Multivariable Feedback Controller Design on Reactive Ion Etching of Silicon Wafers

EECS 565 Final Project Report

Enxu LIU

Qinwen QIAN

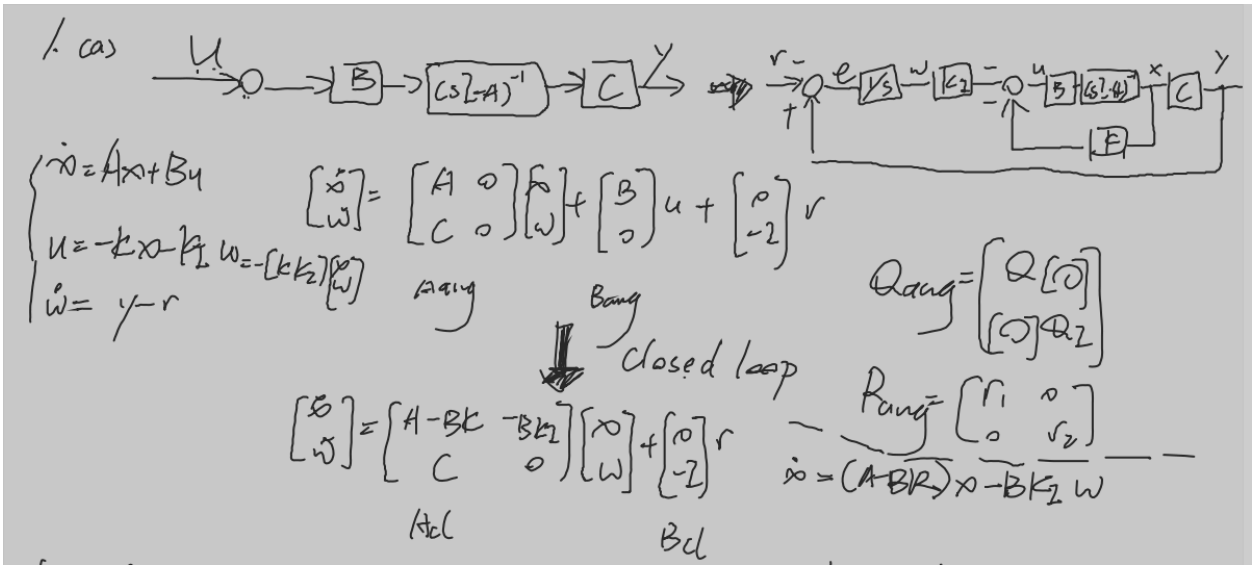
Table of Content

The MATLAB Code is attached at the end of this report.

Part I. Multivariable Feedback Controller Design

(a) Linear Quadratic Regulator with Integrator

We calculated the closed-loop state space function as following:



By choosing the LQR weighting matrix Q = diag(Q,QI) and R = [1,0; 0,2] where QI = diag([⅓, ⅓]) and QI = CP'*diag([3,3])*CP, we are able to achieve the target step responses with the multivariable controller shown in Figure 1.

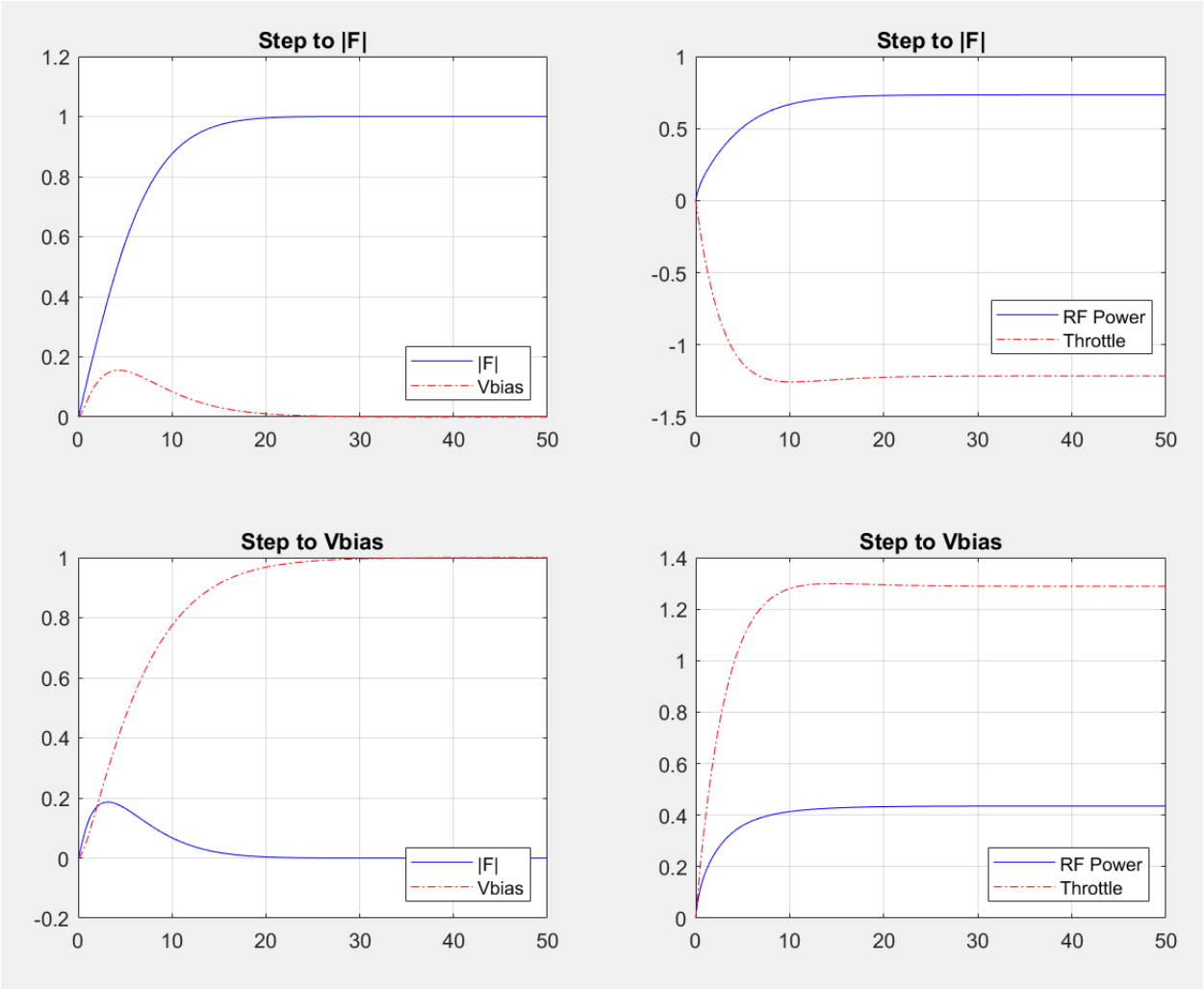


Figure 1. Step Responses with Multivariable Controller.

Set R = diag([R11,R22]). By decreasing the value of R22, the response of [F] to a step command to Vbias become smaller but the overshoot of the step input response will increase, especially the response of from RF Power to Vbias, shown in Figure 2. With overshoot of 5%, the smallest value of R22 we can get is 0.6.

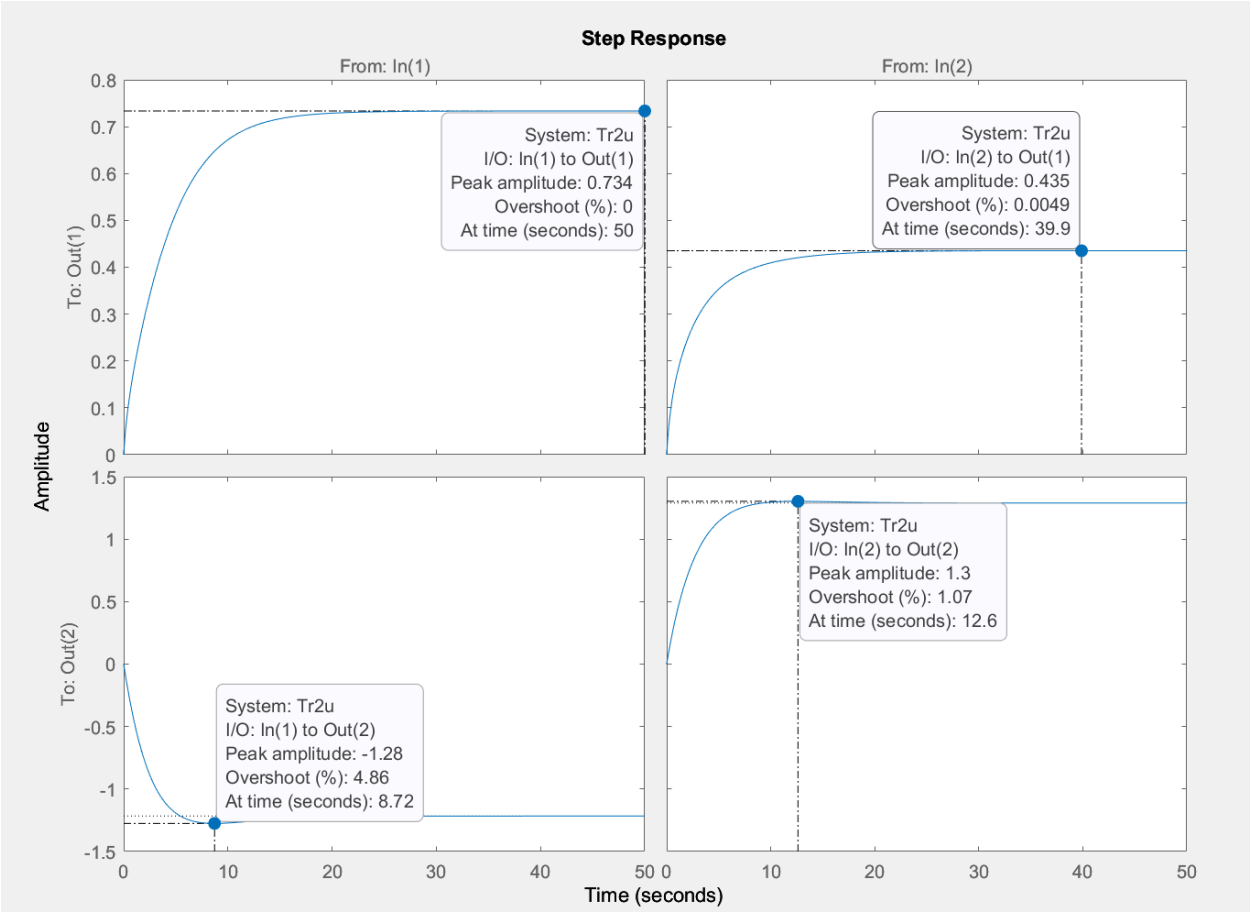
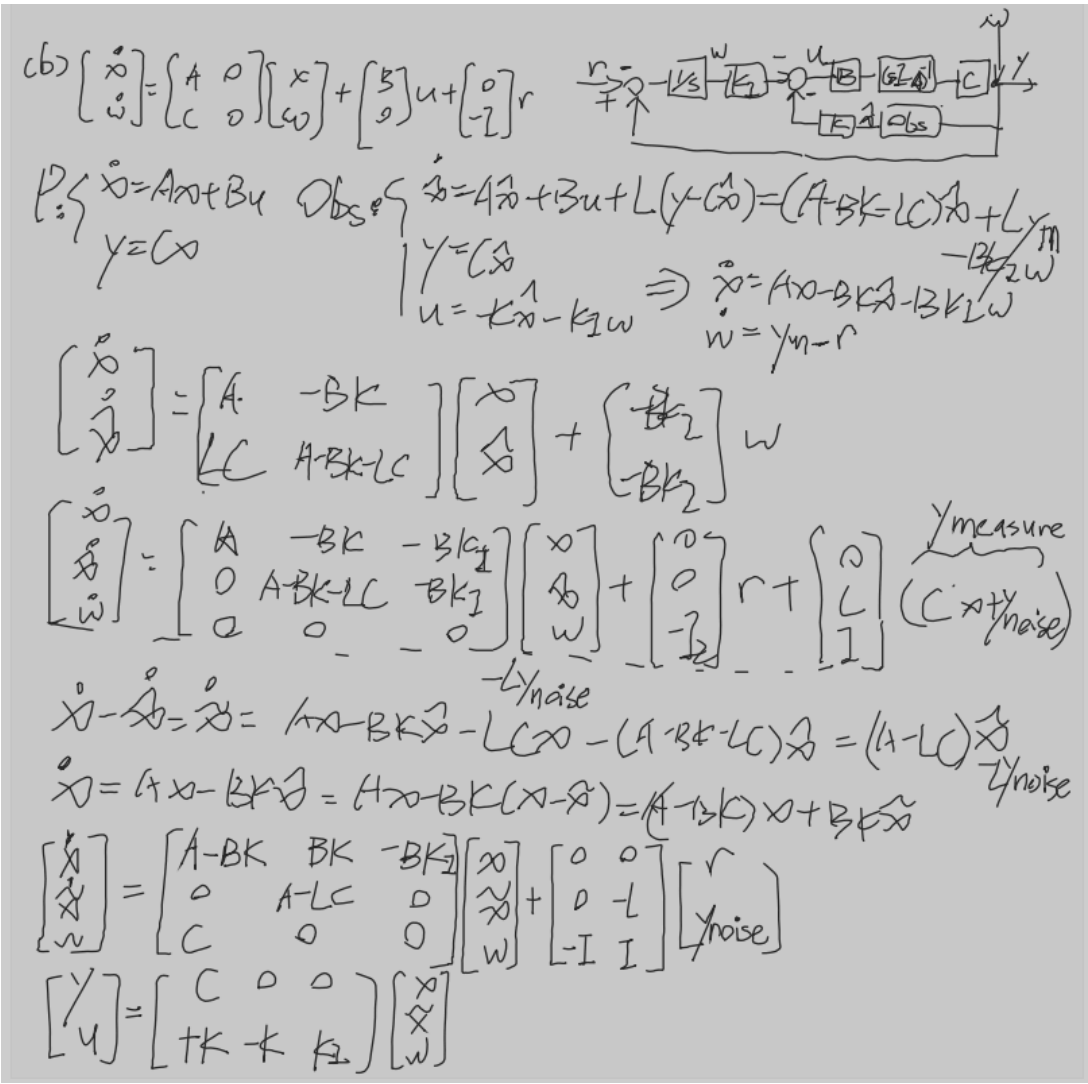


