# Spacecraft Launch Vehicle Attitude Control System Design

Jyot Buch, Alex Hayes, Sepehr Seyedi

Group 14 : AEM 8421 : Robust Multivariable Control Systems

## Abstract

The launch vehicle attitude control system design is an extremely complex process requiring many years of research and development. Role of a attitude control design engineer is to design a control system that satisfies controllability, stability, transient response, GNC feedback loop schedule, etc. This project involves study of a spacecraft launch vehicle attitude control problem. Several control design objectives of a reduced order launch vehicle (LV) are explored by understanding and applying robust control concepts. Fundamental trade off between performance and robustness will be studied in this project. Most of the discussion in this paper is based on derivation of a model and explanation provided in reference [1]. MATLAB code for this project is attached in appendix [B] for reference.

# Contents

## Nomenclature

|  |  |
|---|---|
| $\alpha$ | Angle of Attack (rad) |
| $\delta$ | Engine Deflection Angle or Gimbal Angle (rad) |
| $\dot{\theta}$ | Attitude Angle Rate (rad/s) |
| $\dot{\theta}_{RG}$ | Derivative of Rate Gyro Output (rad/$s^2$) |
| $\theta$ | Attitude Angle (rad) |
| $\theta_{PG}$ | Position Gyro Output (rad) |
| $\theta_{RG}$ | Rate Gyro Output (rad/s) |
| $W_v$ | Lateral Velocity due to wind (ft/s) |

## 1 Dynamics

The nonlinear dynamics of the spacecraft launch vehicle were derived using Euler's equations of motion in [1] under the rigid body assumption. In this work, linearized time invariant

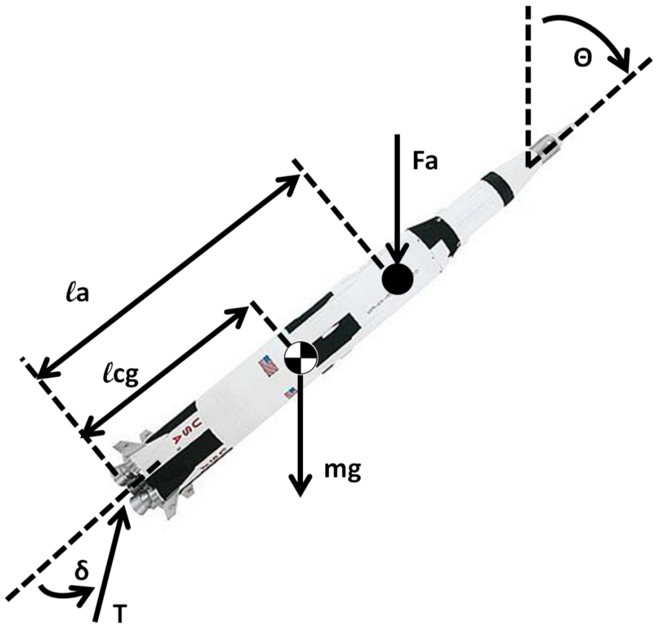


Figure 1: Rocket Attitude Free Body Diagram

control oriented model for attitude dynamics is used for analysis. Attitude of the launch vehicle is defined by its orientation in space. Here, $\theta$ is attitude angle, $\dot{\theta}$ is angular rate, and $\delta$ is the control input related to gimbal angle or an engine deflection angle.

## 1.1 LTI State Space Model

In this section, reduced order spacecraft launch vehicle attitude model is revisited [1]. This Multi-Input Multi-Output (MIMO) model has 3 inputs and 2 outputs. The general LTI state space can be written as,

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u} \\ \mathbf{y} &= C\mathbf{x} + D\mathbf{u} \end{aligned} \tag{1}$$

State vector including sensor dynamics $\theta_{PG}$ and $\theta_{RG}$ is defined as,

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}^T = \begin{bmatrix} \theta & \dot{\theta} & \theta_{PG} & \theta_{RG} & \dot{\theta}_{RG} \end{bmatrix}^T \tag{2}$$

The only exogenous control input is gimbal angle or attitude command $\delta$, including external disturbance inputs as angle of attack $\alpha$ and wind gust $W_v$, the overall input vector can be written as,

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \delta \\ \alpha \\ W_v \end{bmatrix} \tag{3}$$

Where, $A$, $B$, $C$ and $D$ matrices are given by,

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 3.9848 & -0.00794 & 0 & 0 & 0 \\ 25 & 0 & -25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 21199.36 & 0 & -21199.36 & -116.48 \end{bmatrix} \tag{4}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ -7.8235 & -9.413 & 0.00245 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5}$$

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{6}$$

$$D = 0_{2\times 3} \tag{7}$$

### 1.2 Transfer Function

The transfer function of the state space model presented by equation 1 is given by,

$$G(s) = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] = C(sI - A)^{-1}B + D \tag{8}$$

## 2 Nominal Plant Analysis

### 2.1 Open Loop Poles, Natural Frequency and Damping Ratio

Open loop eigenvalues, corresponding damping ratio and natural frequency were computed using MATLAB command `eig()` and `damp()` respectively, which are summarized in the following table. Results show that one of the eigenvalues has a positive real part i.e. $Re(\lambda_1) > 0$, resulting in unstable dynamics.

| Summary of Open Loop Poles | | | | |
|---|---|---|---|---|
| Poles ($\underline{\lambda}$) | Damping ($\zeta$) | Frequency (rad/s) | Time Constant (s) | Behavior |
| 1.99 | -1 | 1.99 | 0.502 | Unstable |
| -2.00 | 1 | 2.00 | 0.5 | Stable, Dominant |
| -25 | 1 | 25 | 0.04 | Stable, Non-Dominant |
| -58.24 + 133.4$j$ | 0.4 | 146 | 0.0172 | Fast-Oscillatory |
| -58.24 - 133.4$j$ | 0.4 | 146 | 0.0172 | Fast-Oscillatory |

It should be noted that, sensor dynamics (last 3 states) are relatively high frequency dynamics i.e. 25 rad/s and 146 rad/s respectively. Since these dynamics are having faster time constant in order of $10^{-2}$ s, we can ignore them for control analysis and derive the simplified $2^{nd}$ order model, which is done in section 3.4. Unstable pole at s = 1.99 has a negative damping, which justifies its unstable behavior with positive real part.

### 2.2 Controllability

Using `ctrb(A,B)` command in MATLAB, controllability matrix $\mathcal{C}$ was computed and found to be full rank of $n = 5$. Therefore, the system is full state controllable. Note that for controllability matrix, columns of B corresponding to disturbance were ignored.

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & A^3B & A^4B \end{bmatrix} \tag{9}$$

## 2.3 Observability

Using `obsv(A,B)` command in MATLAB, observability matrix $\mathcal{O}$ was computed and found to be full rank. Therefore, the system is full state observable.

$$\mathcal{O} = \begin{bmatrix} C & CA & CA^2 & CA^3 & CA^4 \end{bmatrix}^T \tag{10}$$

Since, open loop system is both controllable and observable, it is the minimal realization of the system. It is worth remembering that above definition of controllability and observability provides binary answer such as whether the system is controllable/observable or not.

## 2.4 Open Loop Frequency Response

Open Loop Frequency response of output 1 i.e. $\theta_{PG}$ to all the 3 input (1 control and 2 disturbance inputs) is shown in the Figure 2. Similarly, output 2 i.e. $\theta_{RG}$ to all the input frequency response is shown in 3.

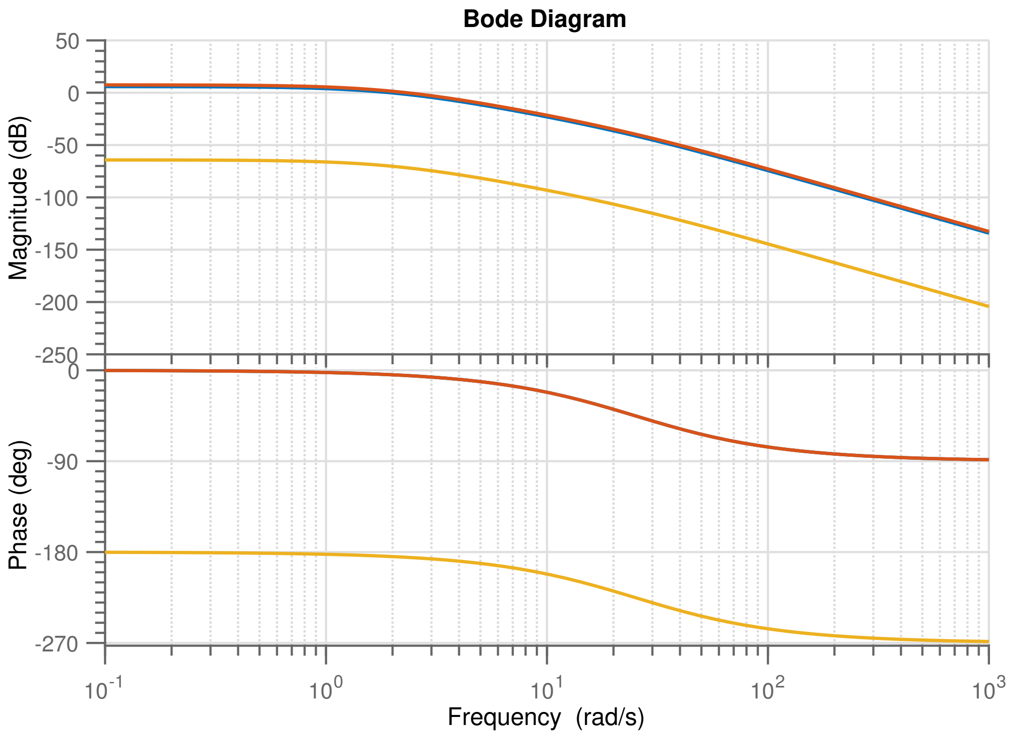


Figure 2: $y_1$ to $u_1$(Blue), $u_2$(Red), $u_3$(Green) Open Loop Frequency Response

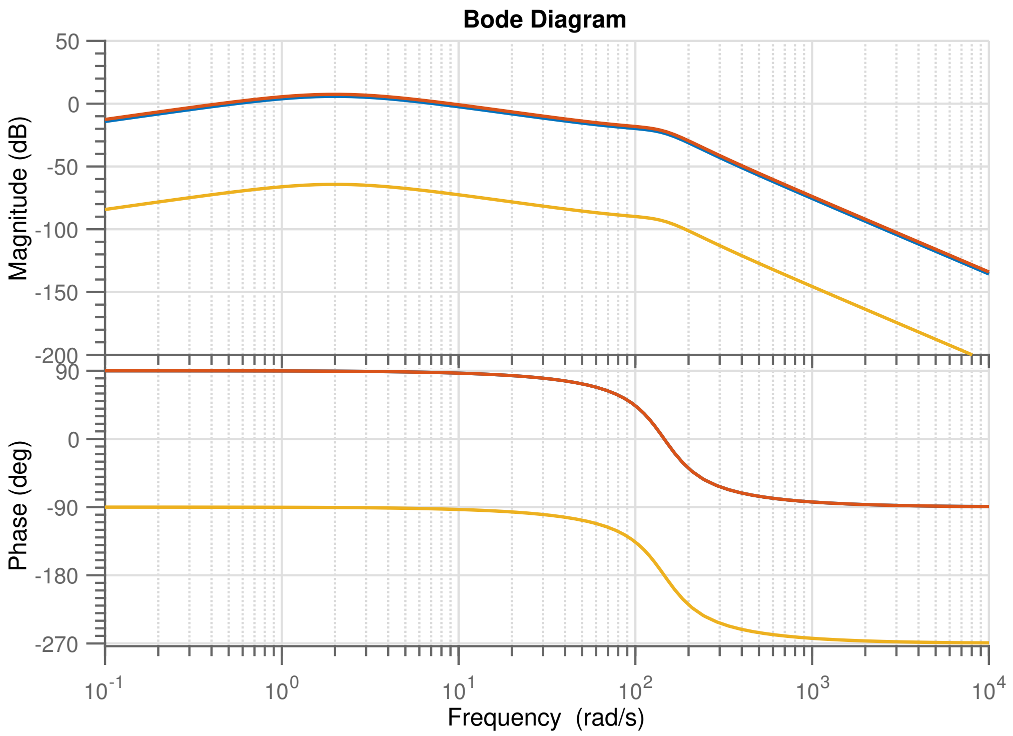


Figure 3: $y_2$ to $u_1$(Blue), $u_2$(Red), $u_3$(Green) Open Loop Frequency Response

### 2.5 Singular Values vs Frequency

Open loop singular values were computed using `svd()` function in MATLAB. It can be observed from the following singular values that largest open loop singular value $\bar{\sigma} = 29981$ and lowest singular value $\underline{\sigma} = 0.7071$. Further analysis is to be done in robust stability and performance section, since we need to bound the largest singular value of the overall system to ensure the robust stability. To reproduce the result shown in [1], the plot of singular values of the frequency response is shown in Figure 4 using MATLAB function `sigma()`. It is observed that the peak singular value happens at approximately 1.5 rad/s with magnitude of 3.54. It is worth noticing that disturbance was also included in this analysis.

## 3 Control Design

We now begin with the general control architecture and deign objectives followed by actual control design in MATLAB/Simulink.

### 3.1 General Control Architecture

The generic control system is shown in Figure 5. Actuator, Plant and Measurement Sensors are not perfect as modeled by the mathematical equations, they exhibit uncertain behavior. (Right side of the red dotted line). Lets consider the special case of the following control architecture which has a unity feedback.

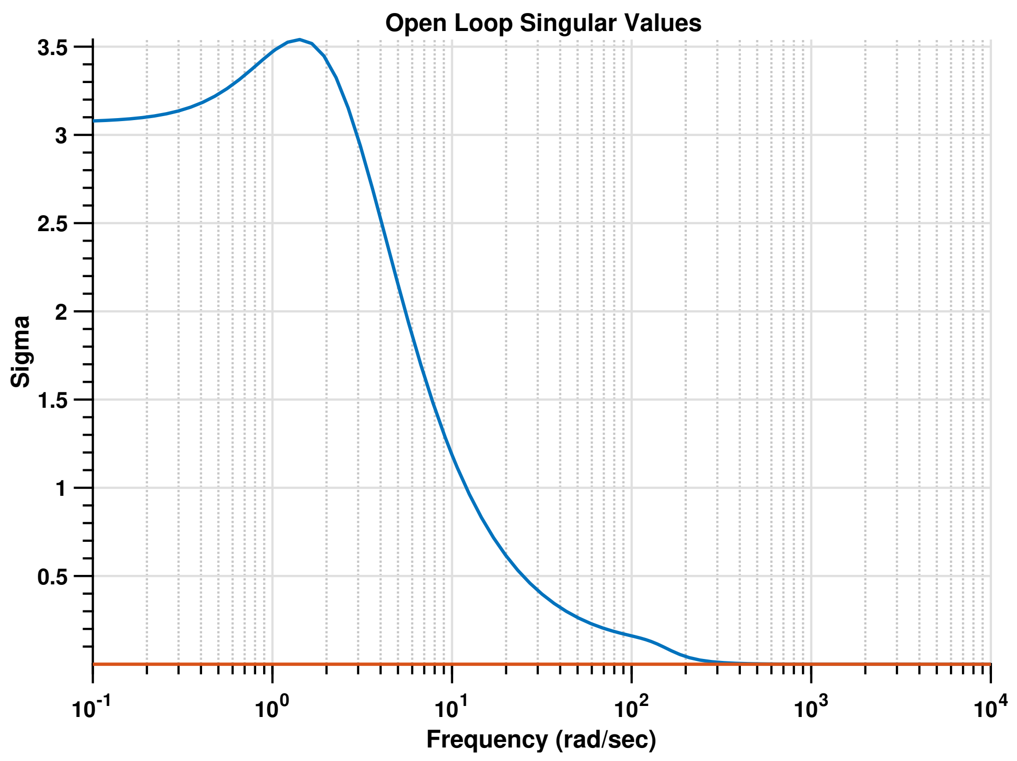


Figure 4: Singular Values of the Frequency Response

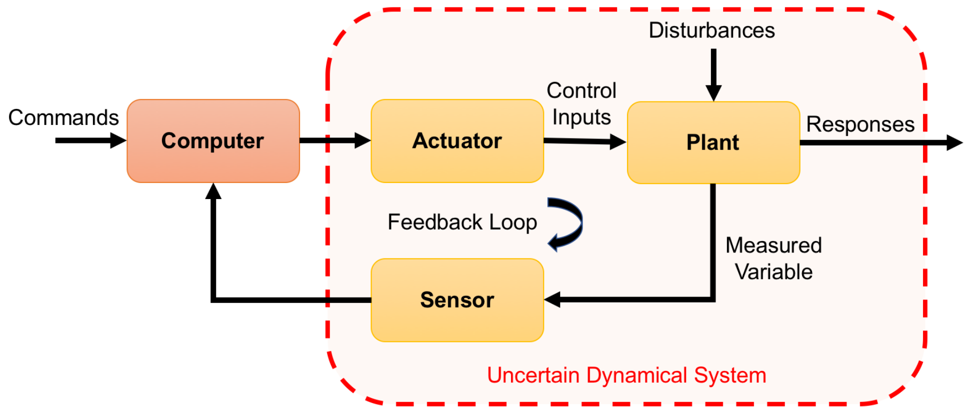


Figure 5: Generic Control System Layout

Here,
G := MIMO Plant Transfer Function
$K$ := Controller Transfer Function
$r$ := Reference Signal Input
$d$ := Disturbance Input
$n$ := Measurement Noise

$$y = \underbrace{(I+GK)^{-1}GK}_{T}\, r + \underbrace{(I+GK)^{-1}}_{S}\, G_d d - \underbrace{(I+GK)^{-1}GK}_{T}\, n \tag{11}$$

