EECS 565 Linear Feedback Control Systems Analysis

Final Rroject

Name : Yifeng.Xu

UM ID : 38049774

Contents

1 Multivariable Feedback Controller Design **2**
1.1 Sub-problem (a) 2
1.2 Sub-problem (b) 4
1.3 Sub-problem (c) 5
1.4 Sub-problem (d) 5

2 Reverse Engineering the Multivariable Controller **8**
2.1 Sub-problem (a) 8
2.2 Sub-problem (b) 9
2.3 Sub-problem (c) 9

3 Oxygen as an Additional Actuator **11**
3.1 Sub-problem (a) 11
3.2 Sub-problem (b) 12

4 Appendix: MATLAB Codes **13**
4.1 Complete MATLAB Codes for Part 1 13
4.2 Complete MATLAB Codes for Part 2 17
4.3 Complete MATLAB Codes for Part 3 18

1 Multivariable Feedback Controller Design

1.1 Sub-problem (a)

The cost function for LQR state feedback design is expressed as:

$$\begin{aligned} J &= \int_0^\infty x_{Aug}^T \cdot Q_{Aug} \cdot x_{Aug} + u^T \cdot R \cdot u \\ &= \int_0^\infty x^T Q_x x + \omega^T Q_\omega \omega + u^T \cdot R \cdot u \\ &= \int_0^\infty x^T \cdot C^T Q_y C \cdot x + \omega^T Q_\omega \omega + u^T \cdot R \cdot u \end{aligned} \tag{1}$$

By transforming $Q_x = C^T Q_y C$, the LQR weighting matrices can be tunned based on physical meanings. The corresponding design parameters are shown below:

$$Q_y = \begin{bmatrix} 7 & 0 \\ 0 & 12 \end{bmatrix} \quad Q_\omega = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.3 \end{bmatrix} \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2}$$

Tuning process description: The initial weighting selection is $Q_y = Q_\omega = R = I_{2\times2}$. (1) By penalizing Q_y, the step response speeds of both $|F|$ and V_{bias} can be adjusted and it is effective to keep response $|F|$ to step command V_{bias} small and vice versa. (2) Adjusting the integrator weighting matrix Q_ω contributes to reducing overshoot from the steady state; hence, Q_ω was tuned to prevent overshoot.

Applying state feedback to the system, the subsequent closed-loop state equation with integrator can be determined:

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \dot{\omega} \end{bmatrix} &= \begin{bmatrix} A & O \\ C & O \end{bmatrix} \begin{bmatrix} x \\ \omega \end{bmatrix} + \begin{bmatrix} B \\ O \end{bmatrix} \begin{bmatrix} -K_1 & -K_2 \end{bmatrix} \begin{bmatrix} x \\ \omega \end{bmatrix} + \begin{bmatrix} O \\ -I \end{bmatrix} r \\ &= \begin{bmatrix} A - BK_1 & -BK_2 \\ C & O \end{bmatrix} \begin{bmatrix} x \\ \omega \end{bmatrix} + \begin{bmatrix} O \\ -I \end{bmatrix} r \end{aligned} \tag{3}$$

Then, the augmented output state equation is design as follows to output step response of both input u and output y.

$$\begin{bmatrix} y \\ u \end{bmatrix} = \begin{bmatrix} C & O_{2\times2} \\ -K_1 & -K_2 \end{bmatrix} \begin{bmatrix} x \\ \omega \end{bmatrix} \tag{4}$$

```
%% Part 1(A): Linear Quadratic Regulator with Integrators
% Augment state equations so that you can do integral control
Aaug = [AP zeros(8,2);
        CP zeros(2,2)];
```

```
Baug = [BP; zeros(2,2)];

% LQR Weighting Matrices
Qy = [7 0; 0 12];
Qx = CP'*Qy*CP;
Qw = [0.5 0; 0 0.3]
Qaug = [Qx zeros(8,2);
        zeros(2,8) Qw];
R = 1*[1 0; 0 1];

% LQ state feedback gain
K = lqr(Aaug,Baug,Qaug,R);
K1 = K(:,1:8);
K2 = K(:,9:10);

% Closed loop state equations with state feedback and integrators.
Asf = Aaug-Baug*K;
Bnew = [zeros(8,2);-eye(2)];
Cnew = [CP, zeros(2,2);
        -K1 -K2];

Psf = ss(Aaug-Baug*K,Bnew,Cnew,0);
Tf_sf = tf(Psf);
```

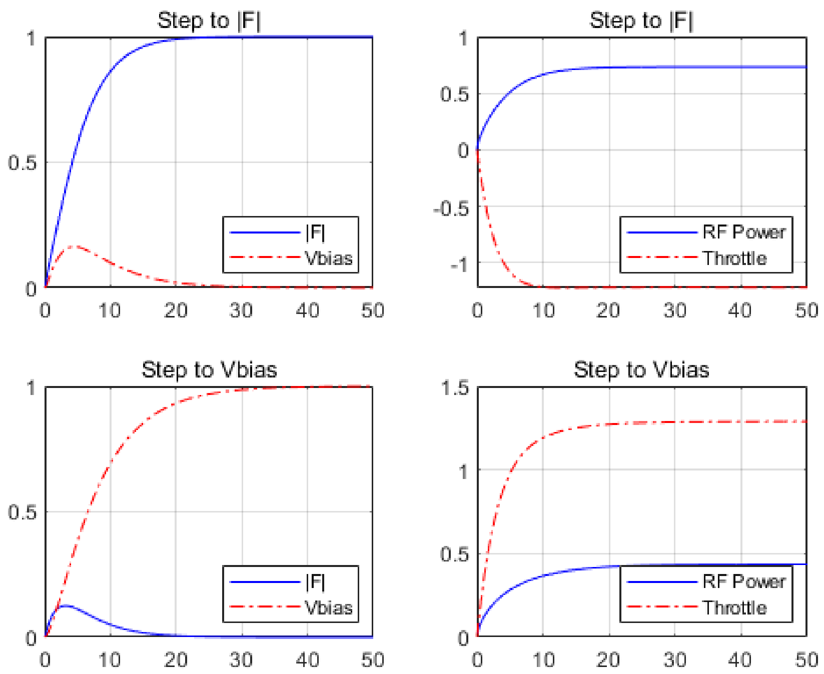

Figure 1: System Step Response based on LQR

1.2 Sub-problem (b)

The observer model with state feedback is represented below, where L is the observer gain that will be determined from loop transfer recovery process:

$$\begin{cases} \dot{\hat{x}} = A\hat{x} + Bu + L\left(y_m - C\hat{x}\right) \\ u = -K_1\hat{x} - K_2\omega \end{cases} \tag{5}$$