Figure 2. Step Response from Input [Power; Throttle] to Output [|F|; Vbias] when R22 = 0.6.

(b) LQR with Observer

Recall that in LQR design, y = Cx + w where w is the sensor noise. The output y is measured y value which equals to Cx + y_noise. We add new state w and define a new variable $\tilde{x} = x - \hat{x}$. Then we can set input of controller as [[F] reference; Vbias reference; [F] noise; Vbias noise]. The detailed calculation of the state space function is shown below:



We take q to be 1 in Figure 3. The largest value of q we can define is 10^10 until MATLAB gives an error. When we increase q, the response to the sensor noise becomes faster (shown in Figure 4 when q = 1000).

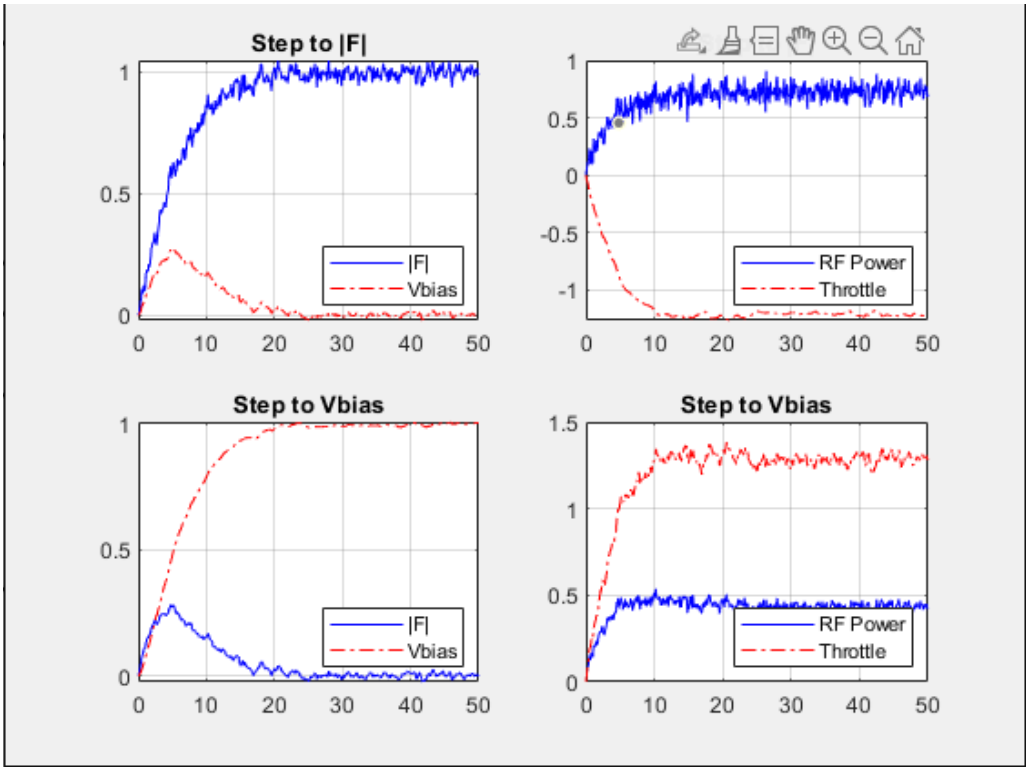


Figure 3. Step Responses with Multivariable Controller and [F] Sensor Noise with q = 1.

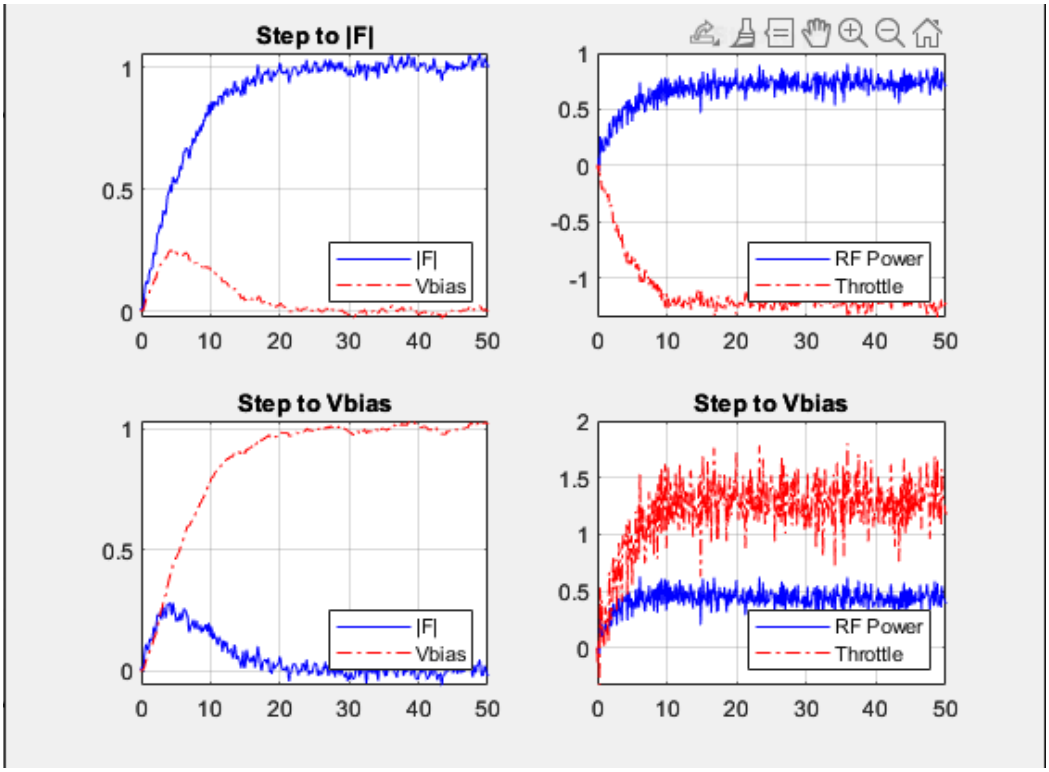


Figure 4. Step Responses with Multivariable Controller and [F] Sensor Noise with q = 1000.

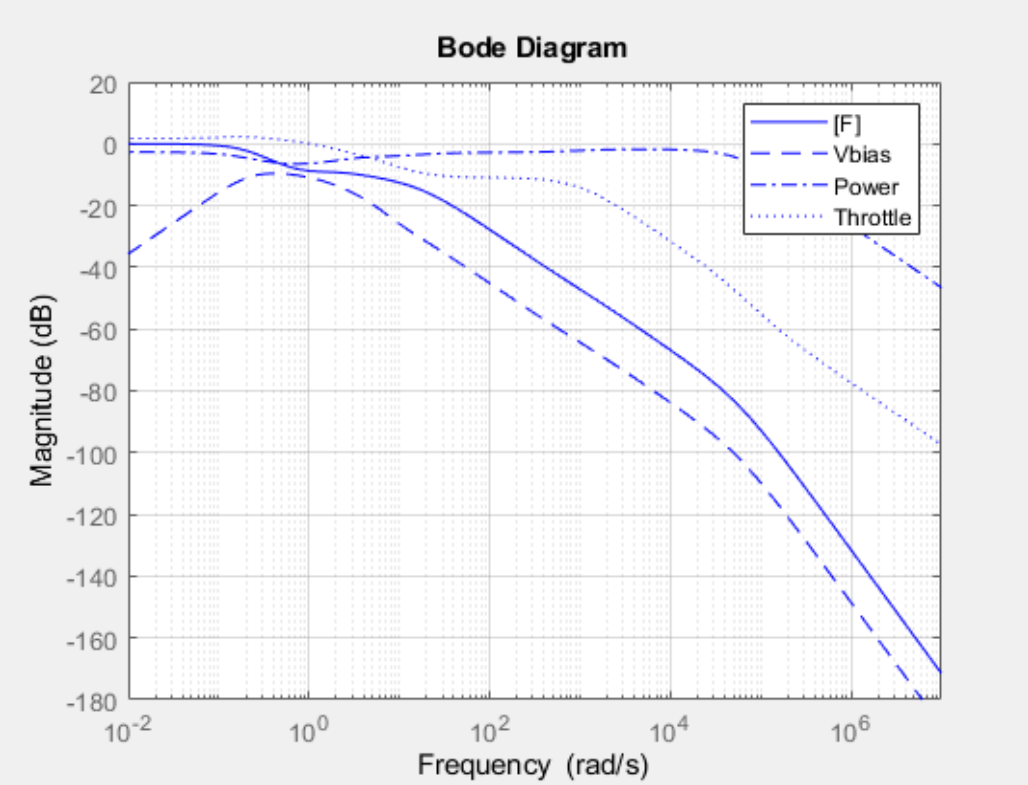


Figure 5. Bode Plots of the Closed Loop Transfer Functions from [F] Noise to [F], Vbias, Power, and Throttle, beta=1000

(c) Whether an arbitrarily good recovery

It will not be able to achieve arbitrarily good recovery, there will still be noise and fluctuation left in the result and the attenuation of noise in [F] and Bias comes at the cost of the amplification of noise of control inputs. This is because no matter how the beta values (scaling factor of the process noise) are tuned to change the observer gain, the poles of estimators will always approach either the poles or zeros of the open loop, or mirrors of them if in ORHP, and the state feedback gain will also amplify the unfiltered process noise which again worsen the results.