Lets define the following,

$$\begin{aligned}
G &= \text{Plant Transfer Function Matrix} \\
K &= \text{Feedback Compensator Transfer Function} \\
L = GK &= \text{Output Loop Transfer Function} \\
S = (I+GK)^{-1} = (1+L)^{-1} &= \text{Output Sensitivity Transfer Function} \\
T = (I+GK)^{-1}GK = (1+L)^{-1}L &= \text{Output Complementary Sensitivity function} \\
L_I = KG &= \text{Input Loop Transfer Function} \\
S_I = (I+KG)^{-1} = (1+L_I)^{-1} &= \text{Input Sensitivity Transfer Function} \\
T_I = (I+KG)^{-1}KG = (1+L_I)^{-1}L_I &= \text{Input Complementary Sensitivity function} \\
\text{Note that, } S+T = I &\text{ or } S_I + T_I = I
\end{aligned}$$

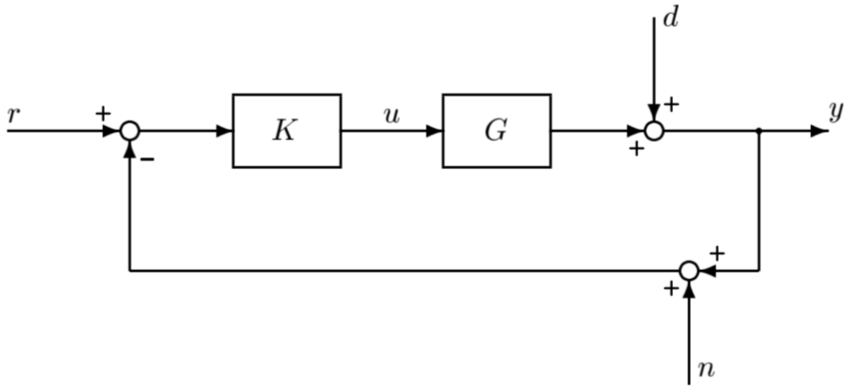


Figure 6: Unity Feedback System

We have following important relationships [3] in Laplace domain,

$$\begin{aligned}
y(s) &= T(s)r(s) + S(s)d(s) - T(s)n(s) \\
u(s) &= K(s)S(s)[r(s) - n(s) - d(s)]
\end{aligned} \tag{12}$$

These relationships determine several closed-loop objectives as presented in [3], in addition to the requirement that K stabilizes G. In what follows, we discuss some of the approximation of these in to open-loop objectives over specified frequency range.

## 3.2 Design Objectives

It is desired to have following control objectives for design considerations.

| **Control Objectives** | |
|---|---|
| **Qualitative** | **Quantitative** |
| Stabilize the unstable launch vehicle dynamics | Place eigenvalues to the left half of the s-plane |
| Reduce sensitivity to disturbances | Make $\underline{\sigma}(L)$ large; for $\omega$ at which $\underline{\sigma}(L) \gg 1$ |
| Robustness to model uncertainties | Make sure the signal norms are bounded |
| Noise Attenuation | Make $\bar{\sigma}(L)$ small; for $\omega$ at which $\bar{\sigma}(L) \ll 1$ |
| Track attitude angle reference $\theta_{ref}$ | Make $\underline{\sigma}(L)$ large; for $\omega$ at which $\underline{\sigma}(L) \gg 1$ |
| Robust stability to an additive perturbation | Make $\bar{\sigma}(K)$ small; for $\omega$ at which $\bar{\sigma}(L) \ll 1$ |
| Minimize control effort | Make $\bar{\sigma}(K)$ small; for $\omega$ at which $\bar{\sigma}(L) \ll 1$ |

Where, $\underline{\sigma}(L)$ and $\bar{\sigma}(L)$ are the smallest singular value and largest singular value of loop transfer function $L$ respectively.

Classical notion of the robustness such as gain margin $|G(jw_g)| \geq 6$ dB, and phase margin of $\angle G(jw_c) \geq 45^o$ is desired. It is desired to have the system behavior such as low pass filter.

In terms of time domain performance, it is desired to have, ...

- Faster rise time, in the order of 2 s or less
- Settling time in the order of less than 5 seconds
- None or minimum overshoot/undershoot
- With respect to maximum disturbance (i.e. High Altitude Wind $\approx 100$ m/s, Angle of Attack $\alpha \approx 5 - 15^o$) no more than $5^o$ of overshoot in attitude angle $\theta$
- Adequate damping to mitigate any oscillations due to disturbance
- Less than 10% steady state error in reference tracking

## 3.3 Trade-offs in MIMO feedback design

The shaping of multi-variable transfer functions is based on the idea that a satisfactory definition of gain (range of gain) for a matrix transfer function is given by the singular values of the transfer function. By multivariable transfer function shaping, therefore, we mean the shaping of singular values of appropriately specified transfer functions such as the loop transfer function or possibly one or more closed-loop transfer functions. The open-loop requirements specified in above table cannot all be satisfied simultaneously. Feedback design is therefore a trade-off over frequency of conflicting objectives. This is not always as difficult as it sounds because the frequency ranges over which the objectives are important can be quite different [3]. For example, disturbance rejection is typically a low frequency requirement, while noise mitigation is often only relevant at higher frequencies.

## 3.4 Reduced Order Model and Control Design

For the purpose of control design, first a single control input system with two state variables and two outputs, both being $\theta$ and $\dot{\theta}$ was considered for the system shown in section 1.1 (See equation 13, which denotes $A_2$,$B_2$,$C_2$,$D_2$). As stated before, it is a valid argument to ignore the high frequency sensor dynamics for primary control design. The results will be verified in what follows with the higher order model.

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 3.9848 & -0.00794 \end{bmatrix}\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} + \begin{bmatrix} 0 \\ -7.8235 \end{bmatrix}\delta \\ y &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} \end{aligned} \tag{13}$$

### 3.4.1 Proportional Control Design

Initial approach to the control design was attempted by proportional feedback controller obtained via LQR. MATLAB command `lqr()` was used for the design. The control gains were computed by minimizing the following quadratic cost function.

$$J(u) = \int_0^\infty (x^TQx + u^TRu)dt \tag{14}$$

Where, tuning parameters such as input cost R was chosen to be 1 and state cost Q as identity matrix `eye(2)` to begin with the control design. Later, using trial and error controller gain K = [-3 -1.3] was selected to place the closed loop poles and get the desired control behavior. These gains were then plugged in to full 5 state controller and closed loop response was simulated. Thus, if the designed controller for 2 state is $K = [k_1, k_2]$ then following controller matrix is used for full 5 state simulation in Simulink.

$$K_5 = \begin{bmatrix} k_1 & k_2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} -3 & -1.3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{15}$$

Thus in summary, we first begin the design of a controller with reduced order model using 2 states i.e. $\theta$, $\dot{\theta}$ and then use the resulting control gains for the full state model by plugging them in to sensor states i.e. $\theta_{PG}$ and $\theta_{RG}$ instead of the true one. It can be seen from the Figure 7 that proportional feedback does well in terms of tracking 0 attitude angle (solid line) but, it does not reduce the steady state error in $\theta$ due to wind gust or angle of attack step (dashed line) disturbances, resulting in steady state offset in attitude angle. Figure 7 demonstrates the case when angle of attack step disturbance was applied at t = 8 s with step magnitude of $-5^o$. Here, negative angle of attack disturbance results in positive offset for attitude angle $\theta_{RG}$. Initial conditions of [$\theta$,$\dot{\theta}$,$\theta_{PG}$,$\theta_{RG}$,$\dot{\theta}_{RG}$] = [0.087, 0.087, 0.087, 0.087, 0] were provided to the full 5 state model to simulate the closed loop response (0.087 rad $\approx 5^o$). Figure 7 shows that proportional controller fails to track $\theta_{ref} = 0$ angle in the presence of disturbance.

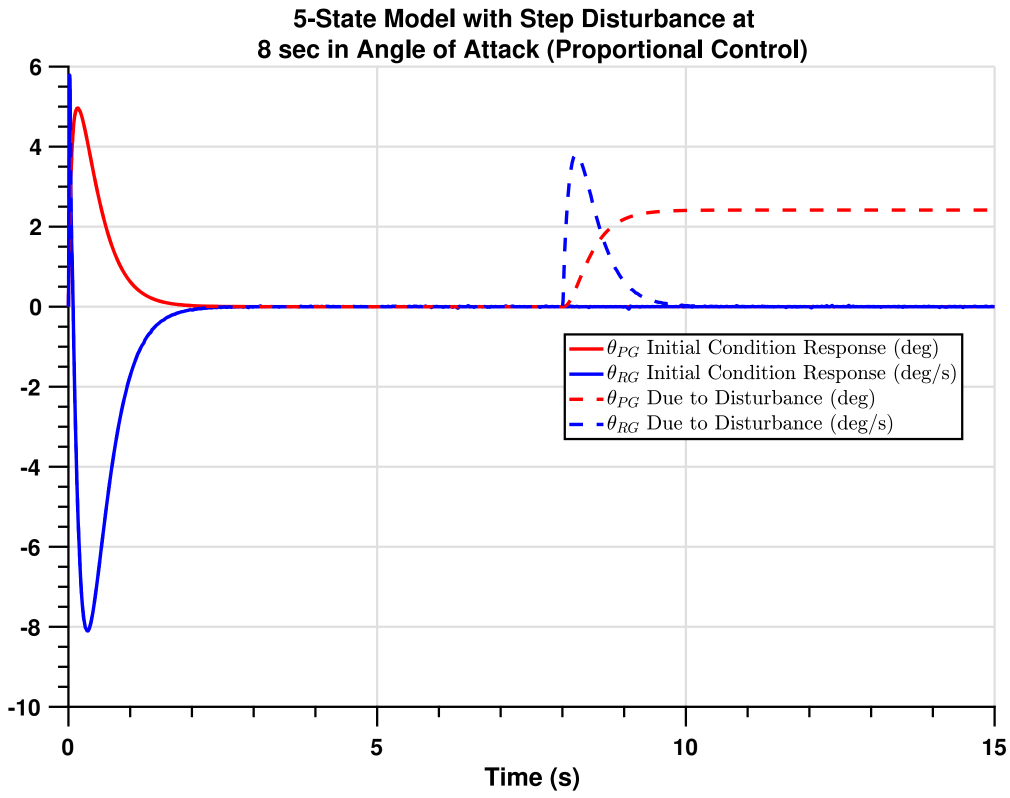


Figure 7: Proportional Control and Steady State Error Due to Step Disturbance

### 3.4.2 Proportional and Integral Feedback Control Design

In order to reduce the steady state error, PI control strategy was considered next. For the two state case, this controller is translated into the following transfer function K(s) matrix. It is worth noticing that $K_P = [k_{p_1}, k_{p_2}]$ and $K_I = [k_{i_1}, 0]$ are both $1 \times 2$ vectors.

$$\begin{aligned} K(s) &= K_I \frac{1}{s} + K_P = \begin{bmatrix} k_{p_1} + \frac{k_{i_1}}{s}, & k_{p_2} \end{bmatrix} \\ &= \left[\begin{array}{c|c} 0_{2\times 2} & I_{2\times 2} \\ \hline K_I & K_P \end{array}\right] \end{aligned} \tag{16}$$

Similar to section 3.4.1, we started with the control design using MATLAB function `lqi()` to compute the gains $K_I$ and $K_P$. Later using trial and error following gains were computed for PI control.

$$\begin{aligned} K_P &= \begin{bmatrix} -3 & -1.3 \end{bmatrix} \\ K_I &= \begin{bmatrix} -2 & 0 \end{bmatrix} \end{aligned} \tag{17}$$

Thus, for the full 5 state model, following gain matrices can be used.

$$\begin{aligned} K_{P_5} &= \begin{bmatrix} k_{p_1} & k_{p_2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} -3 & -1.3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \\ K_{I_5} &= \begin{bmatrix} k_{i_1} & k_{i_2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{18}$$

Full 5 state model was implemented in Simulink to simulate the closed loop response. Figure 8 shows the initial condition response for PI Controller similar to first 2 states of Figure 7. The limitation of Proportional feedback was steady state error to disturbance, which was evaluated in Figure 9 by applying angle of attack disturbance $\alpha = -5^o$ at 1 s. PI Controller result shows that steady state error is now maintained at zero after some transient controller effort. Similar exercise was done with the wind gust disturbance of 100 m/s (maximum possible) as shown in Figure 10 at 1 s. Finally, both the disturbances were applied together at different time instance to see if controller is able to recover from the applied disturbance. Overall, the designed PI controller does well in terms of performance and brings the steady state errors to 0. The primary goal of zero steady state error and stabilization of the plant is fulfilled.