Considering noise as $V = V_0 + \beta^2 B \cdot B^T$, the system model with noise can be expressed as:

$$\begin{cases} \dot{x} = Ax + Bu + V \\ y_m = Cx + W \end{cases} \tag{6}$$

With β increasing, the input loop transfer function with observer plus state feedback L_{obs} will be recovered to L_{SF} from the perspective of pointwise frequency(i.e. $L_{obs}(s) \to L_{SF}(s)$). Figure 2 illustrates the system response with $\beta = 1000$ and 1% noise.

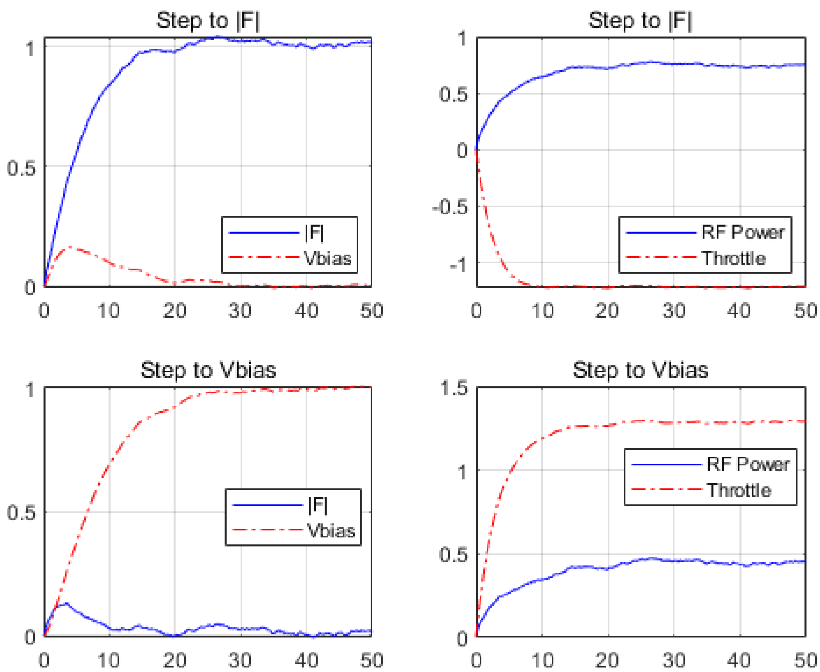

Figure 2: System Step Response with LQG/LTR Controller and 1% Noise

As β becomes large, **an increased noise response can be observed**. After several trials, $\beta \approx 10^6$ will make $|F|$ response not satisfy requirements.

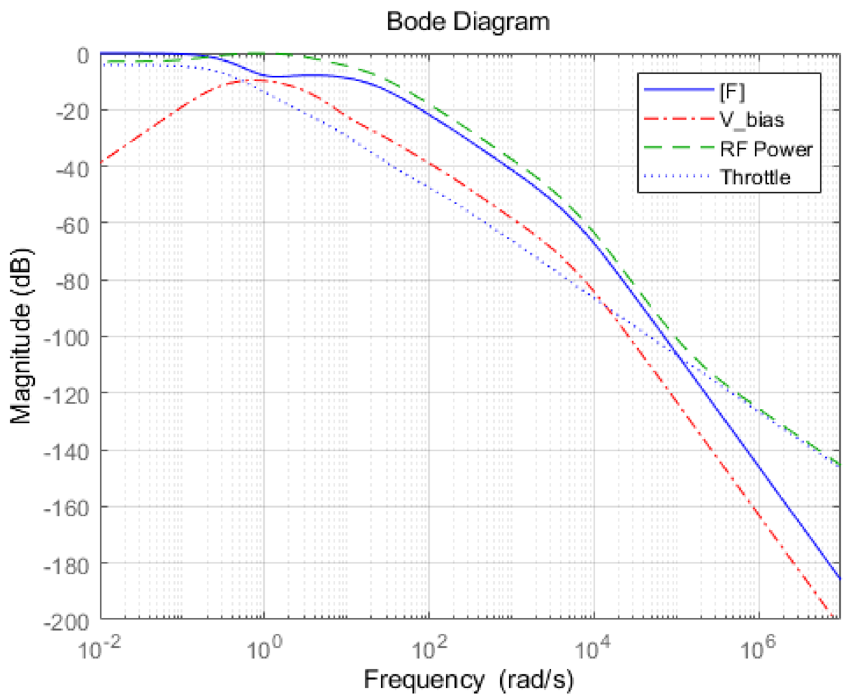

Figure 3: Bode Plots of the Closed-Loop Transfer Function from Noise to Inputs and Outputs

1.3 Sub-problem (c)

Achieving arbitrarily good recovery of the state feedback loop may not be possible in practice. Increasing the observer gain can improve the recovery of the state feedback loop, but it also increases the system's sensitivity to sensor noise. This can lead to degraded performance as sensor noise is amplified, and there's a trade-off between approximating the state feedback design and reducing sensitivity to sensor noise.

