Robust Flight Control Assignment

AE4351

Delft, June 9th 2024

Pierluigi Rinaldi Meneses 5006872
Antonio Minafra ——-

Bell MIM-3 Nike Ajax
Bendix MIM-8 Talos
Bristol Bloodhound Mk 2
CPMIEC HQ-9 Red Banner
English Electric Thunderbird
Lavochkin S-25 Berkut (SA-1 Guild)
Lavochkin S-75 Dvina (SA-2 Guideline)
0
2m
4m
6m
8m
10m

Contents

1 Introduction

2
System Modelling

In this chapter, the modelling of the air-to-air missile short-period pitch axis is done. In Section 2.1 where the general procedure for obtaining the operating point and linearizing a non-linear system using MATLAB is discussed, along with the assumptions used and the validity of the model concerning its flight envelope. Then, in Section 2.2, the model is constructed and analyzed in MATLAB and the results obtained are discussed.

2.1. Flight Dynamics

Linear control theory techniques require the model considered to be a linear, time-invariant model. To obtain such a model, linearization of the non-linear model is required, which means an equilibrium point around which to linearized is needed first. Finding an equilibrium point is appealing since, by slightly perturbing one of the variables while the system is "at rest", one can observe the system's behaviour near this equilibrium point. If the state trajectory quickly diverges from the equilibrium point due to a small disturbance, it indicates the system would be difficult to control. The missile's non-linear system can be described by a set of two non-linear equations:

$$\begin{aligned}\dot{x}(t) &= f_x\left[x(t), u(t), \rho(t)\right]\\ y(t) &= f_y\left[x(t), u(t), \rho(t)\right]\end{aligned}$$

The main assumption behind finding the operating point is that the time derivatives of the states are zero, i.e., $\dot{x}(t) = 0$. We also impose the equilibrium condition $\rho(t) = \bar{\rho}$. It is then necessary to find the equilibrium states, inputs, and outputs $\bar{x}, \bar{u}, \bar{y}$ using the state and output equations. Generally, to solve the system, it is necessary to balance the number of equations with the number of variables by fixing some variables from the state, input, and/or output vector. For example, for the short-period model, one can choose to fix the control δ_m, the state α, or the output n_z.

Computation of the operating point can be performed in MATLAB using two different approaches. The numerical approach consists of using the Simulink Control Design toolset to define an operating point, edit its properties, configure the options of the numerical algorithm (maximum number of steps, type of optimization method, etc.) and finally computing the operating point. Analytical trimming involves solving moment and force equations to find the equilibrium inputs, and states and then substituting those in the output equations to find the equilibrium outputs. This can be done automatically by defining the model in Simulink, and then using the Model Linearizer.

Once the operating point is found, a linearization procedure can be applied to the non-linear model. This is done to simplify the analysis of the system and the development of suitable controllers for it since it reduces the complexity of the model by representing the non-linear system as a linear one. In doing so, it allows for the use of powerful control design (LQG, H-infinity) and stability analysis techniques that are only available for linear systems. This culminates in the ability to effectively design robust controllers that ensure performance and stability in the presence of uncertainties and disturbances.

The general linearization procedure for a non-linear system is as follows: first, the trim point (operating point) is obtained following the methods described above. After the trim point is obtained, the non-linear system dynamics are approximated using a 1st-Order Taylor series expansion around the operating point. This is done by computing the Jacobian matrices of the system equations around the operating point, as shown below.

$$A = \left.\frac{\partial f_x}{\partial x}\right|_{\{\bar{x},\bar{u}\}}, B_u = \left.\frac{\partial f_x}{\partial u}\right|_{\{\bar{x},\bar{u}\}}, C_y = \left.\frac{\partial f_y}{\partial x}\right|_{\{\bar{x},\bar{u}\}}, D_{yu} = \left.\frac{\partial f_y}{\partial u}\right|_{\{\bar{x},\bar{u}\}}$$

The Jacobian matrices are then used to formulate a linearized state-space model as shown below.

$$\dot{x} = A \cdot x + B_u \cdot u \quad y = C_y \cdot x + D_{yu} \cdot u \tag{2.1}$$

It is important to note that to achieve the process above several assumptions are made about the system: it is assumed that the linearized system is only valid for small perturbations around the operating point since the higher-order terms of the Taylor series expansion are neglected. Furthermore, it is assumed that the operating point is constant, otherwise, a new linearization procedure would have to be done each time an analysis is made. Finally, non-linear effects are assumed to be negligible in the vicinity of the operating point. It should also be noted that, as stated previously, the validity of the linearized, non-linear system model is valid for a flight envelope that does not have very high deviations from the operating point condition, otherwise further uncertainties would be introduced into the model, which cannot be accounted for.

The linearization procedure described above can be performed in MATLAB by using either numerical or analytical methods. To linearize the non-linear system numerically the Simulink Model Linearizer App is used along with the Simulink Control Design toolset. The approach taken is to first define the inputs and outputs of the non-linear system in Simulink by right-clicking on the input/output port signals, selecting Linear Analysis Points and navigating to the Input Perturbation & Output Measurement option. Afterwards, using the `getlinio` function, the `ModelIO` object can be imported, followed by setting the linearization options for the numerical algorithm through the `linearizeOptions` function, with the option of defining the state ordering of the model. Finally, the linearized model LinModel with I/O `ModelIO` around the trim point can be computed using the `linearize` function.

The above procedure can also be performed analytically in MATLAB by making use of the Symbolic Math toolbox. Using the `syms` function, all variables and relevant parameters of the non-linear equations of motion of the system can be defined as symbolic objects. Afterwards, using the `diff` function in MATLAB, the derivatives of the symbolic functions with respect to each state and control input are obtained and the state space matrices are found by appropriately grouping such derivatives. Next real parametric uncertainties can be added to the resulting state matrices by using the `realp` function of the Robust Control Toolbox, which redefines the uncertain components. The uncertain model can then be formed using the `uss` function of the same Robust Control Toolbox.

2.2. Model Construction & Analysis

In the following section the creation process of the state space models for the missile, actuator and combined system in MATLAB is briefly described. It should be noted that the missile and actuator models considered correspond to the system's short-period motion states (α and q) and the fin deflection/deflection rate, respectively, with the aim of controlling the aiframe's vertical acceleration. This is due to the fact that the airframe is analysed when it is in a high manoeuvrability condition and thereby its most important eigenmotion is the short-period one. Finally, a stability analysis of the open loop airframe model is done by checking the system's poles and zeros.

Figure 2.1 shows the Simulink model of the airframe model, where G_{act} is the state space model of the missile actuators and G_m is the missile short-period state space model.

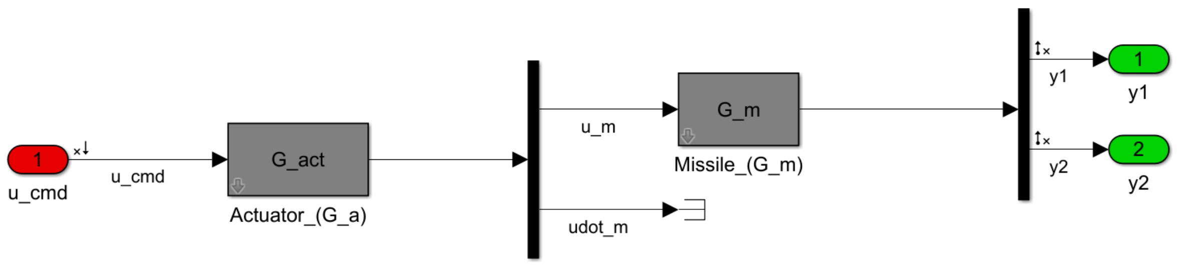


Figure 2.1: Simulink model of the missile airframe

First, an equilibrium point was obtained by using the `findop` function as described in the previous section. Then, the actuator plus missile system G_{am} was linearized around the found equilibrium point. The state-space matrices of the Airframe system are shown below in Equation 2.2 and Equation 2.3, with the states, $x^T = [\alpha, q, \delta_q, \dot{\delta_q}]$, correctly re-arranged to be in order and $y^T = [n_z, q]$.

$$\dot{x} = A_{am} \cdot x + B_{am} \cdot u_{cmd} \qquad y = C_{am} \cdot x + D_{am} \cdot u_{cmd} \tag{2.2}$$

$$\begin{aligned} A_{am} &= \begin{bmatrix} -1.305 & 1 & -0.114 & 0 \\ 0 & -300.4 & 0 & -131.4 \\ 0 & 0 & 0 & 1 \\ 0 & -2.25\times10^4 & 0 & -210 \end{bmatrix}, \quad & B_{am} &= \begin{bmatrix} 0 \\ 0 \\ 2.25\times10^4 \\ 0 \end{bmatrix}, \\ C_{am} &= \begin{bmatrix} -146.3 & 0 & -11.73 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, & D_{am} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned} \tag{2.3}$$

Next, the stability of the system needs to be analysed. This was carried out by obtaining the transfer functions of the command input to the first and second output, which are shown in Equation 2.4 and Equation 2.5 respectively. To visualize the poles and zeros location and their characteristics, the Pole-Zero map of the airframe system was plotted as shown in Figure 2.2

$$H_{u_{\text{cmd}}}^{n_z}(s) = \frac{-2.639\times10^5(s-36.64)(s+36.52)}{(s^2+1.305s+300.4)(s^2+210s+2.25\times10^4)} \tag{2.4}$$

$$H_{u_{\text{cmd}}}^{q}(s) = \frac{-2.9564\times10^6(s+1.044)}{(s^2+1.305s+300.4)(s^2+210s+2.25\times10^4)} \tag{2.5}$$

As can be seen from the plot, all the poles of the system lie in the left-half plane, meaning the Airframe system is stable. The dominant pole is the one corresponding to the missile dynamics, as it is closer to the imaginary axis. This pole corresponds to the missile short-period motion, characterized by a by a low damping ratio, leading to an oscillatory behavior with a slow decay of transients. The actuator dynamics belongs instead to the second pole, with a higher damping ratio. This is in line with the typical dynamics of actuators, which are designed to ensure a quick response with minimal oscillations.

