

FINAL PROJECT REPORT: Linear Feedback Controls (EECS 565)

PROJECT TITLE: CONTROL SYSTEM DESIGN FOR REACTIVE ION ETCHING (RIE) PROCESS

Student Contribution by: (i) Siddhik Reddy Kurapati

(ii) Atharva Chandokar

1. Multivariable Feedback Controller Design

a)

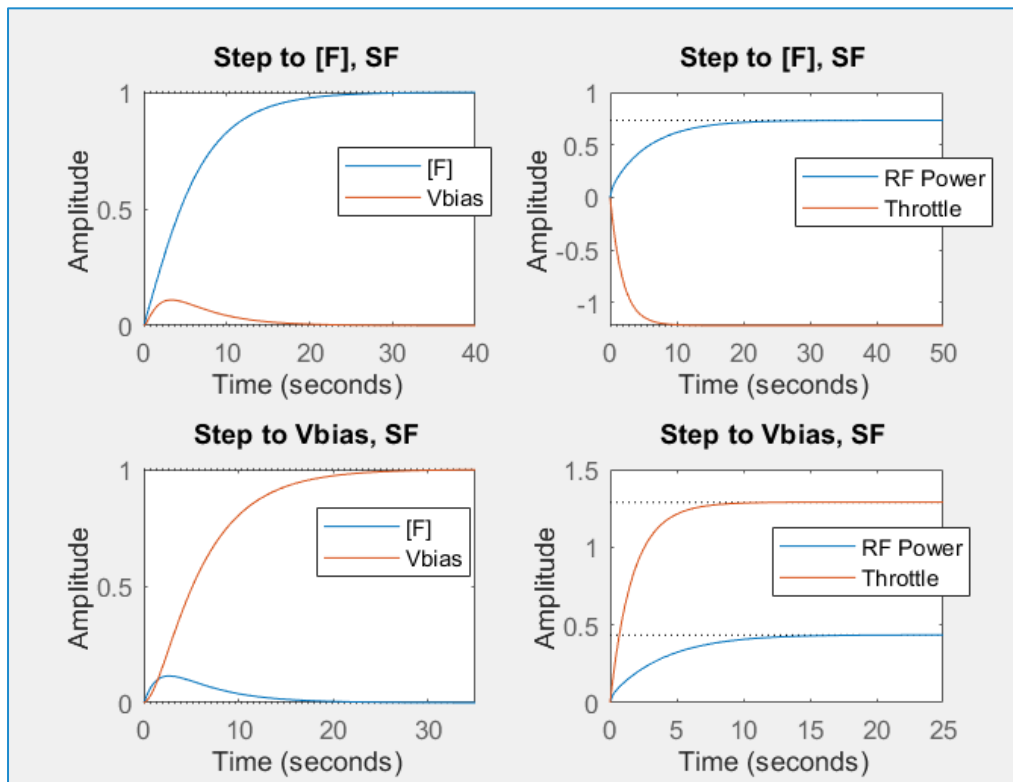


Figure: Step Responses with Multivariable Controller

- Yes, it is possible for us to achieve better compromise among the design goals using a multivariable controller than it was possible for a decentralized controller as shown above.
- We have achieved better step responses than those shown in the Figure(1) of Problem PDF uploaded in Canvas. Our design results in much better settling time, faster response and desired steady state response behavior followed by fulfilling our other design performance specifications.

```

25 %% Part 1(A): Linear Quadratic Regulator with Integrators
26
27 % Augment state equations so that you can do integral control
28 Aaug = [AP zeros(nx,ny);
29         CP zeros(ny,ny)];
30 Baug = [BP;
31         zeros(ny,nu)];
32 Caug = [CP zeros(ny,ny)];
33 %%
34 % LQR Weighting Matrices
35 Q1 = diag([2.5,2.5]);
36 Qi = diag([0.1,0.1]);
37 Q = CP'*Q1*CP;
38 R = diag([0.1,0.1]);
39 % LQ state feedback gain
40 K = lqr(Aaug,Baug,[Q zeros(nx,nu); zeros(nu,nx) Qi],R);
41 %%
42 % Closed loop state equations with state feedback and integrators.
43 sys = ss(Aaug-Baug*K,[zeros(nx,nu); -eye(2)],Caug,0);
44 sys1 = ss(Aaug-Baug*K,[zeros(nx,nu); -eye(2)],-K,0);
45 % Verify that closed-loop is stable (Check to verify no bugs in code)
46 isstable(sys);
47 % Time vector
48 Tf = 50;
49 Nt = 500;
50 t = linspace(0,Tf,Nt);
51
52 % Step responses
53 trans = tf(sys);
54 trans_1 = tf(sys1);
55
56 figure(1)
57 % set(findall(gcf,'type','line'),'linewidth',3);
58 subplot(221)
59 step(trans(1,1))
60 hold on
61 step(trans(2,1))
62 hold on
63 title('Step to [F], SF')
64 legend('[F]', 'Vbias')
65
66 subplot(222)
67 step(trans_1(1,1))
68 hold on
69 step(trans_1(2,1))
70 hold on
71 title('Step to [F], SF')
72 legend('RF Power', 'Throttle')
73
74 subplot(223)
75 step(trans(1,2))
76 hold on
77 step(trans(2,2))
78 hold on
79 title('Step to Vbias, SF')
80 legend('[F]', 'Vbias')
81
82 subplot(224)
83 step(trans_1(1,2))
84 hold on
85 step(trans_1(2,2))
86 hold on
87 title('Step to Vbias, SF')
88 legend('RF Power', 'Throttle')

```

b)

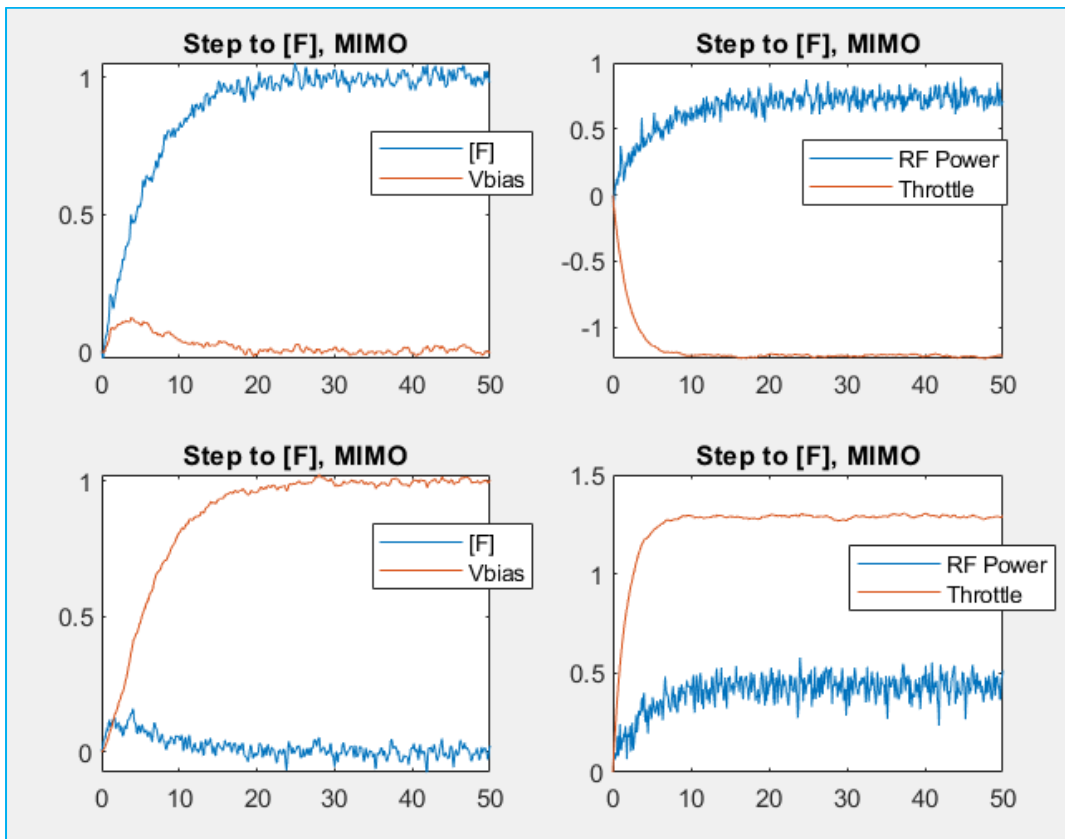


Figure : Step Responses with Multivariable Controller and [F] Sensor Noise.

