Table of Contents

Spacecraft Launch Vehicle Attitude Control System Design

```
% Robust Multivariable Control, Spring 2018
% Authors: Jyot Buch, Alex Hayes, Sepehr Seyedi
% Group 14
```

Clear workspace and Setup Model

```
clear;clc;close all;
bdclose all;
format short g;

% Simulink Model to be used
model = 'LaunchVehicle';
load_system(model);

% Default: Set the Input to Step
set_param(sprintf([model '/InputSwitch']), 'sw', '1');

% Default: Set the Wind Disturbance to Step
set_param(sprintf([model '/DistSwitch']), 'sw', '1');

% Print the the model to PDF
set_param(model, 'PaperType', 'usletter')
set_param(model, 'PaperOrientation', 'landscape')
print(['-s',model],'-dpdf',model)
```

Overall Nominal LTI Plant Linearized Dynamics

```
% State Space
sys = ss(A,B,C,D);

% Transfer Function Matrix
fprintf('============================================================\n');
fprintf('### Transfer Function Matrix of 5th Order Full State Plant: \n');
fprintf('============================================================\n');
G = tf(sys);
```

```
============================================================
### Transfer Function Matrix of 5th Order Full State Plant:
============================================================
```

Nominal Open Loop Analysis

```
% Eigenvalues
[~,Devals] = eig(A);
fprintf('=================\n');
fprintf('### Eigenvalues: \n');
fprintf('=================\n');
disp(diag(Devals));

% Damping and Frequency of Oscillations
fprintf('=========================\n');
fprintf('### Properties of Poles: \n');
fprintf('=========================\n');
damp(sys)

% Controllability
fprintf('====================================\n');
fprintf('### Rank of Controllability Matrix: \n');
fprintf('====================================\n');

% Donot include the disturbance input
Bctrb = B(:,1);
Dctrb = D(:,1);
sysCTRB = ss(A,Bctrb,C,Dctrb);
disp(rank(ctrb(sysCTRB)));

% Observability
fprintf('==================================\n');
fprintf('### Rank of Observability Matrix: \n');
fprintf('==================================\n');
disp(rank(obsv(sys)));

% Minimal Realization
fprintf('=========================\n');
```

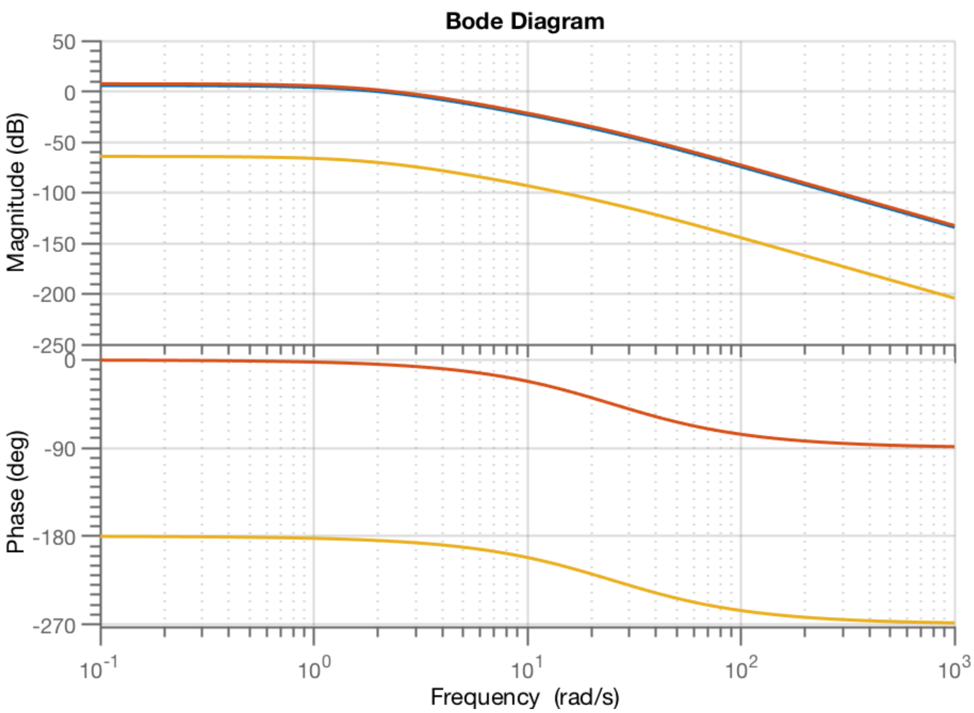
```
fprintf('### Minimal Realization: \n');
fprintf('=========================\n');
minsys = minreal(sys)

% Bode plot of Nominal Plant
h1 = figure;
h2 = figure;
h = {h1,h2};

for i = 1:3
    figure(h1);hold on;
    bode(G(1,i));

    figure(h2);hold on;
    bode(G(2,i));
end
for i = 1:2
    figure(h{i});
    set(findall(gcf,'type','line'),'LineWidth',3);
    grid on;set(findall(0,'type','axes'),'box','off');
    set(gcf,'PaperOrientation','landscape')
    print(sprintf('bodeOpenLoop%d',i),'-depsc');
end

% Singular values vs frequency for Nominal Plant
figure;
[sv,w] = sigma(sys);
semilogx(w,sv);
set(findall(gcf,'type','line'),'LineWidth',3);
axis tight
grid on;set(findall(0,'type','axes'),'box','off');
xlabel('Frequency (rad/sec)');
ylabel('Sigma');
title('Open Loop Singular Values');
print(sprintf('SigmaVsFreq'),'-depsc');

% Singular value decomposition for Nominal Plant
fprintf('==================================\n');
fprintf('### Singular Value Decomposition: \n');
fprintf('==================================\n');
[U,~,V] = svd(A);

% Open Loop Zeros
fprintf('=====================\n');
fprintf('### Open Loop Zeros: \n');
fprintf('=====================\n');
ol_zeros = tzero(sys) ;
disp(' ')
if( isempty(ol_zeros) )
    disp('No finite zeros of v/u')
else
    disp('Zeros of v/u')
    rifd(ol_zeros)
end
```

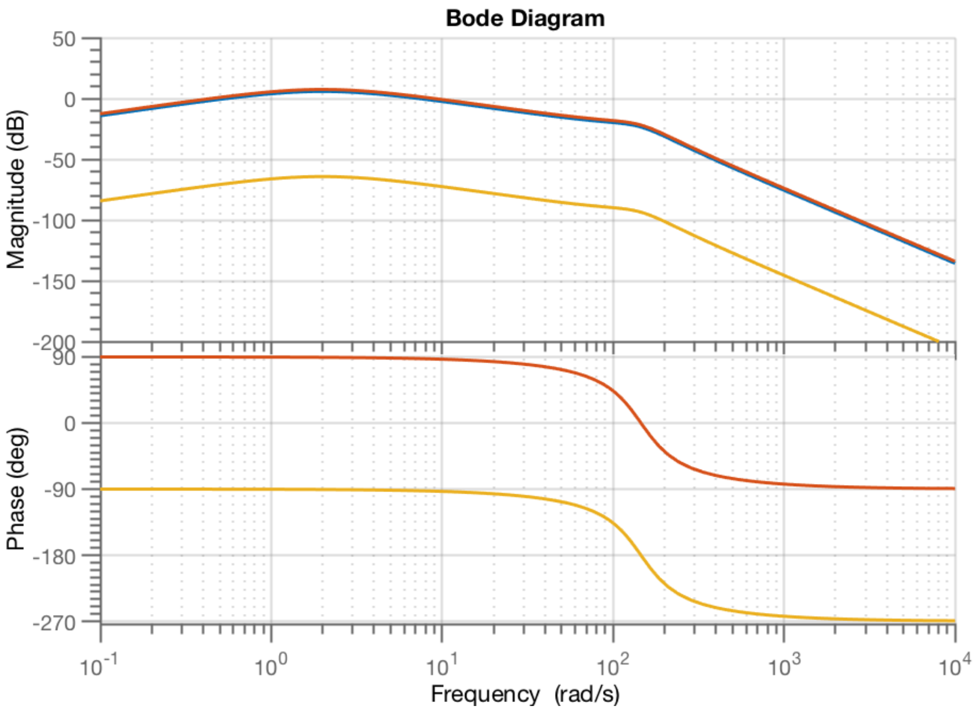
```
=================
### Eigenvalues:
=================
          -25 +          0i
       -58.24 +     133.44i
       -58.24 -     133.44i
       1.9922 +          0i
      -2.0002 +          0i

=========================
### Properties of Poles:
=========================
```