Furthermore, the normal acceleration channel has both minimum and non-minimum phase characteristics due to the presence of zeros on left and right side of the complex plane, respectively. The impact of the non-minimum phase zero can be observed in the system's time response to a step input, where an initial undershoot occurs before the response achieves a positive acceleration.

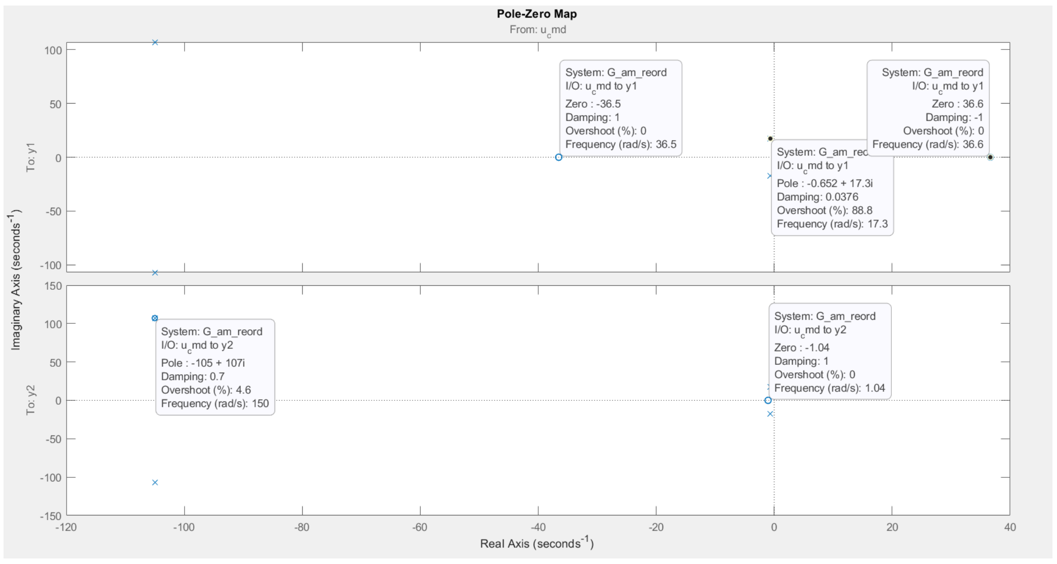
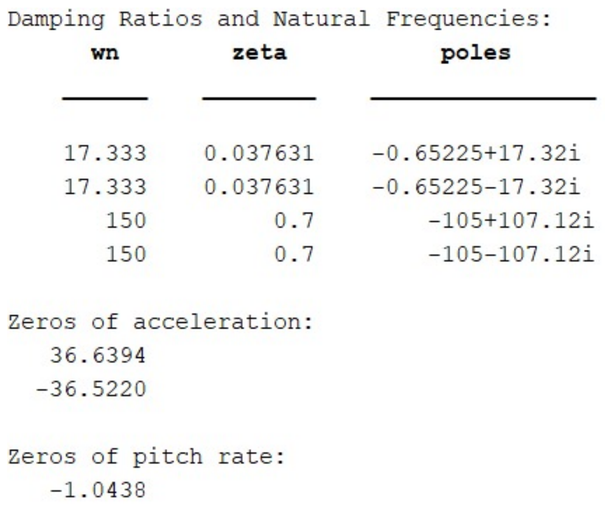


Figure 2.2: Pole-zero map of the missile plus actuator system

The conclusions obtained from the Pole-Zero map used for the stability analysis can be further verified through Figure 2.3. In it, the values of certain open-loop system characteristics such as the poles and zero values as well as the natural frequency, **wn**, and the damping ratio, **zeta**, obtained through the `damp` and `zero` MATLAB functions are shown.



```
Damping Ratios and Natural Frequencies:
      wn          zeta             poles
    ______      ________      _______________

    17.333      0.037631      -0.65225+17.32i
    17.333      0.037631      -0.65225-17.32i
       150           0.7          -105+107.12i
       150           0.7          -105-107.12i

Zeros of acceleration:
   36.6394
  -36.5220

Zeros of pitch rate:
   -1.0438
```

Figure 2.3: Stability analysis verification using the `damp` & `zero` functions

3
Loop Shaping

In the following chapter, two feedback loops were implemented in the system. In Section 3.1, a damping gain is added to the feedback of the pitch rate. This is followed by an open-loop scaling gain in Section 3.2 and an integrator feedback gain for the acceleration channel in Section 3.3.

3.1. Damping Gain Design

In this section, a damping gain, C_q, for the system is designed using simple root locus techniques. This is applied to the pitch rate, q, using negative feedback so as to increase the damping of the missile's poles (airframe dominant poles) [1].

At first, using the transfer function shown in Equation 2.5 and the `rlocusplot` MATLAB function, a value for the damping gain is obtained. This is done by manually moving the airframe poles in such a way that the damping ration becomes 0.7 while still maintaining their stability, which yielded a value of C_q = -0.163. Next a Simulink model implementing such a gain was constructed by incorporating the damping gain using negative feedback into the already existing model shown in Figure 2.1. This can be seen in Figure 3.1.

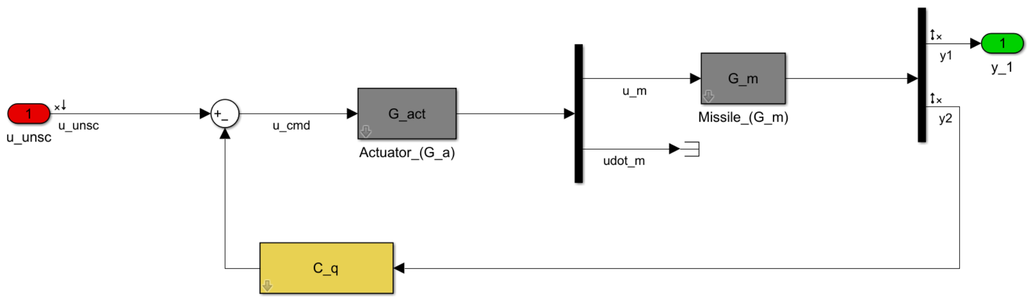


Figure 3.1: Simulink model of the missile airframe with the closed loop gain C_q implemented

After the Simulink model was constructed, the model was linearized using the `linearize` MATLAB function and the new transfer function linking the input, u_{unsc}, to the vertical acceleration, n_z, was obtained as shown in Equation 3.1. This was then compared against the original system normal acceleration transfer function, Equation 3.2, in Figure 3.2 so as to verify the effects of the damping gain.

$$H_{u_{unsc}}^{n_z}(s) = \frac{-2.639 \times 10^5 (s - 36.64)(s + 36.52)}{(s^2 + 28.57s + 416.5)(s^2 + 182.7s + 1.744 \times 10^4)} \tag{3.1}$$

$$H_{u_{\mathrm{cmd}}}^{n_z}(s) = \frac{-2.639 \times 10^5 (s - 36.64)(s + 36.52)}{(s^2 + 1.305s + 300.4)(s^2 + 210s + 2.25 \times 10^4)} \tag{3.2}$$

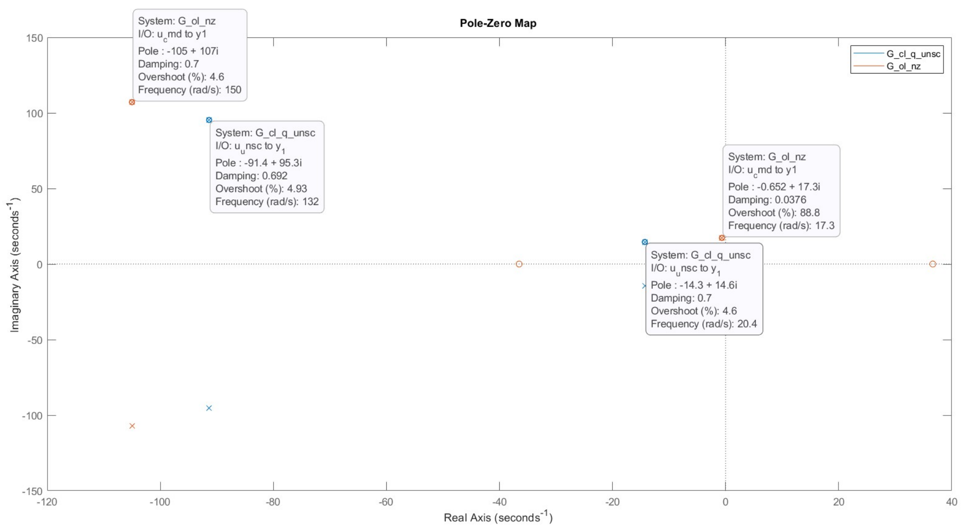


Figure 3.2: Pole Zero map of the model of the missile airframe and the missile airframe with the closed loop gain C_q implemented

As can be seen Figure 3.2, the effects of the addition of the damping gain into the airframe model using negative feedback are twofold. On the one hand, as can be seen by the rectangular boxes of the poles it has a stabilizing effect on the dominant poles since their real parts become more negative. Moreover, the increase in damping ratio can also be seen: in the open-loop system the damping ratio was 0.038 and with the addition of the negative feedback C_q it became 0.7. Secondly, it can be seen that this does not occur for the non-dominant poles since in that case not only does the damping ratio marginally decrease from 0.7 to 0.69 but the real part of the pole also becomes more positive. This seems to imply that the addition of C_q into the airframe model has a destabilizing effect on the actuator poles. Finally, it can also be seen that C_q has no influence on the zeros of the airframe model since no changes can be seen in the Pole Zero map.

