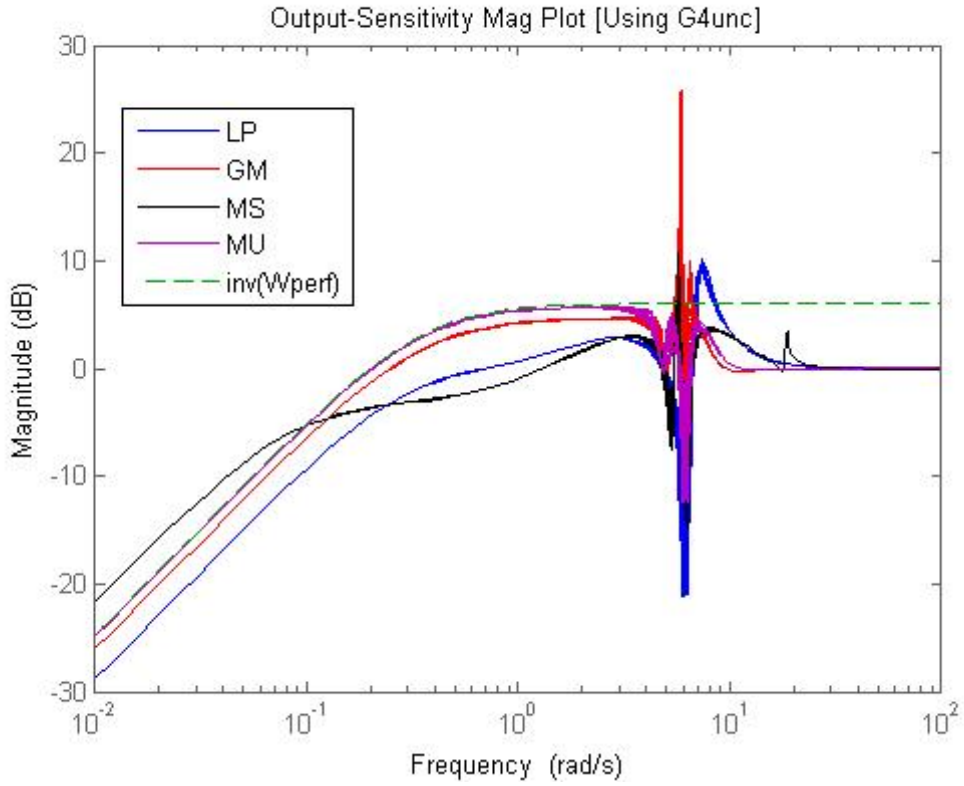


Robustness Analysis

The addition of model uncertainty in the last control design trades off some nominal performance in order to achieve robustness. The robustness of the three control designs can be analyzed in several ways. First, the figure below again shows the Bode magnitude plot of the sensitivity functions, S=1/(1+G*K), for each of the three control designs. In this case the plots are shown using the uncertain plant, `G4unc`, that includes real parametric uncertainty in the first flex mode frequency. The controller `kMU` was designed using the plant `G4unc`. Since `dksyn` achieved `gammaMU < 1.0`, we expect that the sensitivity function with `kMU` should remain below `inv(Wperf)` for all plants in the uncertainty set. The figure below confirms this expectation. The controllers `kLP`, `kGM` and `kMS` did not include this uncertainty in their design. As a result, their sensitivity functions do not robustly satisfy the performance objective. In particular, the classical design `kLP` has an 8dB peak near 0.25rad/sec. Moreover, `kMS` has a very large peak near the first elastic mode which exceed inv(Wperf) because this design tries to invert the uncertain first flex mode.

```
sLPunc = feedback(1,G4unc*kLP);
sGMunc = feedback(1,G4unc*kGM);
sMSunc = feedback(1,G4unc*kMS);
sMUunc = feedback(1,G4unc*kMU);

figure
bodemag(sLPunc,'b',sMSunc,'r',sGMunc,'k',sMUunc,'m',inv(Wperf),'g--',wRange)
legend('LP','GM','MS','MU','inv(Wperf)','Location','Best');
title('Output-Sensitivity Mag Plot [Using G4unc]');
```