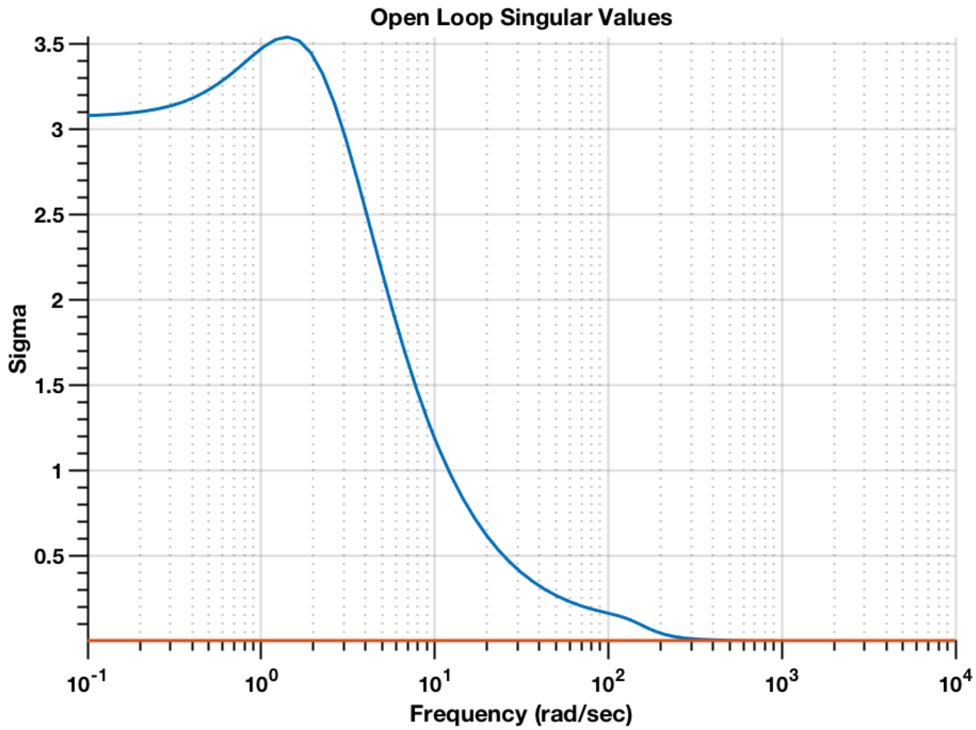
Pole	Damping	Frequency (rad/seconds)	Time Constant (seconds)
1.99e+00	-1.00e+00	1.99e+00	-5.02e-01
-2.00e+00	1.00e+00	2.00e+00	5.00e-01
-2.50e+01	1.00e+00	2.50e+01	4.00e-02
-5.82e+01 + 1.33e+02i	4.00e-01	1.46e+02	1.72e-02
-5.82e+01 - 1.33e+02i	4.00e-01	1.46e+02	1.72e-02

```
====================================
### Rank of Controllability Matrix:
====================================
     5

==================================
### Rank of Observability Matrix:
==================================
     5

=========================
### Minimal Realization:
=========================

minsys =

  A =
              x1         x2         x3         x4         x5
   x1          0          1          0          0          0
   x2      3.985   -0.00794          0          0          0
   x3         25          0        -25          0          0
   x4          0          0          0          0          1
   x5          0   2.12e+04          0  -2.12e+04     -116.5
```

```
  B =
             u1        u2        u3
   x1         0         0         0
   x2    -7.824    -9.413   0.00245
   x3         0         0         0
   x4         0         0         0
   x5         0         0         0

  C =
       x1  x2  x3  x4  x5
   y1   0   0   1   0   0
   y2   0   0   0   1   0

  D =
       u1  u2  u3
   y1   0   0   0
   y2   0   0   0

Continuous-time state-space model.

==================================
### Singular Value Decomposition:
==================================
=====================
### Open Loop Zeros:
=====================

No finite zeros of v/u
```







Reduced Order Model with 2 states

```
% State-Space object of approximated system
sys2 = ss(A2,B2,C2,D2);
```

```
% You can exclude sensor dynamics from the simulink model by doing
thetaPGnum = 1;
thetaPGden = 1;
thetaRGnum = 1;
thetaRGden = 1;
```

Proportional Feedback Control Design

```
% Calculate Gains with LQR
Q2 = diag([1,1]);
R2 = 1;
K2 = lqr(sys2,Q2,R2);
fprintf('=============================================\n');
fprintf('### LQR Control Gains (Approximated System): \n');
fprintf('=============================================\n');
k1 = K2(1) %#ok<*NASGU>
k2 = K2(2)

% Use Pole Placement instead of LQR to manually tune gains
fprintf('=============================================\n');
fprintf('### Pole Placement Gains (Approximated System): \n');
fprintf('=============================================\n');
k1 = -3
k2 = -1.3
K2 = [k1 k2];

% Closed loop A matrix for approximated system
A2_cl = A2-B2*K2;
fprintf('==================================\n');
fprintf('### Closed Loop Damping Ratio: \n');
fprintf('==================================\n');
rifd(eig(A2_cl))

% Closed loop Characteristic Polynomial of Approximated System with P
 Control
phi_cl = vpa(charpoly(A2_cl));
```

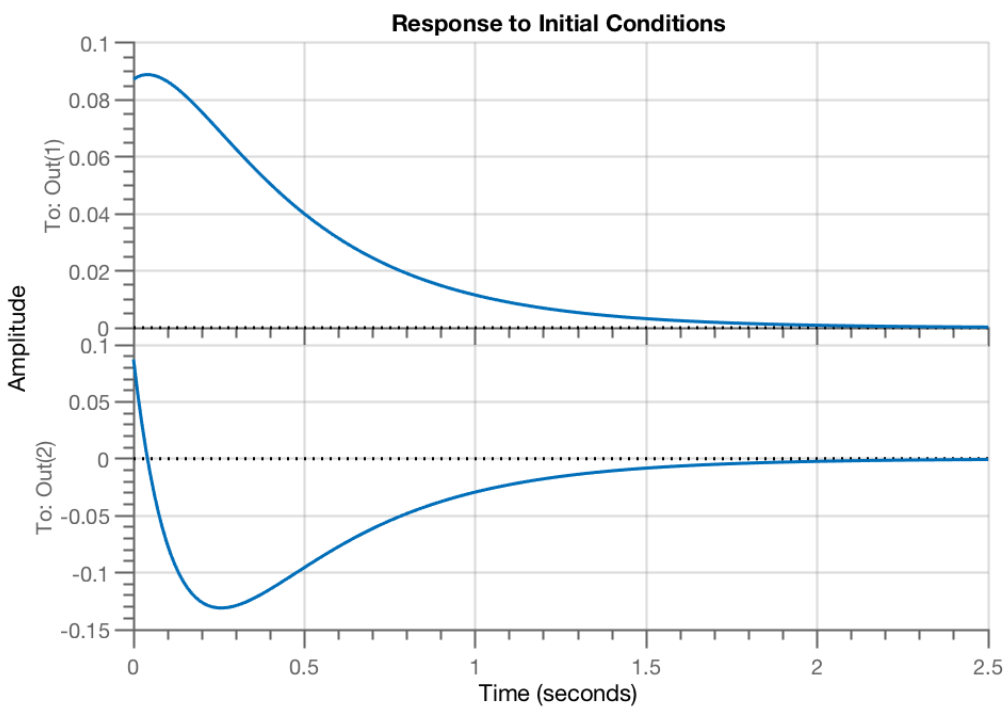
```
=============================================
### LQR Control Gains (Approximated System):
=============================================

k1 =

      -1.6316


k2 =

      -1.1894

=============================================
### Pole Placement Gains (Approximated System):
=============================================
```

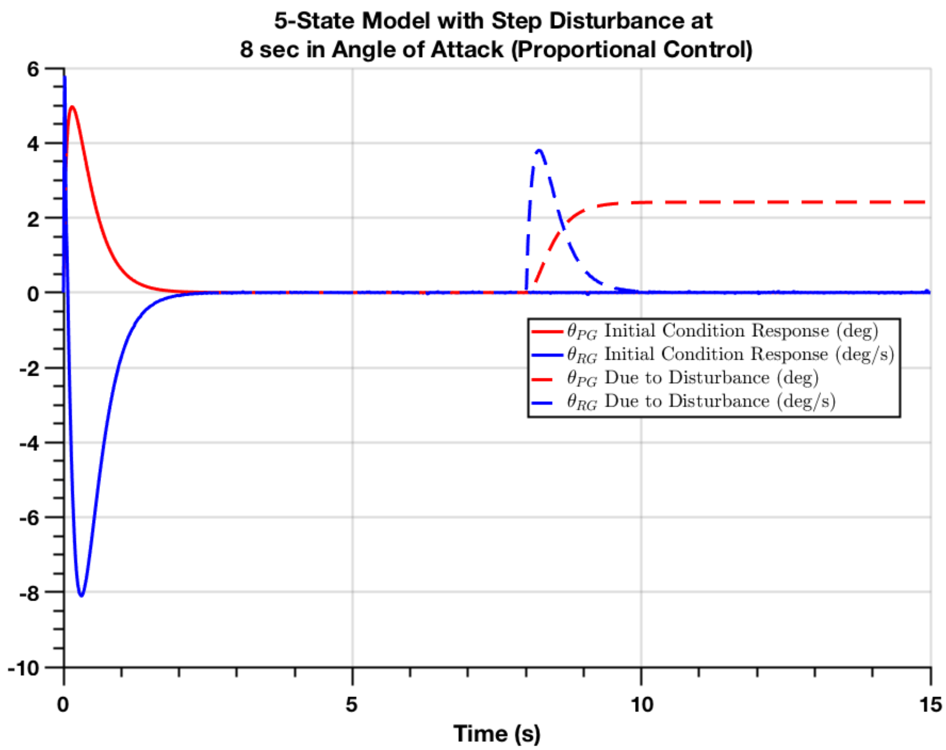
```
k1 =

    -3


k2 =

         -1.3

==================================
### Closed Loop Damping Ratio:
==================================

      real        imaginary      frequency       damping

  -7.6220e+00    0.0000e+00     7.6220e+00    1.0000e+00
  -2.5565e+00    0.0000e+00     2.5565e+00    1.0000e+00
```