3.2. Scaling Gain Design

The objective of this section involves designing a scaling gain C_{sc} to normalize the inner loop transfer function such that its steady-state gain is equal to 1. This means that a unit step input will result in a unit step output, which makes it easier to analyze the performance of the system. This was obtained using the `dcgain` function to compute the DC gain of the unscaled closed-loop transfer function, which was then inverted to yield a value for the scaling gain of 0.0206. The gain was then added to the open loop of the system as shown in Figure 3.3. The output of the step response was checked to be converging to 1, with a 5% settling time of 0.246 [s]. The new system transfer function from u_p to the normal acceleration is given in Equation 3.3.

$$H_{u_p}^{n_z}(s) = \frac{-5427.3(s-36.64)(s+36.52)}{(s^2+28.57s+416.5)(s^2+182.7s+1.744\times10^4)} \tag{3.3}$$

In order to improve the overall robustness of the system, a feedback loop for the normal acceleration is needed. This is further motivated after observing the response of acceleration after introducing a step disturbance in the pitch rate feedback channel, with a zero reference input. This situation is illustrated in the left subfigure of Figure 3.6, which clearly shows that the disturbance is not rejected. To ensure the system can reject disturbances and accurately track the reference input, an integrator with an appropriate gain must be included in the design of the controller.

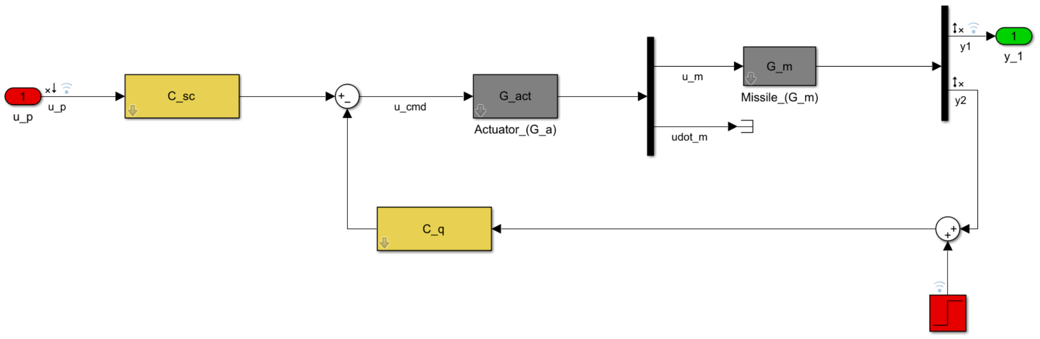


Figure 3.3: Simulink model of the missile airframe with C_q and C_{sc} and the step disturbance

3.3. Integral Gain Design

As explained in the previous section, an integrator with an integral gain is needed in the feedback loop of the normal acceleration for effective disturbance rejection. An integrator term and a gain C_i were added to the forward path of the system, and the value of the gain was set to be unitary (C_i = 1.0) prior to tuning. The resulting Simulink model is shown in Figure 3.4. The system is then linearized and the transfer function from the tracking error input $e1_f$ to the output $y1$ is extracted yielding Equation 3.4.

$$H^{n_z}_{e1_f}(s) = \frac{-5427.3(s+36.52)(s-36.64)}{(s+1.089)(s^2+27.1s+381.1)(s^2+183.1s+1.75\times10^4)} \tag{3.4}$$

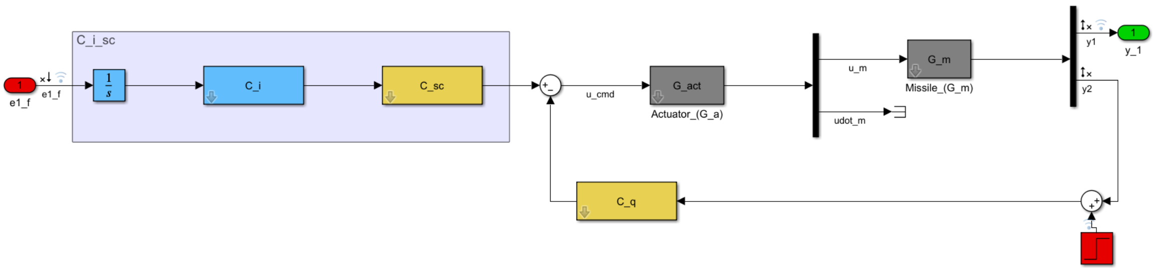


Figure 3.4: Simulink model of the missile airframe with C_q and C_{sc} and the integral gain C_i

The gain tuning of the integral gain was performed using the `sisotool` MATLAB function on the linearized transfer function mentioned above. The specific value for the gain was then obtained by adjusting the original unitary value to achieve either 60°or otherwise the minimum value of the 5 % settling time of the system's step response. A good compromise was found to be C_i = 5.48 as shown in Figure 3.5, where the new transfer function for the normal acceleration of the resulting closed-loop system is as shown in Equation 3.5.

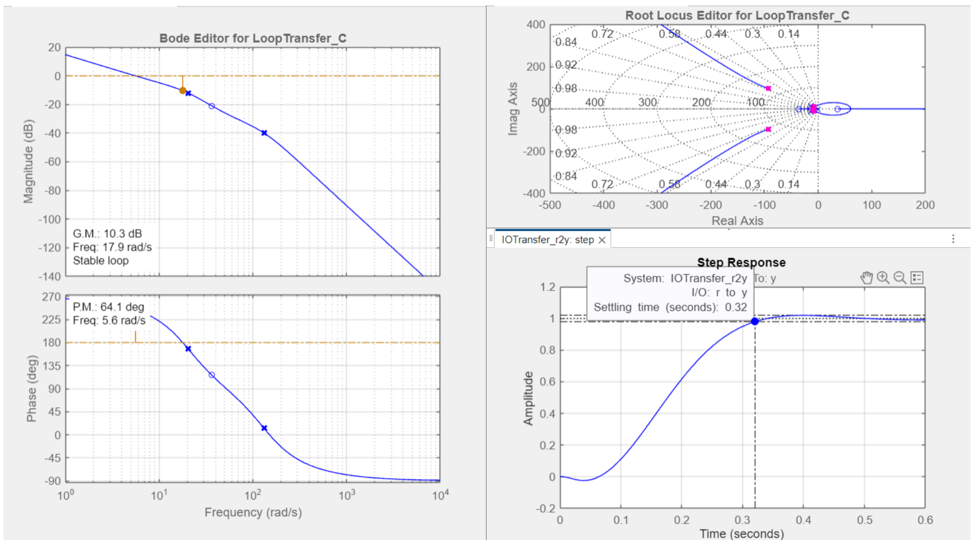


Figure 3.5: Bode plots, root locus and step response of the missile airframe with C_q and C_{sc} and the integral gain C_i

$$H_{e1_f}^{n_z}(s) = \frac{-29741(s+36.52)(s-36.64)}{(s+10.21)(s^2+16.265s+219)(s^2+184.8s+1.78\times10^4)} \tag{3.5}$$

Finally, in order to verify the correct implementation of the pitch rate integral disturbance rejection control system, a final test was made using the Data Inspector feature of Simulink and evaluating the time response of the airframe system with and without the addition of the integral gain. As can be seen in Figure 3.6, the previously damped and scaled system (left step graph) showed a convergence of the vertical acceleration of the missile towards a value of around 8 g's, unlike the 0g reference signal. On the other hand, the new system with integral gain (right) showed the expected behaviour as it can be seen that the system converges to a steady state value of 0 g's, which is the same as the reference signal. Hence, the correct implementation of the pitch rate integral disturbance rejection controller was verified.