1.4 Sub-problem (d)

Stability Margin Summary Table			
	Loop-at-a-time Margin	Multi-loop Disk Margins	Unstructured Margins
L_{obs}	DMI(1)=1.99/DMI(2)=1.55	MMI=1.547	0.9983
L_{sf}	DMI(1)=2.00/DMI(2)=1.99	MMI=1.980	0.9987

It is hence reasonable to state that LQG/LTR controller has been recovered reasonably well. The following codes can be used to determine stability margins.

```
%% Part 1(D): Stability Margins and Comparison With State Feedback
% Loop-at-a-time margins at the plant input
[DMI,MMI] = diskmargin(Lsf);
display(DMI(1))
display(DMI(2))
% Unstructured (fully coupled) margins.
display(MMI)
% Unstructured (fully-coupled) stability margin (USM) at the plant input
[np,wp] = hinfnorm(Tsf);
```

```
StabMarg1 = 1/np
```

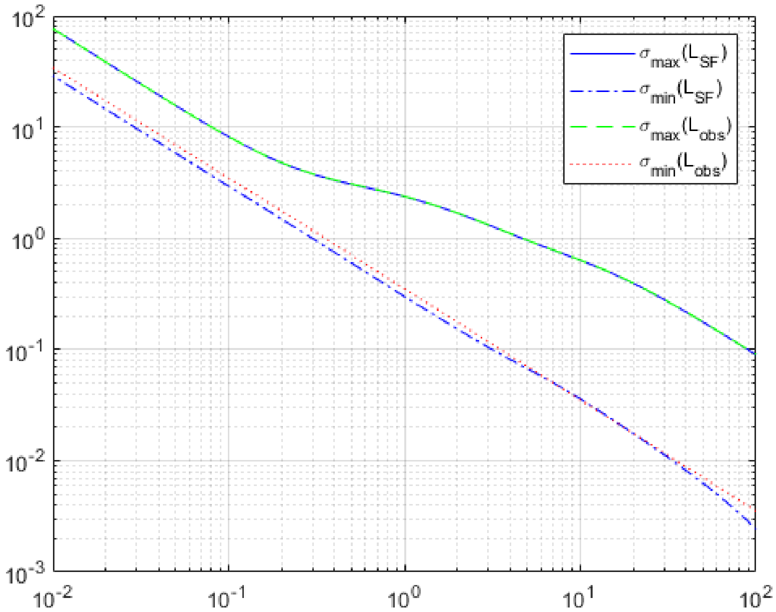

Figure 4: Open Loop Singular Values Plots of L_{sf} and L_I

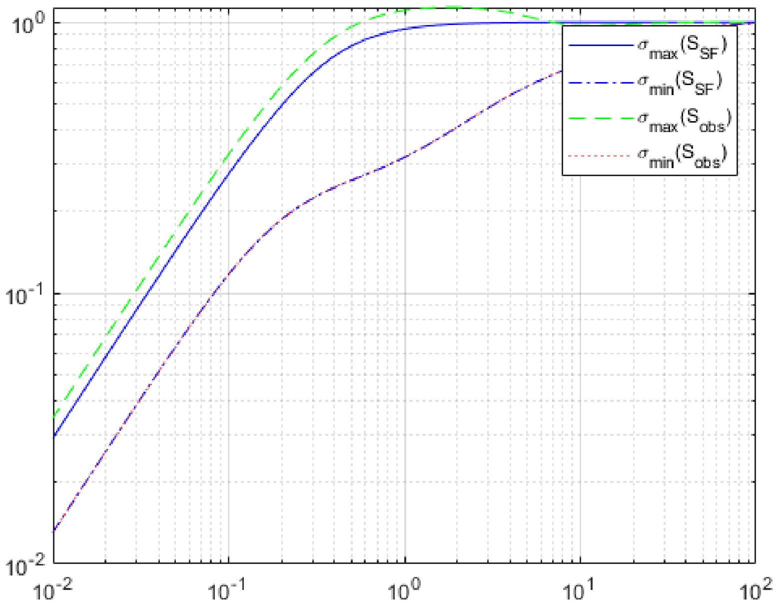

Figure 5: Open Loop Singular Values Plots of S_{sf} and S_I

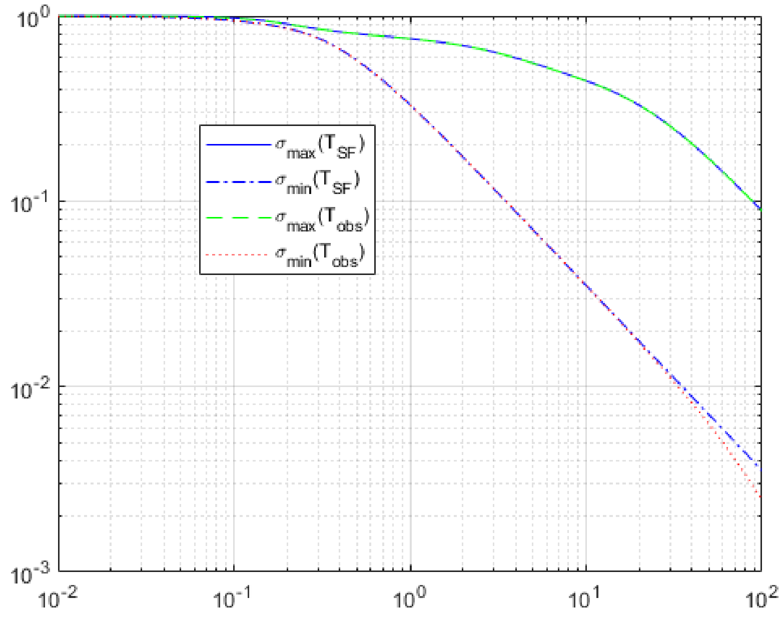

Figure 6: Open Loop Singular Values Plots of T_{sf} and T_I

2 Reverse Engineering the Multivariable Controller

2.1 Sub-problem (a)

Consider the nominal system:

$$\dot{x} = Ax + Bu \tag{7}$$

$$y_m = Cx \tag{8}$$

Taking the Laplace transform of equation 7:

$$X(s) = (SI - A)^{-1} B \cdot U(s) \tag{9}$$

As for the nominal observer-based system, the following equations are important:

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y_m - C\hat{x}) \tag{10}$$

$$u = -K\hat{x} - K_I \omega \tag{11}$$

Define the error dynamics $\tilde{x} = x - \hat{x}$. Then, it can be determined as:

$$\begin{aligned} \dot{\tilde{x}} &= \dot{x} - \dot{\hat{x}} \\ &= Ax + Bu - Ax - Bu - L(y_m - C\hat{x}) \\ &= (A - LC)\tilde{x} \end{aligned} \tag{12}$$

where $A - LC$ is designed to converge and hence asymptotically stable, which implies $x = \hat{x}$.
Taking the Laplace transform of equation 11 and plug in 9:

$$\begin{aligned} U(s) &= -K \cdot X(s) - K_I \cdot W(s) \\ &= -K \cdot (SI - A)^{-1} B \cdot U(s) - K_I \cdot W(s) \\ U(s) &= -\left(I + K \cdot (SI - A)^{-1} B\right)^{-1} K_I \cdot W(s) \end{aligned} \tag{13}$$