Evaluate Closed Loop Response with Proportional Feedback Control

```
% Simulate Initial Condition Response for Approximated System with P
 Control
sys2_cl = ss(A2_cl,B2,C2,D2);
theta0 = deg2rad(5);
thetadot0 = deg2rad(5);
ic = [theta0 thetadot0]';
initial(sys2_cl,ic);
grid on;
set(findall(gcf,'type','line'),'LineWidth',3);
set(findall(0,'type','axes'),'box','off');

% Show that Disturbance Rejection is not good with Proportional
 controller 5 state

% Setup Simulink Model Parameters
defaultModelParams;
Kp2 = K2;
Ki2 = [0 0];
theta0 = deg2rad(5);
thetadot0 = deg2rad(5);
tSim = 15;

% Simulate
sim(model);
initResp = y;
tInit = t;

% Add disturbance now and Simulate
alphaDisturbanceValue = deg2rad(-5);
alphaStepTime = 8;
```

```
sim(model);
disturbanceResp = y;
t1 = t;

% Plot Steady State Error due to Disturbance
figure;
plot(tInit,rad2deg(initResp(:,1)),'r-');hold on;
plot(tInit,rad2deg(initResp(:,2)),'b-');
plot(t1,rad2deg(disturbanceResp(:,1)),'r--');
plot(t1,rad2deg(disturbanceResp(:,2)),'b--');
legend('$\theta_{PG}$ Initial Condition Response (deg)',...
    '$\theta_{RG}$ Initial Condition Response (deg/s)',...
    '$\theta_{PG}$ Due to Disturbance (deg)',...
    '$\theta_{RG}$ Due to Disturbance (deg/s)');
title(sprintf('5-State Model with Step Disturbance at \n8 sec in Angle
 of Attack (Proportional Control)'))
set(findall(gcf,'type','line'),'LineWidth',3);
grid on;set(findall(0,'type','axes'),'box','off');
xlabel('Time (s)');
print(sprintf('disturbanceRejectionPControl'),'-depsc');
```





Proportional + Integral Feedback Control

```
% Calculate Gains with LQI
Q2 = diag([1,1,1,1]);
R2 = 1;
K2 = lqi(sys2,Q2,R2);
fprintf('============================================\n');
fprintf('### LQI Control Gains (Approximated System): \n');
fprintf('============================================\n');

% Gains calculated by LQI
kp1 = K2(1)
kp2 = K2(2)
ki1 = -K2(3)
ki2 = -K2(4)
Kp2 = [kp1 kp2];
Ki2 = [ki1 ki2];

% Use Pole Placement instead of LQI
fprintf('=========================================================
\n');
fprintf('### Pole Placement Controller Gains (Approximated System):
 \n');
fprintf('=========================================================
\n');
kp1 = -3
kp2 = -1.3
ki1 = -2
```

```
ki2 = 0
Kp2 = [kp1 kp2];
Ki2 = [ki1 ki2];
Kp5 = [Kp2;zeros(2)];
Ki5 = [Ki2;zeros(2)];

=============================================
### LQI Control Gains (Approximated System):
=============================================

kp1 =

      -2.7144


kp2 =

      -1.3005


ki1 =

           -1


ki2 =

   -2.227e-07

===========================================================
### Pole Placement Controller Gains (Approximated System):
===========================================================

kp1 =

    -3


kp2 =

         -1.3


ki1 =

    -2


ki2 =

     0
```

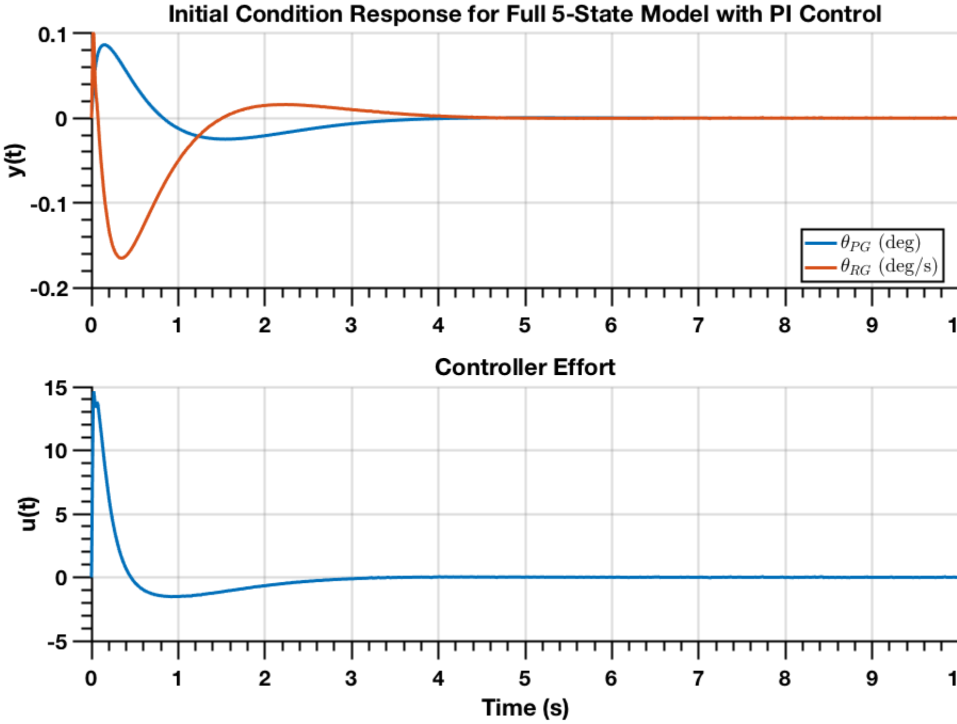
Evaluate Closed Loop Response with Proportional+Integral Feedback Control

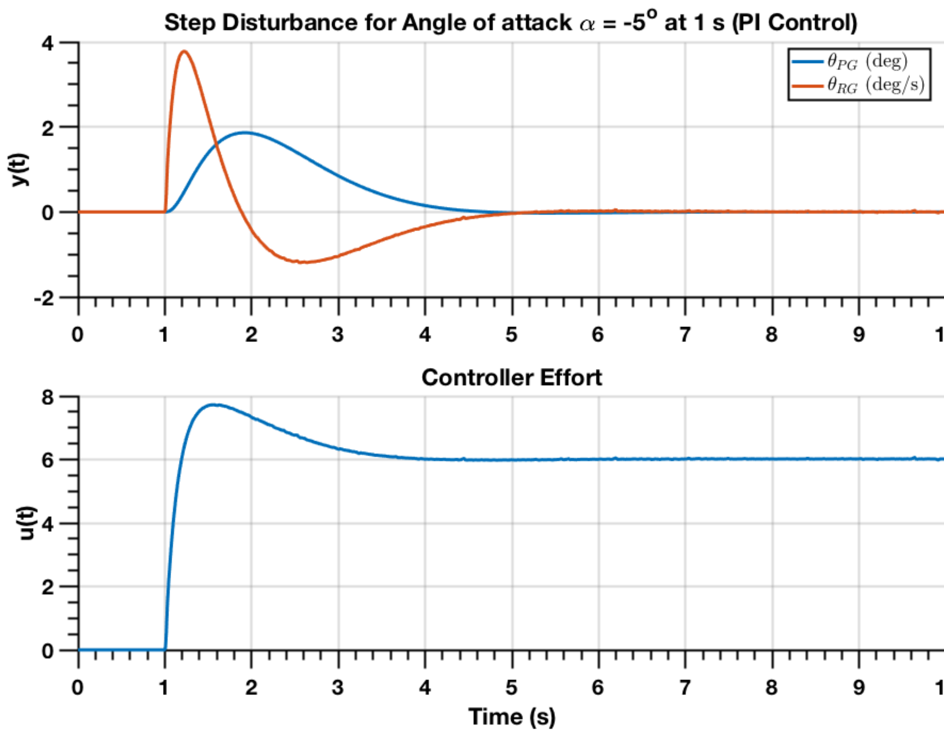
```
%
 ======================================================================
% 1) Simulate Initial Condition Response with 5-State System (Closed
 loop)
%
 ======================================================================

% Setup Simulink Model Parameters
defaultModelParams;
theta0 = deg2rad(5); % rad
thetadot0 = deg2rad(5); % rad/sec

% Simulate
sim(model);

% Plot
figure;
subplot(2,1,1);
plot(t,y);
title('Initial Condition Response for Full 5-State Model with PI
 Control');
legend('$\theta_{PG}$ (deg)','$\theta_{RG}$ (deg/s)');
ylabel('y(t)');
set(findall(0,'type','axes'),'box','off');

subplot(2,1,2);
plot(t,rad2deg(u));
set(findall(0,'type','axes'),'box','off');
xlabel('Time (s)');
ylabel('u(t)');
title('Controller Effort');
print(sprintf([model '_initialCond']),'-depsc');

%
 ======================================================================
% 2) Disturbance in Angle of Attack
%
 ======================================================================

% Setup Simulink Model Parameters
defaultModelParams
alphaDisturbanceValue = deg2rad(-5);
alphaStepTime = 1;

% Simulate
sim(model);

% Plot
figure;
```