(d) Multivariable stability margins

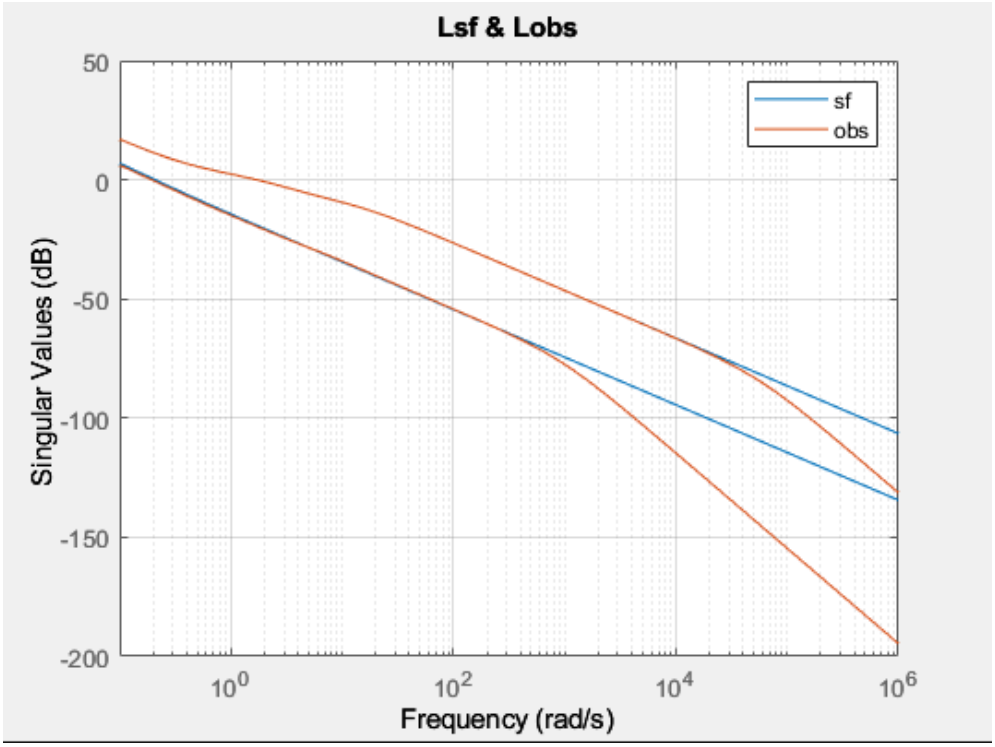


Figure 6: The State Feedback Loop Transfer Function, Lsf and the Input Loop Transfer Function, LI

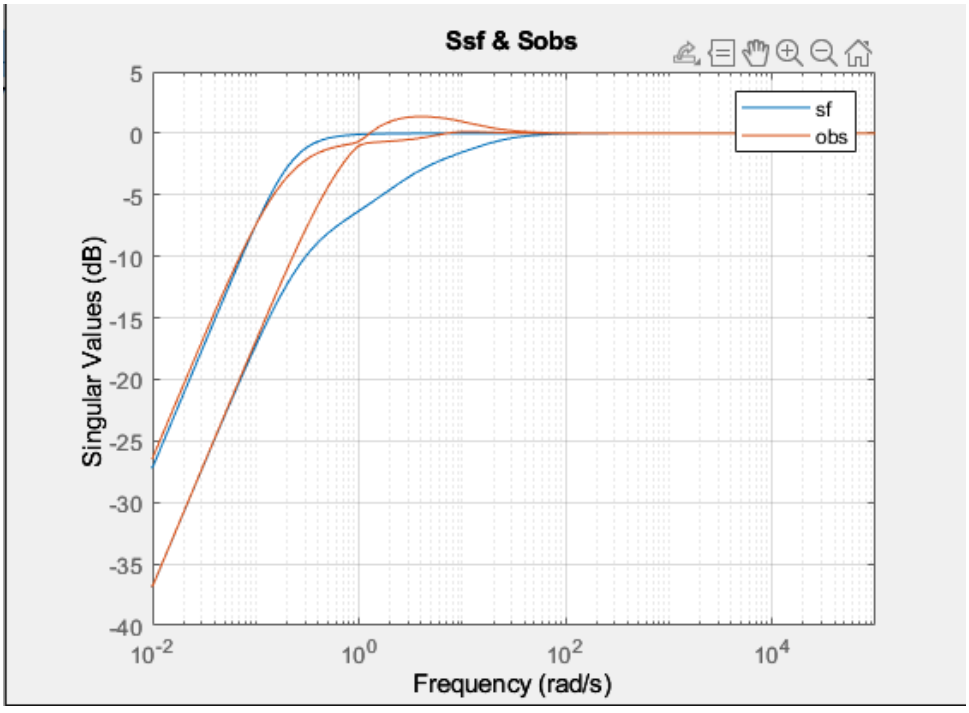


Figure 7: The State Feedback Sensitivity Function, Ssf and the Input Sensitivity Function, SI

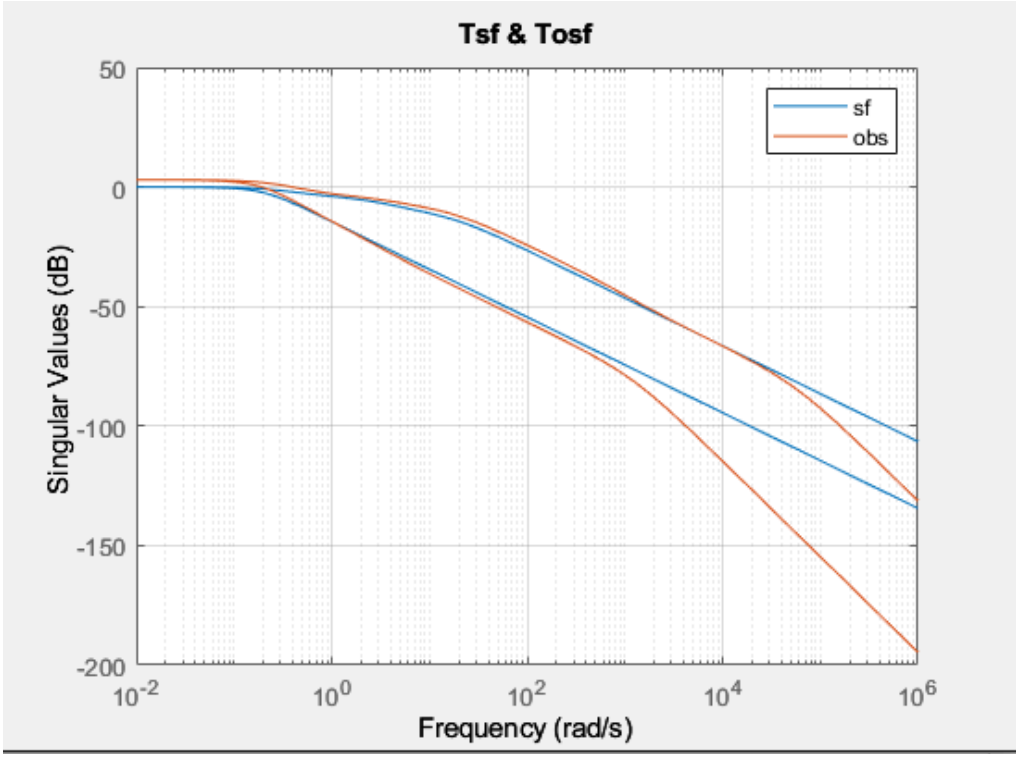


Figure 8: The State Feedback Complementary Sensitivity Function, Tsf and the Input Complementary Sensitivity Function, TI

Computation of stability margins are shown in Figure 9 to 11. The loop-at-a-time margin has an extremely small gain margin and phase margin of 84.5 degrees. The multi-loop disk margin is 1.81. The unstructured (fully coupled) margin is smaller than the loop-at-a-time margin . Thus the multi-loop design has better stability margins than the original system and the decentralized system.

```
258   % Loop-at-a-time margins at the plant input
259   AM = allmargin(Lsf);
260   AM(1)
261   AM(2)
262
```

```
Command Window
  struct with fields:

     GainMargin: [1×0 double]
    GMFrequency: [1×0 double]
    PhaseMargin: 124.9694
    PMFrequency: 1.7044
    DelayMargin: 1.2797
    DMFrequency: 1.7044
         Stable: 1


ans =

  struct with fields:

     GainMargin: 4.1895e-16
    GMFrequency: 0
    PhaseMargin: 84.5248
    PMFrequency: 0.2101
    DelayMargin: 7.0222
    DMFrequency: 0.2101
         Stable: 1
```

Figure 9: Loop-at-a-time Margin

```
262   [DMI, MMI] = diskmargin(Lsf);
263   DMI(1)
264   DMI(2)
265   MMI
```

```
ommand Window
 ans =

  struct with fields:

          GainMargin: [0 Inf]
         PhaseMargin: [-90 90]
          DiskMargin: 2
          LowerBound: 2
          UpperBound: 2
           Frequency: Inf
    WorstPerturbation: [2×2 ss]


ans =

  struct with fields:

          GainMargin: [0.0478 20.9070]
         PhaseMargin: [-84.5232 84.5232]
          DiskMargin: 1.8174
          LowerBound: 1.8174
          UpperBound: 1.8174
           Frequency: 0.2137
    WorstPerturbation: [2×2 ss]

MMI =

  struct with fields:

          GainMargin: [0.0497 20.1340]
         PhaseMargin: [-84.3132 84.3132]
          DiskMargin: 1.8107
          LowerBound: 1.8107
          UpperBound: 1.8144
           Frequency: 0.2222
    WorstPerturbation: [2×2 ss]
```

Figure 10: Disk Margins

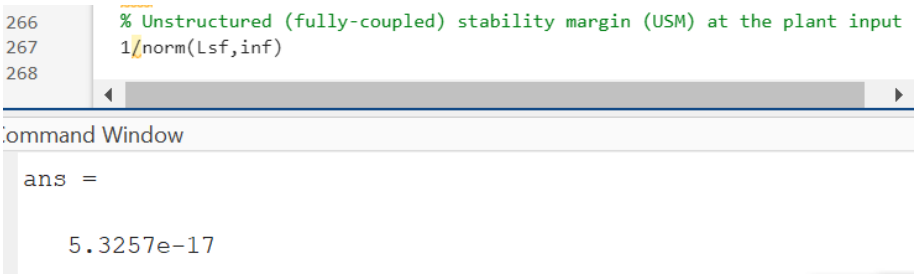


Figure 11: Unstructured (fully coupled) margins

Part II. Reverse Engineering the Multivariable Controller

(a) Prove of equation(1) Tr2y(s)

$\dot{x} = Ax + Bu$

$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x})$

$u = -K\hat{x} + v$

$v = -\frac{K_I}{s}(y - r) = \frac{K_I}{s}(r - y)$

then $\begin{bmatrix} \dot{x} \\ \dot{\hat{x}} \end{bmatrix} = \begin{bmatrix} A & 0 \\ LC & A - LC \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} + \begin{bmatrix} B \\ B \end{bmatrix} u$

Define $e = \hat{x} - x$

$\Rightarrow \begin{bmatrix} \dot{e} \\ \dot{\hat{x}} \end{bmatrix} = \begin{bmatrix} A - LC & 0 \\ -LC & A \end{bmatrix} \begin{bmatrix} e \\ \hat{x} \end{bmatrix} + \begin{bmatrix} 0 \\ B \end{bmatrix} u$

There is no effect of u on e term $\Rightarrow$ not controllable from r

The minimal realization from $v \to y$ is

$$X(s) = (sI - (A - BK))^{-1} B V(s).$$

$$= (sI - A + BK)^{-1} B V(s)$$

$$\Rightarrow Y(s) = C(sI - A + BK)^{-1} B V(s)$$

$$Y(s) = C(sI - A + BK)^{-1} B \frac{K_I}{s} [R(s) - Y(s)]$$

$$[I + C(sI - A + BK)^{-1} B \frac{K_I}{s}] Y(s) = C(sI - A + BK)^{-1} B \frac{K_I}{s} R(s)$$

Rearrange the term $C(sI - A + BK)^{-1} B$

$$C(sI - A + BK)^{-1} B = C[(sI - A)^{-1} \cdot (sI - A)](sI - A + BK)^{-1} B$$

$$= \frac{C(sI - A)^{-1} B}{\frac{sI - A + BK}{sI - A}}$$

$$= \frac{C(sI - A)^{-1} B}{I + (sI - A)^{-1} B}$$

$$= P(s)[I + (sI - A)^{-1} B]^{-1}$$

Then $\{I + P(s)[I + (sI - A)^{-1} B]^{-1} \frac{K_I}{s}\} Y(s) = P(s)[I + (sI - A)^{-1} B]^{-1} \frac{K_I}{s} R(s)$

$$\Rightarrow T_{r \to y}(s) = [I + P(s)(I + (sI - A)^{-1} B)^{-1} \frac{K_I}{s}]^{-1} P(s)[I + (sI - A)^{-1} B]^{-1} \frac{K_I}{s}$$

