HDD Robustness Analysis

This problem analyzes the robustness of a feedback system with the HDD and a classical controller

Contents

Load Plant and Define Controller

`[A0,B0,C0,D0]: State space data for a nominal design fit`

```
% Plant Design Model
load('HDDdata');
G0 = ss(A0,B0,C0,D0);

% Uncertainty Weight
Wu = tf([1 1],[0.5 3])^4;

% Control Law Design:
% Use Proportional gain to set the loop crossover frequency.  Then use a
% lead controller to add phase at the loop crossover.
wb = 0.4;
Kp = 1/abs( freqresp(G0,wb) );
beta = 3;
Klead = tf([beta wb],[1 beta*wb]);
K = Kp*Klead;
```

(A) Nominal Performance

Evaluate performance of nominal closed-loop system

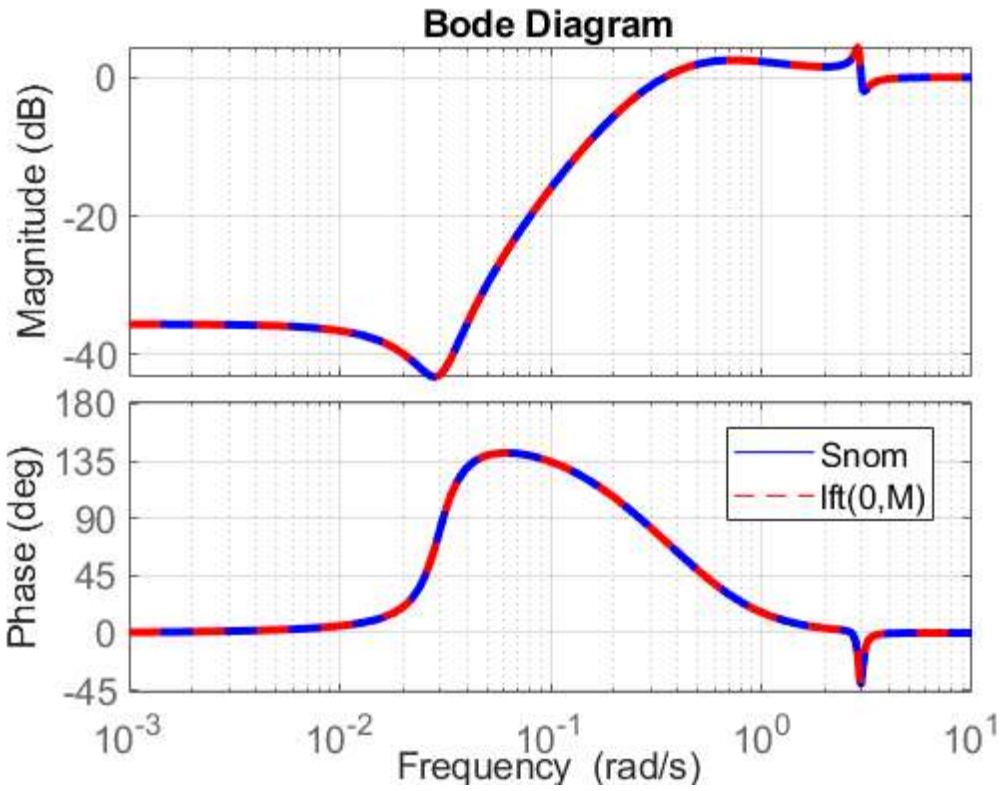
```
% Closed-loop sensitivity S and complementary sensitivity T functions
Snom = feedback(1,G0*K);  % S = 1/(1+G0*K)

% Nominal gain from reference r to error e
gnom = norm(Snom,inf);
```

(B) Construct LFT Form

Form uncertain closed-loop sensitivity as an LFT Fu(G,Delta) Note that: e = r - y = r - G0(w + K e) --> [1+G0 K] e = r - G0 w e = S r - G0*S w where S:= 1/(1+G0 K) is the nominal sensitivity (without uncertainty) It also follows that v = Wu K e = Wu K*S r - Wu T w where T:= (G0 K)/(1+ G0 K) is the nominal complementary sensitivity. Finally we have [v;e] = [-Wu*T Wu*K*S; -G0*S S] [w;r] = G [w;r] We can also construct this in Matlab using uncertain objects (e.g. udyn) and lftdata. This is useful for more complicated systems with many sources of nonlinearity and uncertainty.

```
% Nominal System
Tnom = 1-Snom;
G0Snom = feedback(G0,K);   % P0*Snom
M = [-Wu*Tnom Wu*K*Snom; -G0Snom Snom];
M =  minreal(ss(M));

% Create uncertain sensivity system
Delta = ultidyn('Delta',[1 1]);
Sunc = feedback(1,G0*(1+Delta*Wu)*K);

% Extract LFT nominal system
Mtest=lftdata(Sunc);

% We constructed G by hand and Gtest using Matlab objects. The difference
% between these two should have zero norm.
Merror = norm(M-Mtest,inf)

% Verify that Fu(G,0) corresponds to nominal sensitivity
figure(1)
bode(Snom,'b',lft(0,M),'r--');
grid on;
legend('Snom','lft(0,M)','Location','Best');
if exist('garyfyFigure','file'); garyfyFigure; end
```

```
20 states removed.
Merror =
   1.6591e-11
```