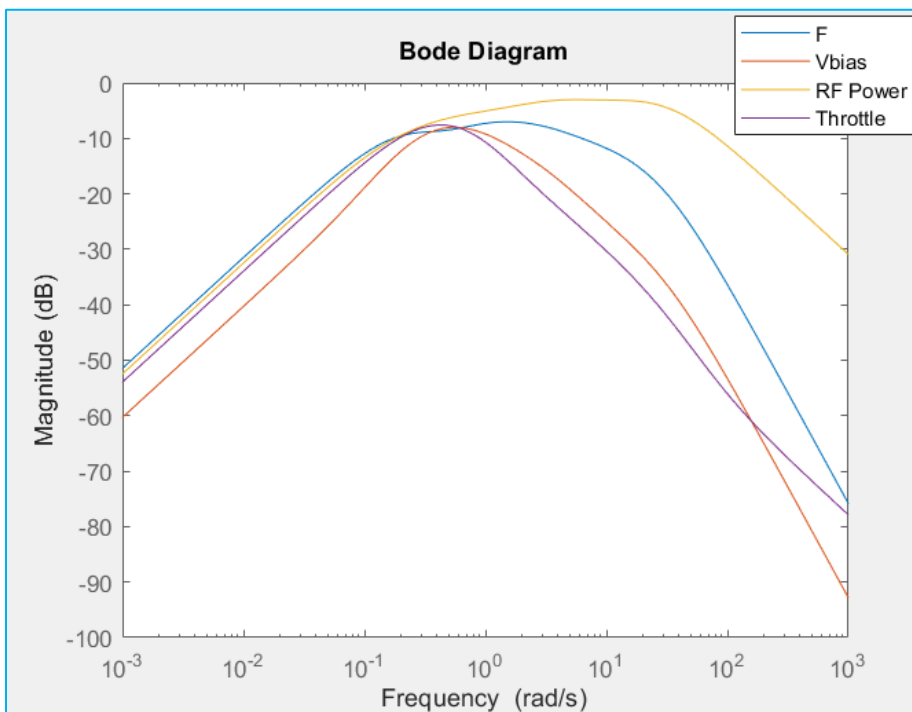


Figure : Bode Plots of the Closed Loop Transfer Functions from [F] Noise to [F], Vbias, Power, and Throttle

- Yes, as 'q' is increasing, there is an increased response to sensor noise. For our design, the largest value of 'q' that is consistent with satisfying the required specification imposed on the response to [F] noise is 2.5 .
- Our design had better step response and Step to [F] for F response, the settling time is around 15.9 secs (corresponding to 1.a graph subplot 1).

NOTE:

One important thing to note is since our step response is faster than that of graph given in the Problem PDF uploaded in Canvas, We have achieved a 'q' value of around 2.5. But if we slow down our step responses then the value of 'q' increases. When it is settling for step to [F] for F response is around 22 secs, the max value of q was determined to be around 9.5.

c)

- No, it is not possible to achieve arbitrarily good recovery of the state feedback loop by tuning the observer only. For our design problem, the plant has a "time delay" similar to the presence of RHP Zeros. Due to the time delay in the plant, the loop bandwidth will be restricted to a certain upper limit.
- So, if the upper limit for bandwidth (loop cross-over frequency) corresponding to the time delay is exceeded, tuning the observer gain will not help to recover the state feedback margins and the loop.
- To achieve the loop transfer recovery of the state feedback loop, we can attempt to increase the penalty on the control input matrix, 'R', This results in smaller feedback gains which corresponds to a small bandwidth.
- So, while designing the state feedback loop, if the loop bandwidth is chosen to be at least reasonably low compared to the time delay, the LTR can converge to the state feedback loop.
- Hence, when there is a time delay (RHP Zeros), it is possible to apply LTR to recover the margins even though the theorem will not hold technically, as long as we design the loop crossover that is consistent with the fundamental limits of Time Delay (RHP Zeroes) in the plant.

d)

Stability Margins:

(i) Loop-at-a-time margins at the plant input:

For state feedback:

Variables - AM_sf							
AM_sf							
2x1 struct with 7 fields							
Fields	GainMargin	GMFrequency	PhaseMargin	PMFrequency	DelayMargin	DMFrequency	Stable
1	3.2716e-17	0	115.4029	17.2177	0.1170	17.2177	1
2	4.4428e-16	0	90.0588	0.5718	2.7489	0.5718	1

For Recovered with observer:

Variables - AM								
AM								
2x1 struct with 7 fields								
Fields	GainMargin	GMFrequency	PhaseMargin	PMFrequency	DelayMargin	DMFrequency	Stable	
1	[3.6864e-15,Inf]	[0,Inf]	127.5615	1.5875	1.4025	1.5875	1	
2	[3.9723e-17,10.5003,Inf]	[0,1.2931,Inf]	74.1737	0.1911	6.7728	0.1911	1	

(ii) Multi-Loop Disk margins at the plant input:

For Symmetric or Regular disk margin:

For State Feedback:

```
MMI_sfs =  
  
    struct with fields:  
  
        GainMargin: [0.0044 228.4483]  
        PhaseMargin: [-89.4984 89.4984]  
        DiskMargin: 1.9826  
        LowerBound: 1.9826  
        UpperBound: 1.9867  
        Frequency: 1.8336  
        WorstPerturbation: [2x2 ss]
```

For Recovered with observer:

```
MMI_s =  
  
    struct with fields:  
  
        GainMargin: [0.2159 4.6323]  
        PhaseMargin: [-65.6364 65.6364]  
        DiskMargin: 1.2898  
        LowerBound: 1.2898  
        UpperBound: 1.2924  
        Frequency: 0.4328  
        WorstPerturbation: [2x2 ss]
```

For T-based disk margin: ("T-based" disk, i.e. a disk centered at 1)

For State Feedback:

```
>> MMI_sf

MMI_sf =

    struct with fields:

        GainMargin: [0.0020 1.9980]
        PhaseMargin: [-59.8650 59.8650]
        DiskMargin: 0.9980
        LowerBound: 0.9980
        UpperBound: 1
        Frequency: 0
        WorstPerturbation: [2×2 ss]
```

For Recovered with observer:

```
>> MMI

MMI =

    struct with fields:

        GainMargin: [0.0020 1.9980]
        PhaseMargin: [-59.8650 59.8650]
        DiskMargin: 0.9980
        LowerBound: 0.9980
        UpperBound: 1.0000
        Frequency: 0
        WorstPerturbation: [2×2 ss]
```

(iii) Unstructured (fully-coupled) stability margin (USM) at the plant input:

For state feedback:

```
>> StabMarg_Tsf

StabMarg_Tsf =

    1.0000
```

For Recovered with observer:

```
>> StabMarg_TI

StabMarg_TI =

    0.9994
```

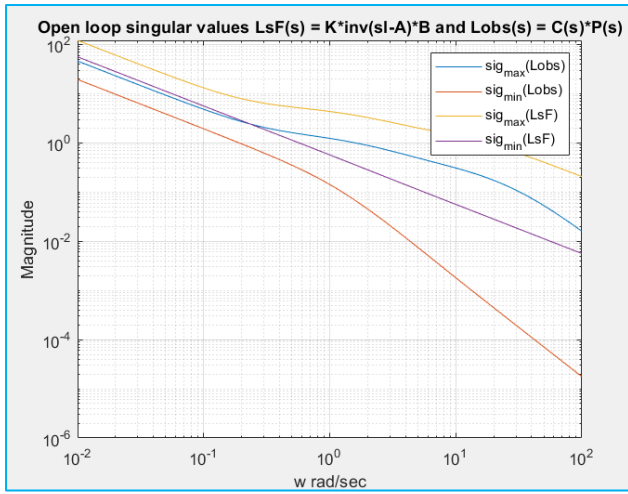


Figure: The State Feedback Loop Transfer Function, Lsf and the Input Loop Transfer Function, LI