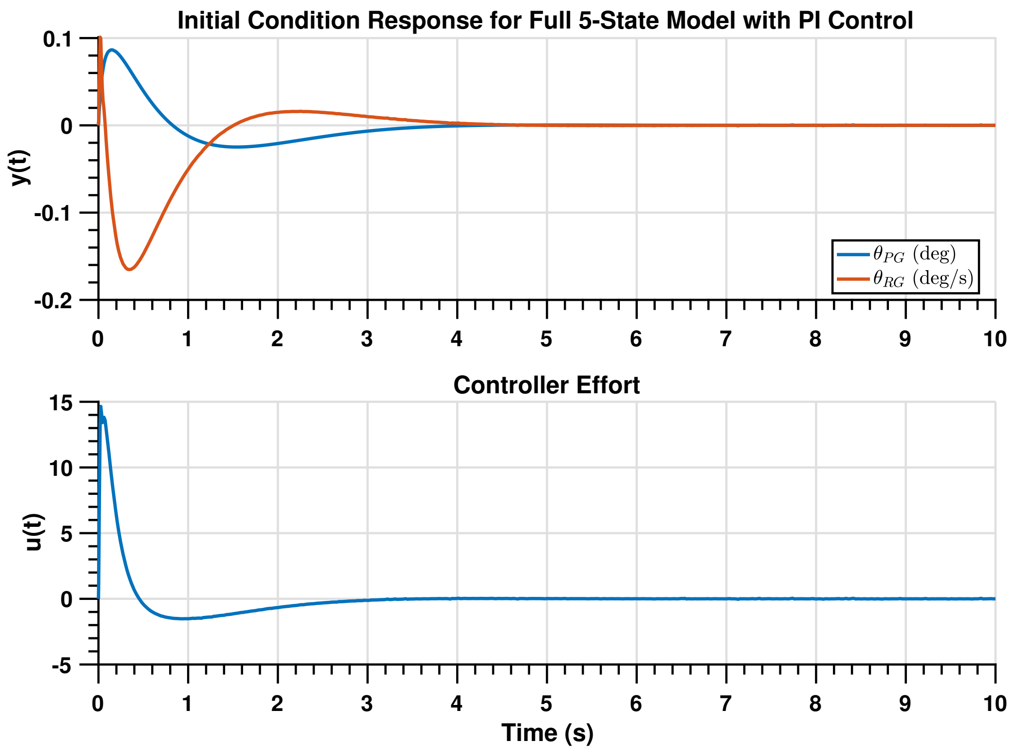


Figure 8: Initial Condition Response with Full State PI Control

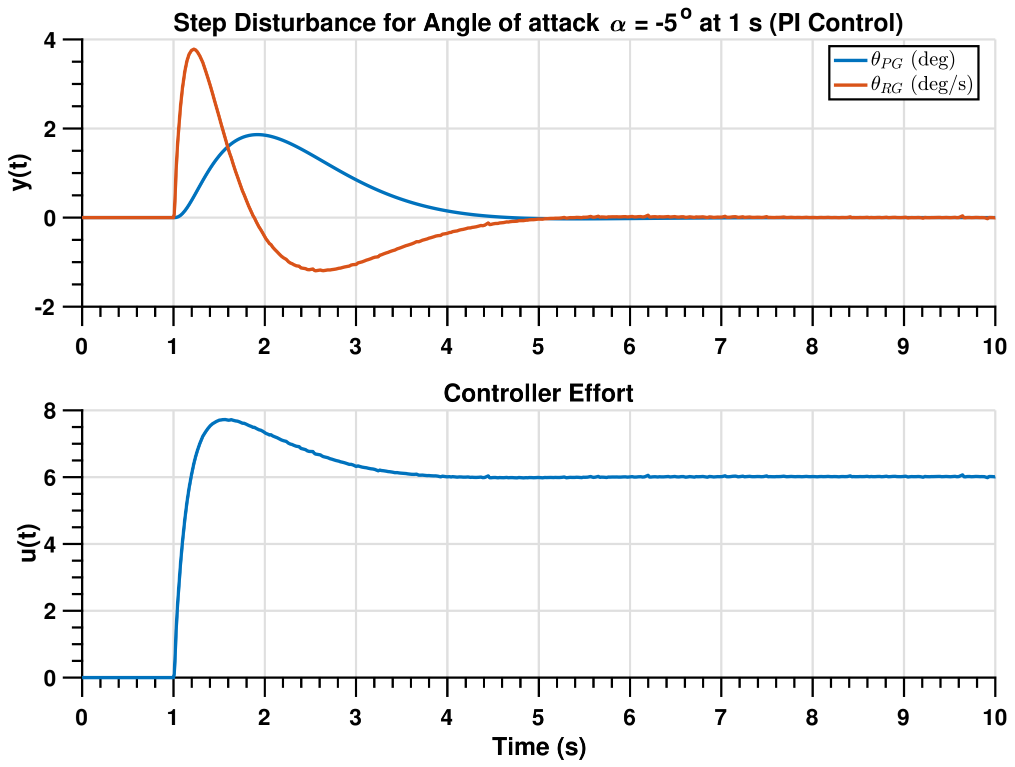


Figure 9: Step Disturbance for $\alpha = -5^o$ at 1 s

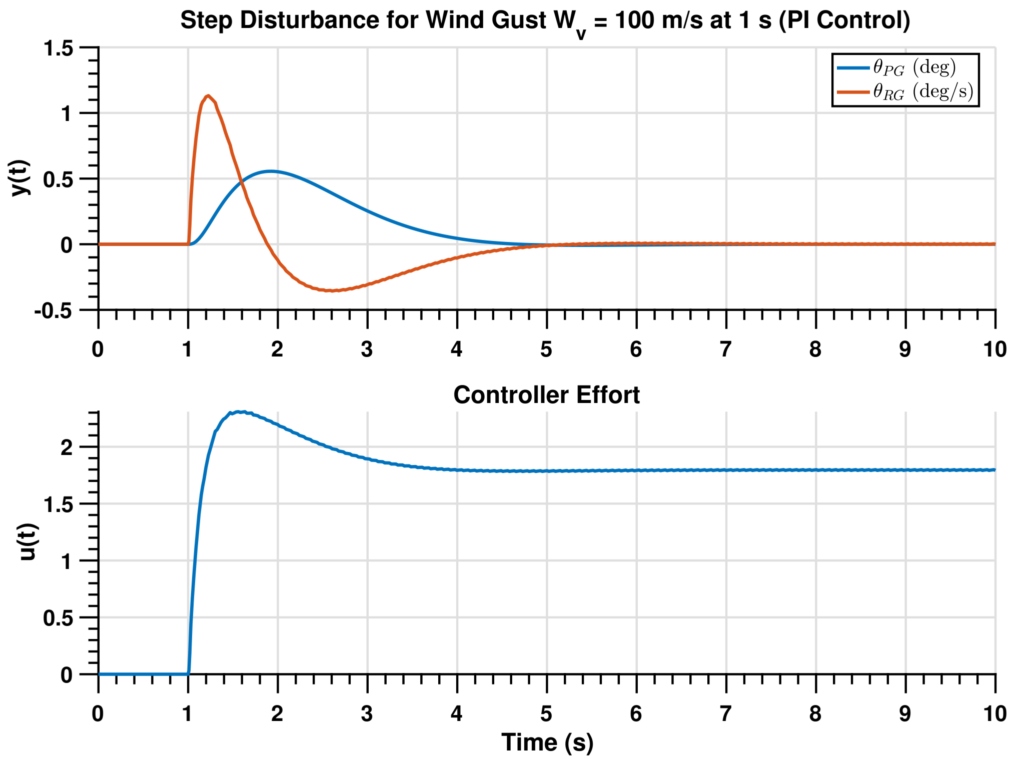


Figure 10: Step Disturbance as Wind Gust $W_v = 100$ m/s at 1 s

## 3.5 Nominal Closed Loop Analysis

### 3.5.1 Poles of Controller K(s)

For Proportional and Integral controller, the poles are located at the origin as can be seen from equation 16.

### 3.5.2 Closed Loop Poles, Damping and Natural Frequency

Compared to open loop poles, it is verified that feedback stabilized the system since all the poles are having negative real part. Following table shows individual pole and damping characteristics. It can be observed that the unstable pole is now transformed in to stable complex conjugate pair of poles with damping ratio of 0.8, which is well damped. `linmod()` function was used to linearize the Simulink model and get state space matrices for closed loop. Further the closed loop poles or eigenvalues were verified from previous calculation.

Summary of Closed Loop Poles

| Poles ($\lambda$) | Damping ($\zeta$) | Frequency (rad/s) | Time Constant (s) | Behavior |
|---|---|---|---|---|
| -52.9 + 131$j$ | 0.373 | 142 | 0.0189 | Fast |
| -52.9 - 131$j$ | 0.373 | 142 | 0.0189 | Fast |
| -1.21 + 0.93$j$ | 0.8 | 1.51 | 0.829 | Well-Damped |
| -1.21 - 0.93$j$ | 0.8 | 1.51 | 0.829 | Well-Damped |
| -6.88 | 1 | 6.88 | 0.145 | Stable |
| -26.5 | 1 | 26.5 | 0.0378 | Non-Dominant |

### 3.5.3 Frequency Response

For 2 states model, the Input loop transfer function is given by,

$$L_{I_2} = KG = \left[\begin{array}{c|c} A_{Li_2} & B_{Li_2} \\ \hline C_{Li_2} & D_{Li_2} \end{array}\right] \tag{19}$$

Where, using cascade state-space manipulations,

$$A_{Li_2} = \begin{bmatrix} 0_{2\times2} & I_2 \\ 0_{2\times2} & A_2 \end{bmatrix}, B_{Li_2} = \begin{bmatrix} 0_{2\times1} & B_2 \end{bmatrix}, C_{Li_2} = \begin{bmatrix} K_I & K_P \end{bmatrix}, D_{Li_2} = 0 \tag{20}$$

Plotting above input loop transfer function, gain cross over frequency was found to be 9.88 rad/sec. It is desired for robust stability for unstable low pass type system that gain crossover frequency $W_c > 2*a$, where $a$ is unstable open loop pole. In this case, it is $1.99 \implies W_c > 4$ criteria is satisfied as shown in the Figure 11.

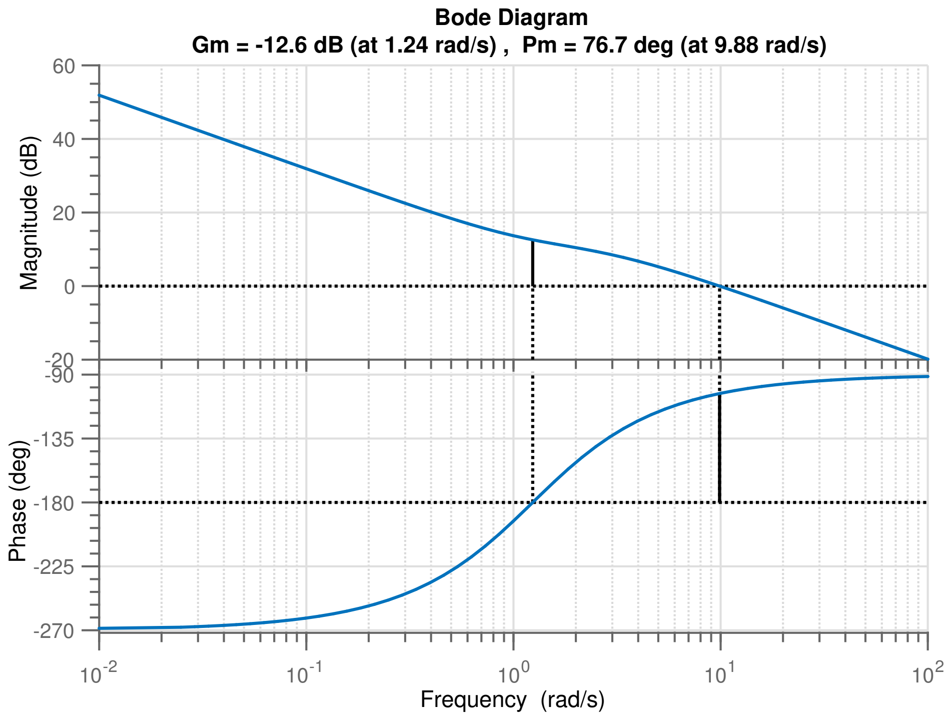


Figure 11: Bode Plot of Input Loop Transfer Function (2-State Model)

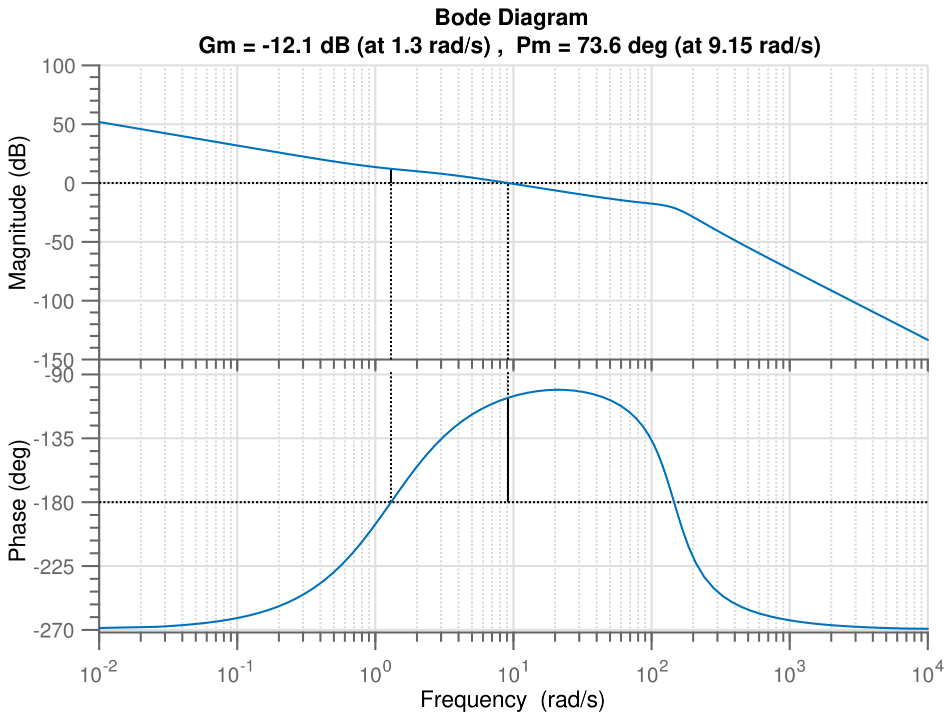


Figure 12: Bode Plot of Input Loop Transfer Function (5-State Model)

### 3.5.4 Time Domain Performance

Step Response was plotted in Simulink for $5^o$ of reference tracking as shown in the Figure 13. It can be seen that steady state error in tracking performance is 0. The response characteristics were computed using MATLAB command `stepinfo(y,t)` and are summarized in the following table.

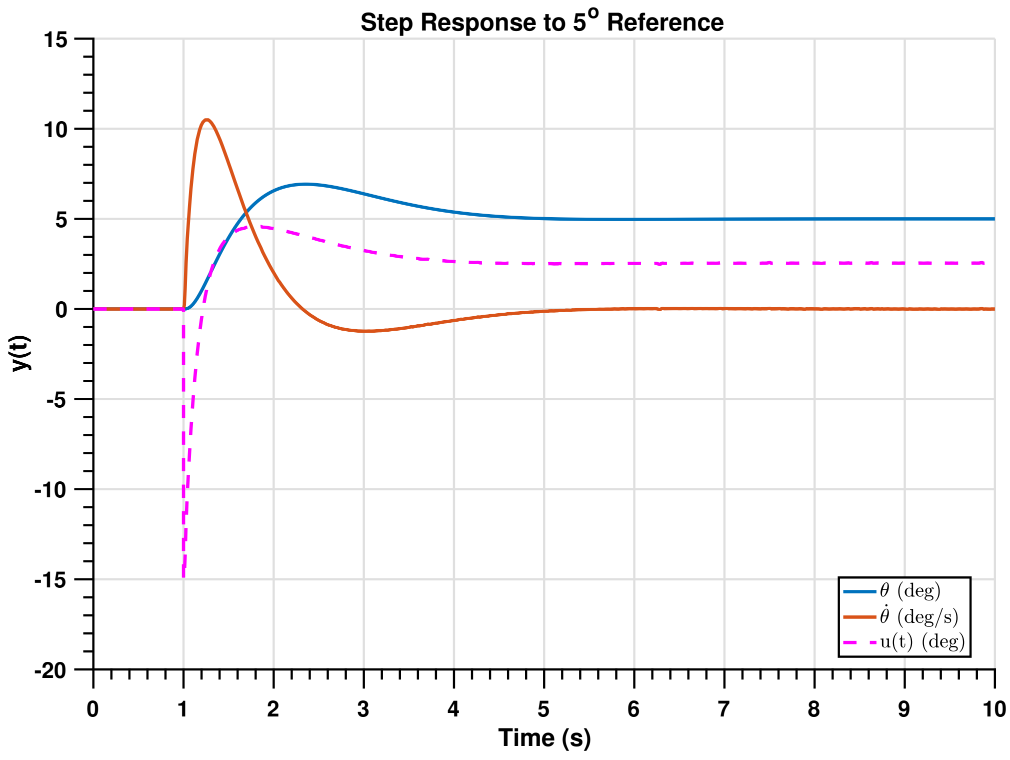


Figure 13: Step Response of Closed Loop System (5-State Model)

| Step Response Statistics | |
|---|---|
| Rise time $(s)$ | 0.42797 |
| Settling Time $(s)$ | 4.5917 |
| Settling Min $(s)$ | 4.5134 |
| Settling Max $(s)$ | 6.9247 |
| Overshoot (%) | 38.48 |
| Undershoot (%) | 0 |
| Peak | 6.9247 |
| Peak Time $(s)$ | 2.3465 |

It is evident that the chosen control strategy provided desired design specifications, such as settling time less than 5 s, rise time less then 2 s. A compromise was made to accept more overshoot ($\approx 38\%$) in return for faster rise time for step reference tracking. Actuator effort was observed to be reasonable which was maximum of $15^o$.