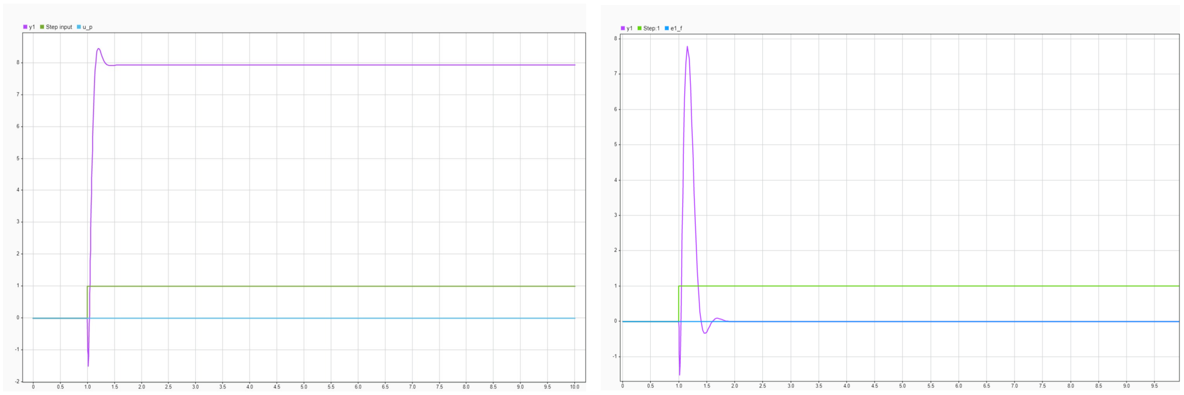


Figure 3.6: Step responses of the airframe system without (left) and with (right) the addition of integral gain

4 Mixed Sensitivity Design (Weighting Filters)

In this chapter the computation of the weighting filers that are to be used during the mixed sensitivity robust controller design is described. This is done by first explaining the rationale behind the characteristics of the filters in Section 4.1, after which the actual computation is described in Section 4.2.

4.1. Weighting Filter

The rocket configuration is an example of the standard form of mixed-sensitivity control. This is basically tracking an input y_c by minimizing its influence on the tracking error e = $yc - y$, on the control input u, and on the output y. To achieve this, the rocket plant is going to be augmented by the use of weighting filters. In general, the weighting filters must be selected as rational, stable, and minimum phase approximations of the singular value responses of the inverse closed-loop transfer functions. For the rocket model, three different weighting functions are considered: W_1, applied to the output sensitivity function S_o, W_2 applied to the control input KS_o and W_3 on the model following error $y_{1_d} - y_1$. It is important to notice that the shaping functions approximate the inverse of the transfer function, so the focus will be in obtaining the inverse of these functions. The inverse weighting filter W_1^{-1} will be used to shape the output sensitivity function:

$$W_S^{-1}(s) = \frac{(s+\epsilon_S)(s+2\zeta_c\omega_c)}{s^2+2\zeta_c\omega_c s+\omega_c^2}$$

which is a measure of the effect of disturbances on the output of the system. At low frequencies, the transfer function is dominated by $\frac{(\epsilon_S)(2\zeta_c\omega_c)}{\omega_c^2}$ which provides some degree of attenuation, since ϵ_S is small. At high frequencies, the opposite is true: the function approaches 1 meaning that $W_1^{-1}(s)$ is a high pass filter. A similar reasoning can be applied for the second weighting filter W_2 employed at the sensitivity function times the controller KS_o. Its function is given by:

$$W_{KS}^{-1}(s) = \frac{\epsilon_{KS}^{-1}s+1}{\omega_{\mathrm{bt}}^{-1}s+1}$$

with a large ϵ_{KS}, which implies a small ϵ_{KS}^{-1}. This is a low pass filter since at low frequencies the function approximates to 1. At high frequencies, instead, the terms ϵ_{KS}^{-1} and $\omega_{\mathrm{bt}}^{-1}$ dominate, but since ϵ_{KS}^{-1} is small, the high-frequency gain is also small. In the next section, the two filters will be designed with the help of the `makeweight` function.

4.2. Weighting Filter Computation

In order to compute the transfer functions of the two filters, the function `makeweight` was employed. The inputs of this function are the DC and HF gain needed, and a specific point, on the magnitude plot where the gain of the function will pass through. For the peak value of the output sensitivity function, an approximated value was obtained from Equation 4.1 with a minimum phase margin of 30°. Moreover, two requirements were set for the specific point of both weighting transfer functions. In the case of W_1, the -3.01 [dB] bandwidth of the sensitivity function had to be 4 [rad/s], whereas for W_2, the -3.01 [dB] bandwidth of the actuator was obtained from its step response (151 [rad/s]) and at that frequency the gain had to be at least -15 [dB]. The parameters used to construct the two filters given the above requirements are given in Table 4.1.

$$PM \geq 2 \arcsin\left(\frac{1}{2M_S}\right) \tag{4.1}$$

Parameter	$W_1^{-}1$	$W_2^{-}1$
DC Gain [dB]	-60	100
HF Gain [dB]	5.72	-40
Gain [dB]	-3.01	-15
Freq [rad/s]	4	151

Table 4.1: Parameters used for the `makeweight` function

Here, is assumed that for both filters, a continuous time filter of order 1 is used, of the form given in Equation 4.2. From the obtained transfer functions using the `makeweight` function, the parameters were recomputed, and are shown in Table 4.2. The functions were then inverted and their singular value magnitude vs frequency plot was obtained as shown in Figure 4.1. In the plot, the parameter ω_i represents the break frequencies of the inverse filters, M_i the low-frequency gain factor, in other words, the slope of the function at low frequencies, and A_i the gain scaling factor at high frequencies.

$$W_1(s) = \frac{s/M_1 + \omega_1}{s + \omega_1 A_1} \text{ and } W_2(s) = \frac{s + \omega_2/A_2}{M_2 s + \omega_2} \tag{4.2}$$

Param.	**Value**
A_1	$10^{-}3$
M_1	1.932
ω_1	5.264
A_2	10^5
M_2	0.01
ω_2	26.81

Table 4.2: Parameter Values for Filters

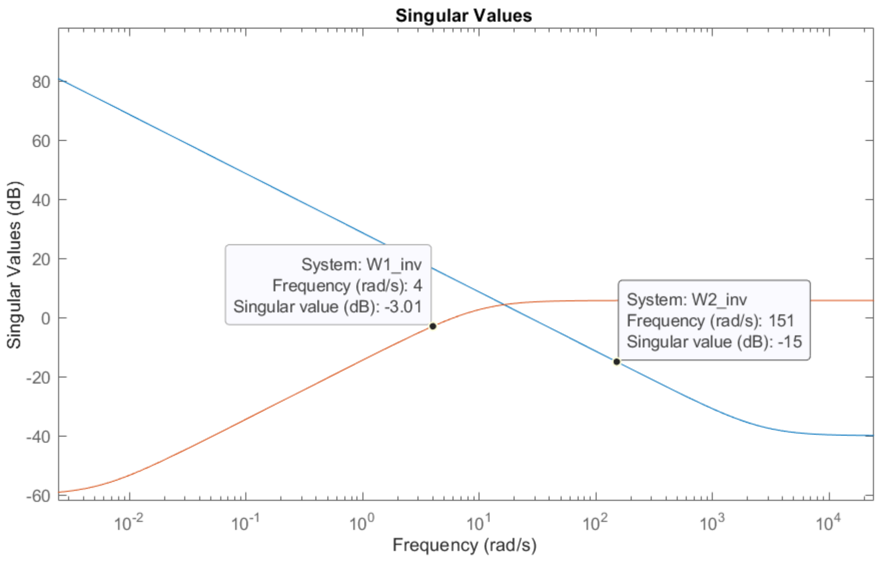


Figure 4.1: Singular value plots of the

5

Mixed Sensitivity Design (Reference Model)

In this chapter, the construction of a reference model that is to be used by the closed-loop controllers for reference tracking is described. This reference model will then be used in conjunction with feedback disturbance rejection controllers in Chapter 6 and Chapter 7 to design a robust feedback disturbance rejection and reference tracking controller.

5.1. Reference Model Computation

In this section, the general process used to obtain the reference model will be described, followed by a brief verification of the resulting transfer function step response.

Starting with the process of obtaining the reference model, as stated in the assignment description [1], it is selected to be modelled as a 2nd order system whose general form is as shown in Equation 5.1.

$$T_d(s) = \frac{\omega_d^2\left(-\frac{s}{z_m}+1\right)}{s^2+2\zeta_d\omega_d s+\omega_d^2} \tag{5.1}$$

Where, ω_d is its natural frequency, ζ_d its damping ratio and z_m the acceleration-channel non-minimum phase zero found in Section 2.2, whose value is 36.6394. To obtain the natural frequency and damping ratio of the system, two targets were set: a 0.18 [s] 5% step response settling time and a 5% step response overshoot. These were then used in conjunction with the MATLAB optimization algorithm `fmincon` to find their smallest possible value that meets the two requirements. In doing so a value of 0.716 for ζ_d and a value of 37.452 [rad/s] for ω_d were found, which yielded a step response that satisfied the requirements, however, this response showed a significant undershoot, which was undesirable. In order to account for this, the values were then adjusted through trial and error so as to give a satisfactory step response while still respecting the requirements mentioned above. This resulted in a value of 0.79 for ζ_d and a value of 22.6 [rad/s] for ω_d. These values provided the transfer function shown by Equation 5.2.

$$T_d(s) = \frac{-13.94\,(s-36.64)}{s^2+35.71s+510.8} \tag{5.2}$$

In order to verify the optimality of the solutions found by the optimization algorithm, the step response of the above transfer function was evaluated and its two relevant characteristics were obtained, as shown in Figure 5.1. As can be seen both the settling time and overshoot requirements were met and, therefore, the reference model transfer function was verified to be correctly calculated.