For integrator W(s), the following relation is defined:

$$\begin{aligned} \dot{\omega} &= y_m - r \\ S \cdot W(s) &= Y(s) - R(s) \\ W(s) &= \frac{Y(s) - R(s)}{S} \end{aligned} \tag{14}$$

Taking the Laplace transform of equation 8 and plug in 9:

$$Y(s) = P(s) \cdot U(s) \tag{15}$$

where $P(s) = C(SI - A)^{-1} B$.
Combining equation 13, 14, and 15, the following expression can be derived:

$$Y(s) = -P(s) \cdot \left(I + K \cdot (SI - A)^{-1} B\right)^{-1} \cdot \frac{K_I}{S} \cdot (Y(s) - R(s)) \tag{16}$$

$$P(s)\left(I + K \cdot (SI - A)^{-1} B\right)^{-1} \frac{K_I}{S} R(s) = \left(I + P(s)\left(I + K(SI - A)^{-1} B\right)^{-1} \frac{K_I}{S}\right) Y(S) \tag{17}$$

Then, the transfer function from reference input r to system output y can be derived, which ends proof:

$$T_{yr}(s) = \frac{Y(s)}{R(s)} = \left(I + P(s)\left(I + K(SI - A)^{-1} B\right)^{-1} \frac{K_I}{S}\right)^{-1} P(s)\left(I + K(SI - A)^{-1} B\right)^{-1} \frac{K_I}{S} \tag{18}$$

This transfer function is independent of the observer due to the fact that the observer error dynamics $\dot{\tilde{x}} = \dot{x} - \dot{\hat{x}} = (A - LC)\tilde{x} = 0$, which implies the optimal estimation of $x = \hat{x}$ and hence makes transfer function independent of the observer.

2.2 Sub-problem (b)

The equivalent compensator with unity output feedback is given as:

$$C_{eq}(s) = \left(I + K(sI - A)^{-1} B\right)^{-1} \frac{K_I}{s} \tag{19}$$

```
%% Part 2(B): Equivalent Controller
% Equivalent controller
%   Ceq = inv[ I+K1 inv(sI-A) B] (KI/s)
figure(7)
sym s;
s = tf('s');
Ceq = inv(eye(2) + K1*inv(s*eye(8)-AP)*BP)*(K2/s);
bode(Ceq(1,1),'b')
hold on
bode(Ceq(1,2),'r-.')
bode(Ceq(2,1),'g--')
bode(Ceq(2,2),'r:')
hold off
legend('Ceq(1,1)','Ceq(1,2)','Ceq(2,1)','Ceq(2,2)');
if exist('garyfyFigure','file'), garyfyFigure, end
title('Equilavlent Compensator');
grid on
```

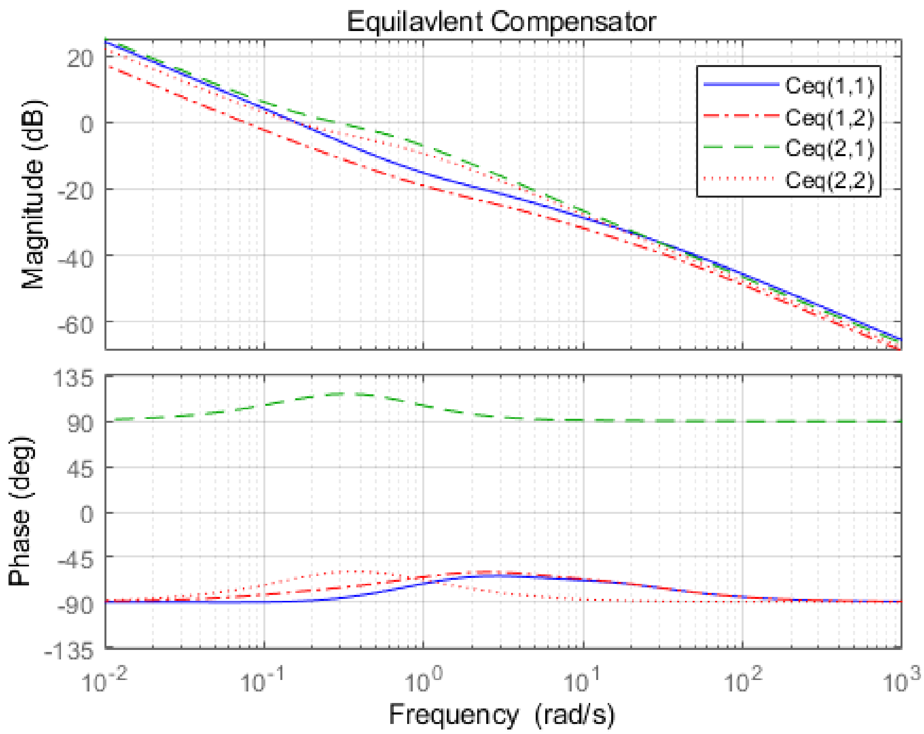

Figure 7: Equivalent Compensator

2.3 Sub-problem (c)

Balanced model reduction methods are used to obtain lower-order approximations of the equivalent compensator. See Appendix for detailed MATLAB implementation:

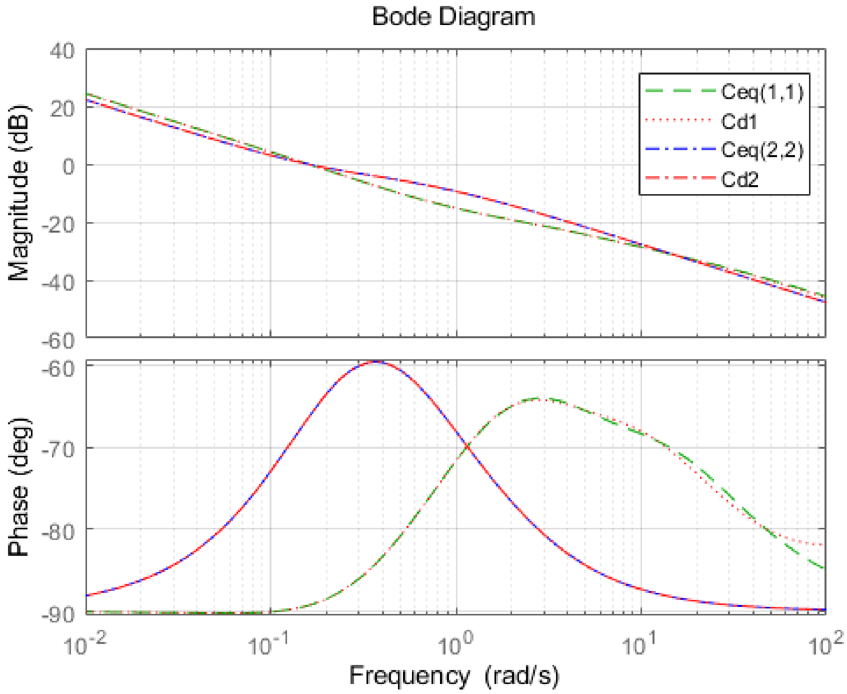

Figure 8: Bode plots of C_{eq} and Approximated C_{d1} and C_{d2}

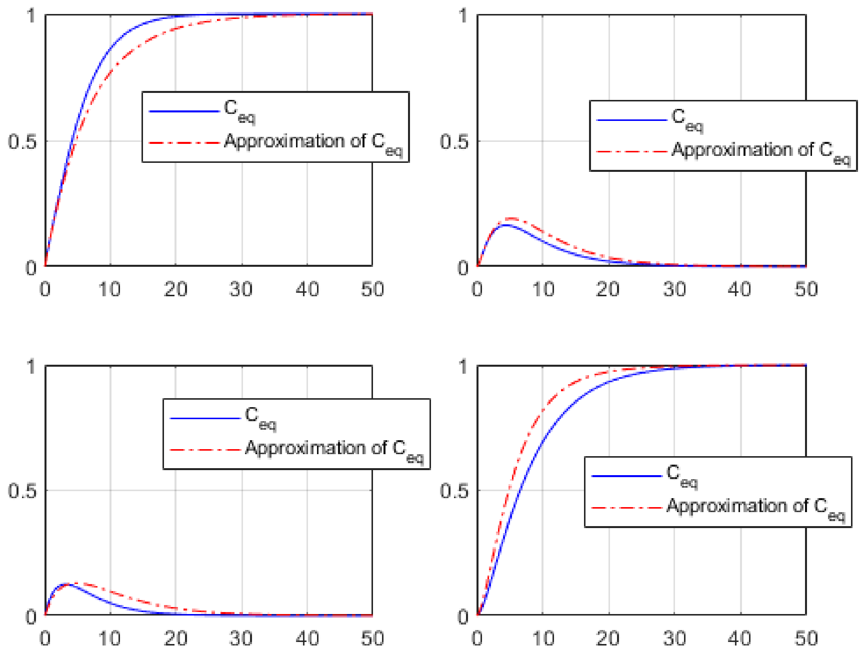

Figure 9: Step Response with C_{eq} and $\hat{C}_{eq}$

3 Oxygen as an Additional Actuator

3.1 Sub-problem (a)

The first step of the feasibility study is to normalize the system so that unitless comparison can be made based on singular values. The input and output scaling matrices are defined as follows:

$$D_I = \begin{bmatrix} 1000 & 0 & 0 \\ 0 & 12.5 & 0 \\ 0 & 0 & 5 \end{bmatrix} \quad D_O = \begin{bmatrix} 16.52 & 0 & 0 \\ 0 & 340 & 0 \\ 0 & 0 & 17.83 \end{bmatrix} \tag{20}$$

Then, the normalized DC gain can be determined as:

$$P_{OXN}(0) = D_O^{-1} P(0) D_I = \begin{bmatrix} 1.0624 & -0.6125 & 0.5875 \\ 0.8189 & 0.3791 & 0.009 \\ 0.0084 & -1.2593 & 0.0168 \end{bmatrix} \tag{21}$$

Then, the corresponding condition number can be determined as:

$$\kappa(P_{OXN}) = \frac{\sigma_{\max}(P_{OXN})}{\sigma_{\min}(P_{OXN})} = 5.5844 \tag{22}$$

The singular value decomposition can be found via the following codes:

```
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
% Normalized System
PoxN = inv(DO)*Pox*DI;
PoxN = ss(PoxN);
% Condition number of DC gain
PN0 = dcgain(PoxN)
cond(PN0)
% Singular Value Decomposition
[U,S,V] = svd(PN0);
S
```

The smallest singular value and the condition number are given by:

$$\sigma_{\min}(P_{OXN}(0)) = 0.2856 \qquad \kappa(P_{OXN}) = 5.5844 \tag{23}$$

These values are reasonable to conclude that it is **feasible** to regulate the system via these three inputs and controllers.

3.2 Sub-problem (b)

The LQG technique utilizes the optimal observer and optimal state feedback, as constructed below:

$$\begin{cases} \dot{\hat{x}} = A\hat{x} + Bu + L\left(y_m - C\hat{x}\right) \\ u = -K_1\hat{x} - K_2\omega \end{cases} \tag{24}$$

Then, considering noise covariance $V = B \cdot B^T$ and $W = I$, the system model is defined by:

$$\begin{cases} \dot{x} = Ax + Bu + V \\ y_m = Cx + W \end{cases} \tag{25}$$

The LQG controller with augmented integrator involves the following optimisation problem:

$$J = \lim_{T \to \infty} \frac{1}{T} \cdot E \cdot \left[\int_0^T x^T \cdot C^T Q_y C \cdot x + \omega^T Q_\omega \omega + u^T \cdot R \cdot u \right] \tag{26}$$

Where the corresponding design parameters are shown below. The V_{bias} weighting entry is penalized to reduce overshoot.

$$Q_y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Q_\omega = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{27}$$

Then, the response is demonstrated below. See Appendix for MATLAB implementation.