(C) Robust Performance: Upper Bound for Nonlinear uncertainty

Compute upper bound on the largest possible gain from r to e over all possible nonlinear uncertainties.

```
% Extract state matrices for nominal part M
[Am,Bm,Cm,Dm]=ssdata(M);
B1 = Bm(:,1);
B2 = Bm(:,2);
C1 = Cm(1,:);
C2 = Cm(2,:);
D11 = Dm(1,1);
D12 = Dm(1,2);
D21 = Dm(2,1);
D22 = Dm(2,2);

% Dimensions
nx = size(Am,1);
nw = 1;
nd = 1;
nv = 1;
ne = 1;

% Form matrices used in SDP
Zwd = zeros(nw,nd);
Zwx = zeros(nw,nx);
Iw = eye(nw);
Id = eye(nd);
Zw = zeros(nw);
Lvw = [C1 D11 D12; Zwx Iw Zwd];
Le = [C2 D21 D22];

% Solve SDP to compute upper bound on gain
beta = 1;
J = [beta^2 0; 0  -1];

cvx_begin sdp quiet
    variable P(nx,nx) semidefinite;
    variable gsq(1,1);
    variable lambda nonnegative;

    [Am'*P+P*Am P*B1 P*B2; B1'*P Zw Zwd; B2'*P Zwd' -gsq*Id] ...
        +lambda*(Lvw'*J*Lvw) + Le'*Le <=0;

    minimize(gsq)
cvx_end
gub = sqrt(gsq);

fprintf('\ngnom = %4.3f, \t gnl (upper bnd) = %4.3f\n',gnom, gub);
```

```
gnom = 1.672,    gnl (upper bnd) = 143.061
```

(D) Robust Performance: Upper Bound for LTI uncertainty

Compute upper bound on the largest possible gain from r to e over all possible LTI uncertainties.

```
% Form Interconnection
M.InputName = {'w';'r'};
M.OutputName = {'v';'e'};

L1 = ss(1);
Psi1 = blkdiag(L1,L1);
Psi1.InputName = {'v','w'};
Psi1.OutputName = {'z1'};

p = 0.5;
L2 = tf(p,[1 p]);
Psi2 = blkdiag(L2,L2);
Psi2.InputName = {'v','w'};
Psi2.OutputName = {'z2'};

Mext = connect(M,Psi1,Psi2,{'w','r'},{'z1','z2','e'});

% Extract state matrices for nominal part G
[Am,Bm,Cm,Dm]=ssdata(Mext);
B1 = Bm(:,1);
B2 = Bm(:,2);
C1 = Cm(1:2,:);
C2 = Cm(3:4,:);
C3 = Cm(5,:);
D1 = Dm(1:2,:);
D2 = Dm(3:4,:);
D3 = Dm(5,:);

% Dimensions
nx = size(Am,1);
nw = 1;
nd = 1;

% Form matrices used in SDP
Zwd = zeros(nw,nd);
Zwx = zeros(nw,nx);
Iw = eye(nw);
Id = eye(nd);
Zw = zeros(nw);
Lv1w1 = [C1 D1];
Lv2w2 = [C2 D2];
Le = [C3 D3];

% Solve SDP to compute upper bound on gain
beta = 1;
J = [beta^2 0; 0  -1];

cvx_begin sdp quiet
    variable P(nx,nx) semidefinite;
    variable gsq(1,1);
    variable lambda1 nonnegative;
    variable lambda2 nonnegative;

    [Am'*P+P*Am P*B1 P*B2; B1'*P Zw Zwd; B2'*P Zwd' -gsq*Id] ...
        +lambda1*(Lv1w1'*J*Lv1w1) + lambda2*(Lv2w2'*J*Lv2w2) +  Le'*Le <=0;

    minimize(gsq)
```

```
cvx_end
gub = sqrt(gsq);

fprintf('\ngnom = %4.3f, \t glin (upper bnd) = %4.3f\n',gnom, gub);
```

```
gnom = 1.672,    glin (upper bnd) = 4.505
```

Special Code: WCGAIN

For comparison, the code below calls the function WCGAIN. This is specialized code that computes the largest gain for LTI uncertaintainties. This is exact (within numerical tolerances) if there is a single LTI uncertainty. It returns the worst-case gain WCG. (This has both upper and lower bounds in general but these will agree if there is a single LTI uncertainty). It also returns an instance of Delta that achieves the worst-case gain. Note that the CVX code above achieves a similar upper bound as WCGAIN.

```
[WCG,WCU] = wcgain(Sunc);
WCG
```

```
WCG =
  struct with fields:

          LowerBound: 4.3382
          UpperBound: 4.3474
    CriticalFrequency: 2.9062
```

Published with MATLAB® R2023b