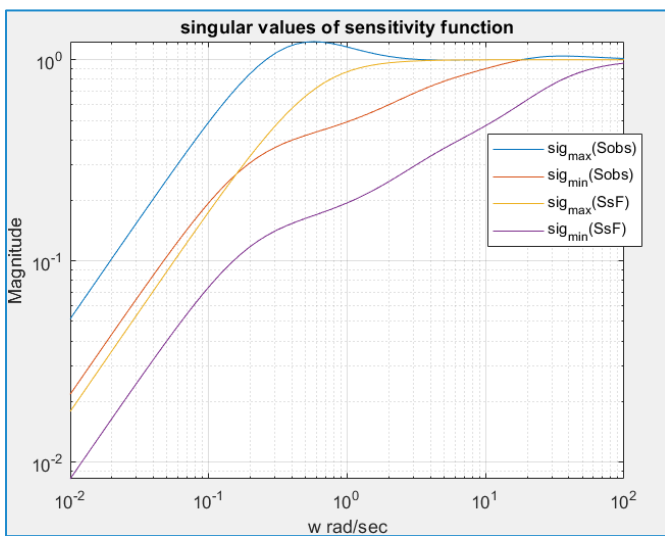


Figure: The State Feedback Sensitivity Function, Ssf and the Input Sensitivity Function, SI

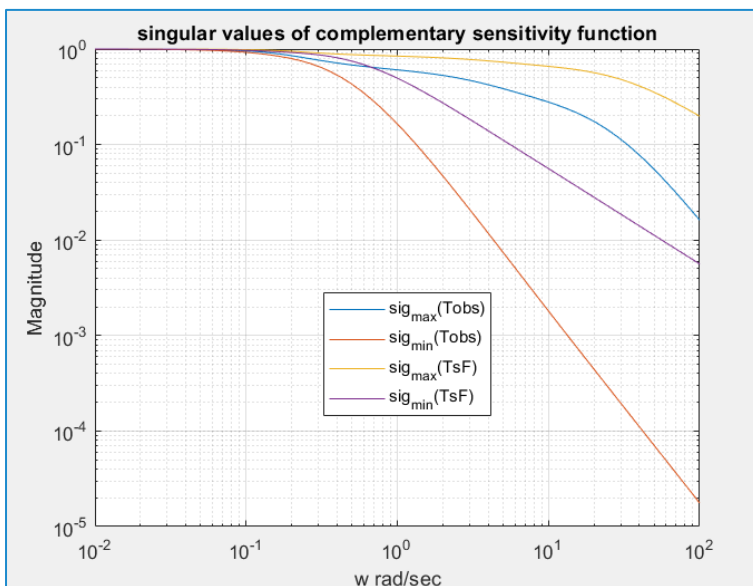


Figure: The State Feedback Complementary Sensitivity Function, Tsf and the Input Complementary Sensitivity Function, TI

```

196 %% Part 1(D): Stability Margins and Comparison With State Feedback
197
198 % Loop-at-a-time margins at the plant input
199
200 % for state feedback
201 Lsf = ss(Aaug,Baug,K,[])
202 AM_sf = allmargin(Lsf)
203
204 AM_sf(1)%For the first channel
205 AM_sf(2)%For the second channel
206
207 %recovered
208 LI = (Cobs * PN);
209
210 AM = allmargin (LI)
211 AM(1)%For the first channel
212 AM(2)%For the second channel
213
214

```

```

215 % Multi-Loop Disk margins at the plant input
216
217 % Regular Disk Margin (symmetric)
218 % for state feedback
219 [DMI_sfs,MMI_sfs] = diskmargin(Lsf);
220 MMI_sfs
221
222 % for recovered
223 [DMI_s,MMI_s] = diskmargin(Cobs*PN); % T-based disk margin
224 MMI_s
225
226 % For T-based Disk margin ("T-based" disk, i.e. a disk centered at 1 )
227 % for state feedback
228 [DMI_sf,MMI_sf] = diskmargin(Lsf,-1); % T-based disk margin
229 MMI_sf
230
231 % for recovered
232 [DMI,MMI] = diskmargin(Cobs*PN,-1); % T-based disk margin
233 MMI
234
235

```

```

236 % Unstructured (fully-coupled) stability margin (USM) at the plant input
237
238 %For State Feedback
239 Tsfc = feedback(Lsf,eye(size(Lsf)));
240 [nsf,wsf] = hinfnorm(Tsfc);
241 StabMarg_Tsf = 1/nsf % unstructured stability margin
242
243 %For Recovered
244 TI = feedback(LI,eye(size(LI)));
245 [np,wp] = hinfnorm(TI);
246 StabMarg_TI = 1/np % unstructured stability margin
247
248 % Input loop transfer function: Compare Lsf to LI
249 Lsf = ss(Aaug,Baug,K,[])
250 Tsfc = feedback(Lsf,eye(size(Lsf)));
251 TI = feedback(LI,eye(size(LI)));

```



```

223 figure (5) % for Loop Gain
224 [sv,wout]=sigma(LI,{1e-2, 1e2});
225 loglog(wout,sv(1,:),wout,sv(2,:))
226 grid on
227 hold on
228 [sv,wout]=sigma(Lsf,{1e-2, 1e2});
229 loglog(wout,sv(1,:),wout,sv(2,:))
230 hold off
231 title('Open loop singular values LsF(s) = K*inv(sI-A)*B and Lobs(s) = C(s)*P(s)')
232 legend('sig_{max}(Lobs)', 'sig_{min}(Lobs)', 'sig_{max}(LsF)', 'sig_{min}(LsF)')
233 xlabel('w rad/sec')
234 ylabel('Magnitude')
235
236 figure(6) % for Complimentary sensitivity
237 [sv,wout]=sigma(TI,{1e-2, 1e2});
238 loglog(wout,sv(1,:),wout,sv(2,:))
239 grid on
240 hold on
241 [sv,wout]=sigma(Tsf,{1e-2, 1e2});
242 loglog(wout,sv(1,:),wout,sv(2,:))
243 hold off
244 title('singular values of complementary sensitivity function')
245 legend('sig_{max}(Tobs)', 'sig_{min}(Tobs)', 'sig_{max}(Tsf)', 'sig_{min}(Tsf)')
246 xlabel('w rad/sec')
247 ylabel('Magnitude')
248 Ssf = feedback(eye(size(Lsf)),Lsf);
249 SI = feedback(eye(size(LI)),LI);
250
251 figure (7) % for Sensitivity
252 [sv,wout]=sigma(SI,{1e-2, 1e2});
253 loglog(wout,sv(1,:),wout,sv(2,:))
254 grid on
255 hold on
256 [sv,wout]=sigma(Ssf,{1e-2, 1e2});
257 loglog(wout,sv(1,:),wout,sv(2,:))
258 hold off
259 title('singular values of sensitivity function')
260 legend('sig_{max}(Sobs)', 'sig_{min}(Sobs)', 'sig_{max}(Ssf)', 'sig_{min}(Ssf)')
261 xlabel('w rad/sec')
262 ylabel('Magnitude')

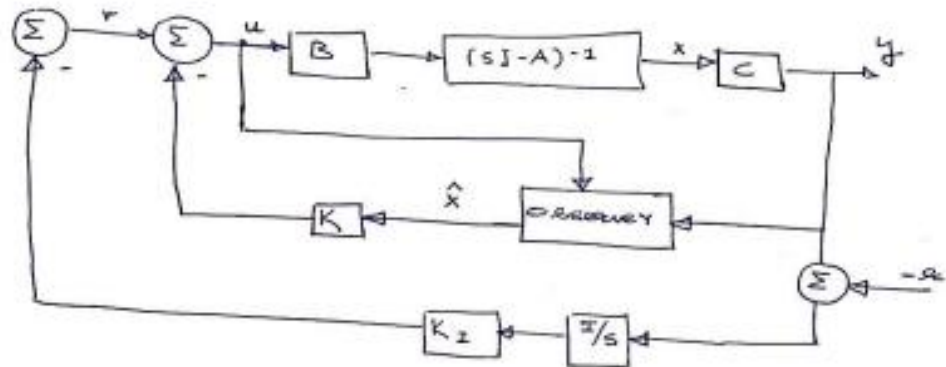
```

- Based on the above values for stability margins for loop-at-a-time, Multi-loop disk margins (symmetric and T-based), and Unstructured margins for both feedback design and recovered design with an observer, we see that the margins for the state feedback design are good, and we are able to recover the margins sufficiently well with an observer as desired.

2) Reverse Engineering the Multivariable Controller

a)

2.]
a.]



→ From the above figure, we observe that there is 'no' disturbance being added to the system.

→ Also, as per linear systems theory, the command response of the system is unaffected by the presence of an observer.

From the above conclusions we can conclude that $x = \hat{x}$.