(b) Equivalent Controller

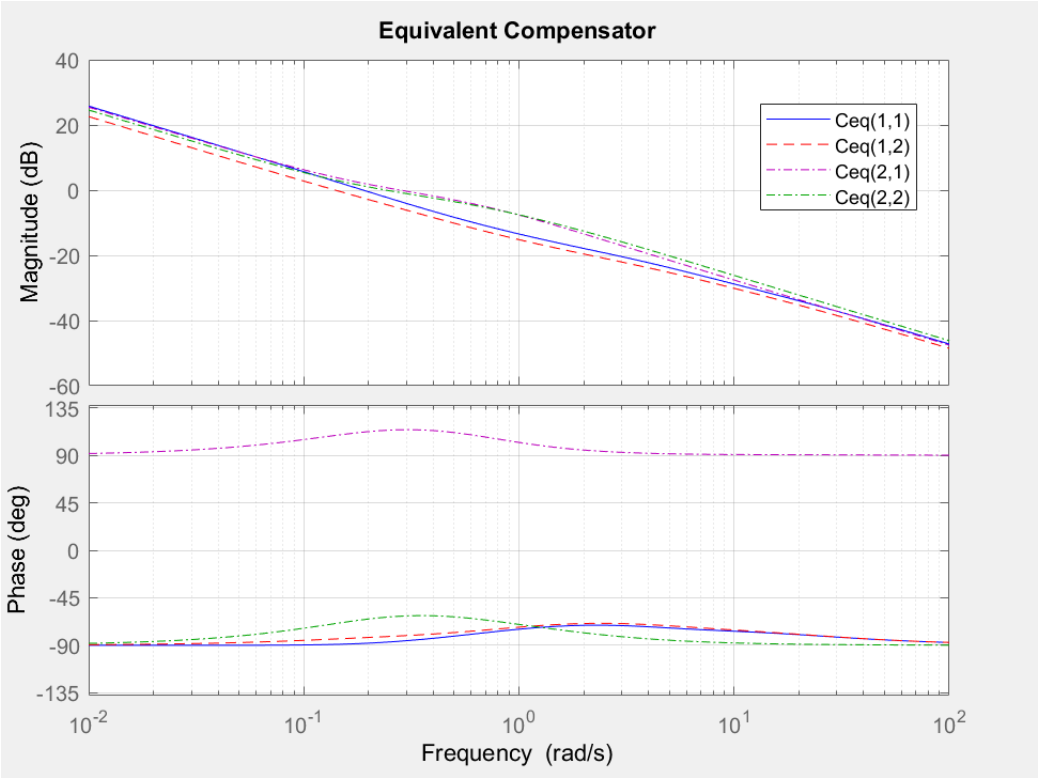


Figure 12. Bode Plot of Equivalent Compensator

(c) Decentralized Approximation

Set both Cd1 and Cd2 equal to second order Pade approximation. From Figure 13, the bode plots of Ceq diagonal elements and their approximations are aligned with each other respectively. However, the step response of Ceq and Ceq approximation still has some difference between them. By changing the order of Pade approximation, the step response of the two controllers was about the same, shown in Figure 14.

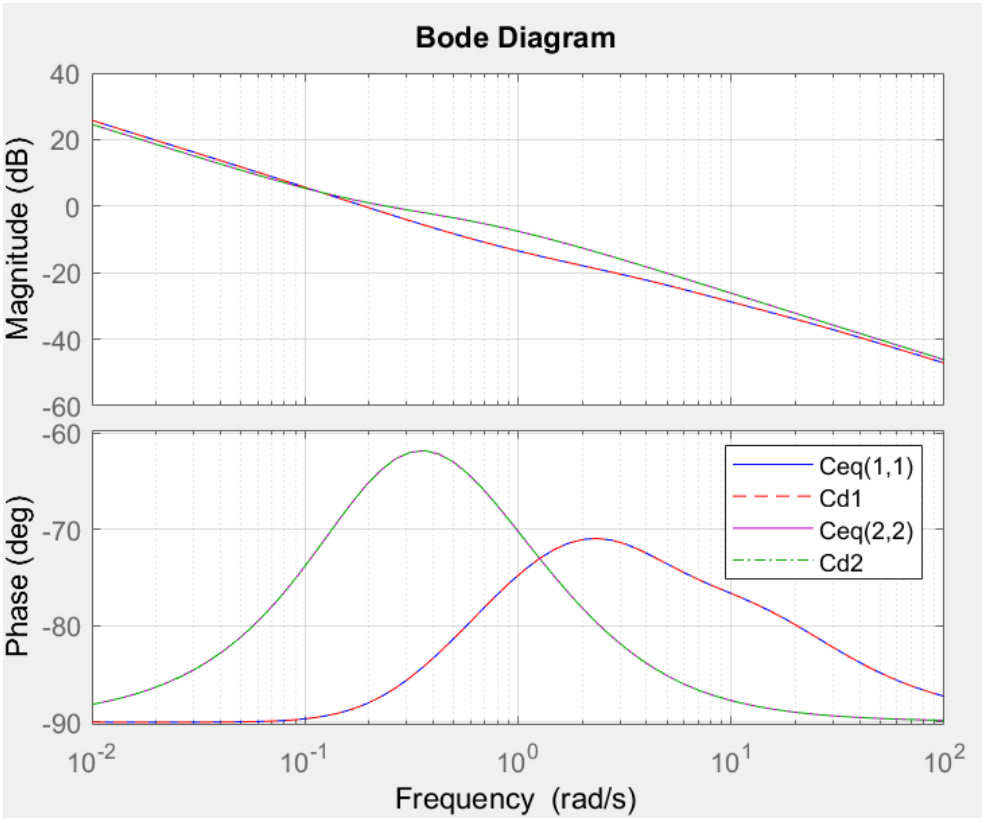


Figure 13. Bode Plot Comparison of diagonal elements of Ceq with Approximations Cd1 and Cd2.

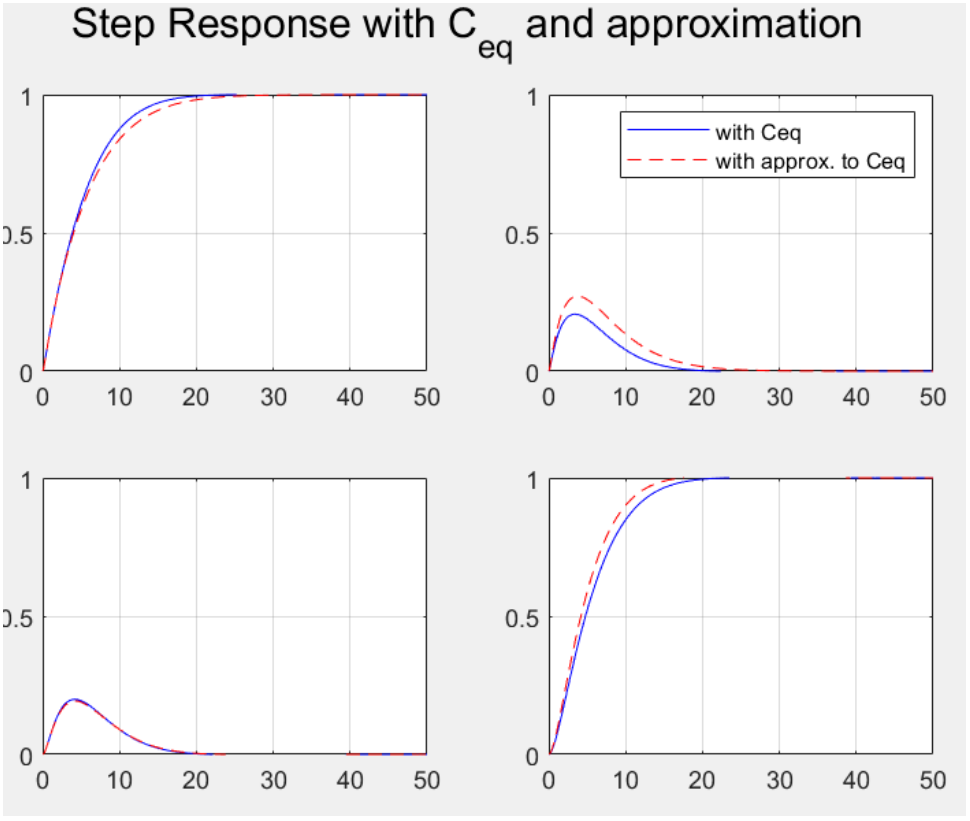


Figure 14. Step Response with Ceq and Approximation.

(d) Comment of Plant Transformation

By redefining the plant output and applying sequential loop closure, we can have a new controller to allow the power control signal to depend on both input tracking errors. Such that the controller drives [F] to a desired value will not have trade off against the desire to minimize the resulting disturbance to Vbias.

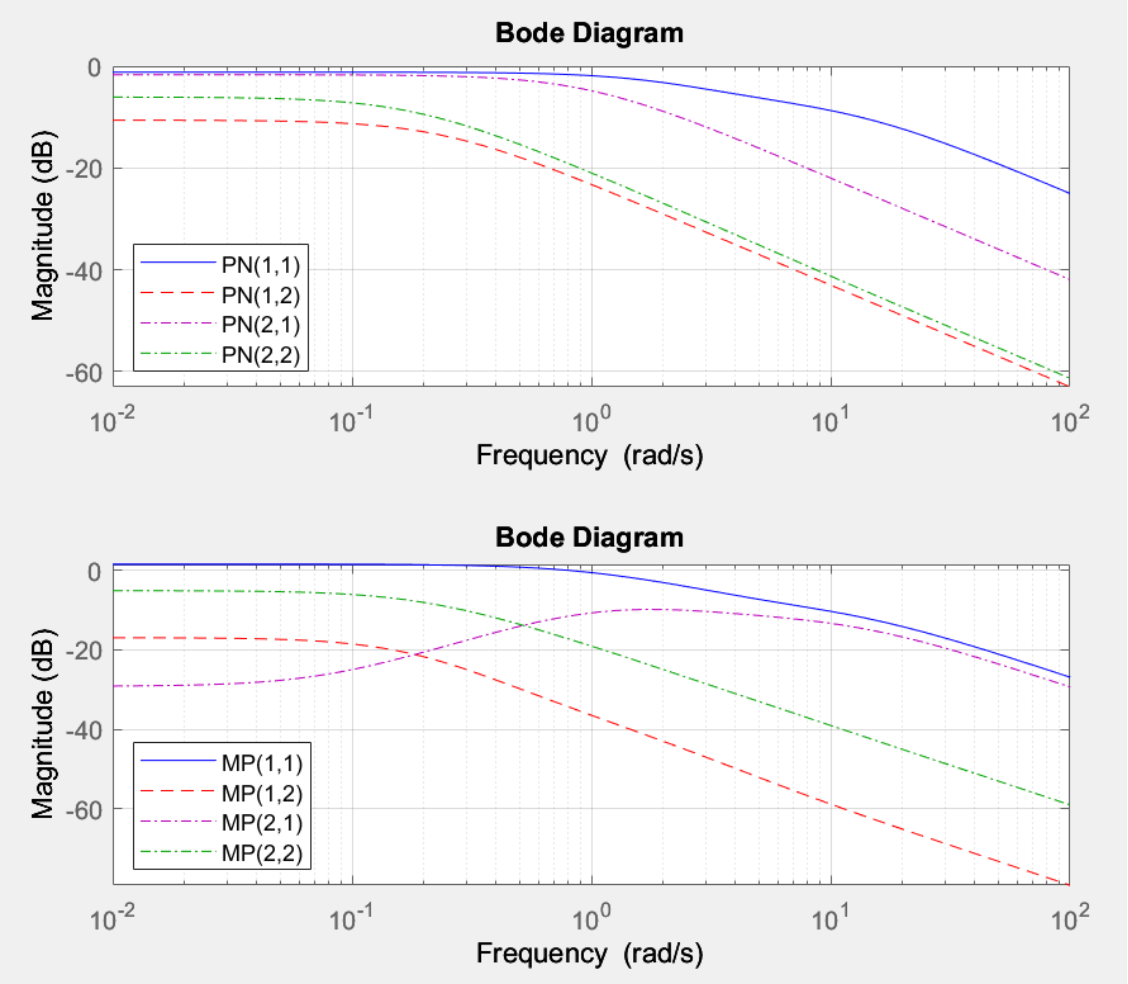


Figure 15. Bode Plots of scaled plant and transformed plant MP.