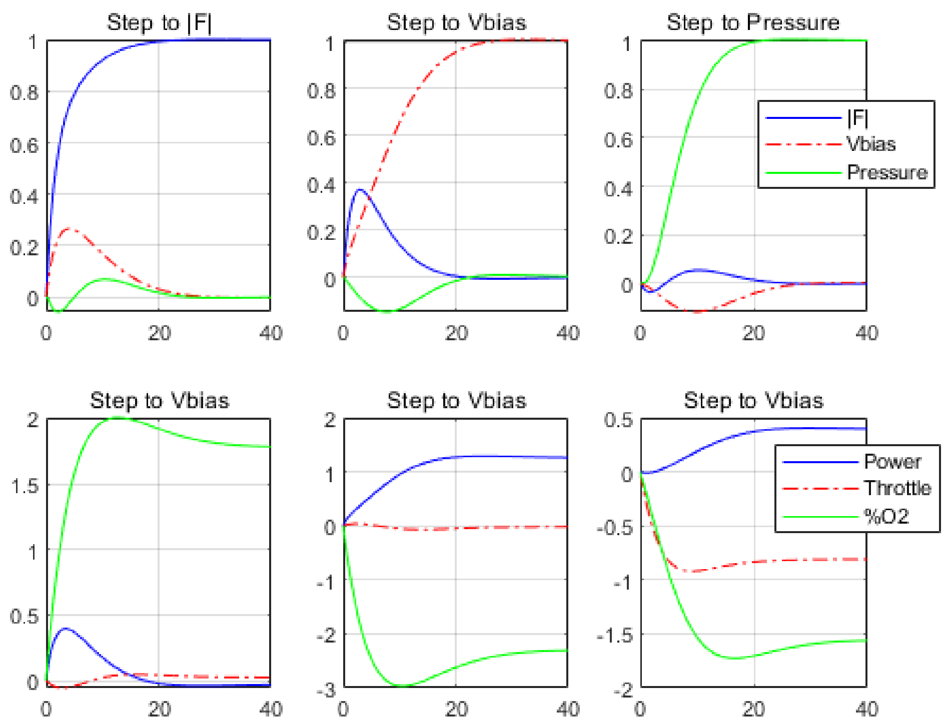

Figure 10: System Step Response with Additional Oxygen %O2 Actuator

4 Appendix: MATLAB Codes

4.1 Complete MATLAB Codes for Part 1

```
%% Final Project Parts 1 and 2:  Reactive Ion Etching with MIMO Control
clc;clear
% 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
% Augment state equations so that you can do integral control
Aaug = [AP zeros(8,2);
        CP zeros(2,2)];
Baug = [BP; zeros(2,2)];

% LQR Weighting Matrices
Qy = [7 0; 0 12];
Qx = CP'*Qy*CP;
Qw = [0.5 0; 0 0.3]
Qaug = [Qx zeros(8,2);
        zeros(2,8) Qw];
R = 1*[1 0; 0 1];

% LQ state feedback gain
K = lqr(Aaug,Baug,Qaug,R);
K1 = K(:,1:8);
K2 = K(:,9:10);

% Closed loop state equations with state feedback and integrators.
Asf = Aaug-Baug*K;
Bnew = [zeros(8,2);-eye(2)];
Cnew = [CP, zeros(2,2);
        -K1 -K2];

Psf = ss(Aaug-Baug*K,Bnew,Cnew,0);
Tf_sf = tf(Psf);

% Verify that closed-loop is stable (Check to verify no bugs in code)
pole(Psf)
% Time vector
Tf = 50;
Nt = 500;
t = linspace(0,Tf,Nt);

% Step Response
y11 = step(Tf_sf(1,1),t);
y21 = step(Tf_sf(2,1),t);
y12 = step(Tf_sf(1,2),t);
y22 = step(Tf_sf(2,2),t);
u11 = step(Tf_sf(3,1),t);
```

```
u21 = step(Tf_sf(4,1),t);
u12 = step(Tf_sf(3,2),t);
u22 = step(Tf_sf(4,2),t);

figure(1)
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

% Part 1(B): Linear Quadratic Regulator with Integrators
% beta = 10000000000;
%%
rng(0);
beta = 1000;
% Covariance matrices for loop transfer recovery observer
V = 0 + beta^2*(BP*BP'); % V=V0+beta^2 BB' with V0=0
% W = [0*randn(Nt,1) 0;
%       0 0*randn(Nt,1)];
W = [1 0; 0 1];
% 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.
L = lqr(AP',CP',V,W)'     % V=V0+beta^2 BB' with V0=0
% 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]

Aaug_obs = [AP -BP*K1 -BP*K2;L*CP AP-BP*K1-L*CP -BP*K2; CP zeros(2,10)];
Baug_obs = [zeros(8,2); zeros(8,2); -eye(2)];
Caug_obs = [CP zeros(2,10); -K1 zeros(2,8) -K2];

% Caug_obs = [Cnew; -K1 -K2];

% Form Closed-Loop
%   Inputs are [|F| Ref.; Vbias Ref; |F| noise; Vbias noise]
%   Outputs: [|F|; Vbias; Power; Throttle]
Psf_obs = ss(Aaug_obs,Baug_obs,Caug_obs,0);
Tf_obs = tf(Psf_obs);
% 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.
eig(Aaug_obs)
eig_L = eig(AP-L*CP)
```

```
eig_K = eig(Aaug-Baug*K)


% Step responses with noise on |F|
Fnoise = randn(Nt,1)*0.1;
y11 = step(Tf_obs(1,1),t);
y21 = step(Tf_obs(2,1),t);
y12 = step(Tf_obs(1,2),t);
y22 = step(Tf_obs(2,2),t);
u11 = step(Tf_obs(3,1),t);
u21 = step(Tf_obs(4,1),t);
u12 = step(Tf_obs(3,2),t);
u22 = step(Tf_obs(4,2),t);

y1n = lsim(Tf_obs(1,1),Fnoise,t);
y2n = lsim(Tf_obs(2,1),Fnoise,t);
u1n = lsim(Tf_obs(1,1),Fnoise,t);
u2n = lsim(Tf_obs(2,1),Fnoise,t);
figure(2)
subplot(2,2,1);
plot(t,y11+y1n,'b',t,y21+y2n,'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+u1n,'b',t,u21+u2n,'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+y1n,'b',t,y22+y2n,'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+u1n,'b',t,u22+u2n,'r-.');
grid on; xlim([0, Tf])
legend('RF Power','Throttle','Location','Southeast');
title('Step to Vbias');
if exist('garyfyFigure','file'), garyfyFigure, end
% Bode magnitude from |F| noise to [|F|; Vbias; Power; Throttle]
An2yu = [AP -BP*K1 -BP*K2;L*CP AP-BP*K1-L*CP -BP*K2; CP zeros(2,10)];
Bn2yu  = [zeros(8,2); L; eye(2,2)];
Cn2yu = [CP zeros(2,10); -K1 zeros(2,8) -K2];
Psf_2yu = ss(An2yu,Bn2yu,Cn2yu,0);
Tf_2yu = tf(Psf_2yu);
figure(3)
bodemag(Psf_2yu(1,1),'b')
hold on
bodemag(Psf_2yu(2,1),'r-.')
bodemag(Psf_2yu(3,1),'g--')
bodemag(Psf_2yu(4,1),':')