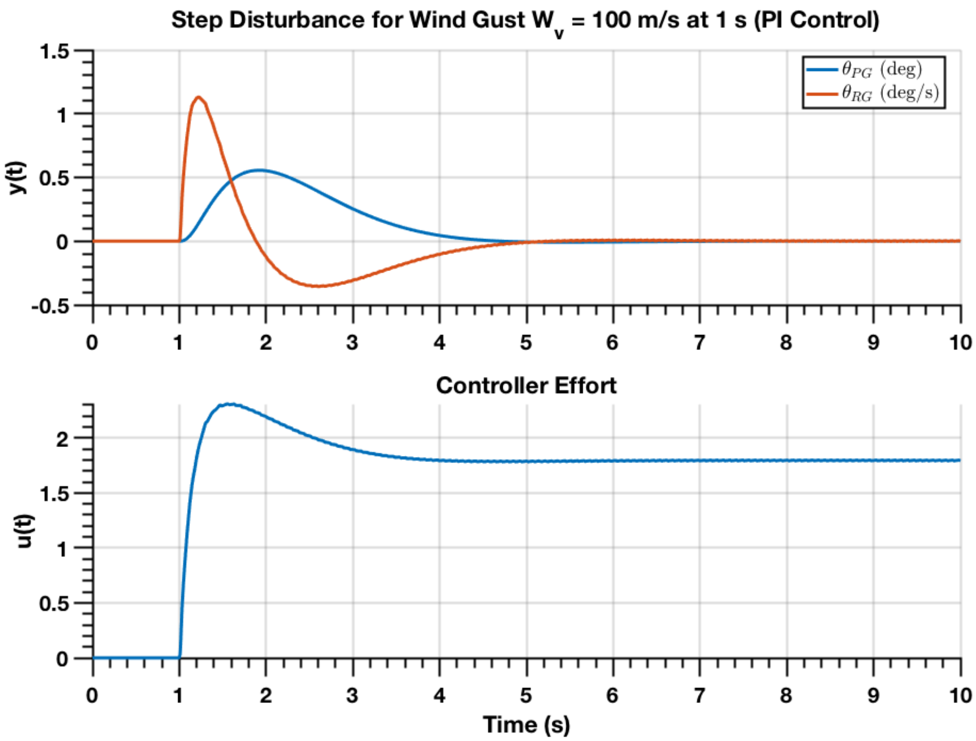
```
subplot(2,1,1);
plot(t,rad2deg(y));
set(findall(0,'type','axes'),'box','off');
title('Step Disturbance for Angle of attack \alpha = -5^{o} at 1 s (PI
 Control)');
legend('$\theta_{PG}$ (deg)','$\theta_{RG}$ (deg/s)');
ylabel('y(t)');

subplot(2,1,2);
plot(t,rad2deg(u));
set(findall(0,'type','axes'),'box','off');
xlabel('Time (s)');ylabel('u(t)');
title('Controller Effort');
print(sprintf([model '_DisturbanceInAlpha']),'-depsc');

%
 ========================================================================
% 3) Disturbance in Wind
%
 ========================================================================

% Setup Simulink Model Parameters
defaultModelParams;
windStepTime = 1;
windDisturbanceValue = 100; % m/s

% Simulate
sim(model);

% Plot
figure;
subplot(2,1,1)
plot(t,rad2deg(y));
set(findall(0,'type','axes'),'box','off');
title('Step Disturbance for Wind Gust W_v = 100 m/s at 1 s (PI
 Control)');
legend('$\theta_{PG}$ (deg)','$\theta_{RG}$ (deg/s)');
ylabel('y(t)');

subplot(2,1,2);
plot(t,rad2deg(u));
set(findall(0,'type','axes'),'box','off');
xlabel('Time (s)');ylabel('u(t)');
title('Controller Effort');
print(sprintf([model '_DisturbanceInWind']),'-depsc');

%
 ========================================================================
% 4) Step Response and Time Domain Performance
%
 ========================================================================

% Setup Simulink Model Parameters
defaultModelParams;
```

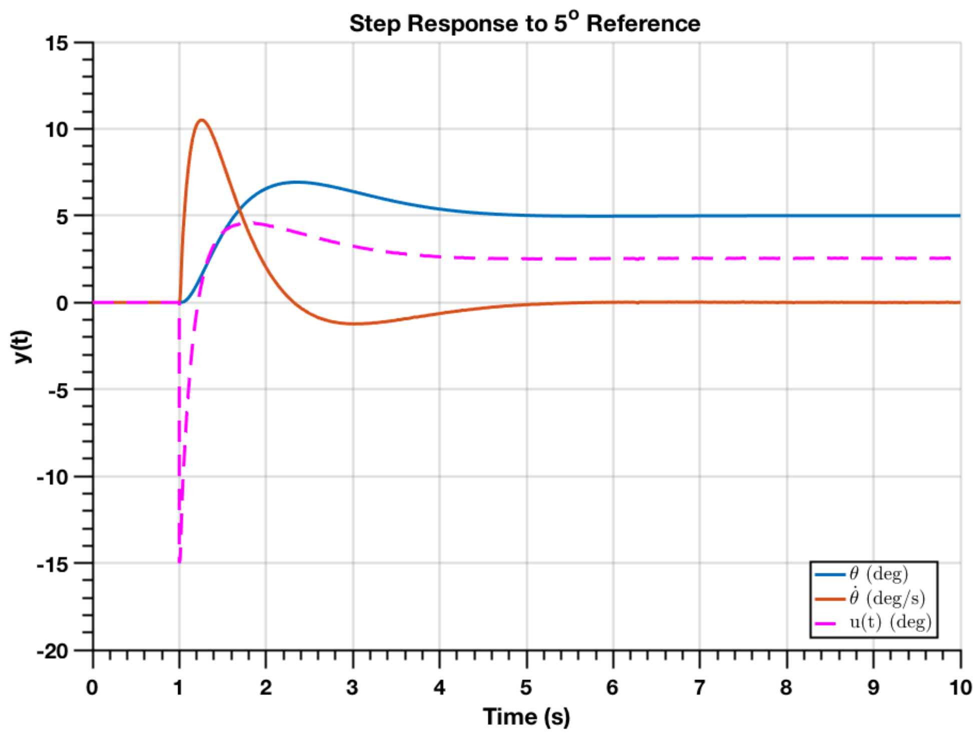
```
stepTime = 1;
thetaStep = deg2rad(5);

% Simulate
sim(model);

% Plot
figure;
plot(t,rad2deg(y));
set(findall(0,'type','axes'),'box','off');
hold on;
plot(t,rad2deg(u),'--m');
xlabel('Time (s)');ylabel('y(t)');
title('Step Response to 5^{o} Reference');
legend('$\theta$ (deg)','$\dot{\theta}$ (deg/s)','u(t) (deg)');
print(sprintf([model '_StepResponse']),'-depsc');

% Display Results
fprintf('=============================\n');
fprintf('### Time Domain Perfromance: \n');
fprintf('=============================\n');
stepInfo = stepinfo(rad2deg(y(:,1)),t)
```

```
=============================
### Time Domain Perfromance:
=============================

stepInfo = 

  struct with fields:

        RiseTime: 0.42797
    SettlingTime: 4.5917
     SettlingMin: 4.5134
     SettlingMax: 6.9247
       Overshoot: 38.485
      Undershoot: 0
            Peak: 6.9247
        PeakTime: 2.3465
```


Initial Condition Response for Full 5-State Model with PI Control
y(t)
θ_{PG} (deg)
θ_{RG} (deg/s)
Controller Effort
u(t)
Time (s)


Step Disturbance for Angle of attack α = -5^{o} at 1 s (PI Control)
y(t)
θ_{PG} (deg)
θ_{RG} (deg/s)
Controller Effort
u(t)
Time (s)