Given, $P(s) = C(sI - A)^{-1}B$

or

For the system, $\dot{x} = Ax + Bu$, $y = Cx$
 $u = -K\hat{x} - K_1 w$

$$\dot{w}(s) = y - r$$

$$\dot{w} = y - r$$

Taking the Laplace Transform,

$$sW(s) = Y(s) - R(s)$$

$$\Rightarrow W(s) = \frac{1}{s} [Y(s) - R(s)]$$

$$\dot{x} = Ax + Bu$$

Taking the Laplace Transform,

$$sX(s) = AX(s) + BU(s)$$

$$\therefore X(s) = (sI - A)^{-1} BU(s)$$

$$\text{From } U(s) = -KX(s) - K_I W(s)$$

$$U(s) = -K(sI - A)^{-1} B \left\{ U(s) \right\} - \frac{K_I}{s} [Y(s) - R(s)]$$

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

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

$$Y(s) = P(s) \cdot U(s)$$

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

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

$$\therefore \frac{Y(s)}{R(s)} = T_{yr}(s) = \left(I + P(s) (I + K(sI - A)^{-1} B)^{-1} \frac{K_I}{s} \right)^{-1} \cdot P(s) \cdot (I + K(sI - A)^{-1} B)^{-1} \frac{K_I}{s}$$

- From the above formulation derived for the transfer function from the reference input r to the system output y , we can observe that the transfer function does not depend on the observer.
- The primary reason is the absence of any disturbance to the system which may require us to use the observer to estimate the states. We know from linear systems theory, that the command response for the system is unaffected by the presence of an observer, hence we do not need to estimate the states for this case.

b)

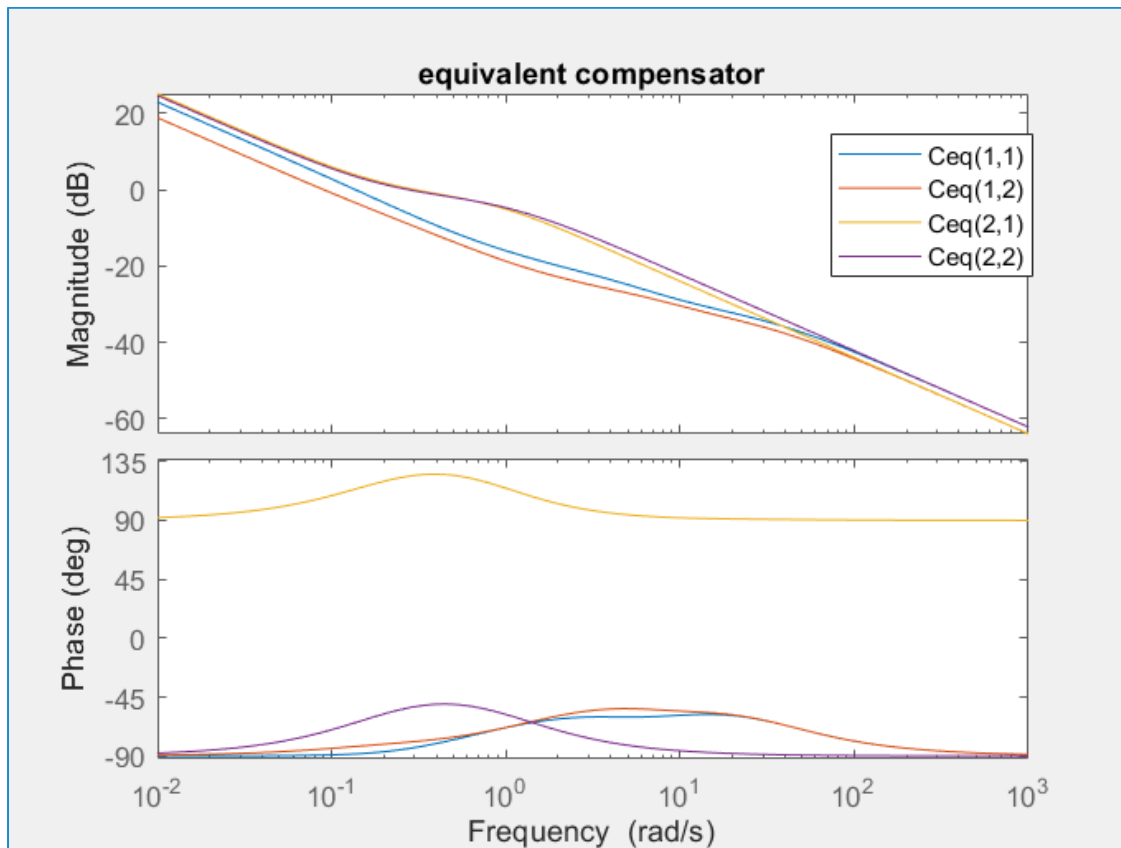


Figure: Equivalent Compensator

- From the above bode plot we can conclude that $Ceq(1,1)$ and $Ceq(1,2)$ are more or less the same (similar ss / tf) with slight differences as the Bode magnitude and phase plot are very similar to each other.
- Also, for $Ceq(2,1)$ and $Ceq(2,2)$'s bode magnitude are similar and from the corresponding phase plot, we can see that $Ceq(2,1) = -Ceq(2,2)$ has a phase of 180 degrees meaning negative complement to each other.

```

251 %% Part 2(B): Equivalent Controller
252
253 % Equivalent controller
254 % Ceq = inv[ I+K1 inv(sI-A) B ] (KI/s)
255 sys = ss(AP,BP,K(:,1:8),0);
256 tr_ = tf(1,[1 0]);
257 % Ceq = tf(inv( eye(2) + K(:,1:8)*inv(s*eye(8) - AP)*BP)*K(:,9:10)/s);
258 Ceq = inv(eye(2) + sys)*K(:,9:10)*tr_;
259 figure(8)
260 bode(Ceq(1,1))
261 hold on
262 bode(Ceq(1,2))
263 hold on
264 bode(Ceq(2,1))
265 hold on
266 bode(Ceq(2,2))
267 hold on
268 legend('Ceq(1,1)', 'Ceq(1,2)', 'Ceq(2,1)', 'Ceq(2,2)')
269 title('equivalent compensator')
270

```

c)