### 3.5.5 Nominal Performance

For Nominal Performance, sensitivity $S(j\omega)$ plot is shown in the Figure 14. Designed bandwidth frequency was chosen to be $\omega_b^* = 1.05$ rad/s. Based on the steady state error requirement $A$ was selected to be 0.1 and to suppress or account for the high frequency neglected dynamics $M$ was chosen to be 2. Following guidelines from page 62 of [3], weights for the sensitivity transfer function then becomes,

$$W_p = \frac{s/M + \omega_b^*}{s + \omega_b^* A} = \frac{0.5s + 1.05}{s + 0.105} \tag{21}$$

Nominal performance test is given by,

$$\|W_P S\|_\infty < 1 \tag{22}$$

Sensitivity and weighted sensitivity transfer functions are plotted in the Figure 14. It is evident that singular values (i.e. magnitude for SISO transfer function) of the weighted sensitivity stays below 1 or 0 dB. That satisfies nominal performance criteria as discussed in [3].

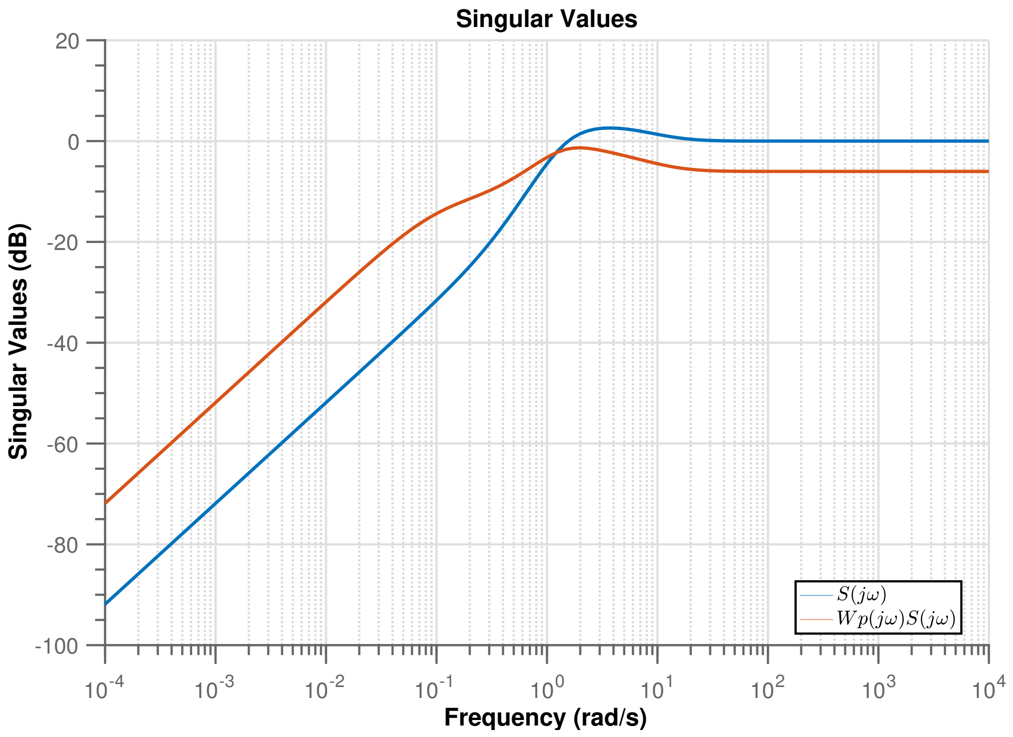


Figure 14: Singular Values as function of frequency

### 3.5.6 Robust Stability

Robust Stability describes a system being stable in the presence of uncertainties. Based on the [3] it can be verified by checking the following test.

$$\|W_I T_I\|_\infty < 1 \tag{23}$$

Using `linmod` command in MATLAB, Simulink model was linearized to find the complementary sensitivity transfer function $T_I$. $W_I$ was chosen as following,

$$W_I = \frac{\tau s + r_0}{(\tau / r_\infty)s + 1} \tag{24}$$

Where, $r_0$ is relative uncertainty at steady-state, $1/\tau$ is the approximate frequency at which the relative uncertainty reaches 100%, $r_\infty$ is the magnitude of the weight at high frequency. Based on the provided guidelines in [3], we choose, $\tau = 1/25$, $r_0 = 0.05$, $r_\infty = 2$. Thus the weighting transfer function becomes,

$$W_I = \frac{(1/25)s + 0.05}{(1/24)s + 1} \tag{25}$$

Launch vehicle elastic mode gets excited at higher frequency than 25 rad/sec, thus it can be assumed to be 100% uncertainty above that frequency. Singular value plot for $W_I T_I$ is shown in the Figure below. The relative uncertainty at steady state is approximately 5% (-26 dB) and high frequency magnitude is acceptable with the factor of 2 (compromise at higher frequency).

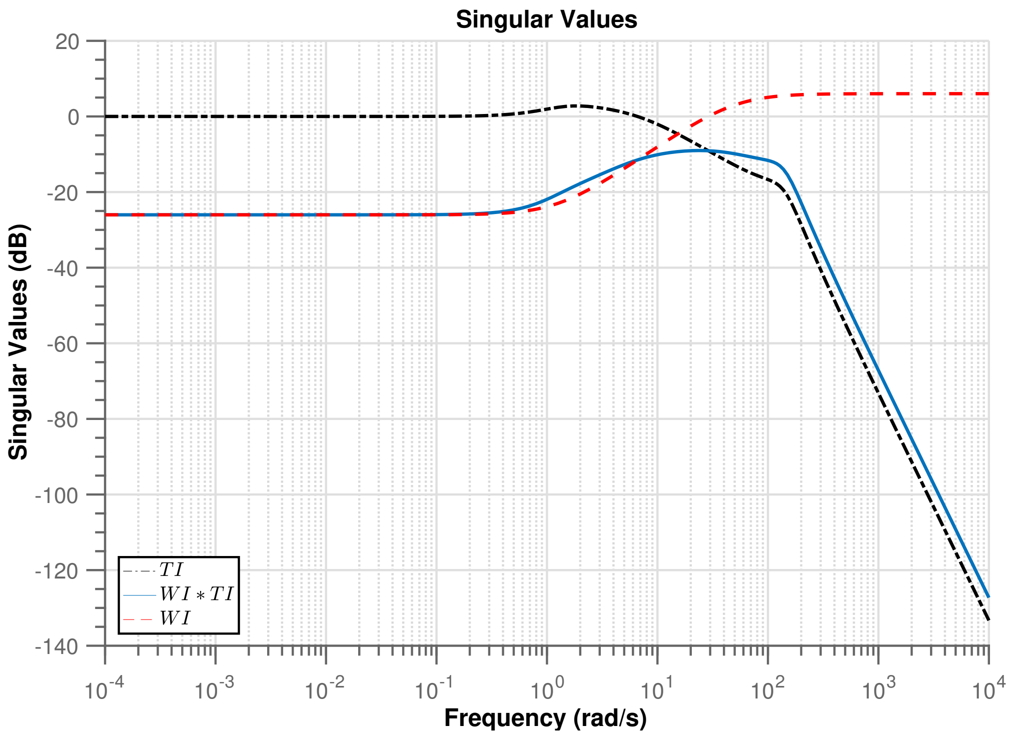


Figure 15: Singular Value Plot for WI*TI

It can be seen from Figure 15 that the $H_\infty$ norm of $W_I T_I$ is less than 1, thus satisfying the Robust Stability test. The peak value for $W_I$ is 6.02 dB. Using the provided RS analysis script, structured singular value upper bounds for 2 Sensors and 1 Actuator were computed for multiplicative perturbation which are shown in the Figure 16. Typically, the picking behavior as shown in this plot is expected. Following section describes the block diagonal uncertainty that was considered in this analysis.

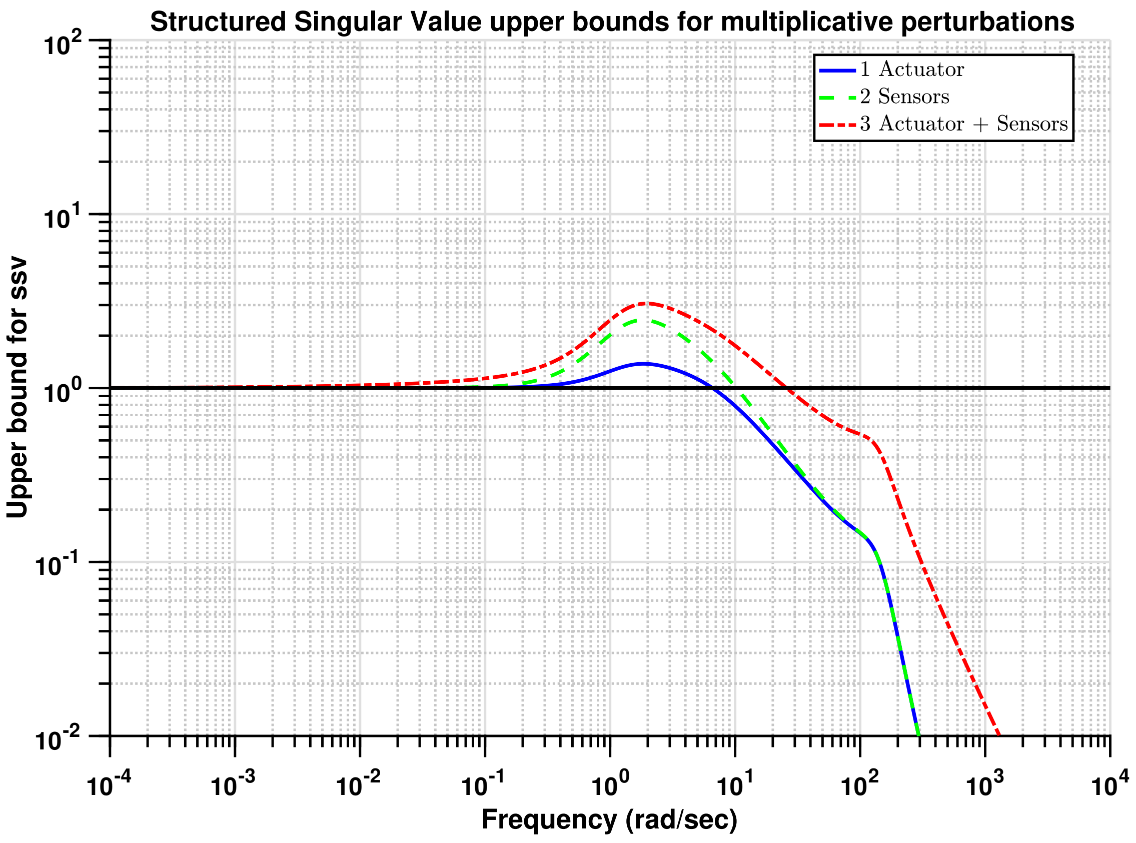


Figure 16: Structured Singular Value Upper Bounds for Sensors and Actuators

### 3.5.7 Uncertainty Blocks

For Single-Loop-At-A-Time analysis, the uncertainty configuration considered here is a structured block diagonal consisting of two scalar complex (Transfer Functions) $1 \times 1$ blocks.

$$\Delta = \begin{bmatrix} \delta_1 & 0 \\ 0 & \delta_2 \end{bmatrix} \tag{26}$$

### 3.5.8 Single-Loop-At-A-Time Analysis

Loops were broken at one actuator and two sensors ($\theta_{PG}$ and $\theta_{RG}$) to find the single-loop-at-a-time (SLAT) loop transfer functions.