Part III. Oxygen as an Additional Actuator

(a) Condition Number of the DC Gain Matrix

The condition number of DC gain the the scaled plant is 5.5844, implying that the small control signals will be required to track certain commands. Thus it is possible to use a 3x3 integral control.

(b) DC Analysis with Oxygen Sensor

We defined the weighting matrix as the following:

```
Q1 = diag([1.5,2,1]);
Qw = diag([1,1.5,1]);
Q = blkdiag(Cs'*Q1*Cs, Qw);
R = diag([2,2,1]);
V = Bs*Bs';
W = eye(3);
```

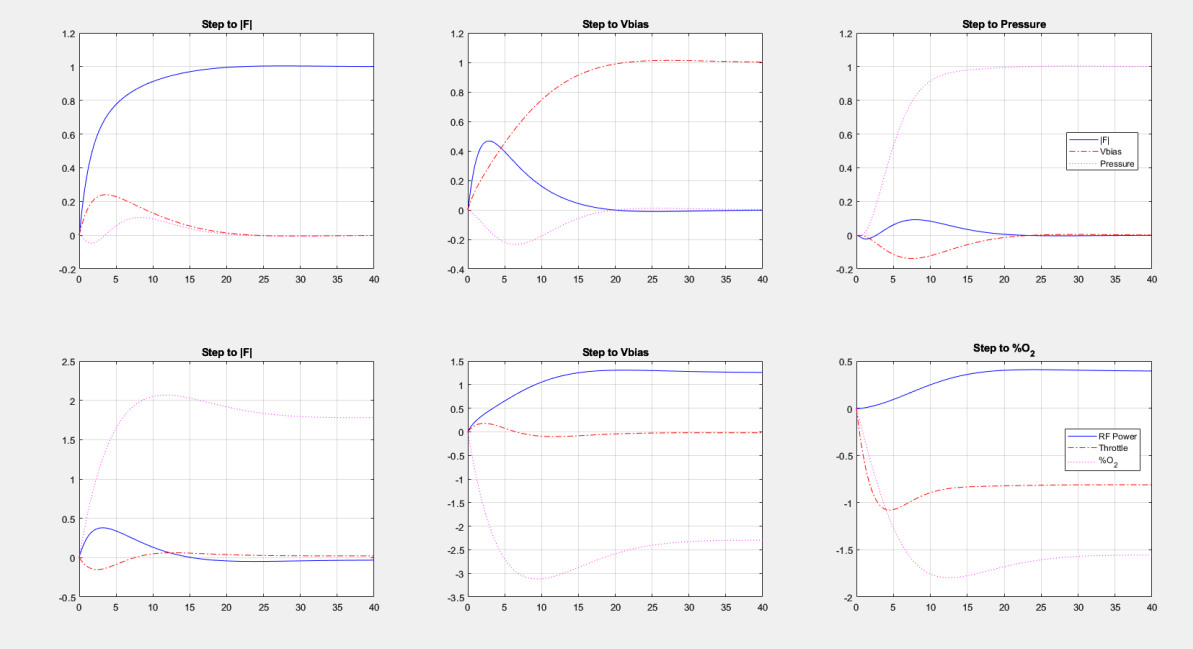


Figure 16: Step Responses with %O2 Added as an Actuator

Contents

Final Project Parts 1 and 2: Reactive Ion Etching with MIMO Control

```
clc
close all
% Plant Model from [Power; Throttle] to [|F|; Vbias]
P1 = [tf([0.17 0.7],[1 15 26.7]); tf(0.28,[1 0.97])];
P2 = [tf(-0.17,[1 0.24]); tf([2.41 9.75],[1 4 0.7])];
P2.InputDelay = 0.5;
P = [P1 P2];

% Normalized System
DO = diag([30 350]);
DI = diag([1000 12.5]);
PN = inv(DO)*P*DI;
PN = ss(PN);

% Use second-order Pade approximation for input delay
PN = pade(PN,2);
PN.InputName = 'u';
PN.OutputName = 'y';

% State-space matrices and dimensions
[AP,BP,CP,DP] = ssdata(PN);
[nx,nu] = size(BP);
ny = size(CP,1);
```

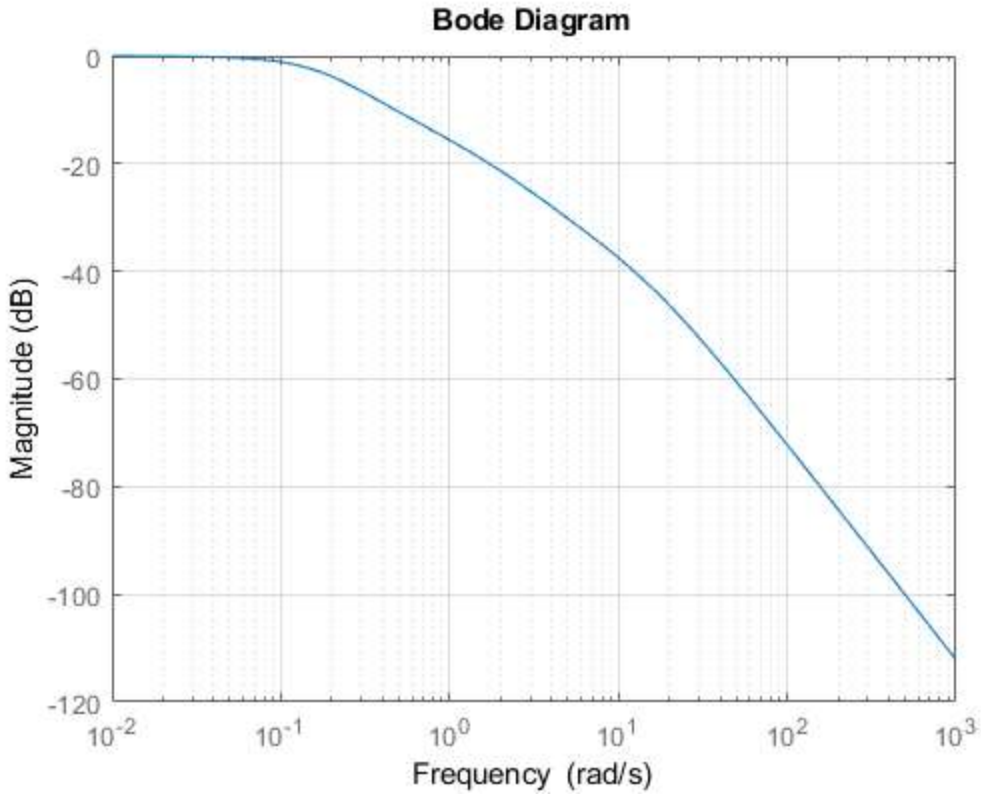
Part 1(A): Linear Quadratic Regulator with Integrators

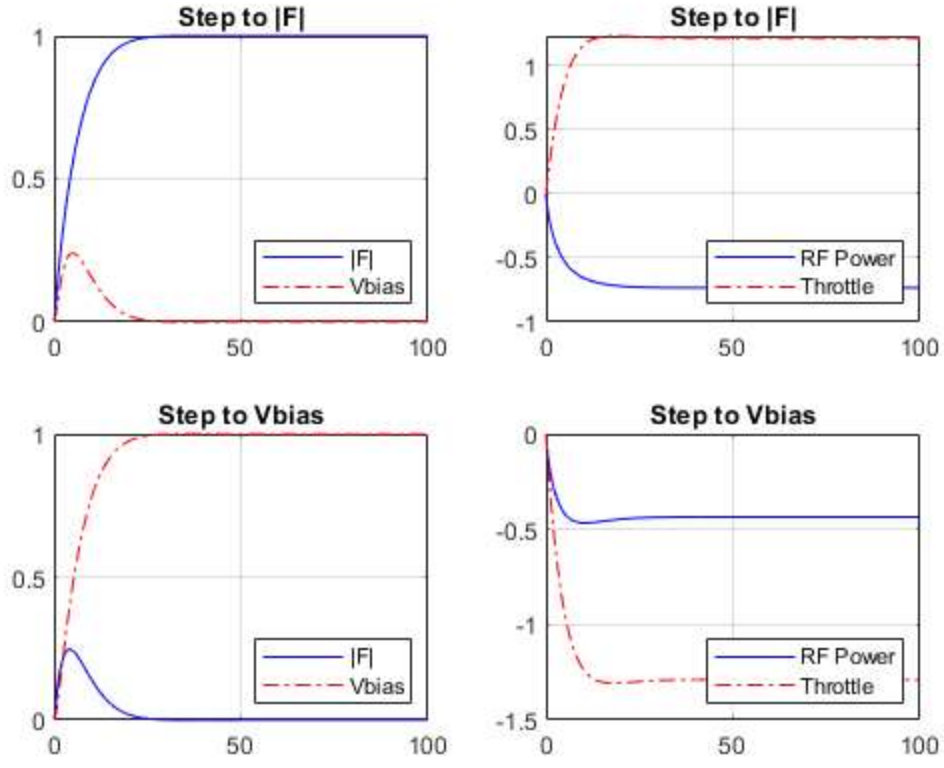
```
close all;
clc;
% Augment state equations so that you can do integral control
Aaug = [AP, zeros(nx,ny);
        CP, zeros(ny,ny)];
Baug = [BP;zeros(ny,nu)];
Caug = [CP,zeros(ny,ny)];
Daug = DP;
% LQR Weighting Matrices

Q1 = [3,0;
      0,3];
Q = CP'*Q1*CP;
Q_I = [1/3,0
       0,1/3];
```

```
Qaug = [Q,zeros(nx,ny);
        zeros(ny,nx),Q_I];
Raug = [1,0;
        0,2];

% Q1 = [1,0;
%       0,1];
% Q = CP'*Q1*CP;
% Q_I = [1,0
%        0,1];
% Qaug = [Q,zeros(nx,ny);
%         zeros(ny,nx),Q_I];
% Raug = eye(2);

% Qaug = eye(10);
% Raug = eye(2);


% LQ state feedback gain
Klqr = lqr(Aaug,Baug,Qaug,Raug);
K = Klqr(1:ny,1:nx);
KI = Klqr(1:ny,nx+1:size(Klqr,2));
% Closed loop state equations with state feedback and integrators.
Acl = Aaug-Baug*Klqr;
Bcl = [zeros(nx,ny);-eye(2)];
Ccl = Caug;
Cks = Klqr;
Dcl = Daug;
% Verify that closed-loop is stable (Check to verify no bugs in code)
T = ss(Acl,Bcl,Ccl,Dcl);
KS = ss(Acl,Bcl,Cks,Dcl);
isstable(T(1,1))
isstable(T(2,2))

% Lsf = ss(Aaug,Baug,Kaug,0);
% Tsf = feedback(Lsf,1);

% Time vector
% Tf = 50;
% Nt = 500;
Tf = 100;
Nt = 1000;
t = linspace(0,Tf,Nt);

% Step responses
figure;
bodemag(T(1,1));grid on;

y11 = step(T(1,1),t);
y21 = step(T(2,1),t);
u11 = step(KS(1,1),t);
u21 = step(KS(2,1),t);
y12 = step(T(1,2),t);
y22 = step(T(2,2),t);
u12 = step(KS(1,2),t);
u22 = step(KS(2,2),t);

figure();
```

```
subplot(2,2,1);
plot(t,y11,'b',t,y21,'r-.');
grid on; xlim([0, Tf])
legend('|F|','Vbias','Location','Southeast');
title('Step to |F|');
if exist('garyfyFigure','file'), garyfyFigure, end

subplot(2,2,2);
plot(t,u11,'b',t,u21,'r-.');
grid on; xlim([0, Tf])
legend('RF Power','Throttle','Location','Southeast');
title('Step to |F|');
if exist('garyfyFigure','file'), garyfyFigure, end

subplot(2,2,3);
plot(t,y12,'b',t,y22,'r-.');
grid on; xlim([0, Tf])
legend('|F|','Vbias','Location','Southeast');
title('Step to Vbias');
if exist('garyfyFigure','file'), garyfyFigure, end

subplot(2,2,4);
plot(t,u12,'b',t,u22,'r-.');
grid on; xlim([0, Tf])
legend('RF Power','Throttle','Location','Southeast');
title('Step to Vbias');
if exist('garyfyFigure','file'), garyfyFigure, end
```

```
ans =

  logical

   1


ans =

  logical

   1
```