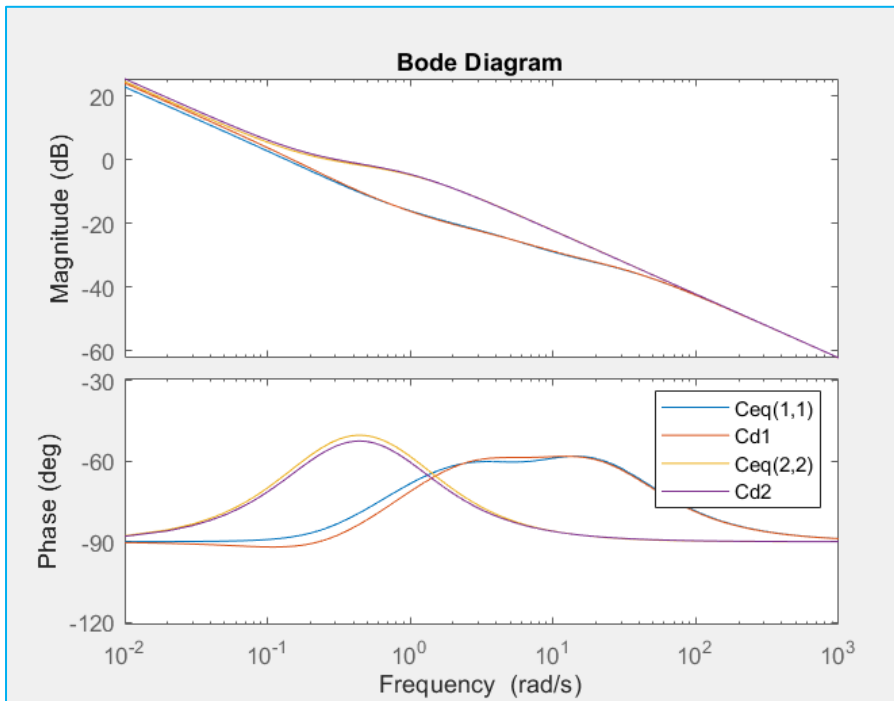


Figure: Bode plots of diagonal elements of Ceq and approximations Cd1 and Cd2

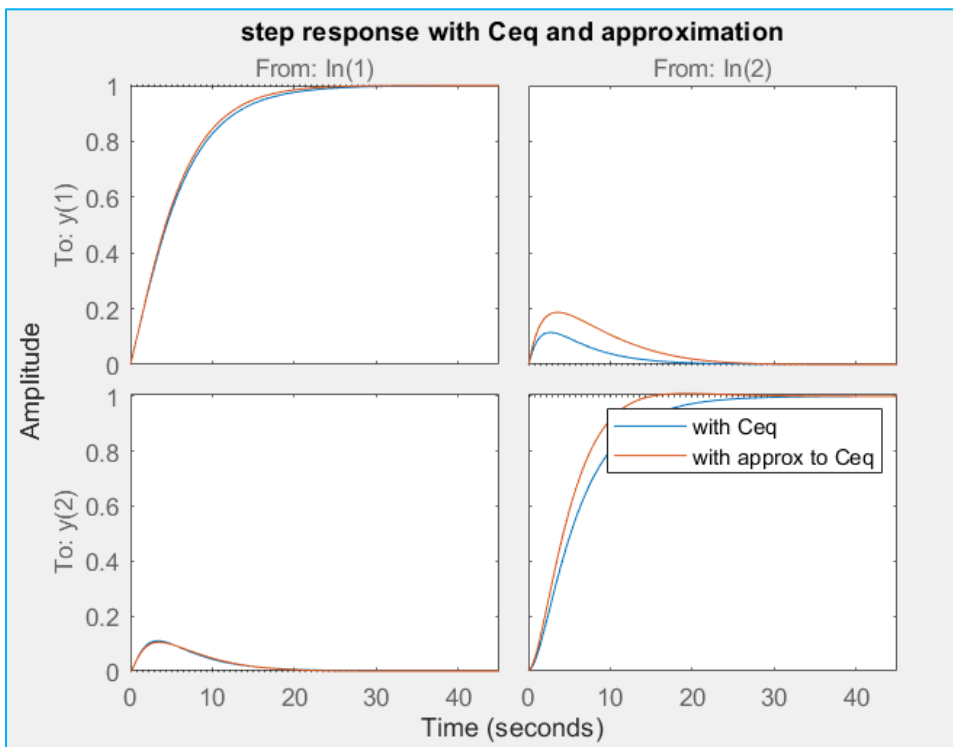


Figure: Step response with Ceq and \hat{C} eq.

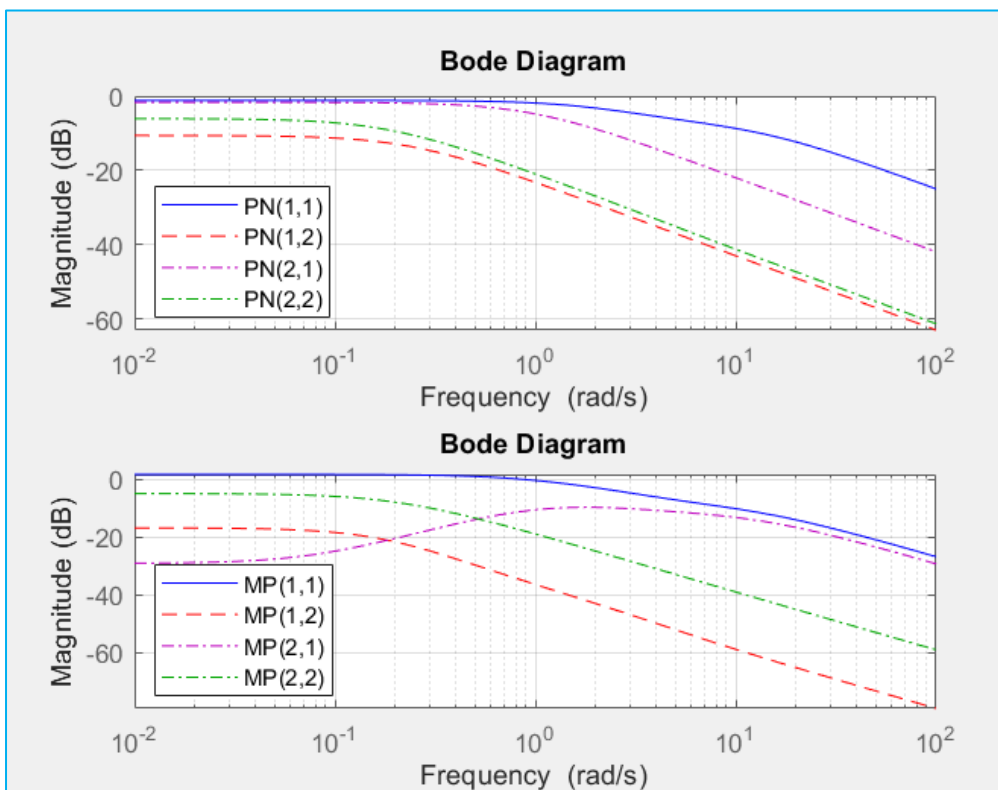
- Based on our design, the Ceq approximation is much better than that shown in the Problem PDF shown in Canvas.
- For our Ceq(1,1) approximation, we have considered a 5th order system (5 poles and 4 zeros).
- For our Ceq(2,2) approximation, we have considered a 3rd order system. This results in a response closer to the actual Ceq.

```

271 %% Part 2(C): Decentralized Approximation of Equivalent Controller
272 s = tf('s');
273 Chateq1 = 0.775*((s+13)*(s+0.25)*(s+0.97)*(s+2)/((s+31)*(s+4.1)*(s+1.2)*s*(s+0.2)));
274 % Chateq2 = 0.632*((s+13)*(s+2)*(s+0.96)*(s+0.97)*(s+0.18)/((s+31)*(s)*(s+4.1)*(s+0.9)*(s+0.21)*(s+1.17)));
275 Chateq2 = 0.775*((s+0.24)*(s+0.19)/((s+0.9)*(s)*(s+0.21)));
276 % Chateq2 = tf([1],[1 0])*tf(0.38*[1 .203],0.203*[1 0.38])*tf([0.1275]);
277 Chateq = [Chateq1 Chateq1; -Chateq2 Chateq2];
278 figure(9)
279 bode(Ceq(1,1),10^-2:10^-2:10^3)
280 hold on
281 bode(Chateq(1,1),10^-2:10^-3:10^3)
282 hold on
283 bode(Ceq(2,2),10^-2:10^-2:10^3)
284 bode(Chateq(2,2),10^-2:10^-2:10^3)
285 hold on
286 legend('Ceq(1,1)', 'Cd1', 'Ceq(2,2)', 'Cd2')
287
288 figure(10)
289 step(feedback(PN*Ceq,eye(2)))
290 hold on
291 step(feedback(PN*Chateq,eye(2)))
292 hold on
293 legend('with Ceq', 'with approx to Ceq')
294 title('step response with Ceq and approximation')

```

Comment on the Plant Transformation



3) Oxygen as an Additional Actuator

a) The conditional number of the DC gain matrix of the scaled plant came around **5.5844**, whereas the conditional number for the plant considered in 1. a came around **2.1595**. The conditional number of the DC gain matrix for the unscaled plant considering three inputs came around **235.9343**. If the conditional number is large, then the plant will be susceptible to disturbances.

```
>> num

num =

    5.5844
```

Since the conditional number is small, as compared to other, it is feasible to use 3 x 3 Integral controller.

```
3 %% Part 3(A) -- DC Analysis With Oxygen Sensor
4
5 % Model
6 % Inputs: [Power; Throttle; %O2]
7 % Outputs: [[F]; Vbias; Pressure]
8 P1 = [zpk(-0.067,[-0.095 -19.69],0.49); ...
9       zpk(-0.27,[-0.19 -62.42],12.23); ...
10      zpk(0.006,[-0.19 -2.33],-0.011)];
11
12 P2 = [zpk(0.73,[-0.11; -39.76],4.85); tf(1.65,[1 0.16]); ...
13       zpk([],[-0.18; -3],-0.97)];
14 P2.InputDelay = 0.42;
15
16 P3 = [tf(0.33,[1 0.17]); tf(0.25,[1 0.41]); tf(0.024,[1 0.4])];
17 P3.InputDelay = 0.77;
18
19 Pox = [P1 P2 P3];
20
21 % Use second-order Pade for plant
22 Pox = pade(Pox, 2);
23
24 % Condition number of DC gain for hot scaled plant
25 Pox_0 = freqresp(Pox,0);
26 num = cond(Pox_0,2)
27
28 % Input and output scalings based on equilibrium values
29 DO = diag([16.52 340 17.83]);
30 DI = diag([1000 12.5 5]);
31
32 % Normalize Plant
33 PoxN = inv(DO)*Pox*DI;
34 PoxN = ss(PoxN);
35 % Condition number of DC gain
36 PoxN_0 = freqresp(PoxN,0);
37 num = cond(PoxN_0,2)
38
```

b)