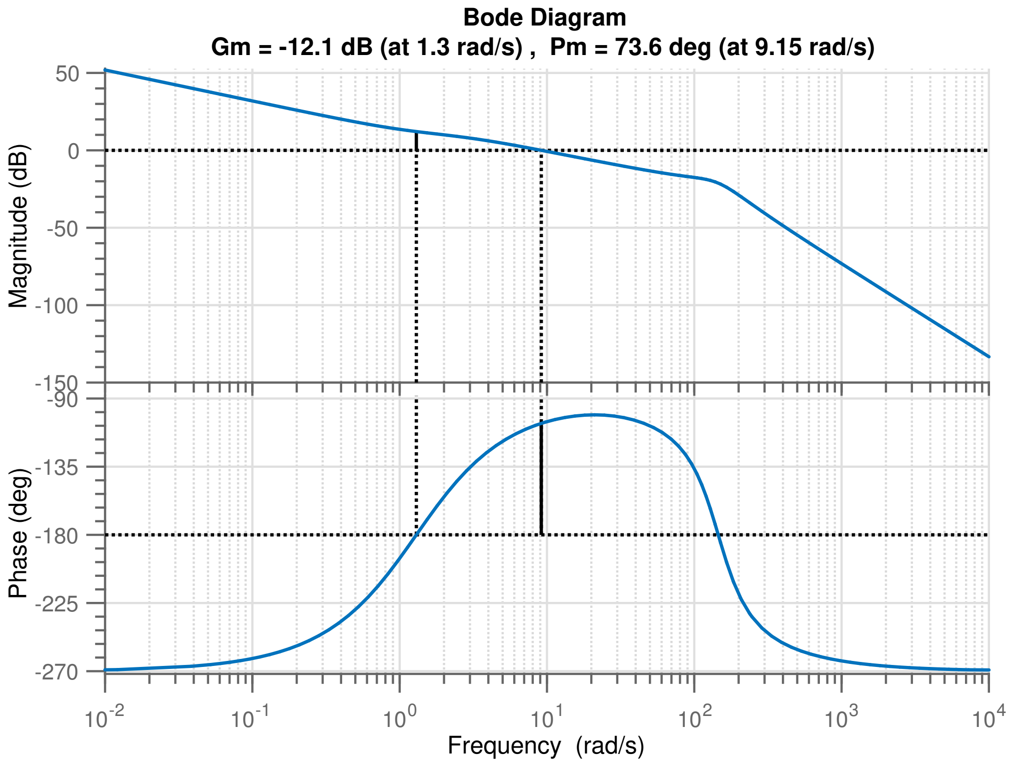


Figure 17: SLAT Analysis at u

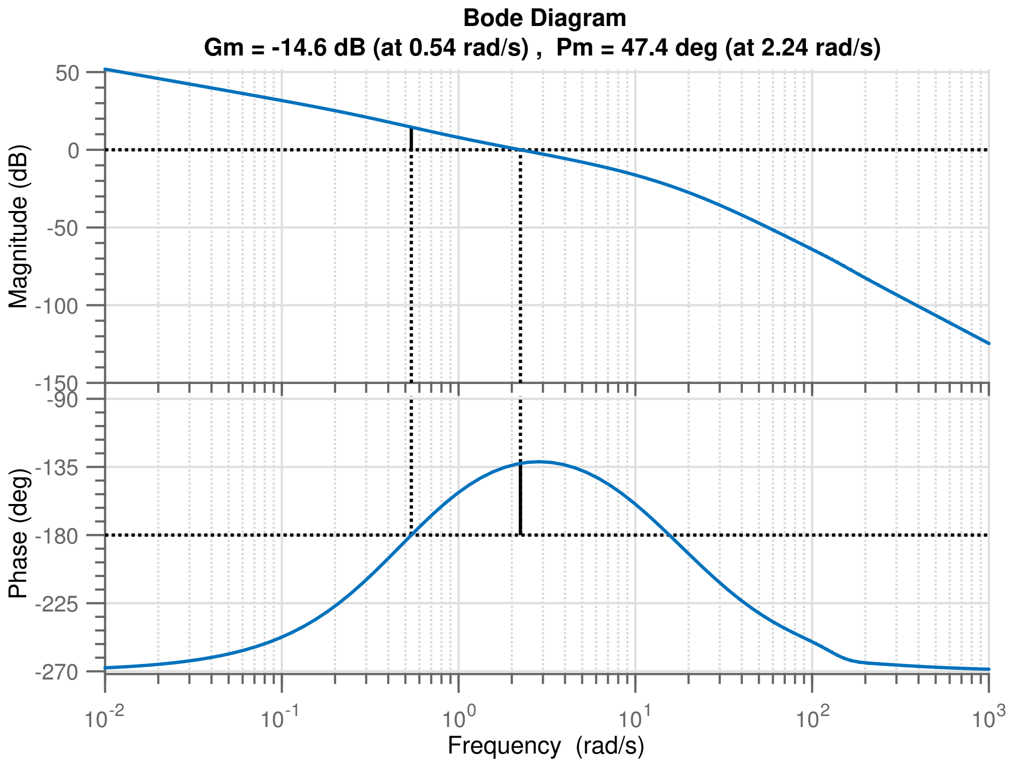


Figure 18: SLAT Analysis at $\theta_{PG}$ measurement

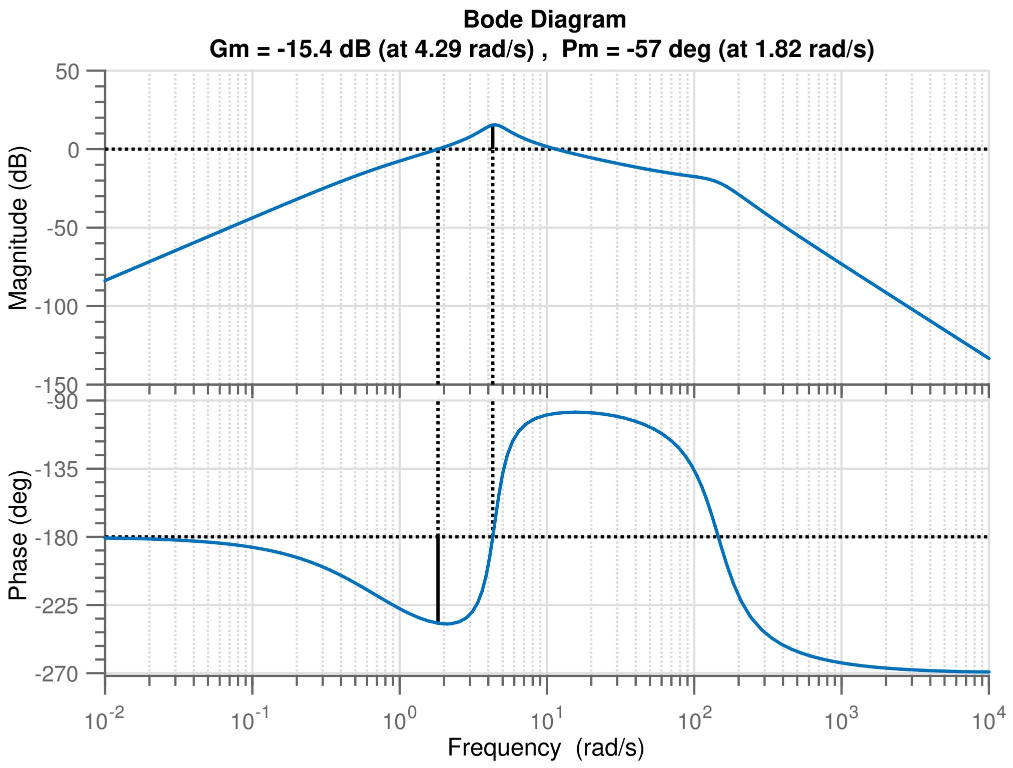


Figure 19: SLAT Analysis at $\theta_{RG}$ measurement

| Summary of Single-Loop-At-A-Time Analysis | | | | |
|---|---|---|---|---|
| Loop Broken At | $\omega_{180}$ (rad/s) | GM (dB) | $\omega_c$ (rad/s) | PM (deg) |
| Input ($u$) | 1.3<br>145 | -12.1<br>21.2 | 9.15 | 73.6 |
| Output ($y_1 = \theta_{PG}$) | 0.54<br>15.7 | -14.6<br>23.1 | 2.24 | 47.4 |
| Output ($y_2 = \theta_{RG}$) | 4.29<br>144<br>0 | -15.4<br>21.1<br>$\infty$ | 1.82<br>11.5 | -57<br>81.5 |

The frequency response of SLAT, when the loop was broken at the input is shown in the Figure 17. Similarly, for first and second output broken loop analysis is shown in the Figure 18 and 19 respectively. `linmod()` was used for obtaining the broken loop transfer functions.

- It is evident that negative phase margin for $\theta_{RG}$ is not as desired but since they are at lower frequency, its not a significant concern. Phase margin frequency is lower than the crossover frequency of the loop. The stability of the loop transfer functions may be a concern due to negative phase margin.

- Overall, all gain margins are found to be greater than 6 dB for either gain increase or gain decrease. In the table, negative gain margin represents gain decrease and positive gain margin represents gain increase margin.

- Moreover, SLAT analysis for $\theta_{RG}$ was considered here because launch vehicles have elastic modes at higher frequencies that were ignored in the modeling.

$$\omega_b^* < \omega_c < 1/\tau \tag{27}$$

- As mentioned before, $\omega_b^* = 1.05$ rad/s is a bandwidth frequency specification for desirable performance. $\omega_c$ is a crossover frequency as shown in the table above. $1/\tau$= 25 rad/s is the frequency above which the model is completely unknown/uncertain. Thus, it is verified that the inequality 27 is satisfied for all the crossover frequencies.

### 3.5.9 Robust Performance

Robust Performance guarantees stability and satisfactory performance i.e. disturbance rejection property of the control system. The test for SISO robust performance is given by,

$$\|W_I T_I\|_\infty + \|W_P S\|_\infty < 1 \tag{28}$$

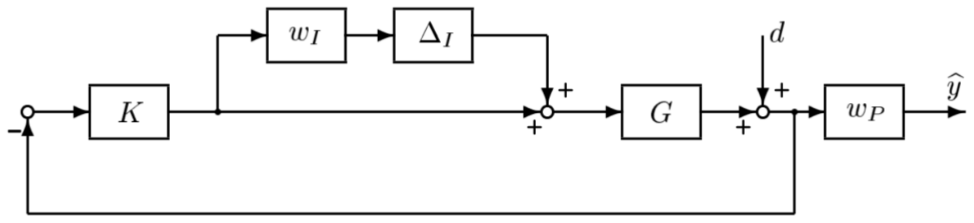


Figure 20: Block diagram for including $W_I$ and $W_p$ as weight matrix

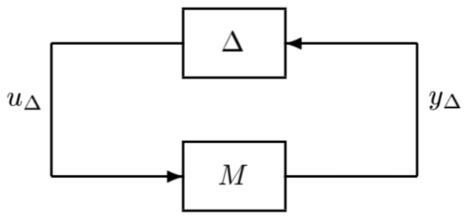


Figure 21: $M - \Delta$ Architecture

As can be seen from attached code, M matrix as shown in Figure 21 can be obtained using Simulink model. For MIMO robust performance, it is desired that $\mu[M] < 1$. For our analysis it is sufficient to check for the SISO test that is presented by inequality 28. From

MATLAB code as attached, $\|W_I T_I\|_\infty + \|W_P S\|_\infty$ was found to be 0.9925, i.e. the RP test is satisfied.

$$\mu[M] = \mu \begin{bmatrix} W_p S & W_p G S_I \\ -W_I S_I K & -W_I T_I \end{bmatrix} \tag{29}$$

## 4 Summary

In this section, a brief summary on the overall project is provided.

- Open loop plant was analyzed for nominal stability and performance, and found to be unstable, but it was controllable and observable.
- In our initial approach, we considered reduced order model that neglected sensor dynamics. Proportional and Integral control was designed for the reduced order model and was implemented on the full $5^{th}$ order model.
- The closed loop system was analyzed for nominal stability and performance. The designed controller was stable and performed well in terms of time domain performance specifications.
- Robust stability and performance measures are summarized in the following table. The closed loop was found to be robust stable for the specified perturbation. Figure 22 shows all the criterion singular values vs. frequency plot.

| Summary of Stability and Performance Measures | | |
|---|---|---|
| | Criterion | Results |
| Nominal Stability | $\mathrm{Re}(\lambda_i) < 0, \forall i$ | Satisfied |
| Nominal Performance | $\|W_P S\|_\infty < 1$ | $\|W_P S\|_\infty = 0.858$ |
| Robust Stability | $\|W_I T_I\|_\infty < 1$ | $\|W_I T_I\|_\infty = 0.629$ |
| Robust Performance | $\|W_I T_I\|_\infty + \|W_P S\|_\infty < 1$ | $\|W_I T_I\|_\infty + \|W_P S\|_\infty = 0.993$ |

- The designed controller works well for vehicles with bending mode frequency at least 25 rad/s. Lower bending mode frequencies will conflict with the assumption that was made. Potentially, controller has to be redesigned for such consideration.

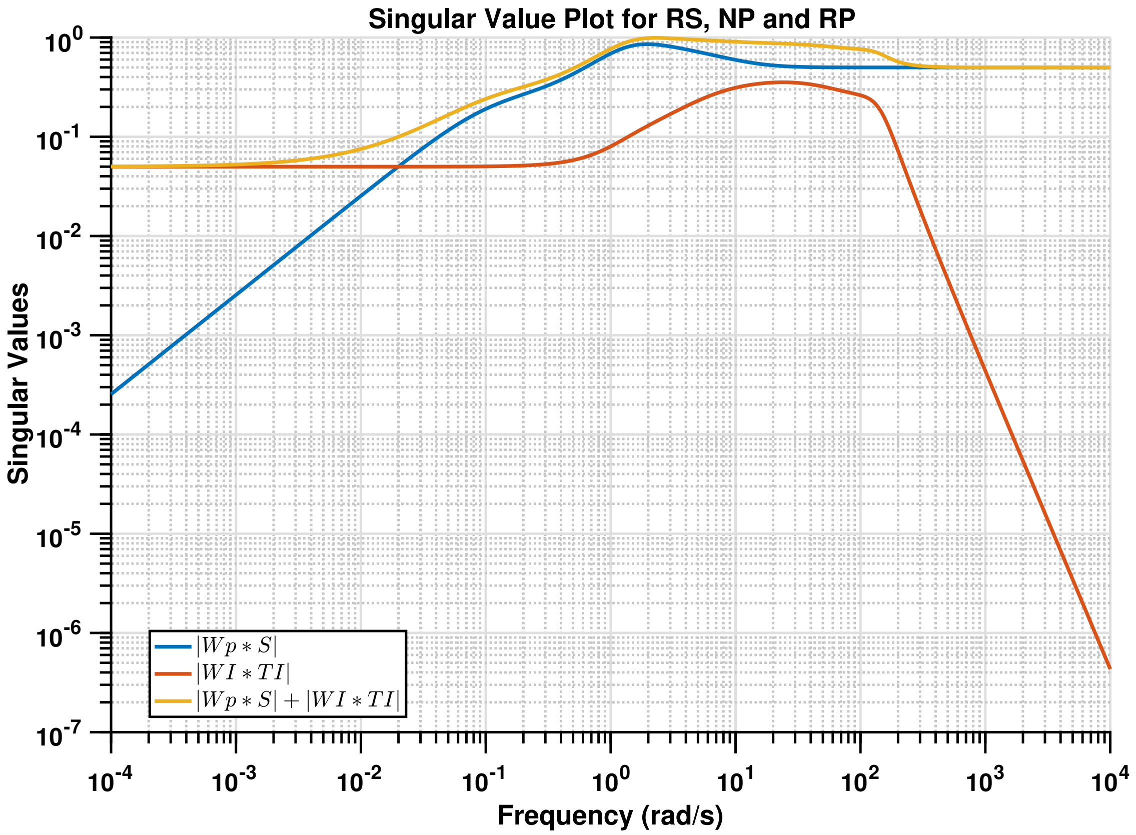


Figure 22: RS, NP and RP Summary

## Conclusion

Control design for the launch vehicle attitude has been studied in this work. Linearized Time Invariant dynamics of Rocket system were used to design an attitude control system using classical and modern control concepts. It was identified that PI feedback compensator can deliver the desired control behavior and performance for this system. Nominal Performance, Nominal Stability, Robust Performance and Robust Stability were analyzed using MATLAB and Simulink tools. It was realized that the controller performance depends on the certain assumptions that were made during the design of a controller. Furthermore, the trade-off between desirable performance at lower frequency and potential compromise at higher frequency was realized during the project. Overall, this was a very good learning experience which allowed us to apply the knowledge that we learned during class.

## Acknowledgment

Our sincere thanks to Prof. Dale Enns for his valuable knowledge, guidance and patience during this project. His classroom discussions and suggestions through email were very helpful for the completion of the project.

## References

[1] Kim, C. H., Commercial Satellite Launch Vehicle Attitude Control Systems Design and Analysis (H-infinity, Loop Shaping, and Coprime Approach): H-infinity, Loop Shaping, and Coprime Factorization Approach, 2007

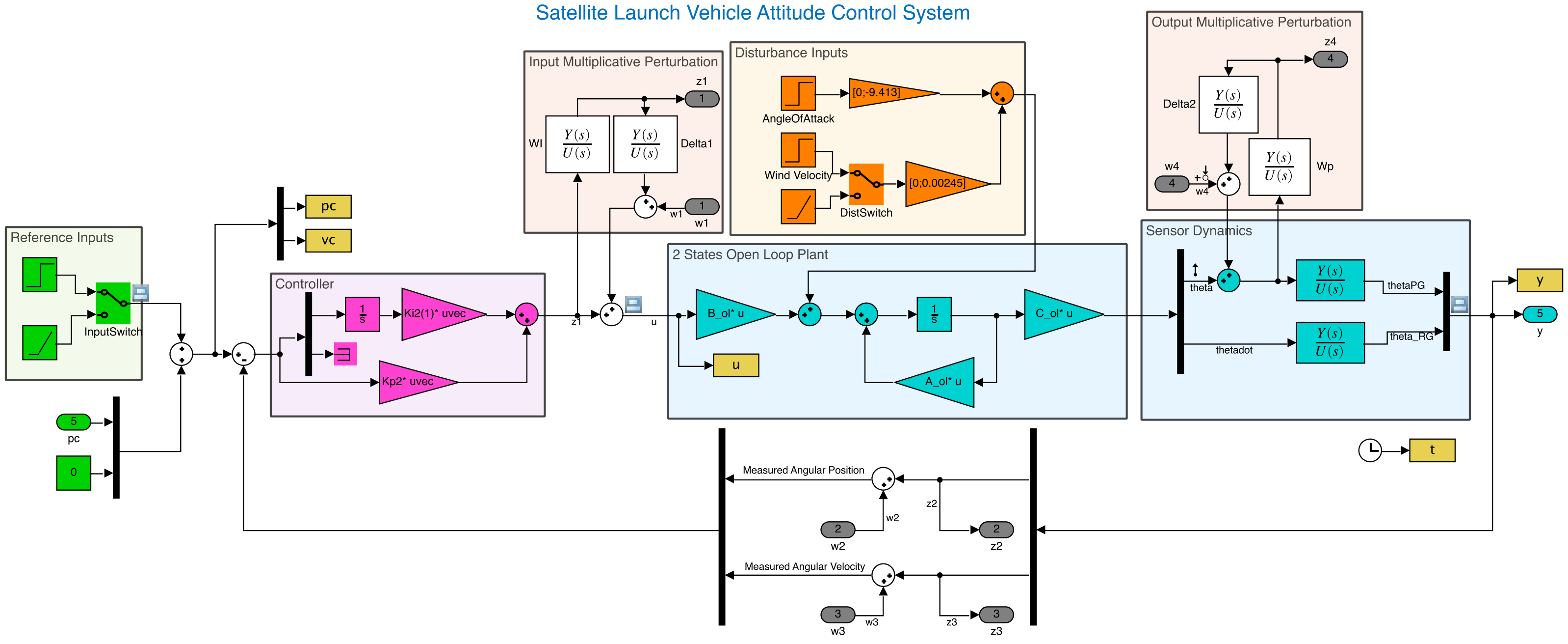
[2] Greensite, A.L., Analysis and design of space vehicle flight control systems, volume 2. National aeronautics and space administration, 1967

[3] Skogestad, S. and Postlethwaite, I., 2007. Multivariable feedback control: analysis and design (Vol. 2, pp. 359-368). New York: Wiley