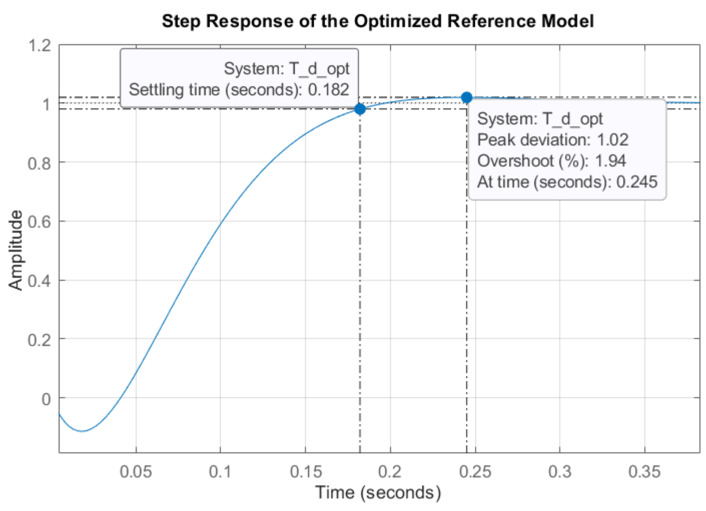


Figure 5.1: Step Response of the reference model with Overshoot and Settling time

6 Mixed Sensitivity Design (Feedback controller design using hinfsyn)

In this chapter the weighting filters designed in Chapter 4 as well as the reference model constructed in Chapter 5 are used in order to design a disturbance rejection/tracking controller using the `hinfsyn` H_∞ norm synthesis MATLAB function. In Section 6.1 the design of the controller is discussed. This is followed by a simplification of the controller by using model order reduction techniques in Section 6.2. Finally, the controller's performance and its simulation will be analysed in Section 6.3.

6.1. Controller Design

As mentioned above, this section will describe the design of the disturbance rejection/tracking controller, however before doing so, the focus will be shifted to proving the derivation of the open and closed loop weighted matrix transfer functions. To prove that the weighted closed-loop matrix transfer function $T_{wz}(s)$ is given by:

$$T_{wz}(s) = \begin{bmatrix} W_1 S_o \\ W_2 C_e S_o \\ W_3 (T_d - T_o) \end{bmatrix}$$

where,

$$S_o = (1 + GC_e)^{-1} \quad \text{and} \quad T_o = GC_e S_o = 1 - S_o$$

The complementary sensitivity function T_o is given by $T_o = 1 - S_o = GC_e S_o = \frac{GC_e}{1+GC_e}$

Then the transfer function from y_1 to w is found $\frac{y_1}{w} = \frac{GC_e}{1+GC_e}$

- Determining z_1:

  $$z_1 = W_1 e_1 = W_1 (w - y_1)$$

  Using $y_1 = \frac{GC_e w}{1+GC_e}$:

  $$z_1 = W_1 \left(w - \frac{GC_e w}{1 + GC_e} \right) W_1 w \left(1 - \frac{GC_e}{1 + GC_e} \right) = W_1 w \cdot S_o$$

  Therefore,

  $$\frac{z_1}{w} = W_1 S_o$$

- Determining z_2:

  $$z_2 = W_2 u = W_2 C_e (w - y_1) = W_2 C_e w \left(1 - \frac{GC_e}{1 + GC_e} \right) = W_2 C_e w \cdot S_o$$

  Therefore,

  $$\frac{z_2}{w} = W_2 C_e S_o$$

Next the error signal $e_{1,d}$ is determined:

$$e_{1,d} = wT_d - y_1 \implies e_{1,d} = w\left(T_d - \frac{GC_e}{1+GC_e}\right)$$

After which it is expressed in terms of w:

$$\frac{e_{1,d}}{w} = T_d - T_o$$

- Finally z_3 can be determined by:

$$z_3 = W_3 e_{1,d}$$

Such that,

$$\frac{z_3}{w} = W_3(T_d - T_o)$$

Combining these results, the weighted closed-loop matrix transfer function $T_{wz}(s)$ is:

$$T_{wz}(s) = \begin{bmatrix} W_1 S_o \\ W_2 C_e S_o \\ W_3(T_d - T_o) \end{bmatrix}$$

The weighted open loop matrix transfer function, P(s), of the model shown in Figure 6.1 is assumed to have the form as shown by Equation 6.1

$$\begin{bmatrix} z \\ v \end{bmatrix} = P \begin{bmatrix} w \\ u \end{bmatrix} \implies \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ v \end{bmatrix} = \begin{bmatrix} W_1 & -W_1 G \\ 0 & W_2 \\ W_3 T_d & -W_3 G \\ 1 & -G \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix} \tag{6.1}$$

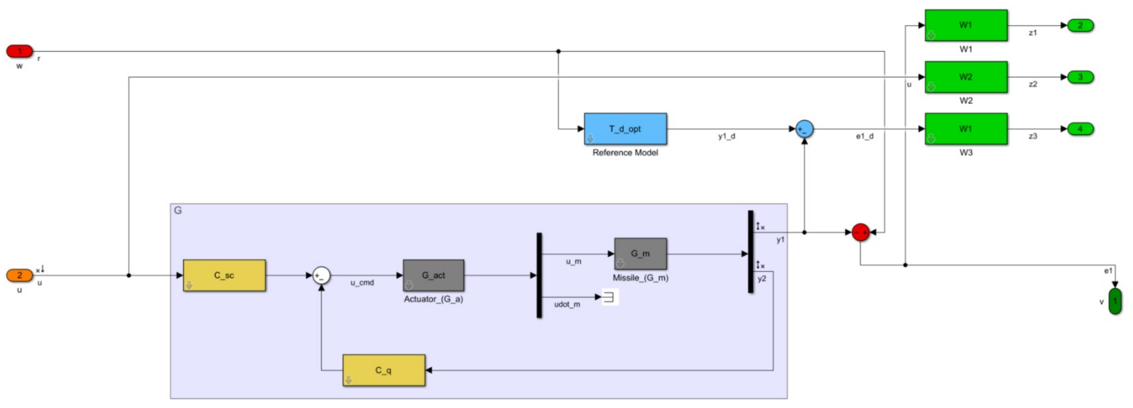


Figure 6.1: Open loop model with weighted filters

Taking a look at the open model with weighted filters shown in Figure 6.1, in order to prove that P(s) is as shown, only the open loop channels are considered resulting in the four equations below:

$$z_1 = W_1 v = W_1 \cdot (w - Gu) = W_1 \cdot w - W_1 \cdot Gu \tag{6.2}$$

$$z_2 = W_2 \cdot u \tag{6.3}$$

$$z_3 = W_3 \cdot (T_D w - Y_1) = W_3 \cdot T_D \cdot w - W_3 \cdot G \cdot u \tag{6.4}$$

$$v = w - G \cdot u \tag{6.5}$$

The system can then be written in a coherent mathematical form, where the actual values of the transfer functions are as shown below:

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ v \end{bmatrix} = \begin{bmatrix} \frac{0.51764(s+10.17)}{s+0.005264} & \frac{2809.4(s-36.64)(s+36.52)(s+10.17)}{(s+0.005264)(s^2+28.57s+416.5)(s^2+182.7s+1.744\cdot 10^4)} \\ 0 & \frac{100(s+0.0002681)}{s+2681} \\ \frac{-11.16(s-36.64)(s+32.01)}{(s+0.02563)(s^2+35.71s+510.8)} & \frac{4345(s-36.64)(s+36.52)(s+32.01)}{(s+0.02563)(s^2+28.57s+416.5)(s^2+182.7s+1.744\cdot 10^4)} \\ 1 & \frac{5427.3(s-36.64)(s+36.52)}{(s^2+28.57s+416.5)(s^2+182.7s+1.744\cdot 10^4)} \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix}$$

The weight matrix transfer functions are given below, where $W_1(s) = W_3(s)$:

$$W_1(s) = \frac{0.51764(s+10.17)}{(s+0.005264)} \tag{6.6}$$

$$W_2(s) = \frac{100(s+0.0002681)}{(s+2681)} \tag{6.7}$$

The correctness of the weighted open loop matrix transfer function, P(s), can then be verified by comparing, for example, the entry of the first row and column with $W_1(s)$ and the entry of the second-row second column with $W_2(s)$.

Now that the derivation of the open and closed-loop weighted matrix transfer functions has been proven, the focus will once again shift to the design of the controller. The controller C_e, corresponding to the integral controller C_i introduced in Chapter 3, was designed by making use of the H_∞ norm synthesis MATLAB function, `hinfsyn`. The `hinfsynOptions` MATLAB function was then used to customize some of the options of the synthesis function such as choosing the Riccati method and a relative tolerance of 10¯6, [1]. This in combination with the weighted open-loop matrix transfer function P(s), shown above, as well as the number of measurements (1) and controls (1), displayed in orange and dark green respectively in Figure 6.1, yielded the state-space representation of the controller, C_e, along with the global performance parameter γ and the closed-loop weighted matrix transfer function $T_{wz}(s)$. It should be noted that the $T_{wz}(s)$ obtained with this method was then verified against the ones obtained from theory to secure the correct implementation of the synthesis function. Table 6.1 and Figure 6.2 show the global&individual performance parameters and the singular value plots of the global&individual closed loop weighted transfer functions respectively.