legend('[F]','V_{bias}','RF Power','Throttle');
if exist('garyfyFigure','file'), garyfyFigure, end
grid on
%%
% Sigma magnitude from [|F| Ref.; Vbias Ref] to [Power; Throttle]
% Cobs = ss(AP - BP*K1 - L*[CP],[L],K1,0);
Cobs = ss(Aaug - Baug*K -[L;zeros(2,2)]*[CP zeros(2,2)],[L;zeros(2,2)],K,0);
% PN_obs = ss(Aaug, Baug, [CP zeros(2,2)], 0);
PN_obs = ss(AP, BP, [CP], 0);
Lobs = Cobs*PN_obs;

% Input loop transfer function: Compare Lsf to LI
```

```
figure(4)
[sv,wout] = sigma(Lobs)
loglog(wout,sv(1,:),'b')
hold on
loglog(wout,sv(2,:),'b-.')
grid on
xlim([0.01 100])


Lsf = ss(Aaug, Baug, K, 0);
[sv,wout] = sigma(Lsf)
loglog(wout,sv(1,:),'g--')
hold on
loglog(wout,sv(2,:),'r:')
grid on
xlim([0.01 100])
legend('\sigma_{max}(L_{SF})','\sigma_{min}(L_{SF})','\sigma_{max}(L_{obs})','\sigma_{min}(L_{obs})'
hold off


figure(5)
% Input sensitivity: Compare Ssf to SI
Ssf = feedback(eye(2),Lsf);
SI = feedback(eye(2),Lobs);
[sv,wout] = sigma(Ssf)
loglog(wout,sv(1,:),'b')
hold on
loglog(wout,sv(2,:),'b-.')
grid on
xlim([0.01 100])

[sv,wout] = sigma(SI)
loglog(wout,sv(1,:),'g--')
hold on
loglog(wout,sv(2,:),'r:')
grid on
xlim([0.01 100])
legend('\sigma_{max}(S_{SF})','\sigma_{min}(S_{SF})','\sigma_{max}(S_{obs})','\sigma_{min}(S_{obs})'


figure(6)
% Input complementary sensitivity: Compare Tsf to TI
Tsf = feedback(Lsf,eye(2));
TI = feedback(Lobs,eye(2));
[sv,wout] = sigma(Tsf)
loglog(wout,sv(1,:),'b')
hold on
loglog(wout,sv(2,:),'b-.')
grid on
xlim([0.01 100])

[sv,wout] = sigma(TI)
loglog(wout,sv(1,:),'g--')
hold on
loglog(wout,sv(2,:),'r:')
grid on
xlim([0.01 100])
legend('\sigma_{max}(T_{SF})','\sigma_{min}(T_{SF})','\sigma_{max}(T_{obs})','\sigma_{min}(T_{obs})'

%% Part 1(D): Stability Margins and Comparison With State Feedback
% Loop-at-a-time margins at the plant input
[DMI,MMI] = diskmargin(Lsf);
display(DMI(1))
display(DMI(2))
% Unstructured (fully coupled) margins.
display(MMI)
% Unstructured (fully-coupled) stability margin (USM) at the plant input
[np,wp] = hinfnorm(Tsf);
StabMarg1 = 1/np
```

4.2 Complete MATLAB Codes for Part 2

```
%% Part 2(B): Equivalent Controller
% Equivalent controller
%   Ceq = inv[ I+K1 inv(sI-A) B] (KI/s)
figure(7)
sym s;
s = tf('s');
Ceq = inv(eye(2) + K1*inv(s*eye(8)-AP)*BP)*(K2/s);
bode(Ceq(1,1),'b')
hold on
bode(Ceq(1,2),'r-.')
bode(Ceq(2,1),'g--')
bode(Ceq(2,2),'r:')
hold off
legend('Ceq(1,1)','Ceq(1,2)','Ceq(2,1)','Ceq(2,2)');
if exist('garyfyFigure','file'), garyfyFigure, end
title('Equilavlent Compensator');
grid on
%% Part 2(C): Decentralized Approximation of Equivalent Controller
%% Cd approximation
figure(8)
Cd1hat = balred(Ceq(1,1), 9);
bode(Ceq(1,1),'g--');
hold on
bode(Cd1hat,'r:')


Cd2hat = balred(Ceq(2,2), 7)
bode(Ceq(2,2),'b-.')
bode(Cd2hat,'r-.')
xlim([0.01 100])
legend('Ceq(1,1)','Cd1','Ceq(2,2)','Cd2');
if exist('garyfyFigure','file'), garyfyFigure, end
grid on


%%

Ceq_hat = [Cd1hat Cd1hat; -Cd2hat Cd2hat];

sys_hat = feedback(PN*Ceq_hat,eye(2));
y11_hat = step(sys_hat(1,1),t);
y21_hat = step(sys_hat(2,1),t);
y12_hat = step(sys_hat(1,2),t);
y22_hat = step(sys_hat(2,2),t);

sys = feedback(PN*Ceq,eye(2));
y11 = step(sys(1,1),t);  %        input
y21 = step(sys(2,1),t);
y12 = step(sys(1,2),t);
y22 = step(sys(2,2),t);


figure(9)
subplot(2,2,1);
plot(t,y11,'b',t,y11_hat,'r-.');
grid on; xlim([0, Tf])
legend('C_{eq}','Approximation of C_{eq}');
if exist('garyfyFigure','file'), garyfyFigure, end

subplot(2,2,2);
plot(t,y21,'b',t,y21_hat,'r-.');
grid on; xlim([0, Tf])
legend('C_{eq}','Approximation of C_{eq}');
ylim([0 1]);
if exist('garyfyFigure','file'), garyfyFigure, end