Nominal Closed Loop Analysis

```
%
 ======================================================================
```

```
% 1) Input-Loop and Output-Loop Transfer Functions for Reduced Order
 Model
%
 ======================================================================

% Input loop state space matrices
A_Li_2 = [zeros(2) eye(2);zeros(2) A2];
B_Li_2 = [zeros(2,1); B2];
C_Li_2 = [Ki2 Kp2];
D_Li_2 = 0;

% Loop state space matrices
A_L_2 = [A2 B2*Ki2; zeros(2,4)];
B_L_2 = [B2*Kp2; eye(2)];
C_L_2 = [eye(2),zeros(2,2)];
D_L_2 = zeros(2);

% Loop and Input Loop TFs
L_2 = tf(ss(A_L_2,B_L_2,C_L_2,D_L_2));
Li_2 = tf(ss(A_Li_2,B_Li_2,C_Li_2,D_Li_2));

% Bode plot
figure;
margin(Li_2);
set(findall(gcf,'type','line'),'LineWidth',3);
grid on;set(findall(0,'type','axes'),'box','off');
print(sprintf('InputLoopBode_2states'),'-depsc');

%
 ======================================================================
% 2) Closed loop Characteristic Polynomial for Reduced Order Model
%
 ======================================================================

A_cl_2 = A_L_2-B_L_2*inv(eye(2)+D_L_2)*C_L_2; %#ok<*MINV>

fprintf('=======================================================================
\n');
fprintf('### Characteristic Polynomial Coefficients (Approximated
 System): \n');
fprintf('=======================================================================
\n');
poly(A_cl_2) % poles_2 = roots(poly(A_cl_2));

fprintf('====================================================================
\n');
fprintf('### Damping Ratio and Natural Frequency (Approximated
 System): \n');
fprintf('====================================================================
\n');
damp(ss(A_cl_2,zeros(4,4),zeros(4,4),zeros(4,4)))

%
 =======================================================================
```

```
% 3) Input-Loop and Output-Loop Transfer Functions for Full 5th Order
 Model
%
 ========================================================================
A_Li_5 = [zeros(2,2) C; zeros(5,2) A];
B_Li_5 = [zeros(2,3);B];
C_Li_5 = [Ki5 Kp5*C];
D_Li_5 = zeros(3,3);

A_L_5 = [A B*Ki5;zeros(2,5) zeros(2,2)];
B_L_5 = [B*Kp5;eye(2)];
C_L_5 = [C zeros(2,2)];
D_L_5 = zeros(2);

L_5 = tf(ss(A_L_5,B_L_5,C_L_5,D_L_5));
Li_5 = tf(ss(A_Li_5,B_Li_5,C_Li_5,D_Li_5));

% Bode plot
figure;
margin(Li_5(1,1));
set(findall(gcf,'type','line'),'LineWidth',2)
grid on;set(findall(0,'type','axes'),'box','off');
print(sprintf('bode_5states1'),'-depsc');

%
 =======================================================================
% 4) Characteristic Polynomial Coefficients of full 5th order system
%
 =======================================================================
% Closed loop Characteristic Polynomial of 5-State System
A_cl_5 = A_L_5-B_L_5*inv(eye(2)+D_L_5)*C_L_5;
fprintf('============================================================
\n')
fprintf('### Characteristic Polynomial Coefficients (5th Order
 System):\n')
fprintf('============================================================
\n')
poly(A_cl_5)
poles_5 = roots(poly(A_cl_5));

fprintf('============================================================
\n');
fprintf('### Damping Ratio and Natural Frequency (5th Order System):
 \n');
fprintf('============================================================
\n');
damp(A_cl_5);

%
 =======================================================================
% 5) Closed loop eigenvalues of full 5th order system from Simulink
%
 =======================================================================
```

```
fprintf('=================================================================
\n')
fprintf('### Closed Loop Eigenvalues from Simulink (5th Order System):
 \n')
fprintf('=================================================================
\n')
[A_cl,B_cl,C_cl,D_cl] = linmod(model);
cltf = tf(ss(A_cl,B_cl,C_cl,D_cl));
damp(A_cl)
```

```
==================================================================
### Characteristic Polynomial Coefficients (Approximated System):
==================================================================

ans =

            1       10.178       19.486       15.647            0

===============================================================
### Damping Ratio and Natural Frequency (Approximated System):
===============================================================


         Pole                Damping       Frequency      Time Constant

                                         (rad/seconds)      (seconds)



  0.00e+00                  -1.00e+00       0.00e+00              Inf

 -1.10e+00 + 8.69e-01i       7.84e-01       1.40e+00         9.11e-01

 -1.10e+00 - 8.69e-01i       7.84e-01       1.40e+00         9.11e-01

 -7.98e+00                   1.00e+00       7.98e+00         1.25e-01

==============================================================
### Characteristic Polynomial Coefficients (5th Order System):
==============================================================

ans =

  Columns 1 through 6

            1       141.49        24108   7.4581e+05   5.3671e+06
 1.0373e+07

  Columns 7 through 8

   8.2927e+06            0

============================================================
### Damping Ratio and Natural Frequency (5th Order System):
```

==

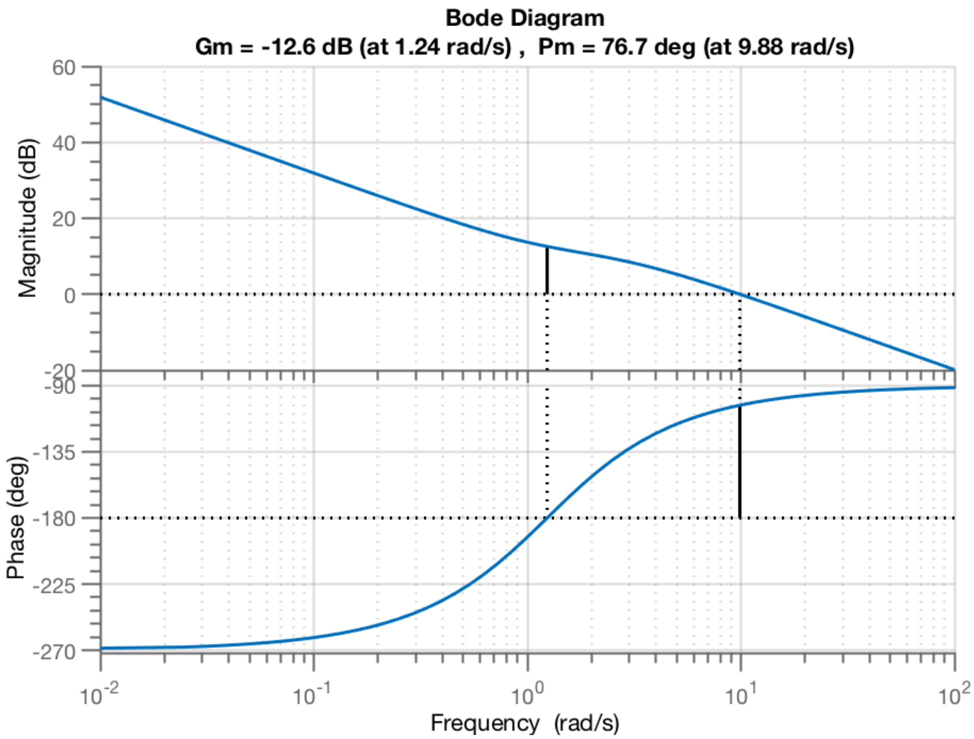
Pole	Damping	Frequency (rad/TimeUnit)	Time Constant (TimeUnit)
0.00e+00	-1.00e+00	0.00e+00	Inf
-5.29e+01 + 1.31e+02i	3.73e-01	1.42e+02	1.89e-02
-5.29e+01 - 1.31e+02i	3.73e-01	1.42e+02	1.89e-02
-2.65e+01	1.00e+00	2.65e+01	3.78e-02
-6.88e+00	1.00e+00	6.88e+00	1.45e-01
-1.21e+00 + 9.03e-01i	8.00e-01	1.51e+00	8.29e-01
-1.21e+00 - 9.03e-01i	8.00e-01	1.51e+00	8.29e-01

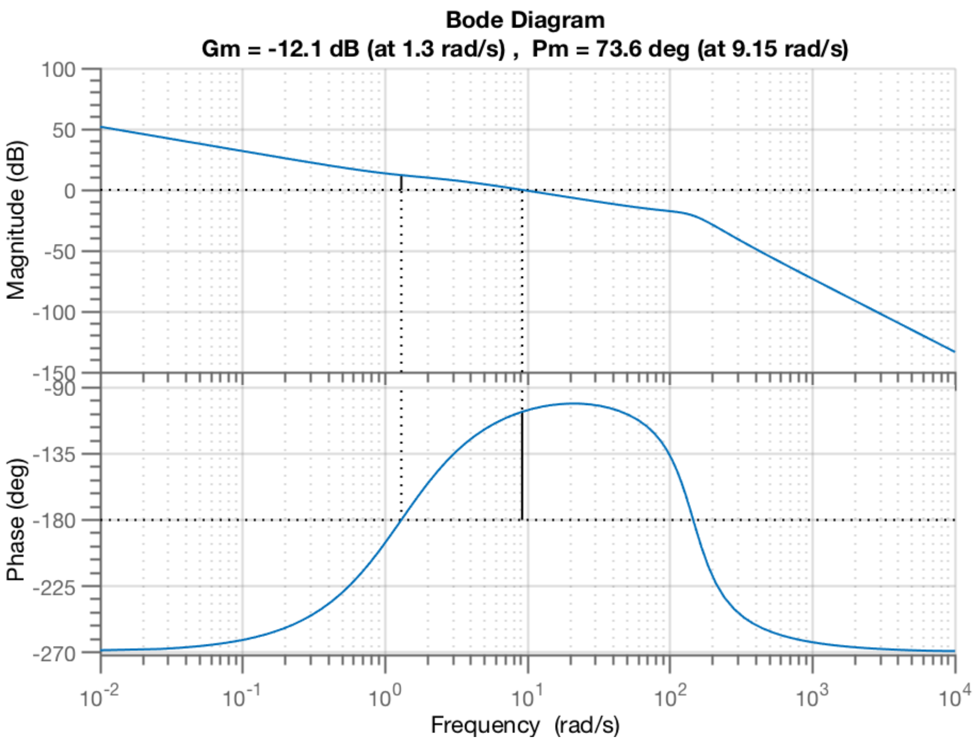
==

Closed Loop Eigenvalues from Simulink (5th Order System):

==

Pole	Damping	Frequency (rad/TimeUnit)	Time Constant (TimeUnit)
-5.29e+01 + 1.31e+02i	3.73e-01	1.42e+02	1.89e-02
-5.29e+01 - 1.31e+02i	3.73e-01	1.42e+02	1.89e-02
-1.21e+00 + 9.03e-01i	8.00e-01	1.51e+00	8.29e-01
-1.21e+00 - 9.03e-01i	8.00e-01	1.51e+00	8.29e-01
-6.88e+00	1.00e+00	6.88e+00	1.45e-01
-2.65e+01	1.00e+00	2.65e+01	3.78e-02