γ	**Value**
$\gamma_{T_{wz}}$	0.9341
γ_1	0.8646
γ_2	0.7860
γ_3	0.6605

Table 6.1: Performance values with $W_3 = W_1$

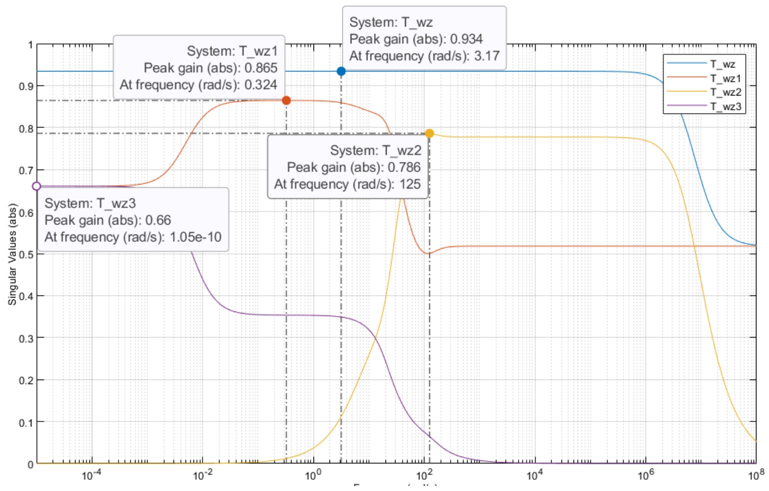


Figure 6.2: Singular value plot of T_{wz} with the individual transfer functions T_{wz_i} ϵ i = 1,2,3

As can be seen from Table 6.1, the model following constraint γ_3 is the 'least' violated as it is the one furthest from 1. This means W_3 can be modified with the objective of tightening the constraint. The filter is manipulated by lowering the gain at the 4 rad/s frequency until one of the individual performance values γ_i becomes higher than one. With this procedure, the model following filter was attenuated from the original $-3.01dB$ to $-16.2dB$. The performance parameter corresponding to the controller input γ_2 breached the threshold of 1, which means that the controller amplifies the input disturbance. The performance values with the modified W_3 are shown in Table 6.2 and the peak values of the singular values can be seen in Figure 6.3. Finally, to recap, the parameters of each weighting filter are given in Table 6.2

Parameter	W_1^{-1}	W_2^{-1}	W_3^{-1}
DC Gain [dB]	-60	100	-60
HF Gain [dB]	5.72	-40	5.72
Gain [dB]	-3.01	-15	-16.2
Freq [rad/s]	4	151	4

Table 6.2: Parameters of the three weighting filters

γ	Value
$\gamma_{T_{wz}}$	1.1212
γ_1	0.8811
γ_2	1.0002
γ_3	0.8735

Table 6.3: Performance values after reducing the magnitude of W_3 to $-16.2dB$

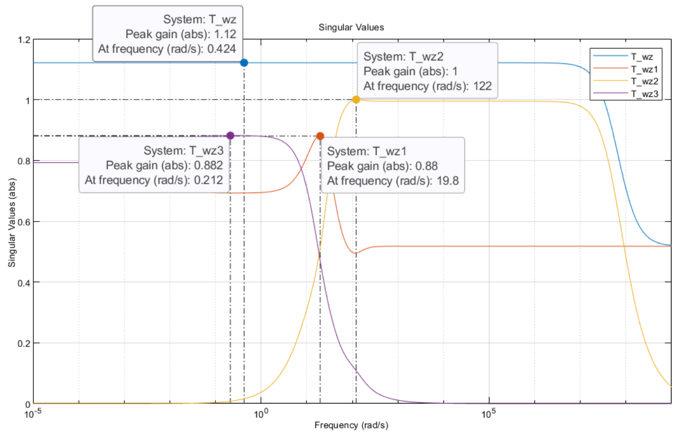


Figure 6.3: Singular value plot of T_{wz} and T_{wz_i} ϵ i = 1,2,3 after reducing the magnitude of W_3

6.2. Controller Order Reduction

The `hinfsys` function, used to design the controller C_e requires the user to express all design requirements in terms of a single weighted transfer function, P in our case. Contrary to other tuning commands such as `hinfstruct`, the synthesis function does not let the user impose any structure on the controller and often results in an high-order controller dynamics [2]. This can be seen by looking at the transfer function of C_e in Equation 6.8 which is of order 9.

$$C0_e(s) = \frac{5.4939 \times 10^5\,(s+2681)\,(s+0.008405)\,(s^2+39.14s+540.3)\,(s^2+28.57s+416.5)\,(s^2+182.7s+1.744\times10^4)}{(s+5.524\times10^7)\,(s+0.009622)\,(s+0.005264)\,(s^2+38.23s+521)\,(s^2+72.61s+1758)\,(s^2+163.9s+1.504\times10^4)} \tag{6.8}$$

From the transfer function, and from its poles and zeros listed in Table 6.4 and Table 6.5 respectively, the following characteristics can be identified:, a HF pole pair at -122.64rad/s, two LF poles at -0.005264rad/s and at -0.00962rad/s, one LF zero at -0.0084rad/s and a HF zero pair at -132.04rad/s. The transfer function can be then minimized by removing these poles and zeros, and by adjusting the gain accordingly to maintain the same magnitude. The result is the minimized transfer function $C_{e_{min}}$ displayed in Equation 6.9. The bode plot in Figure 6.4 further confirms the similarity of the frequency responses.

Also a very HF pole at -5.524×10^7 is present in the model. This pole appears because of the use of the function `hinfsys` with a relative tolerance of 10¯6. The presence of the high frequency pole is also responsible for the appearance of the significant peaks in the phase plots of the model at around 10^6rad/s shown in Figure 6.4 and Figure 6.8. By using a default value of the relative tolerance of 10¯2, the very HF pole moves way closer to the imaginary axis, and the response does not experience such a HF phase shift. It was decided, however, to keep the option set to 10¯6, as this results in more accurate solutions to the Riccati equations used in H_∞ synthesis. Furthermore, it was determined that the phase shift at high frequencies has an unnoticeable impact on the overall performance and robustness of the controller, which provides excellent disturbance rejection capabilities, as it will be shown later in the section.

$$C_{e_{min}} = \frac{6.3685\times10^5\,(s+2681)\,(s+0.008405)\,(s^2+39.14s+540.3)\,(s^2+28.57s+416.5)}{(s+0.005264)\,(s+0.009622)\,(s+5.524\times10^7)\,(s^2+38.23s+521)\,(s^2+72.61s+1758)} \tag{6.9}$$

Natural Frequency [rad/s]	Damping Ratio	Poles
0.005	1.000	-0.005+0i
0.010	1.000	-0.010+0i
22.826	0.837	-19.116+12.474i
22.826	0.837	-19.116-12.474i
41.925	0.866	-36.307+20.963i
41.925	0.866	-36.307-20.963i
122.646	0.668	-81.962+91.231i
122.646	0.668	-81.962-91.231i
55241352.001	1.000	-55241352.001+0i

Table 6.4: Natural Frequency, Damping Ratio, and Poles of C_{0_e}

Natural Frequency [rad/s]	Zeros
2680.953	-2680.953+0i
132.048	-91.366+95.336i
132.048	-91.366-95.336i
20.409	-14.287+14.574i
20.409	-14.287-14.574i
23.245	-19.571+12.543i
23.245	-19.571-12.543i
0.008	-0.008+0i

Table 6.5: Natural Frequency and Zeros of C_{0_e}

The controller can be further reduced by eliminating the pole at -0.005264, since C_i was an integral controller, which can be done by dividing C_e by s. The obtained transfer function $C_{i_{min}}$ is shown in Equation 6.10, where it can be noticed how the controller is of order six. A controller of this order is still too complex to be implemented, therefore the aim will be to reduce the order of the controller while keeping its DC gain intact. This reduction involves retaining only the dominant poles that significantly impact the controller's response. The MATLAB Model Reduction application can help identify these poles by analyzing the Hankel singular values of $C_{i_{min}}$. After looking at the singular values, it was determined that reducing the controller's order to 3 offers the best compromise between having a simple controller and maintaining a step response similar to the original model. This can be noticed by looking at Figure 6.5. The reduced controller transfer function $C_{i_{red}}$ is shown in Equation 6.11 and features three poles. The comparison between the poles of the reduced controller and the original one can be found in the pole-zero map in Figure 6.7, where it can be seen how the poles of the reduced model do not correspond to the original ones. This is not important, as the transfer function has been scaled to keep the same magnitude response. In Figure 6.6 it can be observed how the dynamics of $C_{i_{red}}$ resemble the original response. The reduced controller experiences a difference in phase lead of only 2.6°, when measured at the peak at around 35rad/s.

$$\frac{6.3685\times 10^5(s+2681)(s+0.008405)(s^2+39.14s+540.3)(s^2+28.57s+416.5)}{(s+0.009622)(s+5.524\times 10^7)(s^2+38.23s+521)(s^2+72.61s+1758)} \tag{6.10}$$

$$\frac{6.3685\times 10^5(s+2588)(s+55.48)(s+19.01)}{(s+5.524\times 10^7)(s^2+107.3s+4742)} \tag{6.11}$$

