NLDynamicUncertainty

This problem analyzes the robustness of a classical feedback system with a single, dynamic nonlinear uncertainty.

Contents

(A) Nominal Performance

Evaluate performance of nominal closed-loop system

```
% Open-loop Plant
G = tf(4,[1 8 7]);

% Proportional Controller
Kp = 20;

% Closed-loop sensitivity S and complementary sensitivity T functions
Snom = feedback(1,G*Kp);  % S = 1/(1+F*Kp)

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

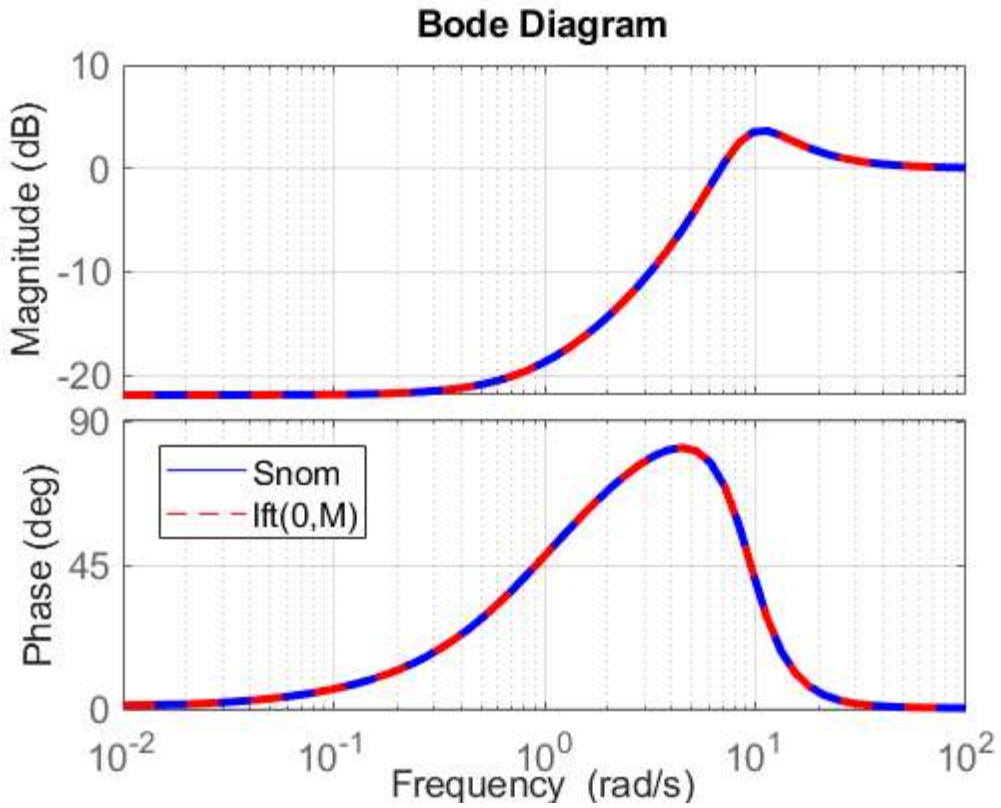
(B) Construct LFT Form

Form uncertain closed-loop sensitivity as an LFT Fu(M,Delta) Note that: e = r - y = r - G(w + Kp e) --> [1+GKp] e = r - G w e = S r - G*S w where S:= 1/(1+FGp) is the nominal sensitivity (without uncertainty) It also follows that v = Kp e = Kp r - T w where T:= (GKp)/(1+G Kp) is the nominal complementary sensitivity. Finally we have [v;e] = [-T Kp*S; -G*S S] [w;r] = M [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;
FSnom = feedback(G,Kp);   % F*Snom
M = [-Tnom Kp*Snom; -FSnom Snom];
M =  minreal(ss(M));

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

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

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

% Verify that Fu(M,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'), garyfyFigure, end
```

```
2 states removed.
Merror =
   1.0120e-13
```



(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 = 2;
nw = 1;
nd = 1;
```

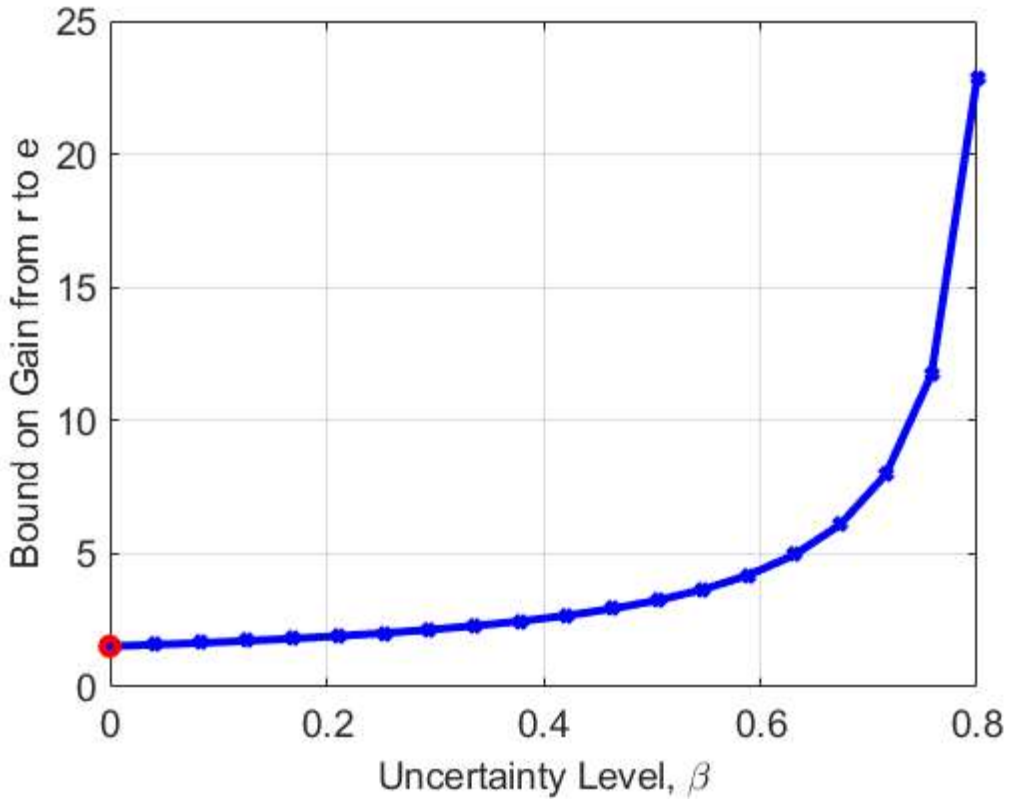
```
nv = 1;
ne = 1;

% 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 = 0.2;
M = [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'*M*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 = 1.528,    gnl (upper bnd) = 1.886
```

(D) Plot of Gain (r to e) vs. Uncertainty Level

```
% Grid of values of alpha
N=20;
beta = linspace(0,0.8,N);

gub = zeros(N,1);
for i=1:N
    % Solve SDP to compute upper bound on gain
    M = [beta(i)^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'*M*Lvw) + Le'*Le <=0;

    minimize(gsq)
    cvx_end
    gub(i) = sqrt(gsq);
end
```

```
% Gain (r to e) vs. uncertainty level.   The value at beta=0 corresponds
% to the nominal gain.
figure(3)
plot(beta,gub,'b-x',0,norm(Snom,inf),'ro');
grid on;
xlabel('Uncertainty Level, \beta');
ylabel('Bound on Gain from r to e');
if exist('garyfyFigure'), garyfyFigure, end
```



Published with MATLAB® R2023b