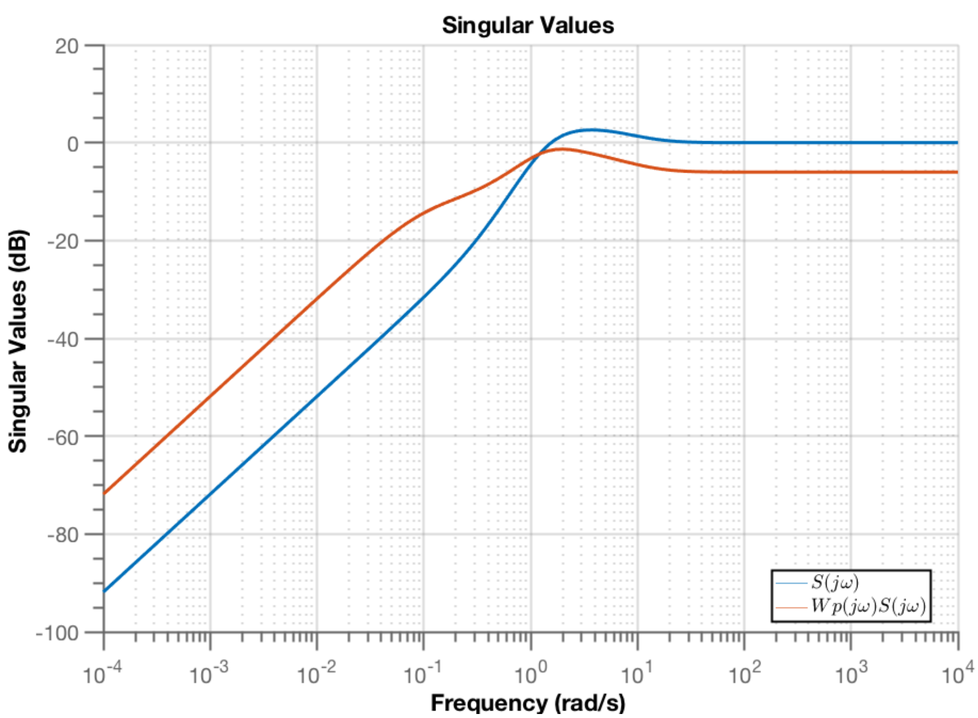




Nominal Performance Test

```
% Get Wb = Minimum Bandwidth Frequency
w = {10^-4,10^4};
S_5 = minreal(inv(eye(2) + L_5));
```

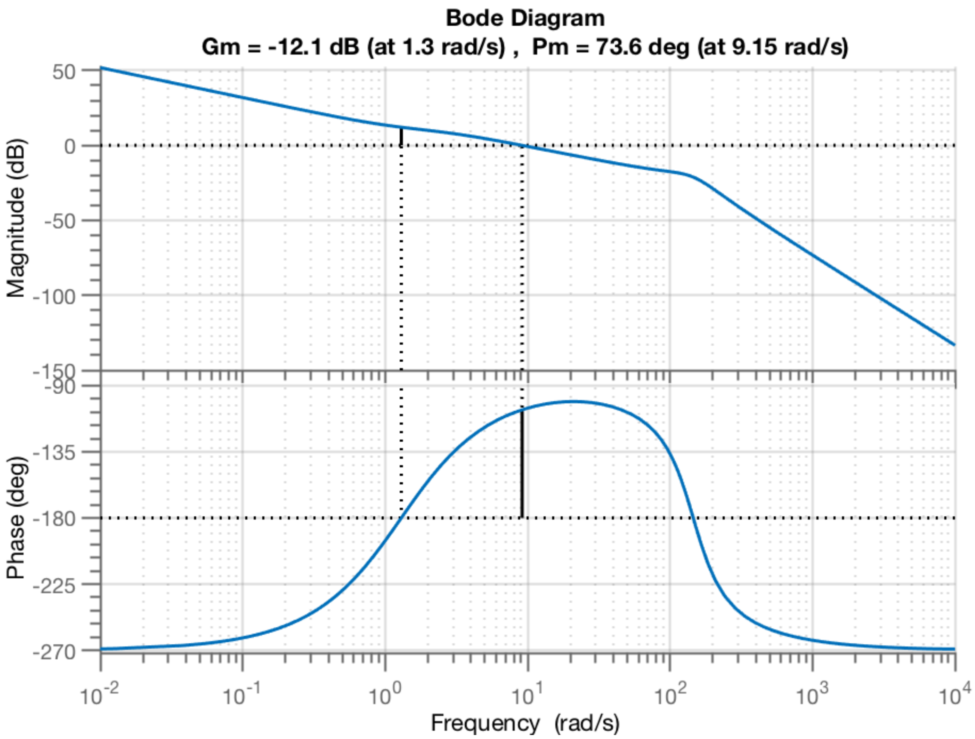
```
% Sensitivity is transfer function related to 4th input and 4th output
S11 = cltf(4,4);

% Plot
figure;
sigma(S11,w);
grid on;set(findall(0,'type','axes'),'box','off');

% Set the bandwidth frequency as per the specification
WbStar = 1.05;

% Weighted Sensitivity Transfer Function
As = 0.1; % Based on the low frequency asymptote
Ms = 2;
Wp = tf([1/Ms WbStar],[1 As*WbStar]);

% Plot
hold on;
sigma(Wp*S11,w);
% sigma(S_5(1,1),w)
set(findall(gcf,'type','line'),'LineWidth',3);
legend('$S(j\omega)$','$Wp(j\omega)S(j\omega)$')
print(sprintf('sigmaFreqPlotForS11'),'-depsc');
```



Single loop at a time analysis for breaking loop at 1-Input

```
% Loop Transfer Function
L_SLAT_U = -inv(1+cltf(1,1))*cltf(1,1);

% Bode Plot
figure;
margin(L_SLAT_U)
set(findall(gcf,'type','line'),'LineWidth',3);
grid on;set(findall(0,'type','axes'),'box','off');
print(sprintf('InputLoopBode_SLAT_U'),'-depsc');

% All margin
fprintf('==========================================================
\n')
fprintf('### Single Loop at a time loop TF (Loop Broken at Input):
 \n')
fprintf('==========================================================
\n')
allmargin(L_SLAT_U)
```

```
==========================================================
### Single Loop at a time loop TF (Loop Broken at Input):
==========================================================

ans =

  struct with fields:

     GainMargin: [0.24779 11.432]
    GMFrequency: [1.2977 145.48]
    PhaseMargin: 73.587
    PMFrequency: 9.1495
    DelayMargin: 0.14037
    DMFrequency: 9.1495
         Stable: 1
```