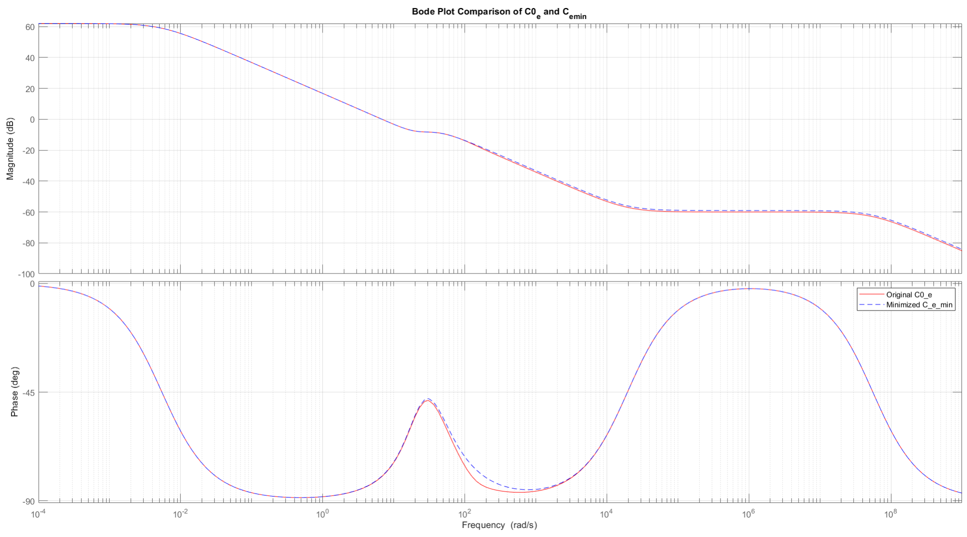


Figure 6.4: Bode plot comparison of H_∞ norm synthesis controller, C_{0_e}, and $C_{0_{min}}$, the minimized controller

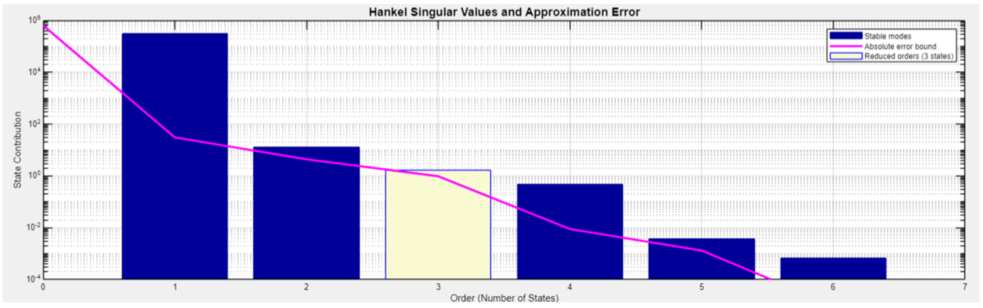


Figure 6.5: Hankel singular value plot of, $C_{i_{min}}$, the integral minimized controller

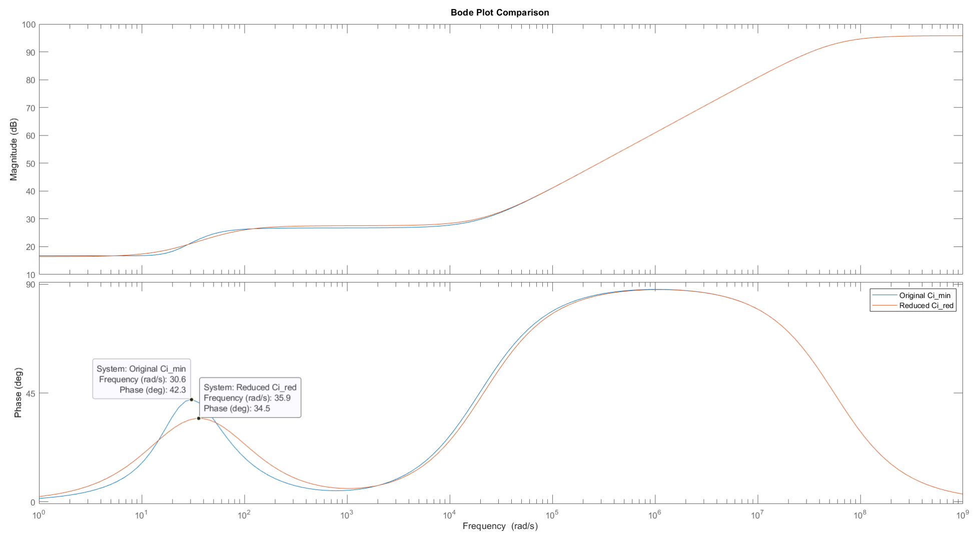
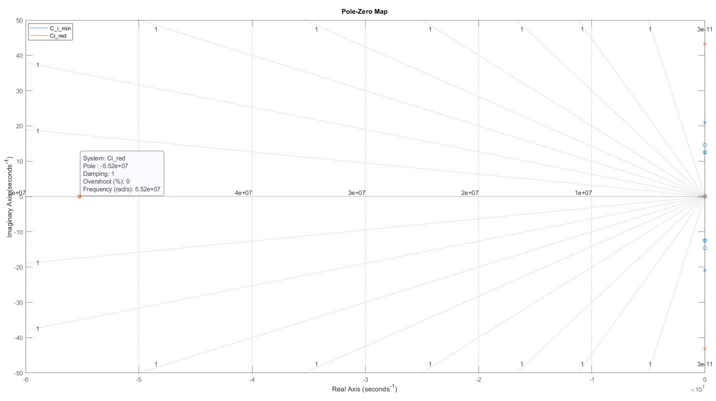
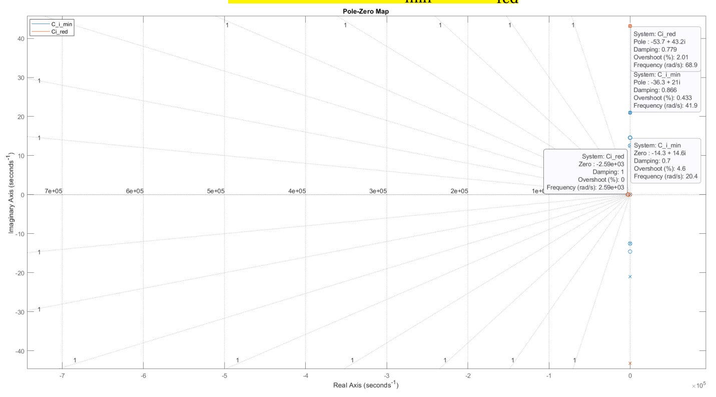


Figure 6.6: Bode plot comparison of integral, $C_{i_{min}}$, and reduced integral, $C_{i_{red}}$, controllers



(a) Pole-Zero Map of $C_{i_{\text{min}}}$ and $C_{i_{\text{red}}}$



(b) Zoomed Pole-Zero Map of $C_{i_{\text{min}}}$ and $C_{i_{\text{red}}}$ zoomed in to show smaller poles

Figure 6.7: Comparison of Pole-Zero Maps for Reduced Integral and Integral Controllers

6.3. Controller Analysis & Simulation

After having obtained the controller, it is needed to analyze the impact on the frequency response of some closed loop transfer functions, and compare these with the inverse of the filters computed in Section 4.2 to check if any violations of the constraints took place. First of all, in order to simulate the performance of the controller, the closed loop model was built in Simulink as shown in Figure 6.8. This model features the constructed filter $C_{i_{red}}$, the Airframe model, a unitary feed forward controller F_f and 6 outputs for analysis.

The outputs are used to evaluate if the constraints defined earlier were respected. In Figure 6.9 six plots show the singular values for six different transfer functions. Focusing on the top row, it can be noticed how the output sensitivity So and the model following error T_m are bounded by the inverse of the filters. For these functions as $\omega \to \infty$, the magnitude of the singular value is less than the one of the filter, which reflects the values of $\gamma_1, \gamma_3 < 1$. For the control input in the top middle plot, the constraint is violated as the magnitude increases beyond W_2, and again this reflects the $\gamma_2 > 1$ found in the previous section. It is of interest to note that at the critical frequency points dictated by the parameters of the weighting filters, unlike for the global performance of the controller, model following error constraint is also violated by the controller as shown in the top right plot. This indicates that the model matching constraint has been tightened too much and should therefore be relaxed a bit more to ensure it is not violated. However, due to the properties of designing a controller with `hinfysn`, it is not possible to decouple the three constraints, meaning that relaxing W3 will also affect the other two.