## A Simulink Model Block Diagram

## B MATLAB Published Code

## Satellite Launch Vehicle Attitude Control System



## 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

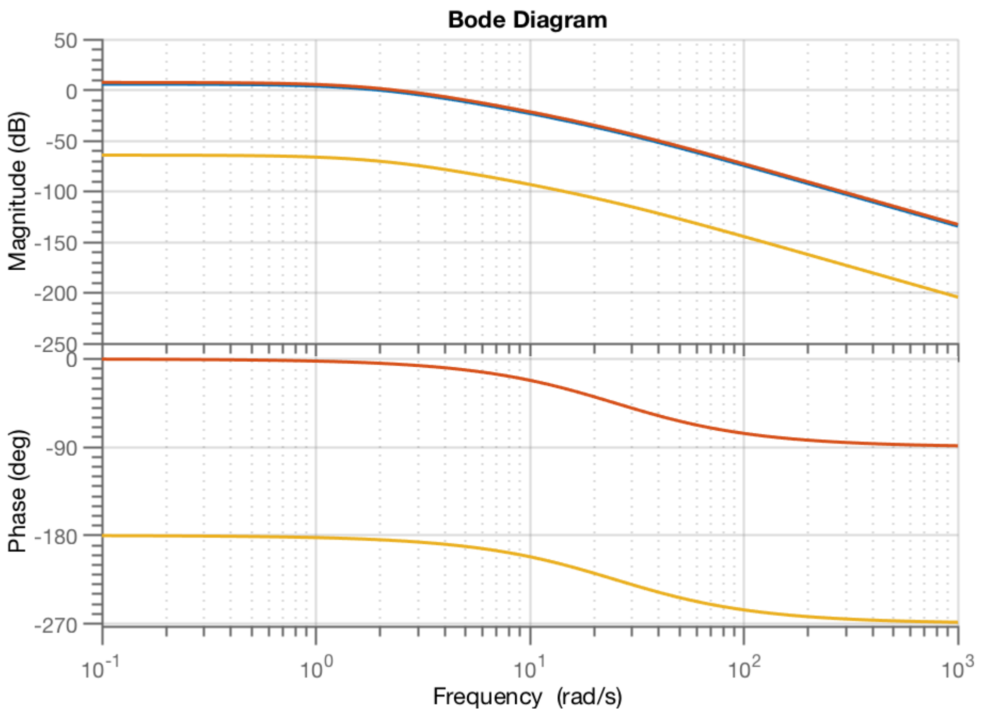
```
% State Space
sys = ss(A,B,C,D);

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

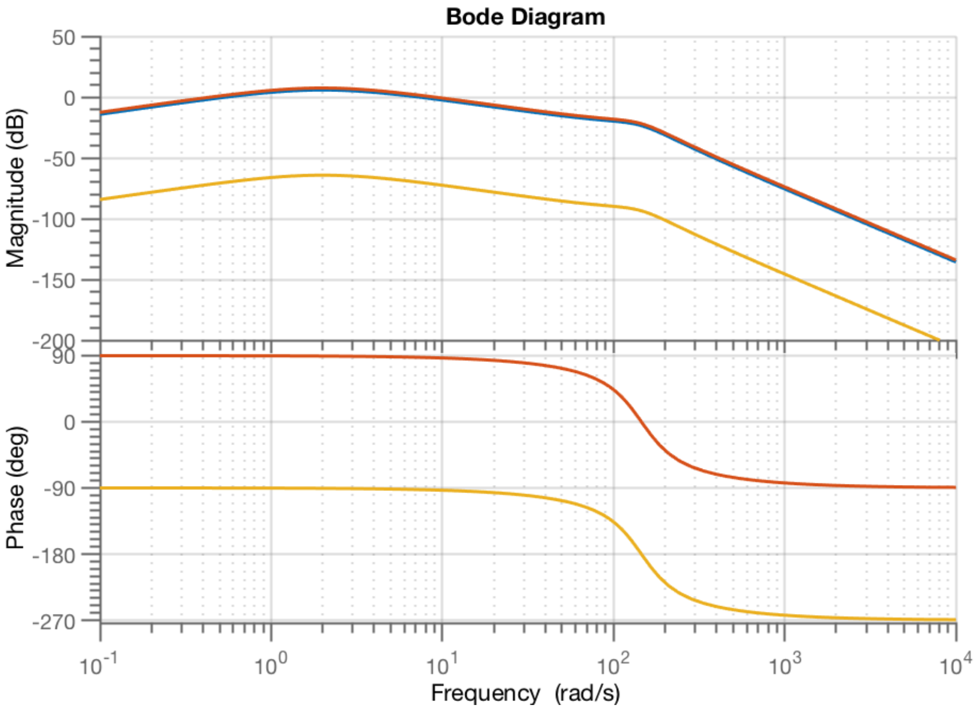
```
============================================================
### Transfer Function Matrix of 5th Order Full State Plant:
============================================================
```

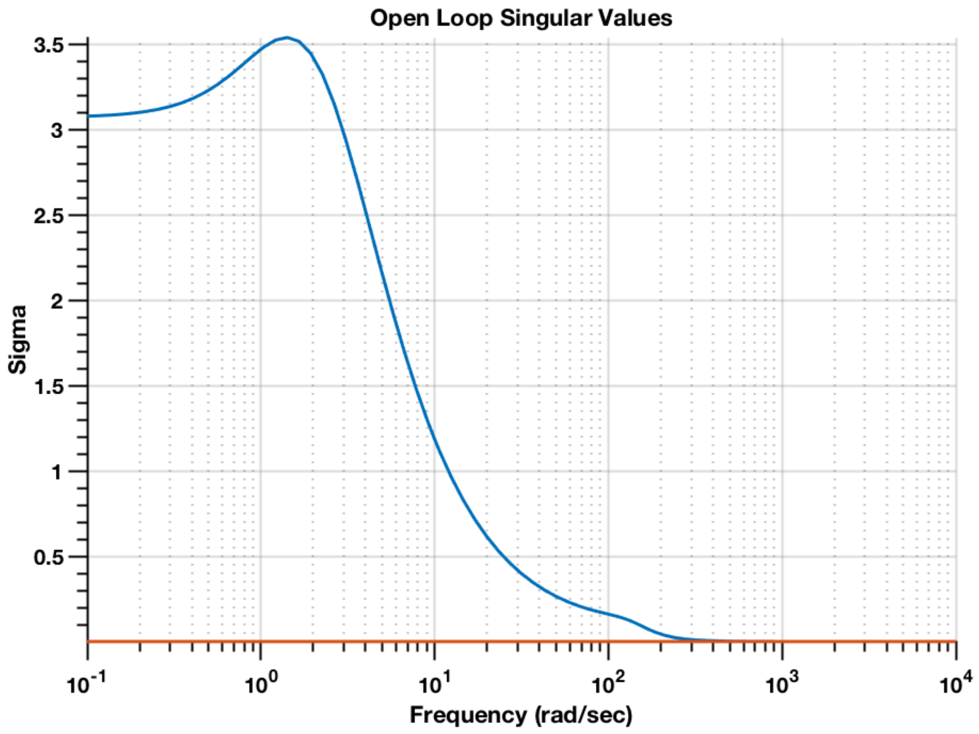
## Nominal Open Loop Analysis

```
% 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      Time Constant

                                         (rad/seconds)      (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

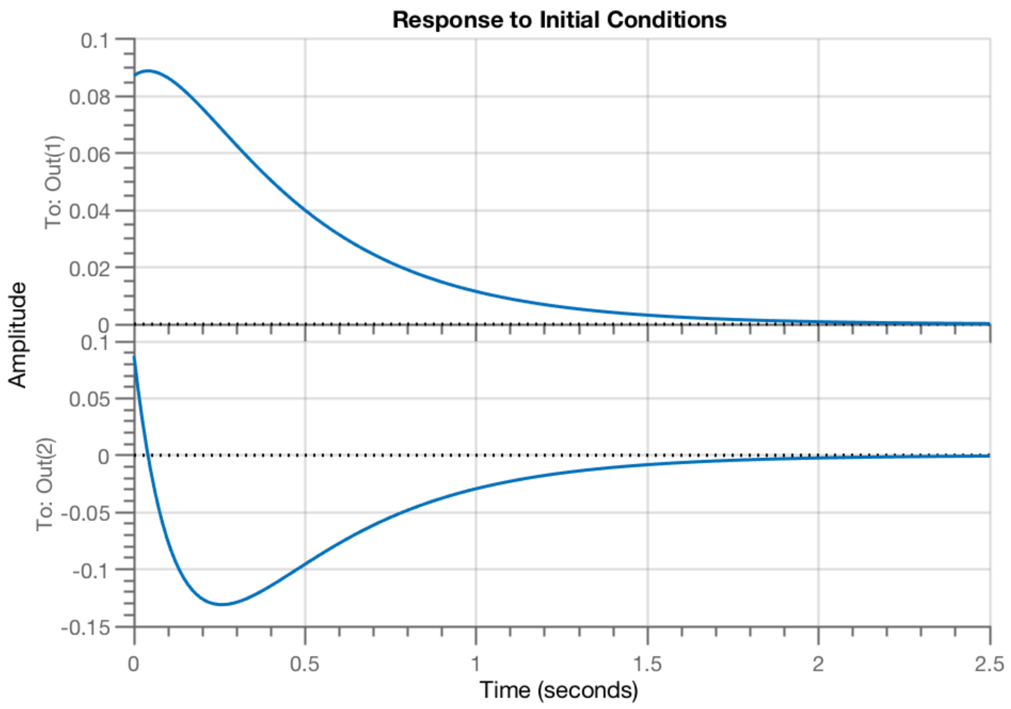
```
% Calculate Gains with LQR
Q2 = diag([1,1]);
R2 = 1;
K2 = lqr(sys2,Q2,R2);
fprintf('=============================================\n');
fprintf('### LQR Control Gains (Approximated System): \n');
fprintf('=============================================\n');
k1 = K2(1) %#ok<*NASGU>
k2 = K2(2)

% 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));
```

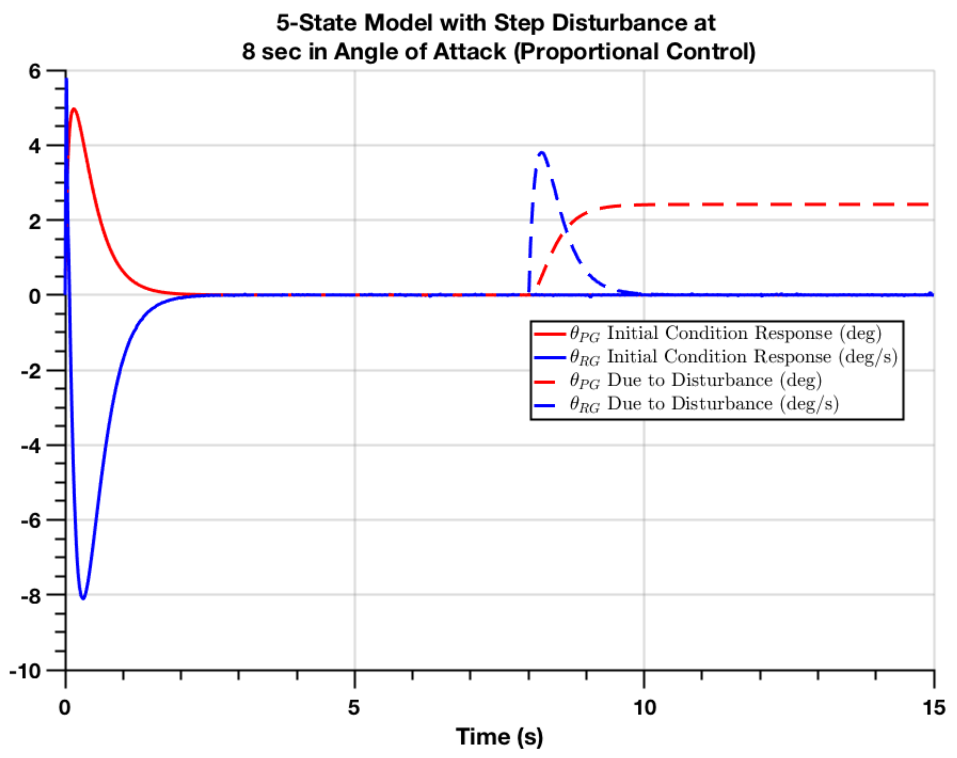
```
=============================================
### LQR Control Gains (Approximated System):
=============================================

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
```

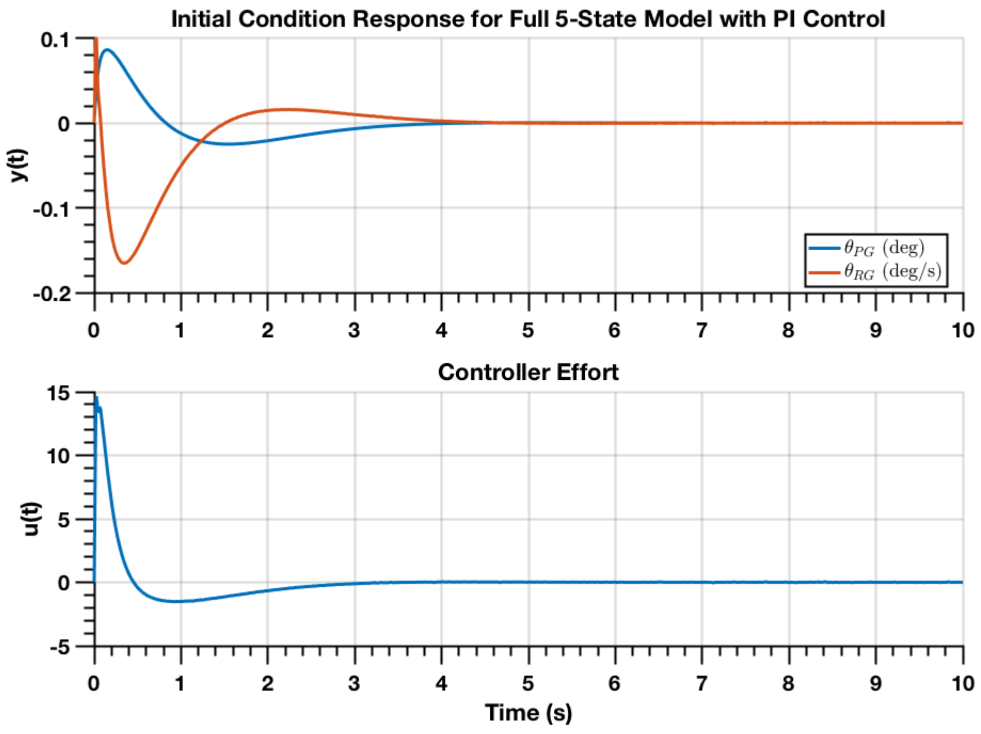
## Evaluate Closed Loop Response with Proportional+Integral Feedback Control

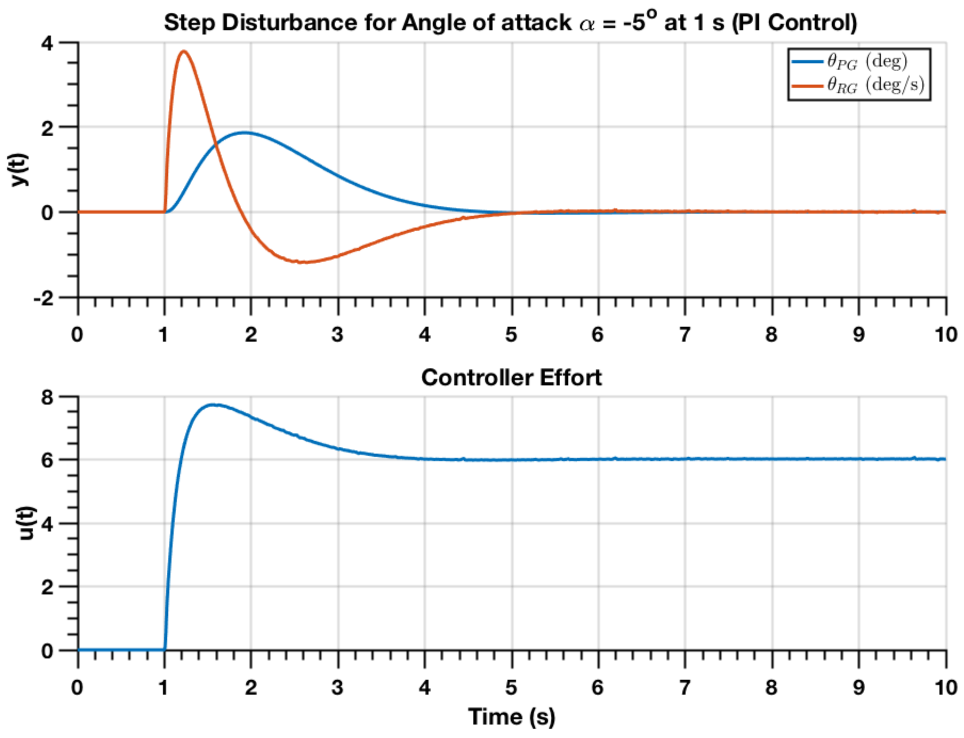
```
%
 =======================================================================
% 1) Simulate Initial Condition Response with 5-State System (Closed
 loop)
%
 =======================================================================

% 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;
```