```

```
subplot(2,2,3);
plot(t,y12,'b',t,y12_hat,'r-.');
grid on; xlim([0, Tf])
legend('C_{eq}','Approximation of C_{eq}');
ylim([0 1]);
if exist('garyfyFigure','file'), garyfyFigure, end

subplot(2,2,4);
plot(t,y22,'b',t,y22_hat,'r-.');
grid on; xlim([0, Tf])
legend('C_{eq}','Approximation of C_{eq}');
ylim([0 1]);
if exist('garyfyFigure','file'), garyfyFigure, end


%% 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
```

4.3 Complete MATLAB Codes for Part 3

```
%% Final Project Part 3:  Reactive Ion Etching with MIMO Control
clc;clear
%% 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);
% Input and output scalings based on equilibrium values
DO = diag([16.52 340 17.83]);
DI = diag([1000 12.5 5]);
% Normalize Plant
% Normalized System
PoxN = inv(DO)*Pox*DI;
PoxN = ss(PoxN);
% Condition number of DC gain
PN0 = dcgain(PoxN)
cond(PN0)
% Singular Value Decomposition
[U,S,V] = svd(PN0);
```

```
S
%% 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 = alpha*Cs'*Cs + Qw
% Augmented Plant with integrators
Aaug = [As zeros(nx,nu);
        Cs zeros(3,3)];
Baug = [Bs; zeros(3,3)];

% LQR Weighting Matrices
Qy = [1 0 0; 0 10 0; 0 0 1];
Qx = Cs'*Qy*Cs;
Qw = [1 0 0; 0 1 0; 0 0 1];

Qaug = [Qx zeros(26,3);
        zeros(3,26) Qw];
R = [1 0 0; 0 10 0; 0 0 1];

% Compute state feedback and observer gains
K = lqr(Aaug,Baug,Qaug,R);
K1 = K(:,1:26);
K2 = K(:,27:29);

% V = 1 + beta^2*(Bs*Bs'); % V=V0+beta^2 BB' with V0=0
V = (Bs*Bs');
W = [1 0 0; 0 1 0; 0 0 1];

L = lqr(As',Cs',V,W)'    % V=V0+beta^2 BB' with V0=0
% Verify that observer error is stable (Check to verify no bugs in code)
eig(As-L*Cs)
% 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]
Aaug_obs = [As -Bs*K1 -Bs*K2;L*Cs As-Bs*K1-L*Cs -Bs*K2; Cs zeros(3,29)];
Baug_obs = [zeros(26,3); zeros(26,3); -eye(3)];
Caug_obs = [Cs zeros(3,29); -K1 zeros(3,26) -K2];


% Form Closed-Loop
%   Inputs: [|F|Ref; Vbias Ref; Press Ref; |F| Noise; Vbias Noise; Press Noise]
%   Outputs: [|F|; Vbias; Press; Power; Throttle; %O2]
Psf_obs = ss(Aaug_obs,Baug_obs,Caug_obs,0);
Tf_obs = tf(Psf_obs);
eig(Aaug_obs)

% 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.
eig(Aaug_obs)

eig_L = eig(As-L*Cs)

eig_K = eig(Aaug-Baug*K)

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

y11 = step(Tf_obs(1,1),t);
y21 = step(Tf_obs(2,1),t);
y31 = step(Tf_obs(3,1),t);

```

```
y12 = step(Tf_obs(1,2),t);
y22 = step(Tf_obs(2,2),t);
y32 = step(Tf_obs(3,2),t);

y13 = step(Tf_obs(1,3),t);
y23 = step(Tf_obs(2,3),t);
y33 = step(Tf_obs(3,3),t);

u11 = step(Tf_obs(4,1),t);
u21 = step(Tf_obs(5,1),t);
u31 = step(Tf_obs(6,1),t);

u12 = step(Tf_obs(4,2),t);
u22 = step(Tf_obs(5,2),t);
u32 = step(Tf_obs(6,2),t);

u13 = step(Tf_obs(4,3),t);
u23 = step(Tf_obs(5,3),t);
u33 = step(Tf_obs(6,3),t);


figure(15)
subplot(2,3,1);
plot(t,y11,'b',t,y21,'r-.',t,y31,'g-');
grid on; xlim([0, Tf])
% legend('|F|','Vbias','Pressure','Location','Southeast');
title('Step to |F|');
if exist('garyfyFigure','file'), garyfyFigure, end

subplot(2,3,2);
plot(t,y12,'b',t,y22,'r-.',t,y32,'g-');
grid on; xlim([0, Tf])
% legend('|F|','Vbias','Pressure','Location','Southeast');
title('Step to Vbias');
if exist('garyfyFigure','file'), garyfyFigure, end

subplot(2,3,3);
plot(t,y13,'b',t,y23,'r-.',t,y33,'g-');
grid on; xlim([0, Tf])
legend('|F|','Vbias','Pressure','Location','Southeast');
title('Step to Pressure');
if exist('garyfyFigure','file'), garyfyFigure, end


subplot(2,3,4);
plot(t,u11,'b',t,u21,'r-.',t,u31,'g-');
grid on; xlim([0, Tf])
title('Step to Vbias');
if exist('garyfyFigure','file'), garyfyFigure, end

subplot(2,3,5);
plot(t,u12,'b',t,u22,'r-.',t,u32,'g-');
grid on; xlim([0, Tf])
title('Step to Vbias');
if exist('garyfyFigure','file'), garyfyFigure, end


subplot(2,3,6);
plot(t,u13,'b',t,u23,'r-.',t,u33,'g-');
grid on; xlim([0, Tf])
legend('Power','Throttle','%O2','Location','Southeast');
title('Step to Vbias');
if exist('garyfyFigure','file'), garyfyFigure, end
```