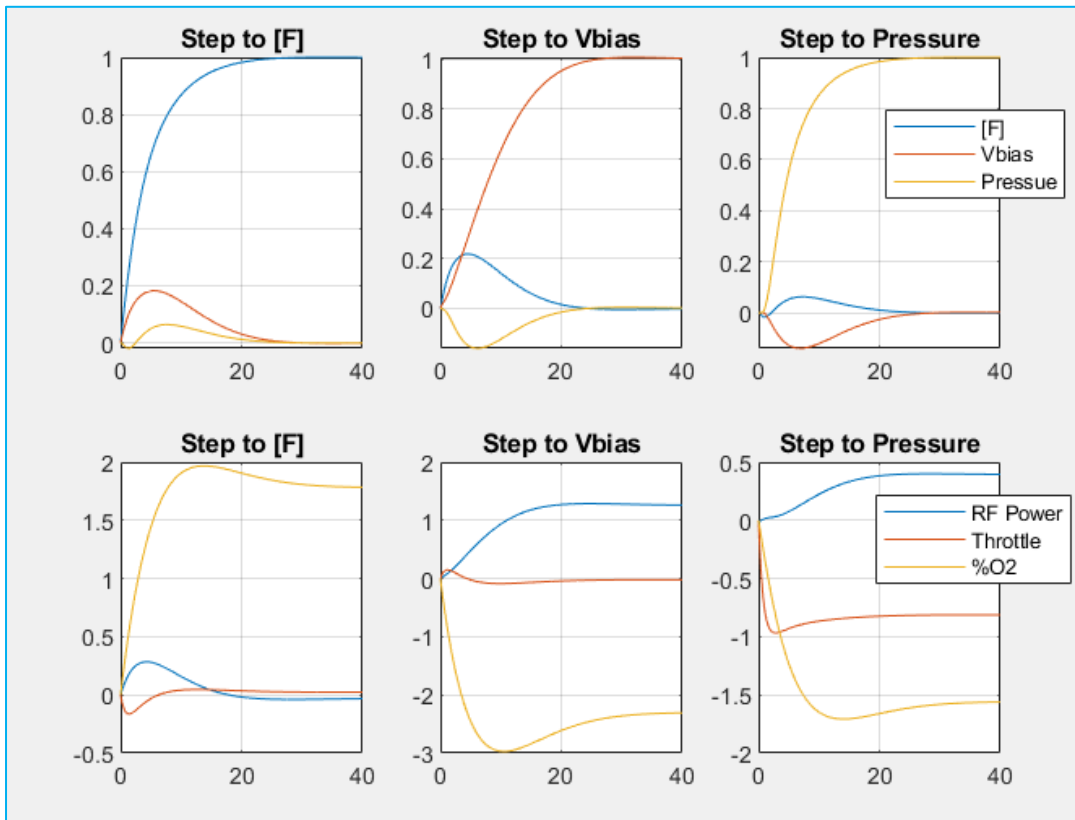


Figure: Step Responses with % O₂ Added as an Actuator

```

39 %% Part 3(B) -- DC Analysis With Oxygen Sensor
40
41 % State-space data for scaled plant
42 [As,Bs,Cs] = ssdata(PoxN);
43 [nx,nu] = size(Bs);
44 ny = size(Cs,1);
45
46 % Weighting matrices (Q,R,V,W)
47 % Assume Q of the form Q = blkdiag(alpha*Cs'*Cs, Qw)
48
49 % Q1 = diag([1,1,1]);
50 % Qi = diag([0.1,0.1,0.1]);
51 % Q = blkdiag(Cs'*Q1*Cs,Qi);
52 % R = diag([1,1,1]);
53
54 Q1 = diag([2.5,2.5,2.5]);
55 Qi = diag([0.2,0.2,0.2]);
56 Q = blkdiag(Cs'*Q1*Cs,Qi);
57 R = diag([0.1,0.1,0.2]);
58
59 % Augmented Plant with integrators
60 Aaug = [As zeros(nx,ny);
61         Cs zeros(ny,ny)];
62 Baug = [Bs;
63         zeros(ny,nu)];
64 Caug = [Cs zeros(ny,ny)];
65
66 % Compute state feedback and observer gains
67 K = lqr(Aaug,Baug,Q,R);
68 q = 3;
69 V = q^2*Baug'*Baug;
70 W = eye(3);
71 G = eye(26);
72 L = lqe(As,G,Cs,V,W);
73
74 sys = ss(Aaug-Baug*K,[zeros(nx,nu); -eye(nu)],Caug,0);
75 sys1 = ss(Aaug-Baug*K,[zeros(nx,nu); -eye(nu)],-K,0);

```



```

140 %%
141 % Construct controller:
142 % This includes the observer, integrators, and feedback gains.
143 % Inputs: [|F|Ref; Vbias Ref; Press Ref; |F| Meas; Vbias Meas; Press Meas]
144 % Outputs: [Power; Throttle; %O2]
145 A_obs = [As -Bs*K(:,1:26) -Bs*K(:,27:29);
146          L*Cs As-Bs*K(:,1:26)-L*Cs -Bs*K(:,27:29);
147          Cs zeros(3,29)];
148 B_obs = [[zeros(52,3); eye(3)] [zeros(26,1); L(:,1); zeros(3,1)] [zeros(26,1); L(:,1); zeros(3,1)]];
149 C_obs= [Cs zeros(size(Cs)) zeros(3,3); zeros(3,26) -K];
150 sys5 = -ss(A_obs,B_obs,C_obs,0);
151 sys6 = ss(A_obs,B_obs,[zeros(3,26) K],0);
152
153 % Form Closed-Loop
154 % Inputs: [|F|Ref; Vbias Ref; Press Ref; |F| Noise; Vbias Noise; Press Noise]
155 % Outputs: [|F|; Vbias; Press; Power; Throttle; %O2]
156 trans = tf(sys5);
157 trans_1 = tf(sys6);
158
159
160 % Verify that closed-loop eigenvalues are the union of the observer
161 % and state-feedback eigenvalues. This is a useful debugging step
162 % to verify that you have correctly formed the closed-loop.
163 eig2 = eig(Aaug-Baug*K);
164 eig3 = eig(AP-L*CP);
165 eig1 = eig(A_obs);
166 isstable(sys5);
167

```

APPENDIX:

Complete MATLAB Code For The Project:

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

```
% 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];
P_0 = freqresp(P,0);
num = cond(P_0,2)
% Normalized System
DO = diag([30 350]);
DI = diag([1000 12.5]);
PN = inv(DO)*P*DI;
PN = ss(PN);
```

```
% Condition number of DC gain
PN_0 = freqresp(PN,0);
num = cond(PN_0,2)
```

```
% 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

% 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)];
```

%%

% LQR Weighting Matrices

```
Q1 = diag([2.5,2.5]);
Qi = diag([0.1,0.1]);
Q = CP'*Q1*CP;
R = diag([0.1,0.1]);
```

% LQ state feedback gain

```
K = lqr(Aaug,Baug,[Q zeros(nx,nu); zeros(nu,nx) Qi],R);
%%
```

% Closed loop state equations with state feedback and integrators.

```
sys = ss(Aaug-Baug*K,[zeros(nx,nu); -eye(2)],Caug,0);
sys1 = ss(Aaug-Baug*K,[zeros(nx,nu); -eye(2)],-K,0);
```

% Verify that closed-loop is stable (Check to verify no bugs in code)

```
isstable(sys);
```