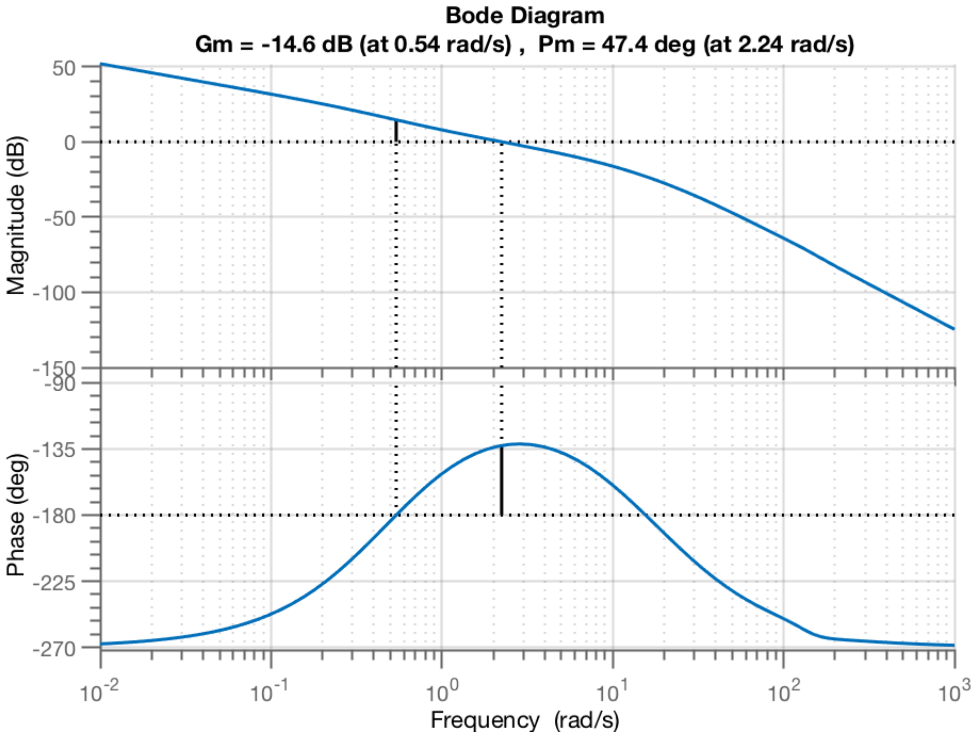


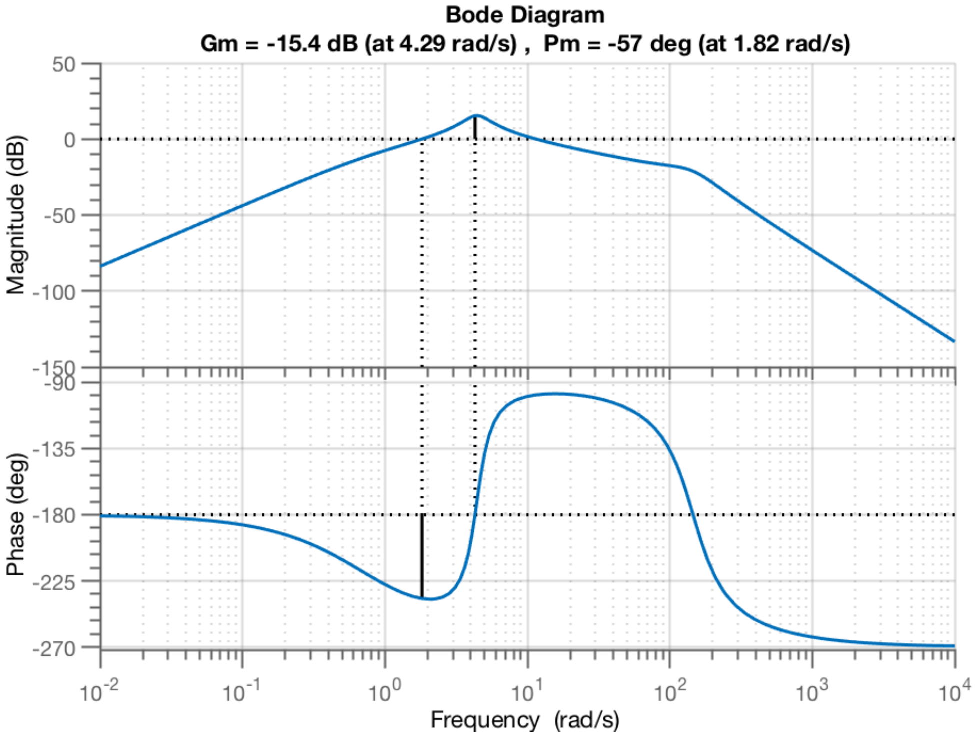
Single loop at a time analysis for breaking loop at 2-Sensors

```
% Loop Transfer Function
L_SLAT_Y1 = -inv(1+cltf(2,2))*cltf(2,2);
L_SLAT_Y2 = -inv(1+cltf(3,3))*cltf(3,3);

% Bode Plot
figure;
margin(L_SLAT_Y1)
set(findall(gcf,'type','line'),'LineWidth',3);
grid on;set(findall(0,'type','axes'),'box','off');
print(sprintf('InputLoopBode_SLAT_Y1'),'-depsc');

figure;
margin(L_SLAT_Y2)
set(findall(gcf,'type','line'),'LineWidth',3);
grid on;set(findall(0,'type','axes'),'box','off');
print(sprintf('InputLoopBode_SLAT_Y2'),'-depsc');

% All margin
fprintf('============================================================
\n')
fprintf('### Single Loop at a time loop TF (Loop Broken at Output1):
 \n')
fprintf('============================================================
\n')
```

```
allmargin(L_SLAT_Y1)
fprintf('============================================================
\n')
fprintf('### Single Loop at a time loop TF (Loop Broken at Output2):
 \n')
fprintf('============================================================
\n')
allmargin(L_SLAT_Y2)
```

```
============================================================
### Single Loop at a time loop TF (Loop Broken at Output1):
============================================================

ans =

  struct with fields:

     GainMargin: [0.18658 14.131]
    GMFrequency: [0.54009 15.568]
    PhaseMargin: 47.366
    PMFrequency: 2.2381
    DelayMargin: 0.36937
    DMFrequency: 2.2381
         Stable: 1

============================================================
### Single Loop at a time loop TF (Loop Broken at Output2):
============================================================

ans =

  struct with fields:

     GainMargin: [1.145e+15 0.17014 11.453]
    GMFrequency: [3.6655e-08 4.2874 145.59]
    PhaseMargin: [-57.006 81.508]
    PMFrequency: [1.8201 11.474]
    DelayMargin: [2.9054 0.12398]
    DMFrequency: [1.8201 11.474]
         Stable: 1
```

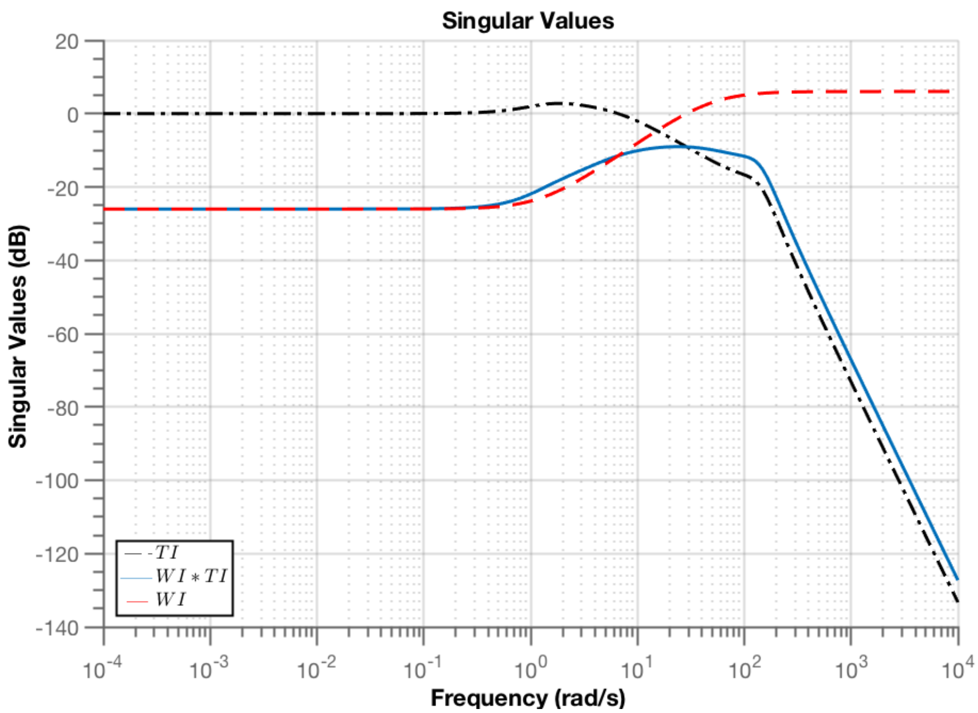




Define Weighting Transfer Function WI(s) and Test Robust Stability

```
tau = 1/25;
```

```
r0 = 0.05;
rInf = 2;
WI = tf([tau r0],[tau/rInf 1]);

% Input Complementory Sensitivity
TI = minreal(Li_5*inv(eye(3)+Li_5));
TI11 = TI(1,1); % Identical with cltf(1,1)

% Plot singular value vs w
figure;
sigma(TI11,w,'-.k');
hold on;
sigma(WI*TI11,w);
sigma(WI,w,'--r');
legend('$TI$','$WI*TI$','$WI$')
set(findall(gcf,'type','line'),'LineWidth',3);
grid on;set(findall(0,'type','axes'),'box','off');
print(sprintf('SigmaOfWiTi'),'-depsc');
```



Robust Performance Test

```
% Get M from simulink model
[A_cl_M,B_cl_M,C_cl_M,D_cl_M] = linmod(model);
cltf_M = minreal(tf(ss(A_cl_M,B_cl_M,C_cl_M,D_cl_M)));

% Define M matrixc
M = [cltf_M(1,1) cltf_M(4,1); cltf_M(1,4) cltf_M(4,4)];
ww = logspace(-4,4,1000);
```

```
% Consider SISO insted
sigmaWpS = sigma(Wp*S11,ww);
sigmaWIT = sigma(WI*TI11,ww);
total = sigmaWpS + sigmaWIT;
fprintf('================================================\n')
fprintf('### SISO Robust Performance |Wp*S| + |WI*T| < 1:\n')
fprintf('================================================\n')
disp(max(total));
```

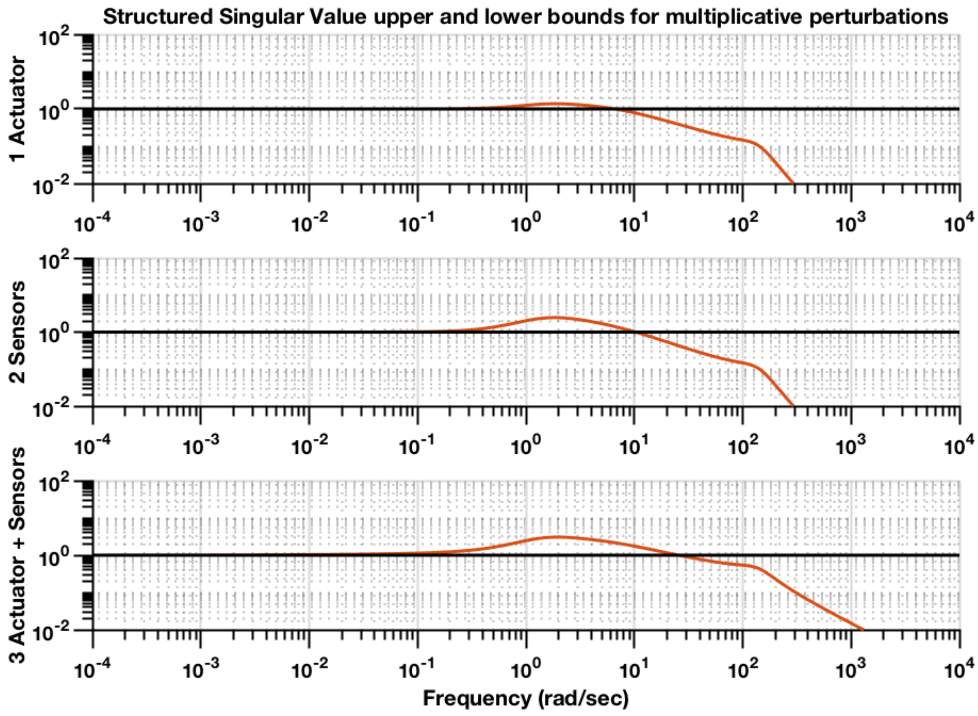
```
================================================
### SISO Robust Performance |Wp*S| + |WI*T| < 1:
================================================
      0.99251
```