We can use `wcgain` to compute the worst-case gain of the closed-loop sensitivity transfer functions. The worst-case is measured using the H-infty norm, i.e. the worst perturbation is the one that causes the largest peak on the Bode magnitude plot of the sensitivity function. As expected from the Bode magnitude plots above, the sensitivity has a much larger worst-case peak with `Kms` than with `Kdk`. The worst-case peak with `Kdk` is approximately 1.9 which is below the peak bound specified by `inv(Wperf)`.

```
[wcgMS,wcuncMS] = wcgain(sMSunc);
```

Compare nominal and worst-case weighted gains

```
[norm(sMSunc.Nominal,inf) wcgMS.LowerBound]
```

```
ans =

    1.7135   35.0333
```

```
[wcgMU,wcuncMU] = wcgain(sMUunc);
```

Compare nominal and worst-case weighted gains

```
[norm(sMUunc.Nominal,inf) wcgMU.LowerBound]
```

```
ans =
```

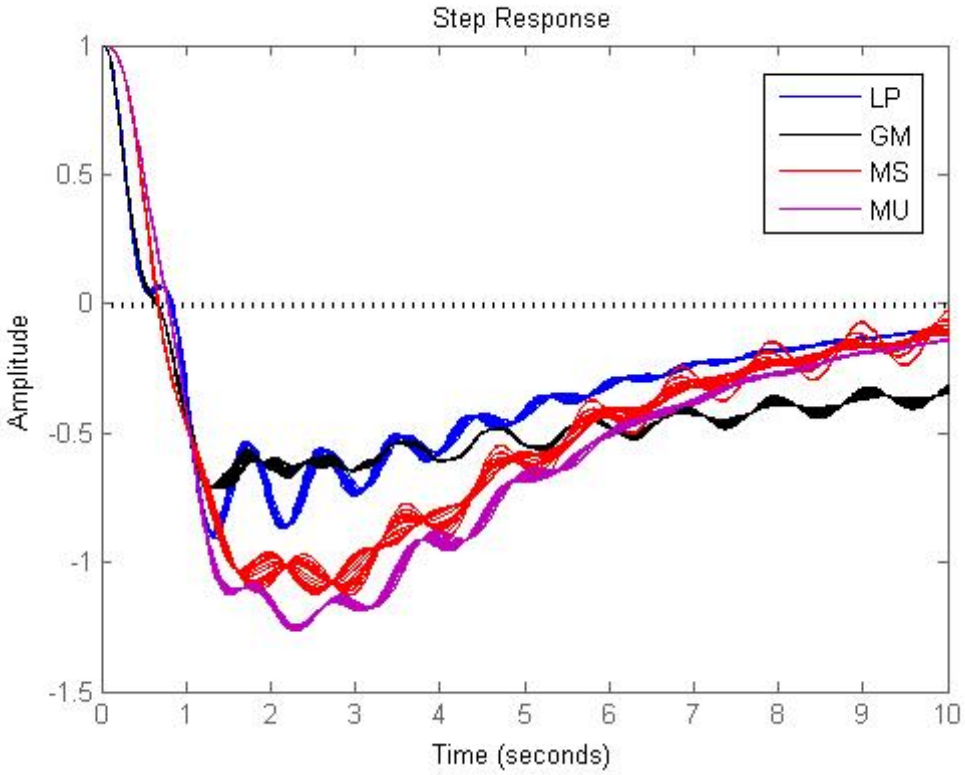
```
    1.9131    1.9210
```

Clearly, kMU sacrifices some nominal performance to achieve better robustness.

Step responses

Based on a few samples, the step responses show the high degree of insensitivity of the kMU design to the low-frequency mode's value.

```
step(feedback(1,G4unc*kLP),'b',feedback(1,G4unc*kGM),'k',...
   feedback(1,G4unc*kMS),'r',feedback(1,G4unc*kMU),'m',10);
legend('LP','GM','MS','MU','Location','Best');
```



Published with MATLAB® R2013b