% Time vector

```
Tf = 50;
Nt = 500;
t = linspace(0,Tf,Nt);
```

% Step responses

```
trans = tf(sys);
trans_1 = tf(sys1);
```

```

figure(1)
% set(findall(gcf,'type','line'),'linewidth',3);
subplot(221)
step(trans(1,1))
hold on
step(trans(2,1))
hold on
title('Step to [F], SF')
legend('[F]', 'Vbias')

subplot(222)
step(trans_1(1,1))
hold on
step(trans_1(2,1))
hold on
title('Step to [F], SF')
legend('RF Power', 'Throttle')

subplot(223)
step(trans(1,2))
hold on
step(trans(2,2))
hold on
title('Step to Vbias, SF')
legend('[F]', 'Vbias')

subplot(224)
step(trans_1(1,2))
hold on
step(trans_1(2,2))
hold on
title('Step to Vbias, SF')
legend('RF Power', 'Throttle')
%% Part 1(B): Linear Quadratic Regulator with Integrators

% Covariance matrices for loop transfer recovery observer
q = 2.5;
V = q^2*BP*BP';
W = eye(2);

% 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.
G = eye(8);
L = lqe(AP,G,CP,V,W);
Laug = [L;
        eye(2)];
Cobs = ss([AP-BP*K(:,1:8)-L*CP -BP*K(:,9:10); zeros(2,10)],Laug,K,0);
% sys3 = ss([AP-L*CP-BP*K(:,1:8) -BP*K(:,9:10); zeros(2,8) zeros(2,2)],)
loop_sys3 = Cobs*PN;
sys3 = feedback(loop_sys3,eye(2));
sys4 = tf(Cobs)*feedback(eye(2),loop_sys3);
%%
% Verify that observer error is stable (Check to verify no bugs in code)
eig2 = eig(Aaug-Baug*K);
eig3 = 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]
A_obs = [AP -BP*K(:,1:8) -BP*K(:,9:10);
         L*CP AP-BP*K(:,1:8)-L*CP -BP*K(:,9:10);
         CP zeros(2,10)];

```

```

B_obs = [[zeros(16,2); eye(2)] [zeros(8,1); L(:,1); zeros(2,1)] [zeros(8,1); L(:,1);
zeros(2,1)]];
C_obs= [CP zeros(size(CP)) zeros(2,2); zeros(2,8) -K];

% Form Closed-Loop
% Inputs are [|F| Ref.; Vbias Ref; |F| noise; Vbias noise]
% Outputs: [|F|; Vbias; Power; Throttle]
sys5 = -ss(A_obs,B_obs,C_obs,0);
sys6 = ss(A_obs,B_obs,[zeros(2,8) K],0);

% 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.
eig1 = eig(A_obs);
isstable(sys5);

% trans = tf(sys3);
% trans_1 = tf(sys4);
%%
% Step responses with noise on |F|
noise=0.1*randn(501,1);
ref1 = [ones(size(noise)) zeros(size(noise)) noise zeros(size(noise))];
noise=0.1*randn(501,1);
ref2 = [zeros(size(noise)) ones(size(noise)) zeros(size(noise)) noise];
[y1 t]=lsim(sys5,ref1,[0:0.1:50]);

[y2 t]=lsim(sys5,ref2,[0:0.1:50]);

figure(1)
subplot(221)
plot(t,y1(:,1),t,y1(:,2))
hold on
title('Step to |F|, MIMO')
legend('|F|','Vbias')
xlim([0 50])

subplot(222)
plot(t,y1(:,3),t,y1(:,4))
hold on
title('Step to |F|, MIMO')
legend('RF Power','Throttle')
xlim([0 50])

subplot(223)
plot(t,y2(:,1),t,y2(:,2))
hold on
title('Step to |F|, MIMO')
legend('|F|','Vbias')
xlim([0 50])

subplot(224)
plot(t,y2(:,3),t,y2(:,4))
hold on
title('Step to |F|, MIMO')
legend('RF Power','Throttle')
xlim([0 50])
%%
% Bode magnitude from |F| noise to [|F|; Vbias; Power; Throttle]
figure(5)
bodemag(sys5(1,3),sys5(2,3),sys5(3,3),sys5(4,3))
legend('F ','Vbias','RF Power','Throttle')
% Sigma magnitude from [|F| Ref.; Vbias Ref] to [Power; Throttle]
figure(6)
sigma(sys5(3,1),sys5(3,2),sys5(4,1),sys5(4,2))
title('Sigma magnitude from [|F| Ref.; Vbias Ref] to [Power; Throttle]')

```

```

legend('|F| Ref. to Power',' Vbias Ref to Power','|F| Ref. to Throttle',' Vbias Ref to Throttle')
% Sigma magnitude from [|F| Noise; Vbias Noise] to [Power; Throttle]
figure(7)
sigma(sys5(3,3),sys5(3,4),sys5(4,3),sys5(4,4))
title('Sigma magnitude from [|F| Noise; Vbias Noise] to [Power; Throttle]')
legend('|F| Noise to Power',' Vbias Noise to Power','|F| Noise to Throttle',' Vbias Noise to Throttle')

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

% Loop-at-a-time margins at the plant input

% for state feedback
Lsf = ss(Aaug,Baug,K,[])
AM_sf = allmargin(Lsf)

AM_sf(1)%For the first channel
AM_sf(2)%For the second channel

%recovered
LI = (Cobs * PN);

AM = allmargin (LI)
AM(1)%For the first channel
AM(2)%For the second channel

% Multi-Loop Disk margins at the plant input

% Regular Disk Margin (symmetric)
% for state feedback
[DMI_sfs,MMI_sfs] = diskmargin(Lsf);
MMI_sfs

% for recovered
[DMI_s,MMI_s] = diskmargin(Cobs*PN); % T-based disk margin
MMI_s

% For T-based Disk margin ("T-based" disk, i.e. a disk centered at 1 )
% for state feedback
[DMI_sf,MMI_sf] = diskmargin(Lsf,-1); % T-based disk margin
MMI_sf

% for recovered
[DMI,MMI] = diskmargin(Cobs*PN,-1); % T-based disk margin
MMI

% Unstructured (fully-coupled) stability margin (USM) at the plant input

%For State Feedback
Tsf = feedback(Lsf,eye(size(Lsf)));
[nsf,wsf] = hinfnorm(Tsf);
StabMarg_Tsf = 1/nsf % unstructured stability margin

%For Recovered
TI = feedback(LI,eye(size(LI)));
[np,wp] = hinfnorm(TI);
StabMarg_TI = 1/np % unstructured stability margin

% Input loop transfer function: Compare Lsf to LI
Lsf = ss(Aaug,Baug,K,[])
Tsf = feedback(Lsf,eye(size(Lsf)));
TI = feedback(LI,eye(size(LI)));

```

```

figure (5) % for Loop Gain
[sv,wout]=sigma(LI,{1e-2, 1e2});
loglog(wout,sv(1,:),wout,sv(2,:))
grid on
hold on
[sv,wout]=sigma(Lsf,{1e-2, 1e2});
loglog(wout,sv(1,:),wout,sv(2,:))
hold off
title('Open loop singular values LsF(s) = K*inv(sI-A)*B and Lobs(s) = C(s)*P(s)')
legend('sig_{max}(Lobs)', 'sig_{min}(Lobs)', 'sig_{max}(LsF)', 'sig_{min}(LsF)')
xlabel('w rad/sec')
ylabel('Magnitude')

```

```

figure(6) % for Complimentary sensitivity
[sv,wout]=sigma(TI,{1e-2, 1e2});
loglog(wout,sv(1,:),wout,sv(2,:))
grid on
hold on
[sv,wout]=sigma(Tsf,{1e-2, 1e2});
loglog(wout,sv(1,:),wout,sv(2,:))
hold off
title('singular values of complementary sensitivity function')
legend('sig_{max}(Tobs)', 'sig_{min}(Tobs)', 'sig_{max}(TsF)', 'sig_{min}(TsF)')
xlabel('w rad/sec')
ylabel('Magnitude')
Ssf = feedback(eye(size(Lsf)),Lsf);
SI = feedback(eye(size(LI)),LI);

```

```

figure (7) % for Sensitivity
[sv,wout]=sigma(SI,{1e-2, 1e2});
loglog(wout,sv(1,:),wout,sv(2,:))
grid on
hold on
[sv,wout]=sigma(Ssf,{1e-2, 1e2});
loglog(wout,sv(1,:),wout,sv(2,:))
hold off
title('singular values of sensitivity function')
legend('sig_{max}(Sobs)', 'sig_{min}(Sobs)', 'sig_{max}(SsF)', 'sig_{min}(SsF)')
xlabel('w rad/sec')
ylabel('Magnitude')

```

```

% figure (11)
% y1= (sigma(db2mag(TI,{1e-2, 1e2})))
% y2= (sigma(db2mag(Tsf,{1e-2, 1e2})))