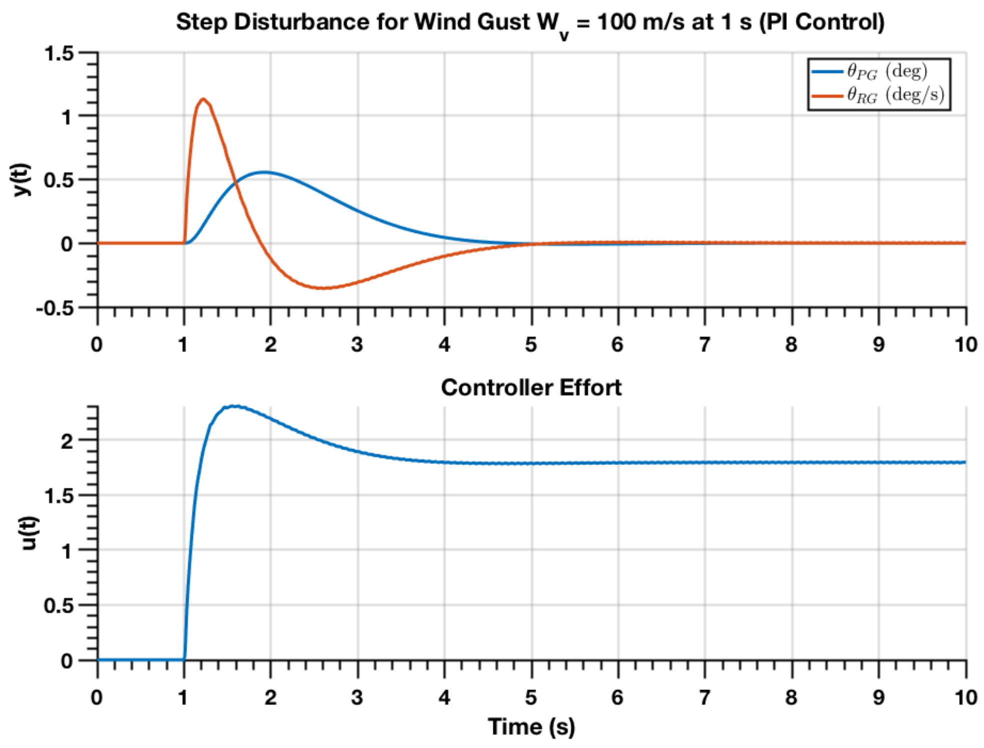
```
subplot(2,1,1);
plot(t,rad2deg(y));
set(findall(0,'type','axes'),'box','off');
title('Step Disturbance for Angle of attack \alpha = -5^{o} at 1 s (PI
 Control)');
legend('$\theta_{PG}$ (deg)','$\theta_{RG}$ (deg/s)');
ylabel('y(t)');

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;
```

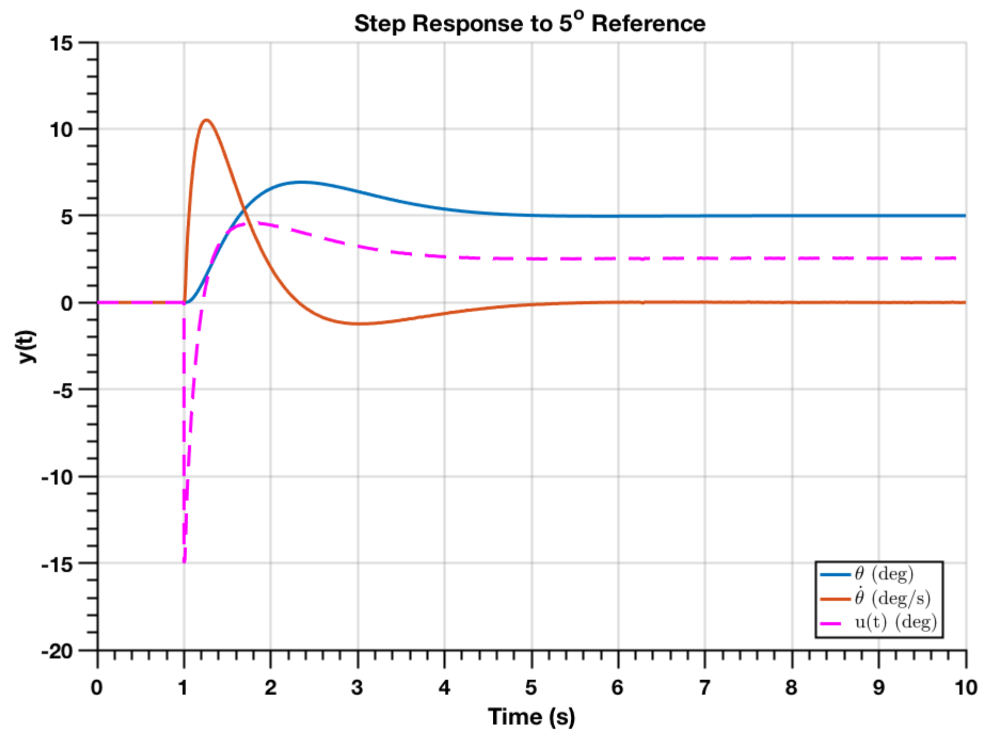
```
stepTime = 1;
thetaStep = deg2rad(5);

% 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)
$\theta_{PG}$ (deg)
$\theta_{RG}$ (deg/s)
Controller Effort
u(t)
Time (s)


Step Disturbance for Angle of attack $\alpha$ = -5° at 1 s (PI Control)
y(t)
$\theta_{PG}$ (deg)
$\theta_{RG}$ (deg/s)
Controller Effort
u(t)
Time (s)



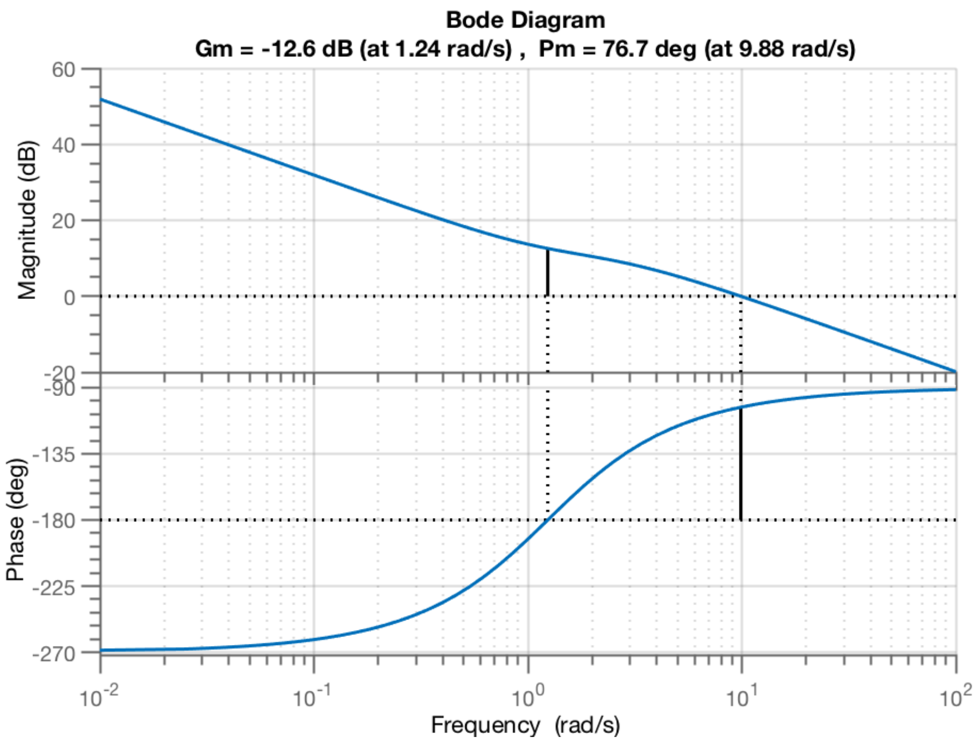


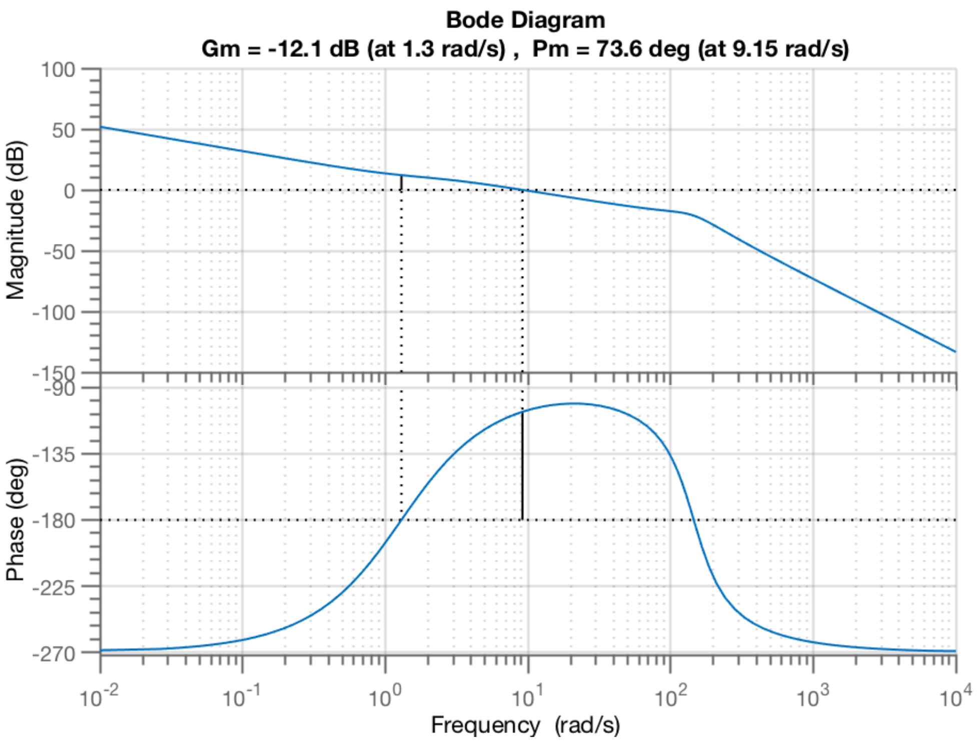
## Nominal Closed Loop Analysis

```
%
  ====================================================================
% 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       Time Constant

                                       (rad/TimeUnit)     (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       Time Constant

                                       (rad/TimeUnit)     (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
```

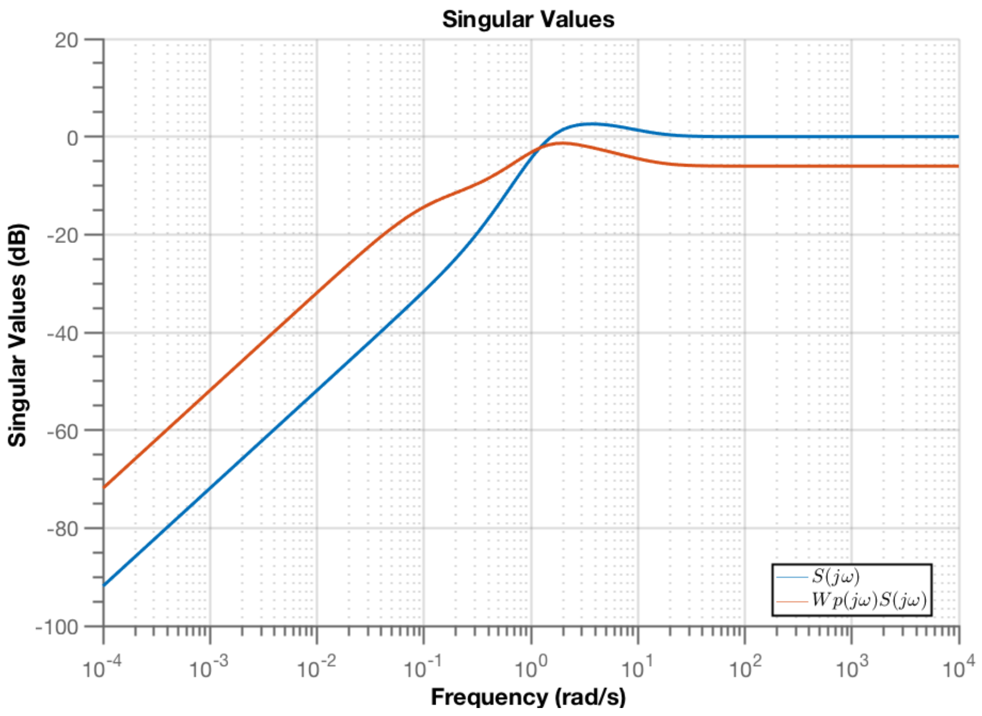




## Nominal Performance Test

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

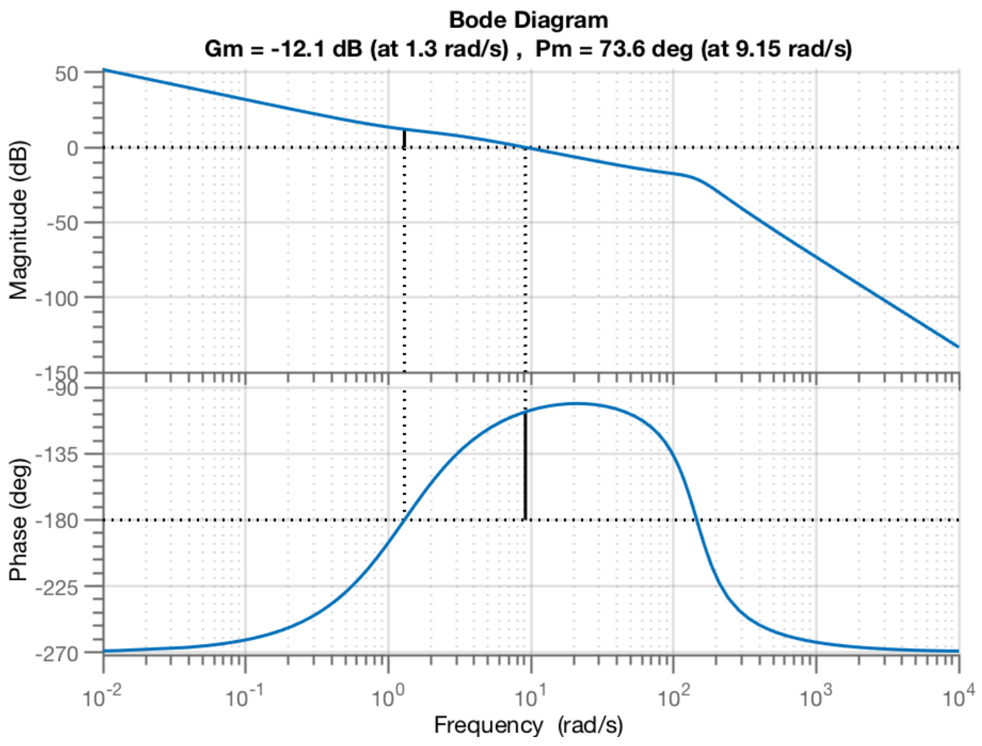
```
% Sensitivity is transfer function related to 4th input and 4th output
S11 = cltf(4,4);