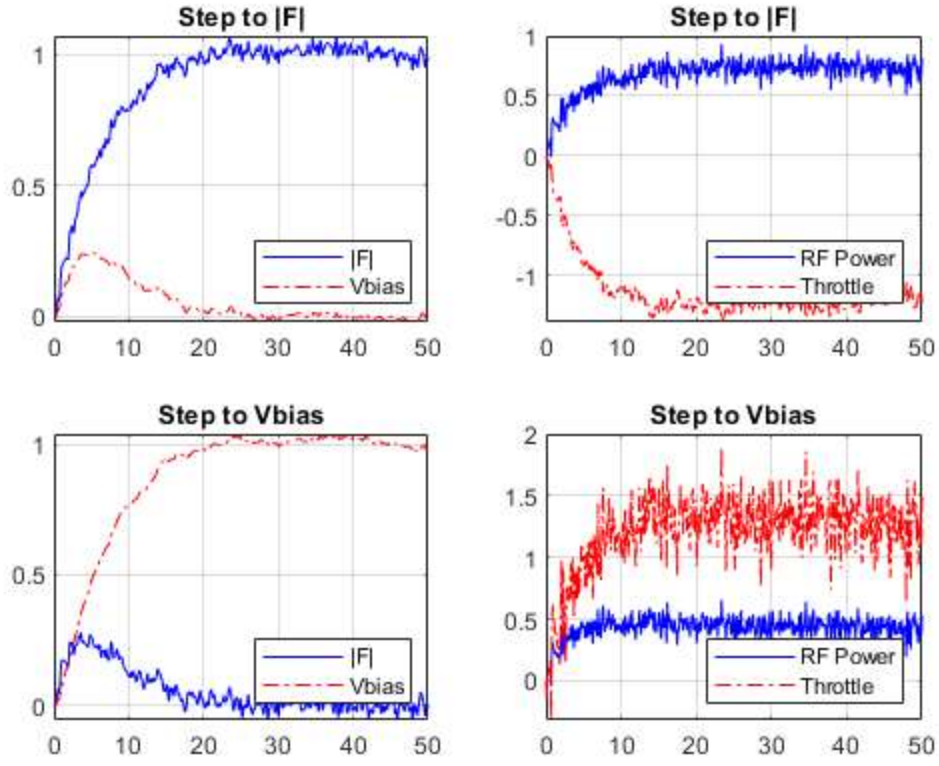
Part 1(B): Linear Quadratic Regulator with Integrators

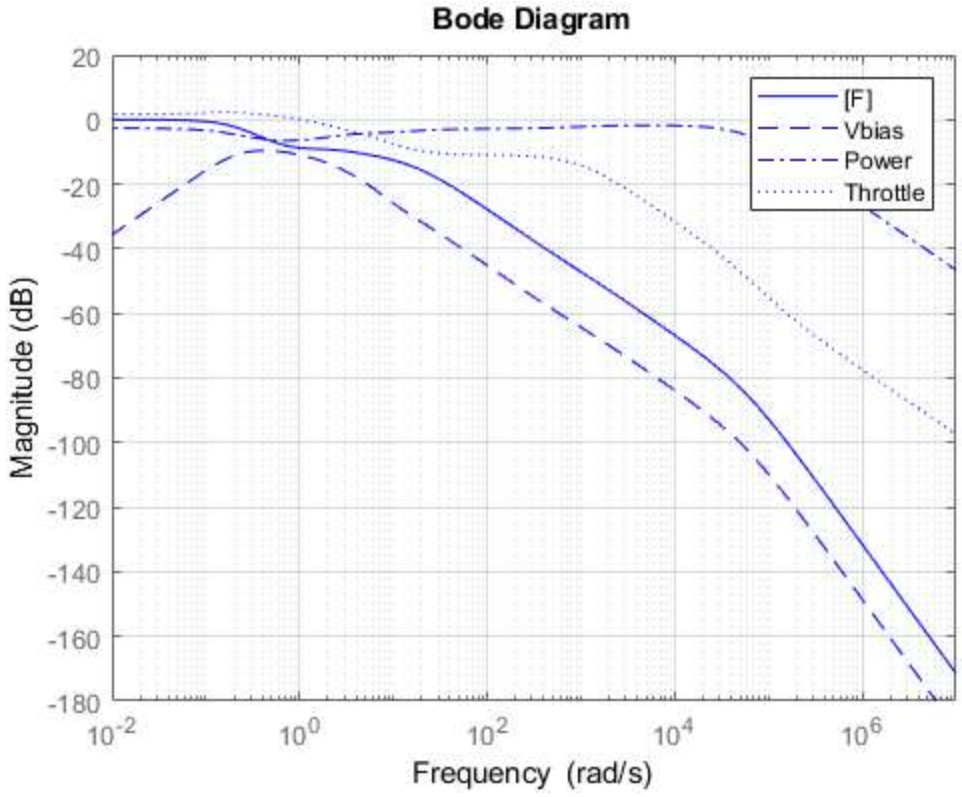
```
close all;
clc;
% Covariance matrices for loop transfer recovery observer
```

```
betavals = [10 100 1000];
Nb = numel(betavals);
% Observer gain
% Note: We only need to estimate the plant states. We do not need the
% observer to construct an estimate of the integrator states.
beta=betavals(1);
W = diag([0.01,0.01]);

L = lqr(AP',CP',1000^2*(BP*BP'),W)';    % 1000^2*(BP*BP'),W

% Verify that observer error is stable (Check to verify no bugs in code)
eig(AP-L*CP);

% Construct controller:
% This includes the observer, integrators, and feedback gains.
%   Inputs: [|F| Ref.; Vbias Ref; |F| measurement; Vbias measurement]
%   Outputs: [Power; Throttle]
Aobs = [AP,-BP*K,-BP*KI;
        zeros(nx,nx),AP-BP*K-L*CP,-BP*KI;
        zeros(ny,nx),zeros(ny,10)]; % CP
Bobs = [zeros(8,2),zeros(8,2);
        zeros(8,2),L;
        -eye(2),eye(2)]; % zeros(2,2)
Cobs = [zeros(2,8),-K,-KI];
Dobs = 0;
Kobs = ss(Aobs,Bobs(:,3:4),Cobs,Dobs);

% Form Closed-Loop
%   Inputs are [|F| Ref.; Vbias Ref; |F| noise; Vbias noise]
%   Outputs: [|F|; Vbias; Power; Throttle]
Aosf = [AP-BP*K,BP*K,-BP*KI;
        zeros(nx,nx),AP-L*CP,zeros(8,2);
        CP,zeros(ny,10)];
Bosf = [zeros(8,2),zeros(8,2);
        zeros(8,2),-L;
        -eye(2),eye(2)]; % zeros(2,2)

Cosf = [CP,zeros(ny,10);
        -K,K,-KI]; % zeros(ny,nx)
Dosf = 0;

Posf = ss(Aosf,Bosf,Cosf,Dosf);

Tosf = Posf(1:2,:);
Kosf = Posf(3:4,:);

% Verify that closed-loop eigenvalues are the union of the observer
% and state-feedback eigenvalues. This is a useful debugging step
% to verify that you have correctly formed the closed-loop.
isstable(Posf)
% Step responses with noise on |F|
Tf = 50;
Nt = 500;
t = linspace(0,Tf,Nt);
stepinput = ones(1,Nt);
stepzeros = zeros(1,Nt);
Fnoise = 0.1*randn(1,Nt);
```

```
y11 = lsim(Tosf(1,1),stepinput',t)+lsim(Tosf(1,3),Fnoise',t);
y21 = lsim(Tosf(2,1),stepinput',t)+lsim(Tosf(2,3),Fnoise',t);
u11 = lsim(Kosf(1,1),stepinput',t)+lsim(Kosf(1,3),Fnoise',t);
u21 = lsim(Kosf(2,1),stepinput',t)+lsim(Kosf(2,3),Fnoise',t);

y12 = lsim(Tosf(1,2),stepinput',t)+lsim(Tosf(1,4),Fnoise',t);
y22 = lsim(Tosf(2,2),stepinput',t)+lsim(Tosf(2,4),Fnoise',t);
u12 = lsim(Kosf(1,2),stepinput',t)+lsim(Kosf(1,4),Fnoise',t);
u22 = lsim(Kosf(2,2),stepinput',t)+lsim(Kosf(2,4),Fnoise',t);

figure();
subplot(2,2,1);
plot(t,y11,'b',t,y21,'r-.');
grid on; xlim([0, Tf])
legend('|F|','Vbias','Location','Southeast');
title('Step to |F|');

subplot(2,2,2);
plot(t,u11,'b',t,u21,'r-.');
grid on; xlim([0, Tf])
legend('RF Power','Throttle','Location','Southeast');
title('Step to |F|');

subplot(2,2,3);
plot(t,y12,'b',t,y22,'r-.');
grid on; xlim([0, Tf])
legend('|F|','Vbias','Location','Southeast');
title('Step to Vbias');

subplot(2,2,4);
plot(t,u12,'b',t,u22,'r-.');
grid on; xlim([0, Tf])
legend('RF Power','Throttle','Location','Southeast');
title('Step to Vbias');

% Bode magnitude from |F| noise to [|F|; Vbias; Power; Throttle]
figure;
bodemag(Posf(1,3),'-');hold on;grid on;
bodemag(Posf(2,3),'--');
bodemag(Posf(3,3),'-.');
bodemag(Posf(4,3),':');hold off;
legend('[F]','Vbias','Power','Throttle');
% Sigma magnitude from [|F| Ref.; Vbias Ref] to [Power; Throttle]

% figure;
% sigma(Kosf(1,1));hold on;grid on;
% sigma(Kosf(1,2));
% sigma(Kosf(2,1));
% sigma(Kosf(2,2));hold off;
% sigma(Kosf);

% Sigma magnitude from [|F| Noise; Vbias Noise] to [Power; Throttle]
```

```
ans =

  logical
```