```

```

% plot({1e-2, 1e2},y1,{1e-2, 1e2},y2)

```

```

% Input sensitivity: Compare Ssf to SI

```

```

% Input complementary sensitivity: Compare TsF to TI

```

```

%% Part 2(B): Equivalent Controller

```

```

% Equivalent controller
% Ceq = inv[ I+K1 inv(sI-A) B] (KI/s)
sys = ss(AP,BP,K(:,1:8),0);
tr_ = tf(1,[1 0]);
% Ceq = tf(inv( eye(2) + K(:,1:8)*inv(s*eye(8) - AP)*BP)*K(:,9:10)/s);
Ceq = inv(eye(2) + sys)*K(:,9:10)*tr_;
figure(8)
bode(Ceq(1,1))
hold on
bode(Ceq(1,2))

```

```

hold on
bode(Ceq(2,1))
hold on
bode(Ceq(2,2))
hold on
legend('Ceq(1,1)', 'Ceq(1,2)', 'Ceq(2,1)', 'Ceq(2,2)')
title('equivalent compensator')

%% Part 2(C): Decentralized Approximation of Equivalent Controller
s = tf('s');
Chateq1 = 0.775*((s+13)*(s+0.25)*(s+0.97)*(s+2)/((s+31)*(s+4.1)*(s+1.2)*s*(s+0.2)));
% Chateq2 =
0.632*((s+13)*(s+2)*(s+0.96)*(s+0.97)*(s+0.18)/((s+31)*(s)*(s+4.1)*(s+0.9)*(s+0.21)*(s+1.17)));
Chateq2 = 0.775*((s+0.24)*(s+0.19)/((s+0.9)*(s)*(s+0.21)))
% Chateq2 = tf([1],[1 0])*tf(0.38*[1 .203],0.203*[1 0.38])*tf([0.1275]);
Chateq = [Chateq1 Chateq1; -Chateq2 Chateq2];
figure(9)
bode(Ceq(1,1),10^-2:10^-2:10^3)
hold on
bode(Chateq(1,1),10^-2:10^-3:10^3)
hold on
bode(Ceq(2,2),10^-2:10^-2:10^3)
bode(Chateq(2,2),10^-2:10^-2:10^3)
hold on
legend('Ceq(1,1)', 'Cd1', 'Ceq(2,2)', 'Cd2')

figure(10)
step(feedback(PN*Ceq,eye(2)))
hold on
step(feedback(PN*Chateq,eye(2)))
hold on
legend('with Ceq', 'with approx to Ceq')
title('step response with Ceq and approximation')
%% 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

%% Final Project Part 3: Reactive Ion Etching with MIMO Control

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

% 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);

% Condition number of DC gain for not scaled plant
Pox_0 = freqresp(Pox,0);
num = cond(Pox_0,2)

% Input and output scalings based on equilibrium values
DO = diag([16.52 340 17.83]);
DI = diag([1000 12.5 5]);

% Normalize Plant
PoxN = inv(DO)*Pox*DI;
PoxN = ss(PoxN);
% Condition number of DC gain
PoxN_0 = freqresp(PoxN,0);
num = cond(PoxN_0,2)

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

% State-space data for scaled plant
[As,Bs,Cs] = ssdata(PoxN);
[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([1,1,1]);
% Qi = diag([0.1,0.1,0.1]);
% Q = blkdiag(Cs'*Q1*Cs,Qi);
% R = diag([1,1,1]);

Q1 = diag([2.5,2.5,2.5]);
Qi = diag([0.2,0.2,0.2]);
Q = blkdiag(Cs'*Q1*Cs,Qi);
R = diag([0.1,0.1,0.2]);

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

% Compute state feedback and observer gains
K = lqr(Aaug,Baug,Q,R);
q = 3;
V = q^2*Bs*Bs';
W = eye(3);
G = eye(26);
L = lqe(As,G,Cs,V,W);

sys = ss(Aaug-Baug*K,[zeros(nx,nu); -eye(nu)],Caug,0);
sys1 = ss(Aaug-Baug*K,[zeros(nx,nu); -eye(nu)],-K,0);
% %%
% % Time vector
% Tf = 50;
% Nt = 500;
% t = linspace(0,Tf,Nt);

```



```

%
% % Step responses
% trans = tf(sys);
% trans_1 = tf(sys1);
%
% figure(1)
%
% subplot(2,3,1)
% step(trans(1,1))
% hold on
% step(trans(2,1))
% hold on
% step(trans(3,1))
% title('Step to [F], SF')
% grid on
%
% subplot(2,3,2)
% step(trans(1,2))
% hold on
% step(trans(2,2))
% hold on
% step(trans(3,2))
% title('Step to [F], SF')
% grid on
%
% subplot(2,3,3)
% step(trans(1,3))
% hold on
% step(trans(2,3))
% hold on
% step(trans(3,3))
% title('Step to Vbias, SF')
%
% grid on
%
% subplot(2,3,4)
% step(trans_1(1,1))
% hold on
% step(trans_1(2,1))
% hold on
% step(trans_1(3,1))
% title('Step to Vbias, SF')
%
% grid on
%
% subplot(2,3,5)
% step(trans_1(1,2))
% hold on
% step(trans_1(2,2))
% hold on
% step(trans_1(3,2))
%
%
% subplot(2,3,6)
% step(trans_1(1,3))
% hold on
% step(trans_1(2,3))
% hold on
% step(trans_1(3,3))
%%
% 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]
A_obs = [As -Bs*K(:,1:26) -Bs*K(:,27:29);

```

```

L*Cs As-Bs*K(:,1:26)-L*Cs -Bs*K(:,27:29);
Cs zeros(3,29)];
B_obs = [[zeros(52,3); eye(3)] [zeros(26,1); L(:,1); zeros(3,1)] [zeros(26,1); L(:,1);
zeros(3,1)] [zeros(26,1); L(:,1); zeros(3,1)]];
C_obs= [Cs zeros(size(Cs)) zeros(3,3); zeros(3,26) -K];
sys5 = -ss(A_obs,B_obs,C_obs,0);
sys6 = ss(A_obs,B_obs,[zeros(3,26) K],0);

% Form Closed-Loop
% Inputs: [|F|Ref; Vbias Ref; Press Ref; |F| Noise; Vbias Noise; Press Noise]
% Outputs: [|F|; Vbias; Press; Power; Throttle; %O2]
trans = tf(sys5);
trans_1 = tf(sys6);

% 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.
eig2 = eig(Aaug-Baug*K);
eig3 = eig(AP-L*CP);
eig1 = eig(A_obs);
isstable(sys5);

%%
% Time vector
Tf = 40;
Nt = 400;
t = linspace(0,Tf,Nt);

% Step Responses (Without Noise)
noise=0.1*randn(401,1);
ref1 = [ones(size(noise)) zeros(size(noise)) zeros(size(noise)) zeros(size(noise))
zeros(size(noise)) zeros(size(noise))];
noise=0.1*randn(401,1);
ref2 = [zeros(size(noise)) ones(size(noise)) zeros(size(noise)) zeros(size(noise))
zeros(size(noise)) zeros(size(noise))];
ref3 = [zeros(size(noise)) zeros(size(noise)) ones(size(noise)) zeros(size(noise))
zeros(size(noise)) zeros(size(noise))];
[y1 t]=lsim(sys5,ref1,[0:0.1:40]);

[y2 t]=lsim(sys5,ref2,[0:0.1:40]);

[y3 t]=lsim(sys5,ref3,[0:0.1:40]);

figure

subplot(231)
plot(t,y1(:,1),t,y1(:,2),t,y1(:,3))
title('Step to [F]')
grid on

subplot(234)
plot(t,y1(:,4),t,y1(:,5),t,y1(:,6))
title('Step to [F]')
grid on

subplot(232)
plot(t,y2(:,1),t,y2(:,2),t,y2(:,3))
title('Step to Vbias')

grid on

subplot(235)
plot(t,y2(:,4),t,y2(:,5),t,y2(:,6))

```

```
title('Step to Vbias')

grid on

subplot(233)
plot(t,y3(:,1),t,y3(:,2),t,y3(:,3))
title('Step to Pressure')
legend('[F]', 'Vbias', 'Pressue')
grid on

subplot(236)
plot(t,y3(:,4),t,y3(:,5),t,y3(:,6))
title('Step to Pressure')
legend('RF Power', 'Throttle', '%O2')
grid on
```