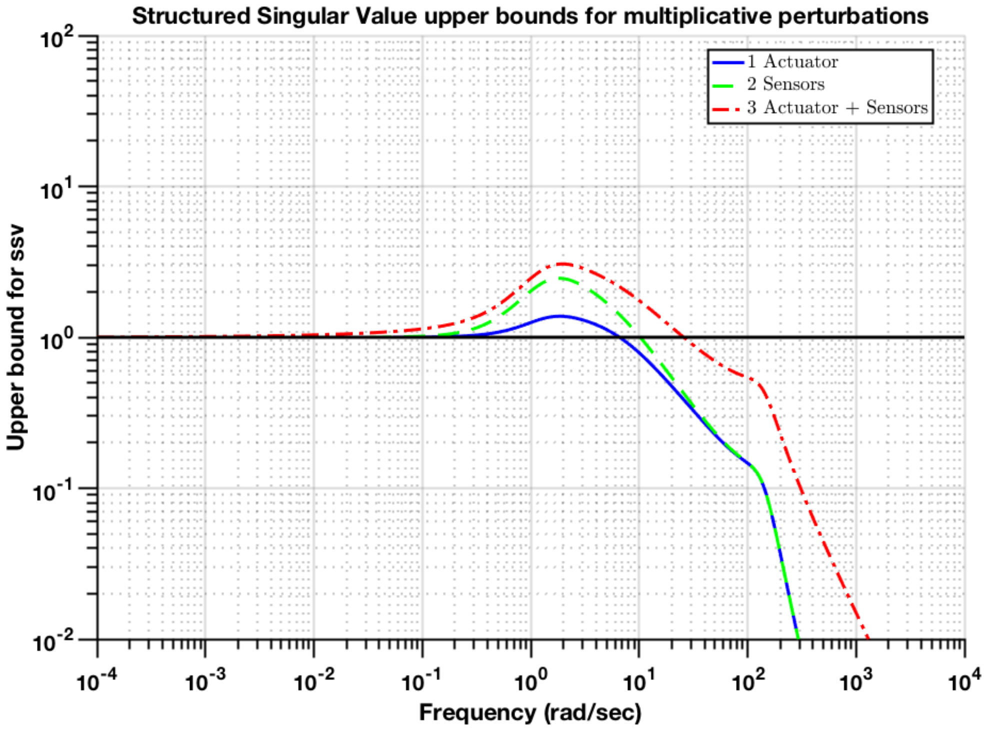
RS Analysis from code provided in class

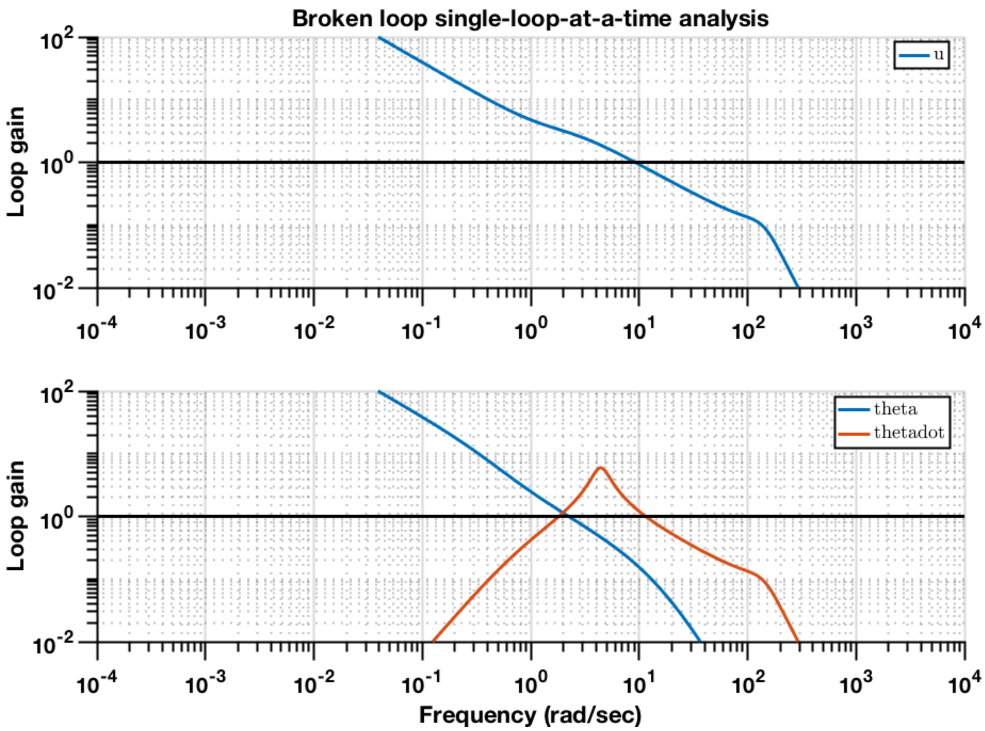
```
fprintf('================================\n')
fprintf('### Robust Stability Analysis: \n')
fprintf('================================\n')

% Set Wp and WI to be static Gain of 1
Wp = tf(1,1);
WI = tf(1,1);

% Get the state-space
[A_cl,B_cl,C_cl,D_cl] = linmod(model);
cltf = tf(ss(A_cl,B_cl,C_cl,D_cl));

% Define inputs
u_names = {'u'} ;
y_names = {'theta','thetadot'} ;
do_prt = 1;
do_plt = 1;
w_scl = 1;
freq_units = 'rad/sec' ;

[Hcl,sort_all_gm,sort_all_pm,mu_results,all_lp] = ...

 do_RSanal(A_cl,B_cl,C_cl,D_cl,u_names,y_names,ww,do_prt,do_plt,w_scl,...
    freq_units);
set(findall(0,'type','axes'),'box','off');
```

```
================================
### Robust Stability Analysis:
================================

Points completed: 1000/1000
Points completed: 1000/1000
Points completed: 1000/1000
--------------------------------------------------------

max_over_w_lower_and_upper_bnds =
```

```
        1.3784        1.3784
        2.4586        2.4586
        3.0619        3.0619

--------------------------------------------------------
Loop        w      max(mu)
     u   1.8547     1.38
     y   1.8208     2.46
    uy   1.9602     3.06
--------------------------------------------------------

Loop       w         GM

--------------------------------------------------------
u           1.2978    12.12
theta       0.5401    14.58
thetadot    4.2870    15.36
u         145.5095   -21.17
thetadot  145.6210   -21.18
theta      15.5742   -23.01
--------------------------------------------------------

Loop       w         PM

--------------------------------------------------------
theta       2.2382    47.36
thetadot    1.8201   -57.01
u           9.1498    73.59
thetadot   11.4745    81.51
```



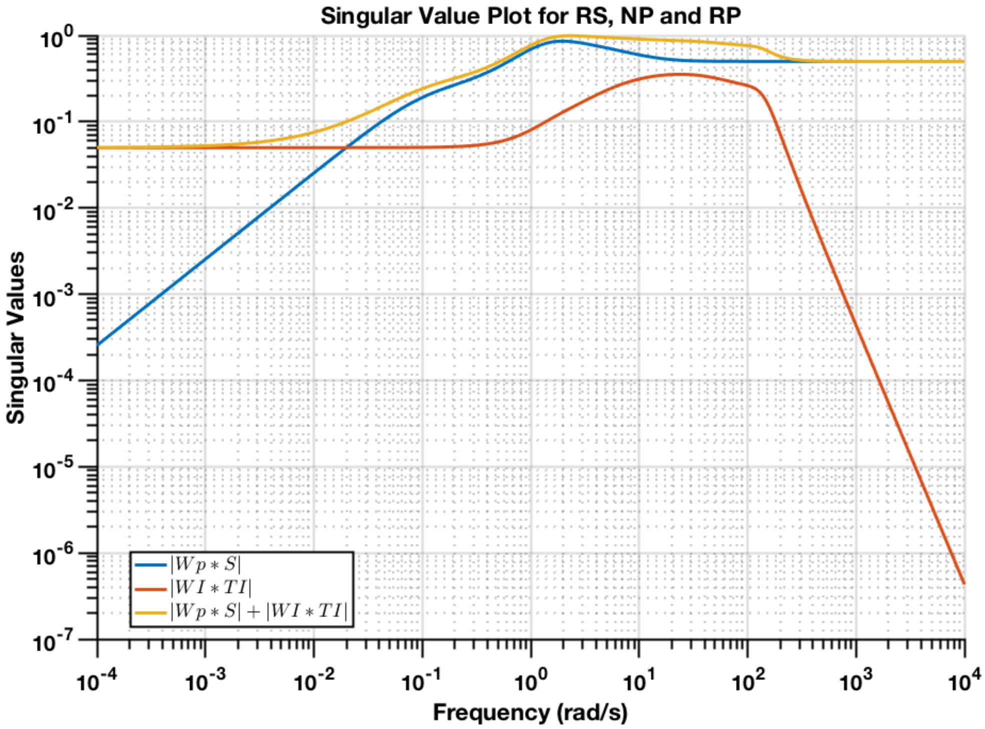




Summary

```
figure;
loglog(ww,sigmaWpS,ww,sigmaWIT,ww,total);
legend('$|Wp*S|$','$|WI*TI|$','$|Wp*S| + |WI*TI|$');
```

```
title('Singular Value Plot for RS, NP and RP')
xlabel('Frequency (rad/s)')
ylabel('Singular Values');
grid on;set(findall(0,'type','axes'),'box','off');
print(sprintf('Summary'),'-depsc');
```



Close Simulink Model Without Saving

```
close_system(model,0);
```

Published with MATLAB® R2018a