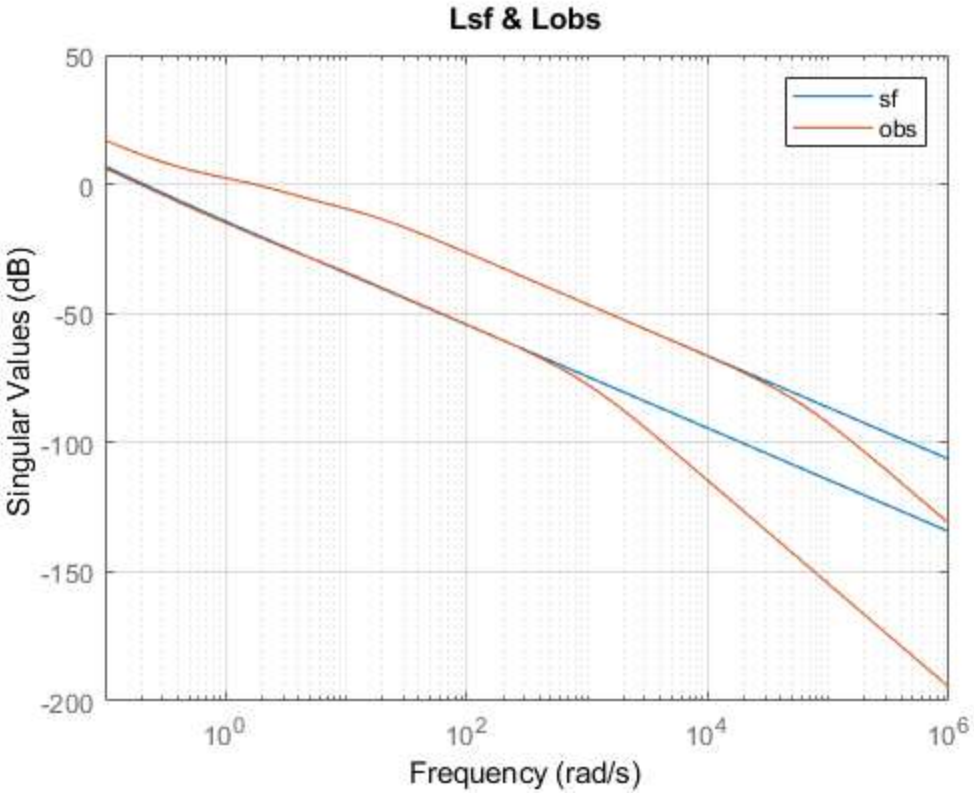


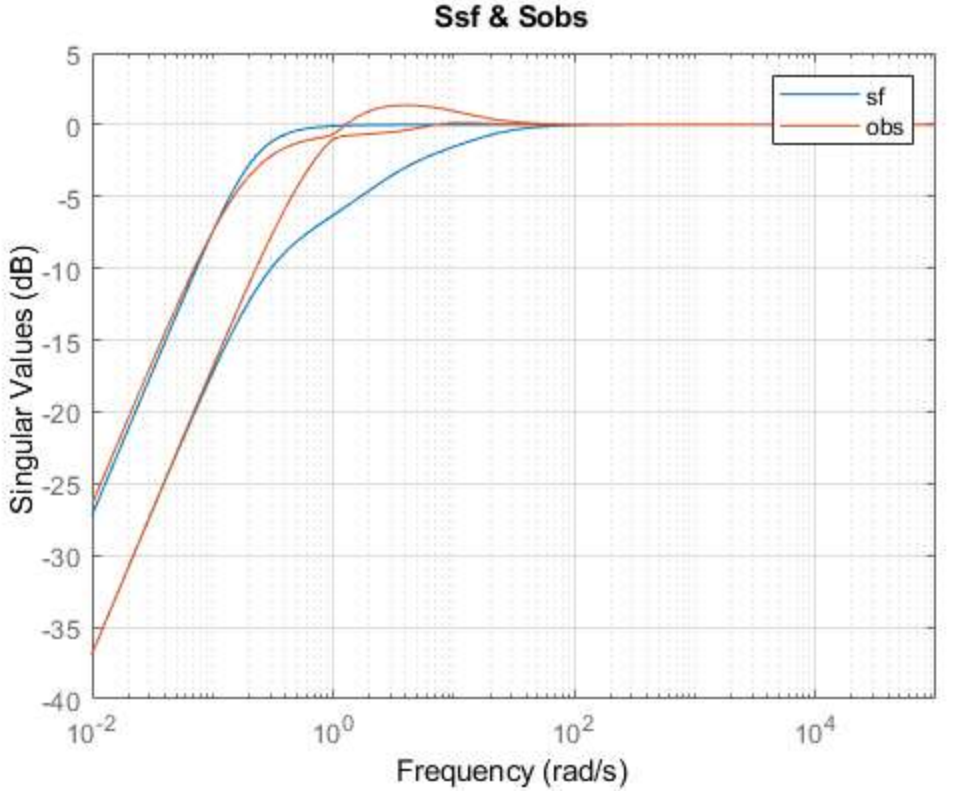
Part 1(D): Stability Margins and Comparison With State Feedback

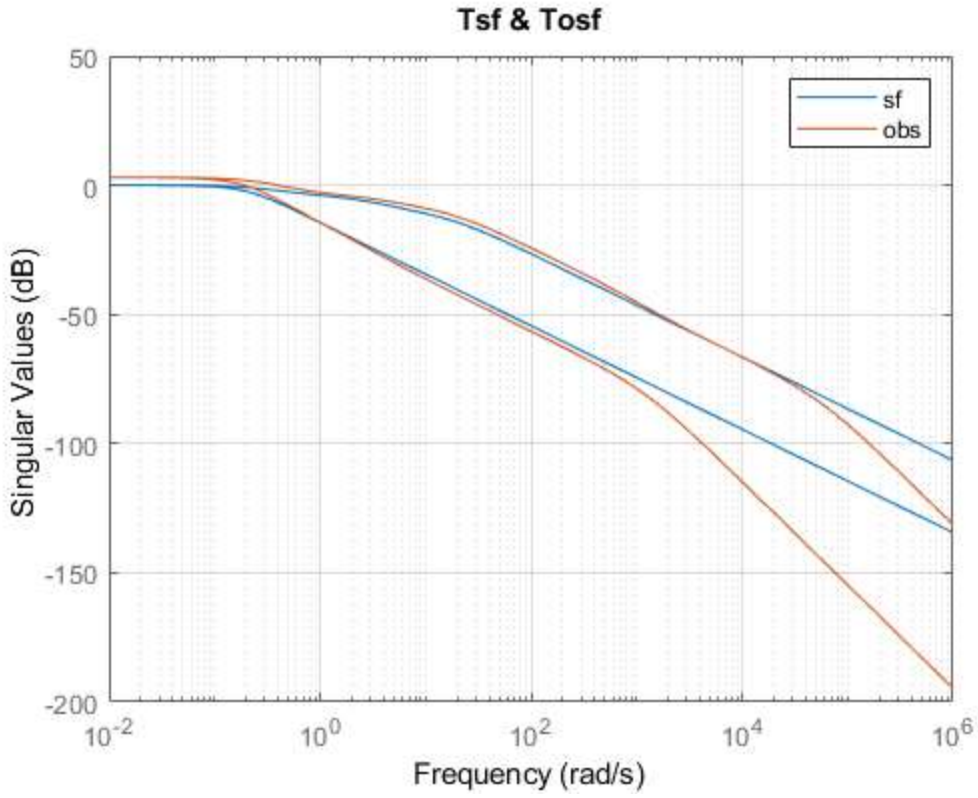
```
clc
close all
Kaug = [K,KI];
Lsf = ss(Aaug,Baug,Kaug,0);
Tsf = feedback(Lsf,eye(2));
Ssf = feedback(eye(2),Lsf);

% ss(AP,BP,CP,DP) ss(Aaug,Baug,Caug,Daug)
Pobs = ss(Aaug,Baug,Caug,Daug) ;
Lobs = Kobs*Pobs;
Tobs = feedback(Lobs,eye(2));
Sobs = feedback(eye(2),Lobs);

% Loop-at-a-time margins at the plant input
AM = allmargin(Lsf);
AM(1)
AM(2)
[DMI, MMI] = diskmargin(Lsf);
DMI(1)
DMI(2)
MMI
% Unstructured (fully-coupled) stability margin (USM) at the plant input
1/norm(Lsf,inf)

% Input loop transfer function: Compare Lsf to Lobs
figure;
sigma(Lsf);hold on;grid on;
sigma(Lobs);
title("Lsf & Lobs");
legend('sf','obs');

% Input sensitivity: Compare Ssf to Sobs
figure;
sigma(Ssf);hold on;grid on;
sigma(Sobs);
title("Ssf & Sobs");
legend('sf','obs');

% Input complementary sensitivity: Compare Tsf to Tosf
figure;
sigma(Tsf);hold on;grid on;
sigma(Tosf);
title("Tsf & Tosf");
legend('sf','obs');
```

```
ans =

  struct with fields:

     GainMargin: [1×0 double]
    GMFrequency: [1×0 double]
    PhaseMargin: 124.9694
    PMFrequency: 1.7044
    DelayMargin: 1.2797
    DMFrequency: 1.7044
         Stable: 1
```

```
ans =

  struct with fields:

     GainMargin: 4.1895e-16
    GMFrequency: 0
    PhaseMargin: 84.5248
    PMFrequency: 0.2101
    DelayMargin: 7.0222
    DMFrequency: 0.2101
         Stable: 1


ans =

  struct with fields:

          GainMargin: [0 Inf]
         PhaseMargin: [-90 90]
          DiskMargin: 2
          LowerBound: 2
          UpperBound: 2
           Frequency: Inf
    WorstPerturbation: [2×2 ss]


ans =

  struct with fields:

          GainMargin: [0.0478 20.9070]
         PhaseMargin: [-84.5232 84.5232]
          DiskMargin: 1.8174
          LowerBound: 1.8174
          UpperBound: 1.8174
           Frequency: 0.2137
    WorstPerturbation: [2×2 ss]


MMI =

  struct with fields:

          GainMargin: [0.0497 20.1340]
         PhaseMargin: [-84.3132 84.3132]
          DiskMargin: 1.8107
          LowerBound: 1.8107
          UpperBound: 1.8144
           Frequency: 0.2222
    WorstPerturbation: [2×2 ss]


ans =

   5.3257e-17
```


Lsf & Lobs
sf
obs
Singular Values (dB)
Frequency (rad/s)
50
0
-50
-100
-150
-200
10^0
10^2
10^4
10^6


Ssf & Sobs
sf
obs
Singular Values (dB)
Frequency (rad/s)
5
0
-5
-10
-15
-20
-25
-30
-35
-40
10^{-2}
10^0
10^2
10^4



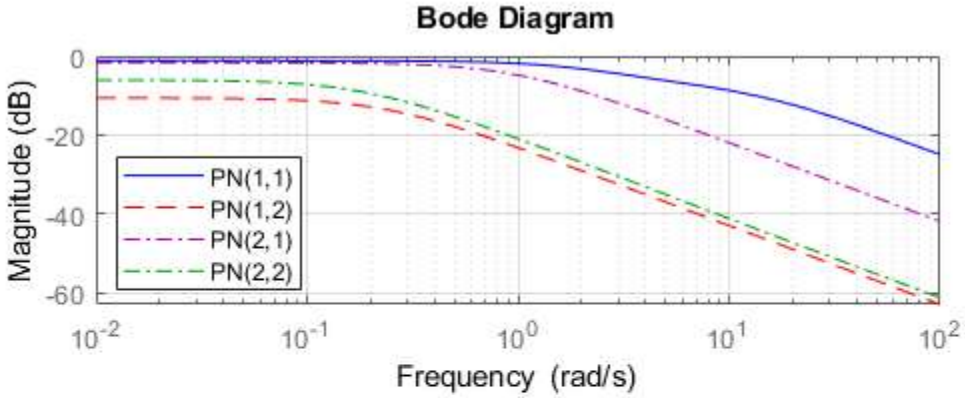
Part 2(B): Equivalent Controller

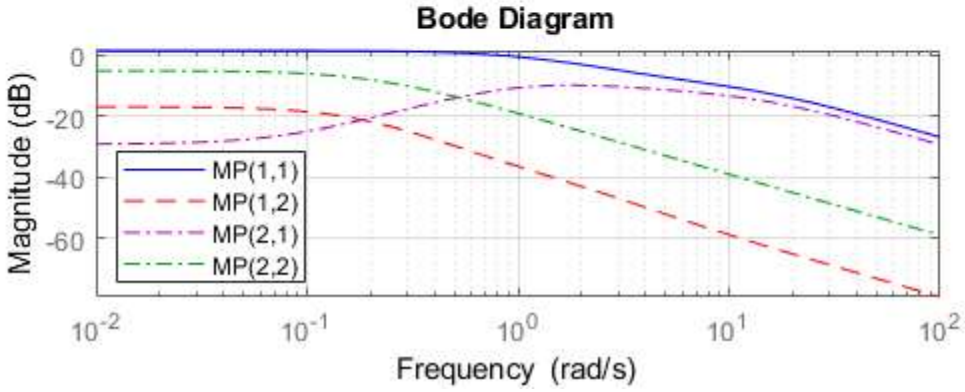
```
% Equivalent controller
%   Ceq = inv[ I+K1 inv(Sobs-A) B] (KI/s)
```

Part 2(C): Decentralized Approximation of Equivalent Controller

Part 2: Comment on Plant Transformation

```
M = [1 1; -1 1]/sqrt(2);
MP = M*PN;

figure(12)
subplot(2,1,1)
bodemag(PN(1,1),'b',PN(1,2),'r--',PN(2,1),'m-.',PN(2,2),'g-.',{1e-2,1e2});
legend('PN(1,1)','PN(1,2)','PN(2,1)','PN(2,2)','Location','Southwest');
grid on;
if exist('garyfyFigure','file'), garyfyFigure, end

subplot(2,1,2)
bodemag(MP(1,1),'b',MP(1,2),'r--',MP(2,1),'m-.',MP(2,2),'g-.',{1e-2,1e2});
legend('MP(1,1)','MP(1,2)','MP(2,1)','MP(2,2)','Location','Southwest');
grid on;
if exist('garyfyFigure','file'), garyfyFigure, end
```