% 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)
```

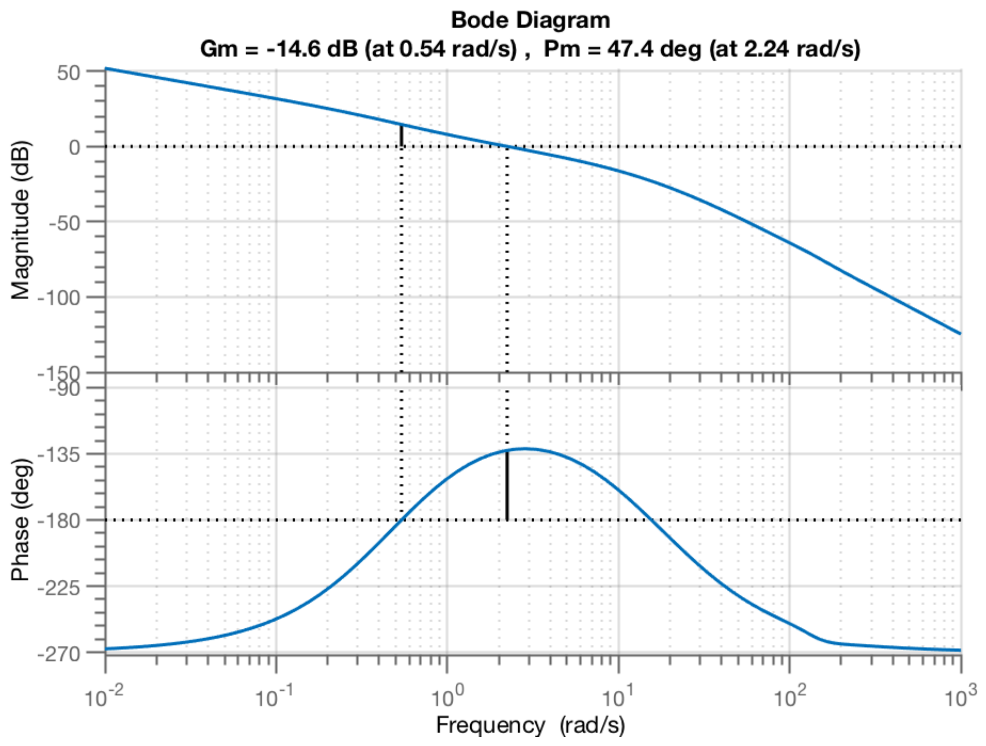
```
==========================================================
### Single Loop at a time loop TF (Loop Broken at Input):
==========================================================

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
```

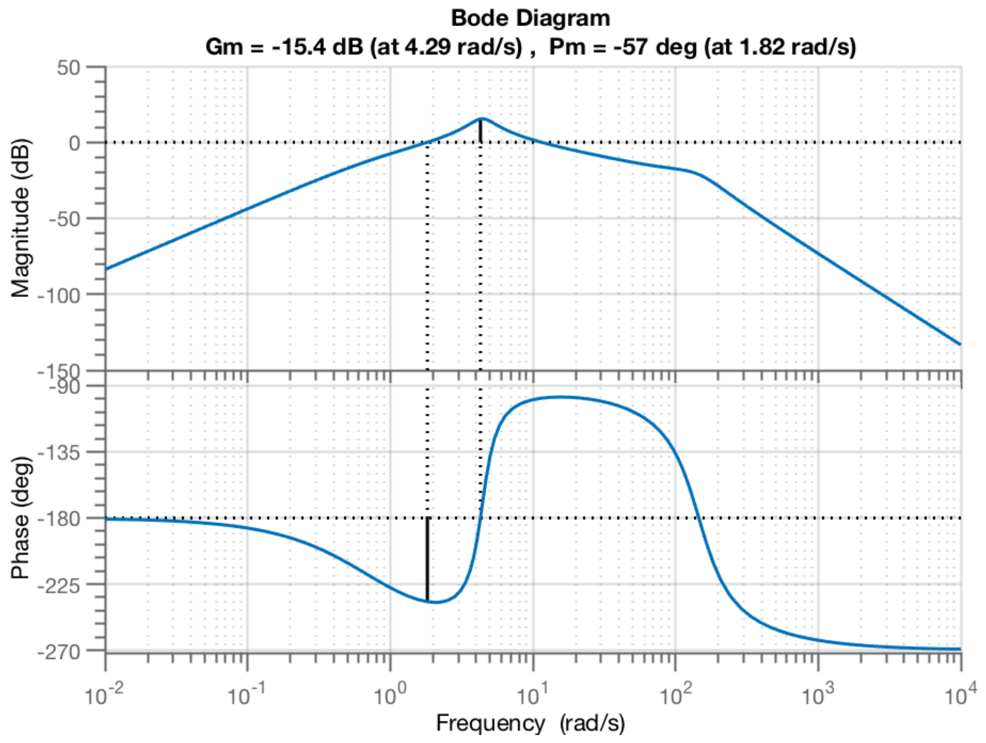


## Single loop at a time analysis for breaking loop at 2-Sensors

```
% 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
```

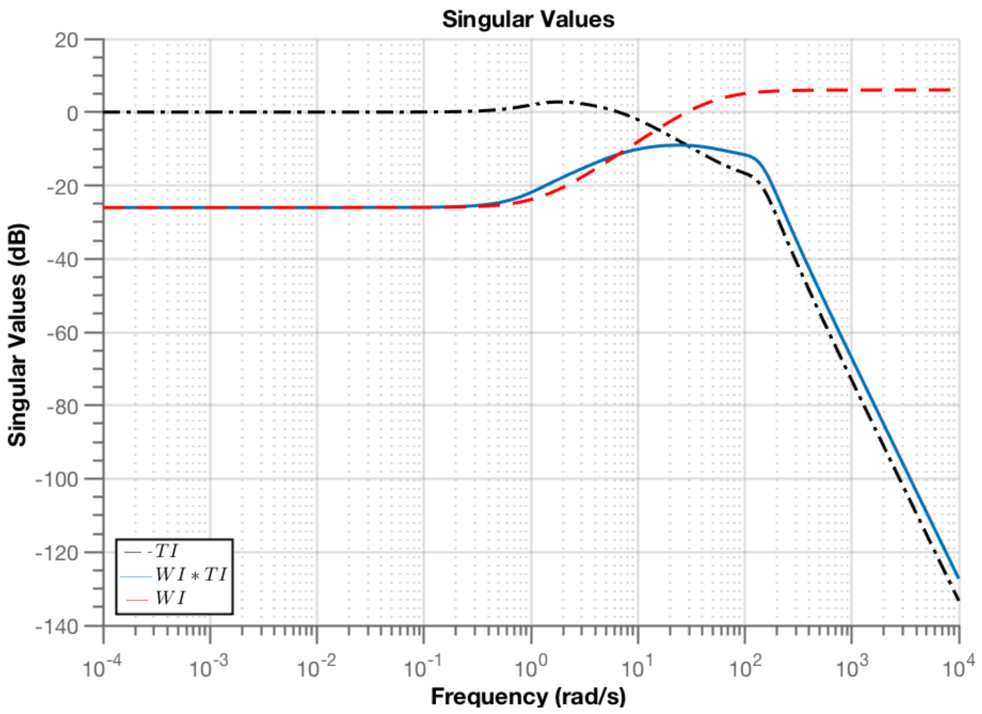




## Define Weighting Transfer Function WI(s) and Test Robust Stability

```
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);
```

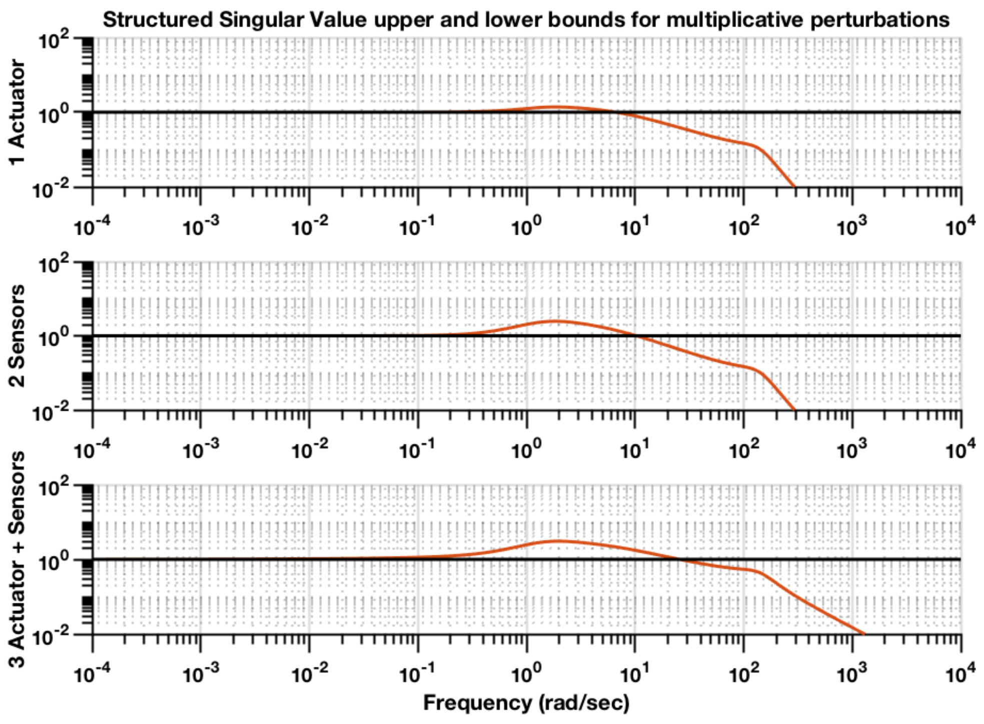
```
% Consider SISO insted
sigmaWpS = sigma(Wp*S11,ww);
sigmaWIT = sigma(WI*TI11,ww);
total = sigmaWpS + sigmaWIT;
fprintf('================================================\n')
fprintf('### SISO Robust Performance |Wp*S| + |WI*T| < 1:\n')
fprintf('================================================\n')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

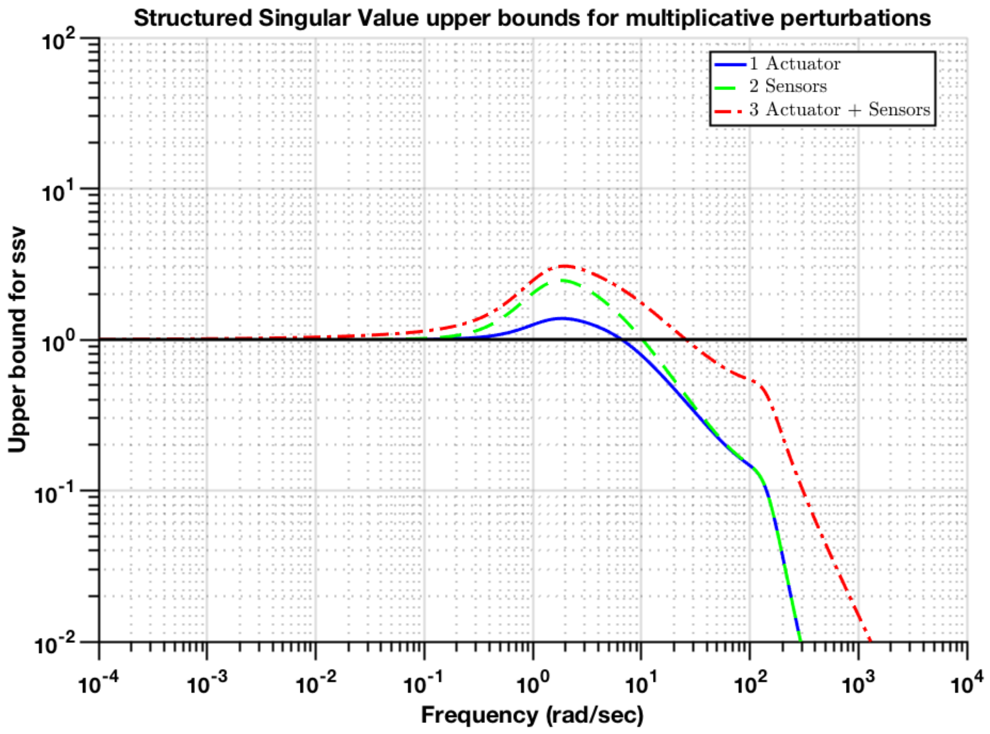
```
================================================
### SISO Robust Performance |Wp*S| + |WI*T| < 1:
================================================
      0.99251
```

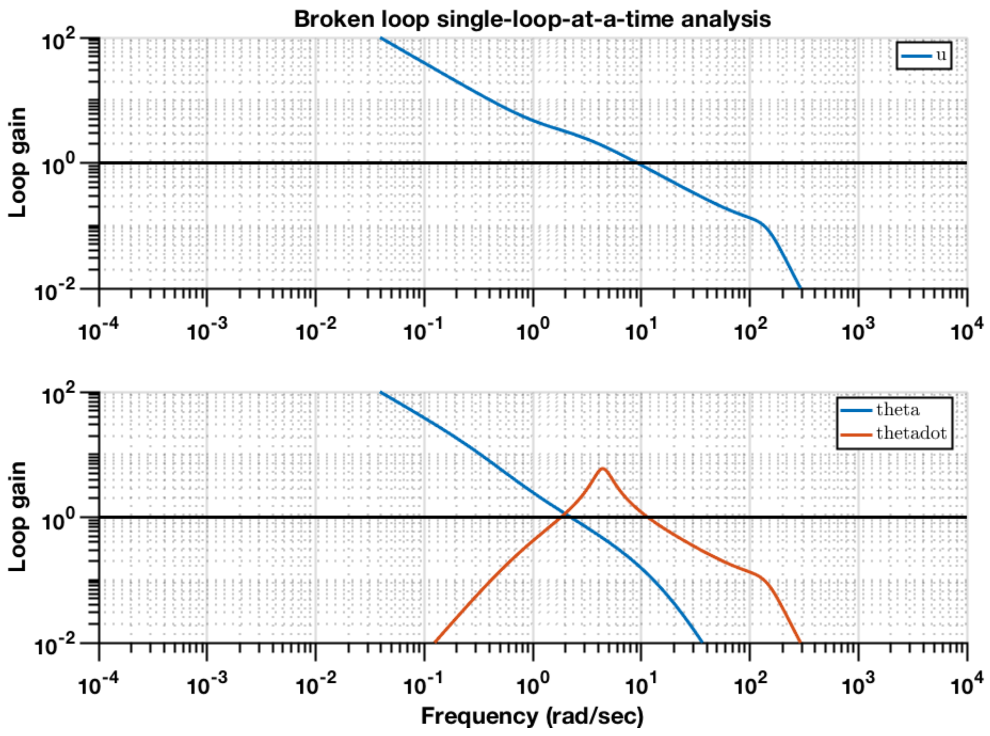
## RS Analysis from code provided in class

```
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
```



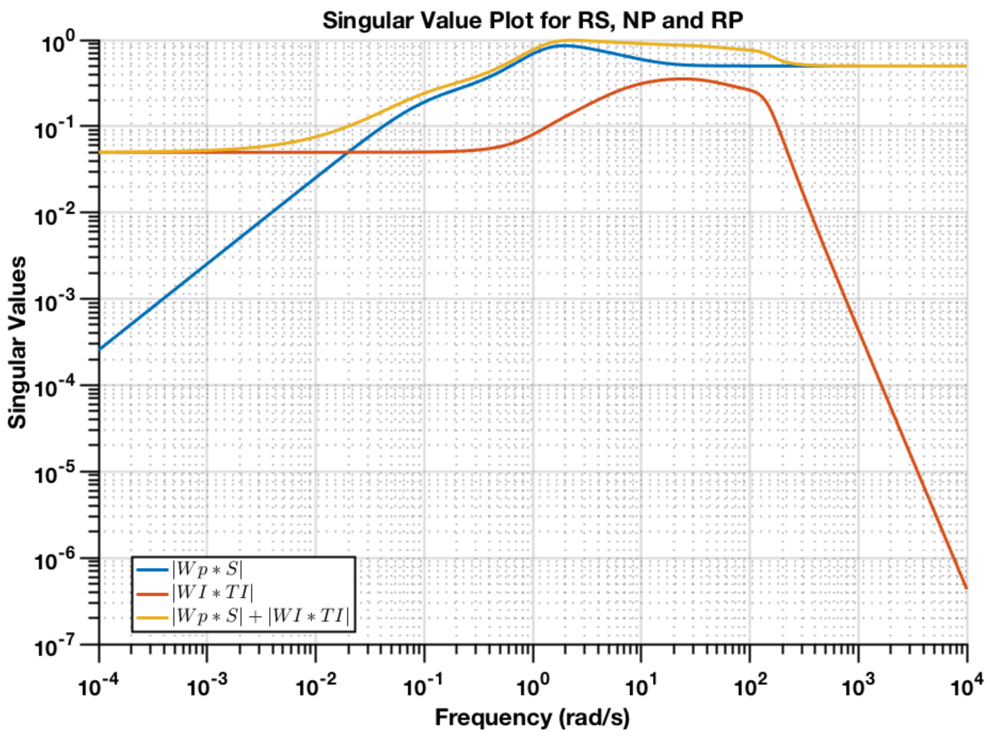




## Summary

```
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*