Published with MATLAB® R2022b

Final Project Part 3: Reactive Ion Etching with MIMO Control

Contents

Part 3(A) -- DC Analysis With Oxygen Sensor

```
close all
clc
% Model
% Inputs: [Power; Throttle; %O2]
% Outputs: [|F|; Vbias; Pressure]
P1 = [zpk(-0.067,[-0.095 -19.69],0.49); ...
    zpk(-0.27,[-0.19 -62.42],12.23); ...
    zpk(0.006,[-0.19 -2.33],-0.011)];

P2 = [zpk(0.73,[-0.11; -39.76],4.85); tf(1.65,[1 0.16]); ...
      zpk([],[-0.18; -3],-0.97)];
P2.InputDelay = 0.42;

P3 = [tf(0.33,[1 0.17]); tf(0.25,[1 0.41]); tf(0.024,[1 0.4])];
P3.InputDelay = 0.77;

Pox = [P1 P2 P3];

% Use second-order Pade for plant
Pox = pade(Pox, 2);

% Input and output scalings based on equilibrium values
DO = diag([16.52 340 17.83]);
DI = diag([1000 12.5 5]);
% Normalize Plant
PN = inv(DO)*Pox*DI;
PN = ss(PN);

% Condition number of DC gain
PNDC = dcgain(PN);
[U,sigma,V] = svd(PNDC);
singular_values = diag(sigma);
condition_number = max(singular_values)/min(singular_values)
```

```
condition_number =

    5.5844
```

Part 3(B) -- DC Analysis With Oxygen Sensor

```
clc
close all
% State-space data for scaled plant
```

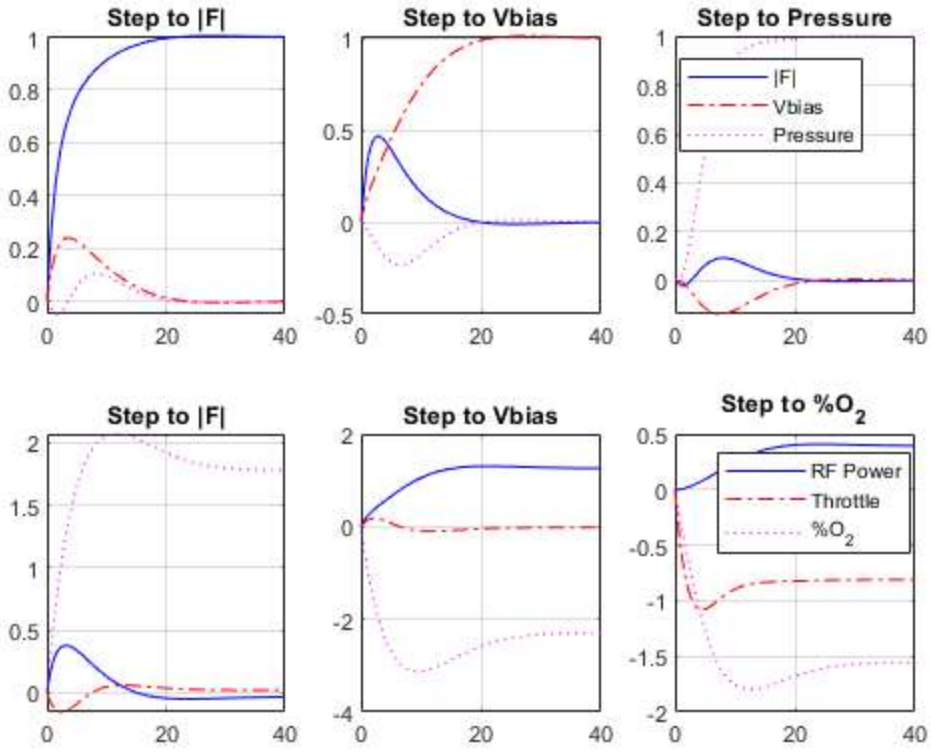
```
[As,Bs,Cs,Ds] = ssdata(PN);
[nx,nu] = size(Bs);
ny = size(Cs,1);

% Weighting matrices (Q,R,V,W)
% Assume Q of the form Q = blkdiag(alpha*Cs'*Cs, Qw)
Q1 = diag([1.5,2,1]);
Qw = diag([1,1.5,1]);
Q = blkdiag(Cs'*Q1*Cs, Qw);
R = diag([2,2,1]);
V = Bs*Bs';
W = eye(3);

% Augmented Plant with integrators
Aaug = [As, zeros(nx,ny);
        Cs, zeros(ny,ny)];
Baug = [Bs; zeros(ny,nu)];
Caug = [Cs, zeros(ny,ny)];
Daug = Ds;

% Compute state feedback and observer gains
K_lqr = lqr(Aaug,Baug,Q,R);
K = K_lqr(1:ny,1:nx);
KI = K_lqr(1:ny,nx+1:size(K_lqr,2));

L = lqr(As',Cs',V, W)';

% Construct controller:
% This includes the observer, integrators, and feedback gains.
%   Inputs: [|F|Ref; Vbias Ref; Press Ref; |F| Meas; Vbias Meas; Press Meas]
%   Outputs: [Power; Throttle; %O2]
Aobs = [As,-Bs*K,-Bs*KI;
        zeros(nx,nx),As-Bs*K-L*Cs,-Bs*KI;
        zeros(ny,55)]; % Cs
Bobs = [zeros(nx,nu),zeros(nx,nu);
        zeros(nx,nu),L;
        -eye(nu),eye(nu)]; % zeros(2,2)
Cobs = [zeros(ny,nx),-K,-KI];
Dobs = 0;
Kobs = ss(Aobs,Bobs(:,3:4),Cobs,Dobs);
% Form Closed-Loop
%   Inputs: [|F|Ref; Vbias Ref; Press Ref; |F| Noise; Vbias Noise; Press Noise]
%   Outputs: [|F|; Vbias; Press; Power; Throttle; %O2]
Aosf = [As-Bs*K,Bs*K,-Bs*KI;
        zeros(nx,nx),As-L*Cs,zeros(nx,nu);
        Cs,zeros(ny,55-26)];
Bosf = [zeros(nx,nu),zeros(nx,nu);
        zeros(nx,nu),-L;
        -eye(nu),eye(nu)]; % zeros(2,2)

Cosf = [Cs,zeros(ny,55-26);
        -K,K,-KI]; % zeros(ny,nx)
Dosf = 0;

Posf = ss(Aosf,Bosf,Cosf,Dosf);

Tobs = Posf(1:3,:);
Kobs = Posf(4:6,:);
```

```
% Verify that closed-loop eigenvalues are the union of the observer
% and state-feedback eigenvalues. This is a useful debugging step
% to verify that you have correctly formed the closed-loop.
sort(eig(Aosf))
union(eig(As-Bs*K),eig(As-L*Cs))
isstable(Posf)
% Time vector
Tf = 40;
Nt = 400;
t = linspace(0,Tf,Nt);
% Step Responses (Without Noise)
y11 = step(Tobs(1,1),t);
y21 = step(Tobs(2,1),t);
y31 = step(Tobs(3,1),t);
u11 = step(Kobs(1,1),t);
u21 = step(Kobs(2,1),t);
u31 = step(Kobs(3,1),t);
y12 = step(Tobs(1,2),t);
y22 = step(Tobs(2,2),t);
y32 = step(Tobs(3,2),t);
u12 = step(Kobs(1,2),t);
u22 = step(Kobs(2,2),t);
u32 = step(Kobs(3,2),t);
y13 = step(Tobs(1,3),t);
y23 = step(Tobs(2,3),t);
y33 = step(Tobs(3,3),t);
u13 = step(Kobs(1,3),t);
u23 = step(Kobs(2,3),t);
u33 = step(Kobs(3,3),t);
figure();
subplot(2,3,1);
plot(t,y11,'b',t,y21,'r-.',t,y31,'m:');
grid on; xlim([0, Tf])
title('Step to |F|');
subplot(2,3,4);
plot(t,u11,'b',t,u21,'r-.',t,u31,'m:');
grid on; xlim([0, Tf])
title('Step to |F|');
subplot(2,3,2);
plot(t,y12,'b',t,y22,'r-.',t,y32,'m:');
grid on; xlim([0, Tf])
title('Step to Vbias');
subplot(2,3,5);
plot(t,u12,'b',t,u22,'r-.',t,u32,'m:');
grid on; xlim([0, Tf])
title('Step to Vbias');
subplot(2,3,3);
plot(t,y13,'b',t,y23,'r-.',t,y33,'m:');
grid on; xlim([0, Tf])
legend('|F|','Vbias','Pressure','Location','best');
title('Step to Pressure');
subplot(2,3,6);
plot(t,u13,'b',t,u23,'r-.',t,u33,'m:');
grid on; xlim([0, Tf])
legend('RF Power','Throttle','%O_2','Location','best');
title('Step to %O_2');
```

```
ans =

  -0.0838 + 0.0000i
  -0.0969 + 0.0000i
  -0.1094 + 0.0000i
  -0.1109 + 0.0000i
  -0.1622 + 0.0000i
  -0.1634 + 0.0000i
  -0.1878 - 0.0070i
  -0.1878 + 0.0070i
  -0.1900 + 0.0000i
  -0.1900 + 0.0000i
  -0.1490 - 0.1266i
  -0.1490 + 0.1266i
  -0.2458 + 0.0000i
  -0.3026 + 0.0000i
  -0.3999 + 0.0000i
  -0.4004 + 0.0000i
  -0.4098 + 0.0000i
  -0.4118 + 0.0000i
  -0.3238 - 0.2708i
  -0.3238 + 0.2708i
  -0.6784 + 0.0000i
  -2.3415 + 0.0000i
  -2.3515 + 0.0000i
  -2.9920 + 0.0000i
  -2.9965 + 0.0000i
  -3.8961 - 2.2494i
  -3.8961 + 2.2494i
  -3.8961 - 2.2494i
  -3.8961 + 2.2494i
  -3.8961 - 2.2494i
  -3.8961 + 2.2494i
  -3.8961 - 2.2494i
  -3.8961 + 2.2494i
  -3.8961 - 2.2494i
  -3.8961 + 2.2494i
  -3.8961 - 2.2494i
  -3.8961 + 2.2494i
  -7.1429 - 4.1239i
  -7.1429 + 4.1239i
  -7.1429 - 4.1239i
  -7.1429 + 4.1239i
  -7.1429 - 4.1239i
  -7.1429 + 4.1239i
  -7.1429 - 4.1239i
  -7.1429 + 4.1239i
  -7.1429 - 4.1239i
  -7.1429 + 4.1239i
  -7.1429 - 4.1239i
  -7.1429 + 4.1239i
 -29.0652 + 0.0000i
 -31.3666 + 0.0000i
 -39.9513 + 0.0000i
 -40.0680 + 0.0000i
 -73.3990 + 0.0000i
 -73.9115 + 0.0000i
```

```
ans =

  -0.0838 + 0.0000i
  -0.0950 + 0.0000i
  -0.1100 + 0.0000i
  -0.1109 + 0.0000i
  -0.1622 + 0.0000i
  -0.1632 + 0.0000i
  -0.1878 - 0.0070i
  -0.1878 + 0.0070i
  -0.1900 + 0.0000i
  -0.1900 + 0.0000i
  -0.3026 + 0.0000i
  -0.3130 + 0.0000i
  -0.3393 - 0.0820i
  -0.3393 + 0.0820i
  -0.4004 + 0.0000i
  -0.4038 + 0.0000i
  -0.4098 + 0.0000i
  -0.8726 + 0.0000i
  -2.3146 + 0.0000i
  -2.3515 + 0.0000i
  -2.7542 + 0.0000i
  -2.9920 + 0.0000i
  -3.8772 - 2.2250i
  -3.8772 + 2.2250i
  -3.8961 - 2.2494i
  -3.8961 + 2.2494i
  -3.8961 - 2.2494i
  -3.8961 + 2.2494i
  -3.8961 - 2.2494i
  -3.8961 + 2.2494i
  -3.8961 - 2.2494i
  -3.8961 + 2.2494i
  -7.1429 - 4.1239i
  -7.1429 + 4.1239i
  -7.1429 - 4.1239i
  -7.1429 + 4.1239i
  -7.1429 - 4.1239i
  -7.1429 + 4.1239i
  -7.1429 - 4.1239i
  -7.1429 + 4.1239i
  -7.1429 - 4.1239i
  -7.1429 + 4.1239i
  -7.1769 - 4.0902i
  -7.1769 + 4.0902i
 -29.4131 + 0.0000i
 -31.3666 + 0.0000i
 -39.8891 + 0.0000i
 -40.0680 + 0.0000i
 -73.7669 + 0.0000i
 -73.9115 + 0.0000i
```

```
ans =

  logical

   1
```



Published with MATLAB® R2022b