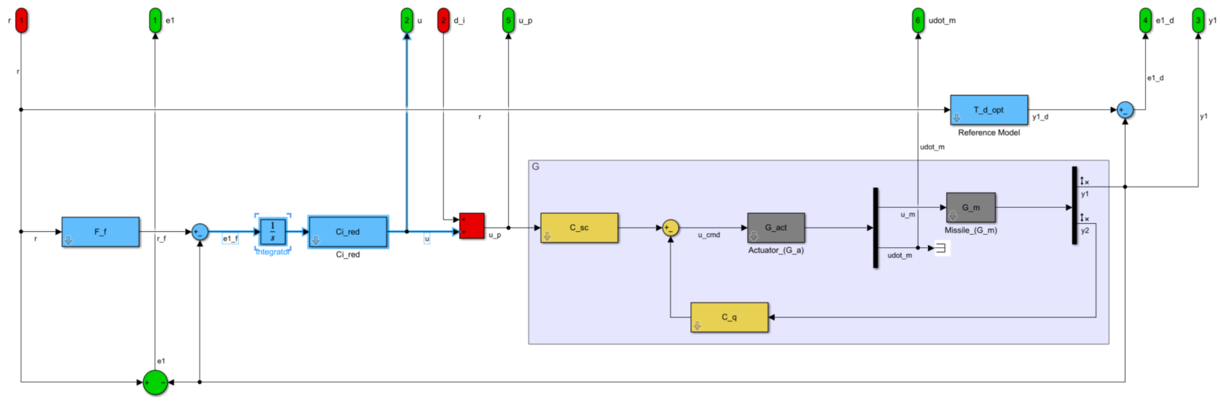
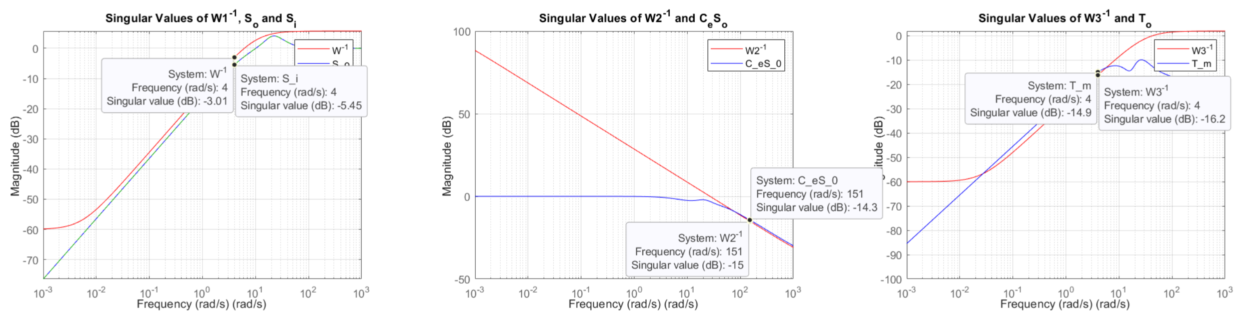


Figure 6.8: Simulink closed loop block diagram showing overall control setup



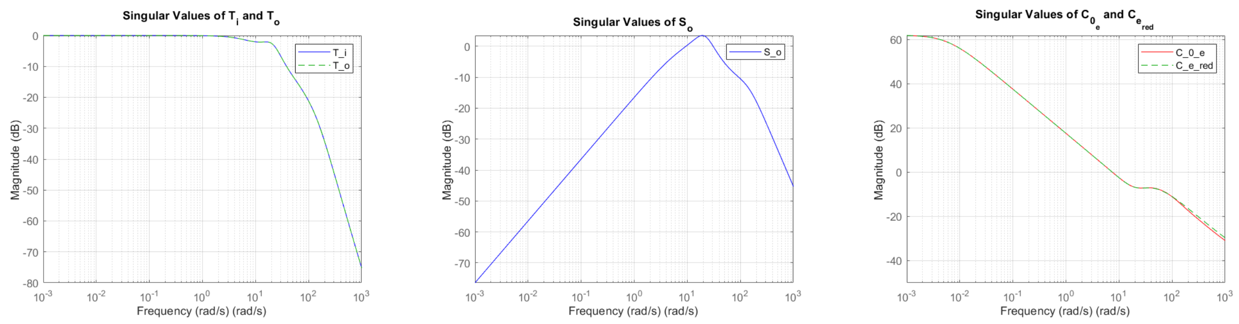


Figure 6.9: Singular value plots of the closed loop transfer functions, weighting filters, $C0_e$ and Ce_{red}

It is fundamental to also look at the open loop response of the system. The previous Simulink model was modified by breaking the loop at the actuator's input as shown in Figure 6.10. The stability margins of the OL system can now be analyzed by looking at the magnitude and phase plots in Figure 6.11. The margins are listed in Table 6.6 for conciseness. The GM of $11.149dB$ indicates that the system can tolerate an increase in gain by a factor of about 3.6, the phase margin indicates a maximum phase lag increase of up to 56.01°which are both satisfactory and robust. The delay margin (DM) is quite small, but it does not pose issues as long as the system operates within the bounds. Also notice how the gain of the system is less than -30dB at 300rad/s.

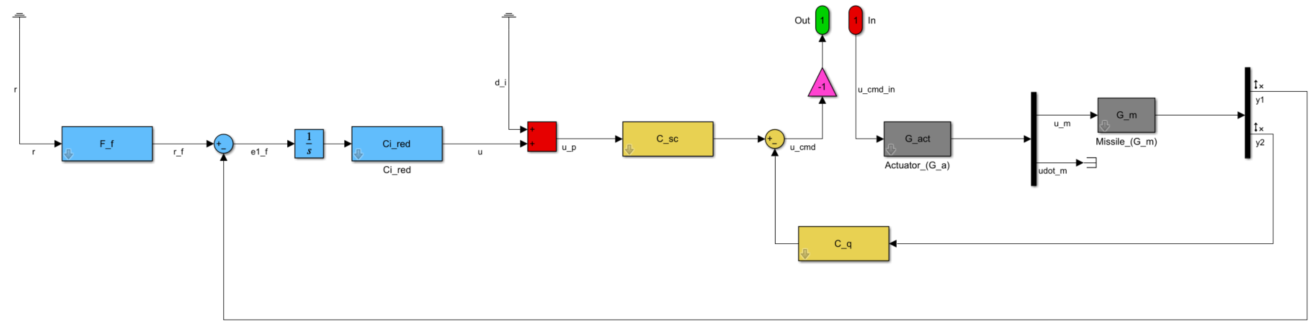


Figure 6.10: Open loop system obtained by breaking the loop at the actuator's input

Metric	**Value**
Gain Margin [dB]	11.149
Phase Margin [deg]	56.017
Delay Margin [s]	0.00746

Table 6.6: Gain margin, phase margin, and delay margin of the Open Loop transfer function

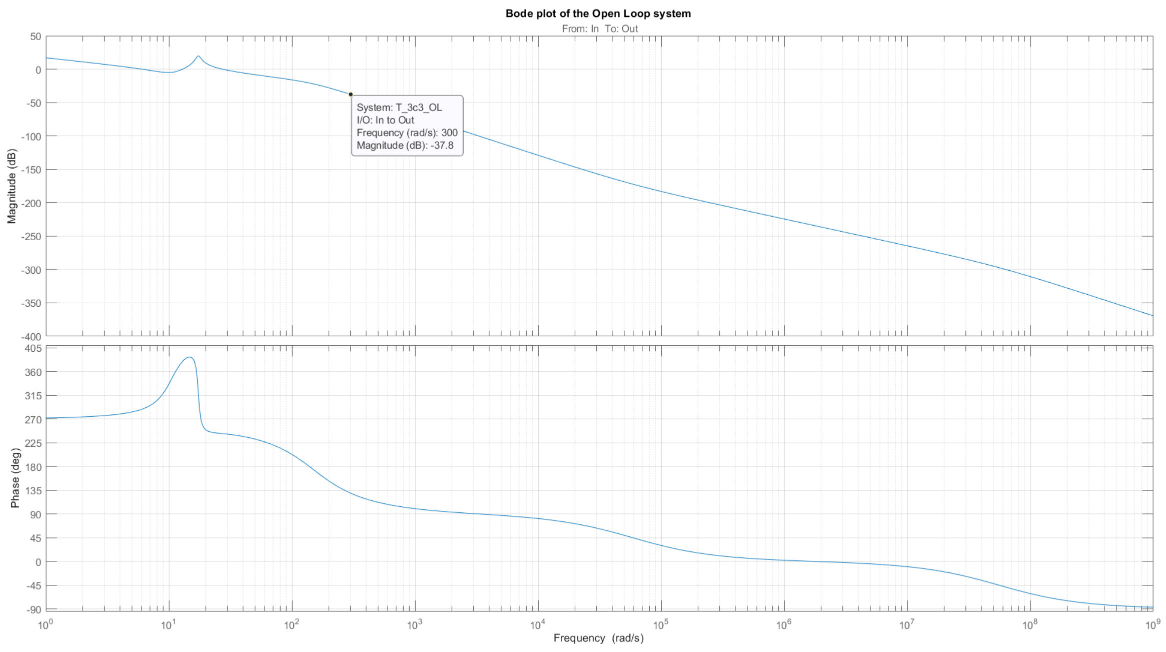


Figure 6.11: Bode plot of the open loop transfer function margin

Finally, it is imperative to evaluate the controller's performance for disturbance rejection, reference tracking and model matching. This is done by simulating the closes loop airframe system's time responses to a step disturbance, as shown in Figure 6.12, where their relevant characteristics are summarised in Table 6.7.

System	**Settling Time [s]**	**Transient Time [s]**	**Peak Response [°]**
S_o	-	0.507	-
T_d	0.182	-	-
T_o	0.514	-	-
S_oG	-	0.607	-
$T_{r_{\dot{u}_m}}$	-	-	22.3

Table 6.7: Settling times, peak response, and transient times for various transfer functions

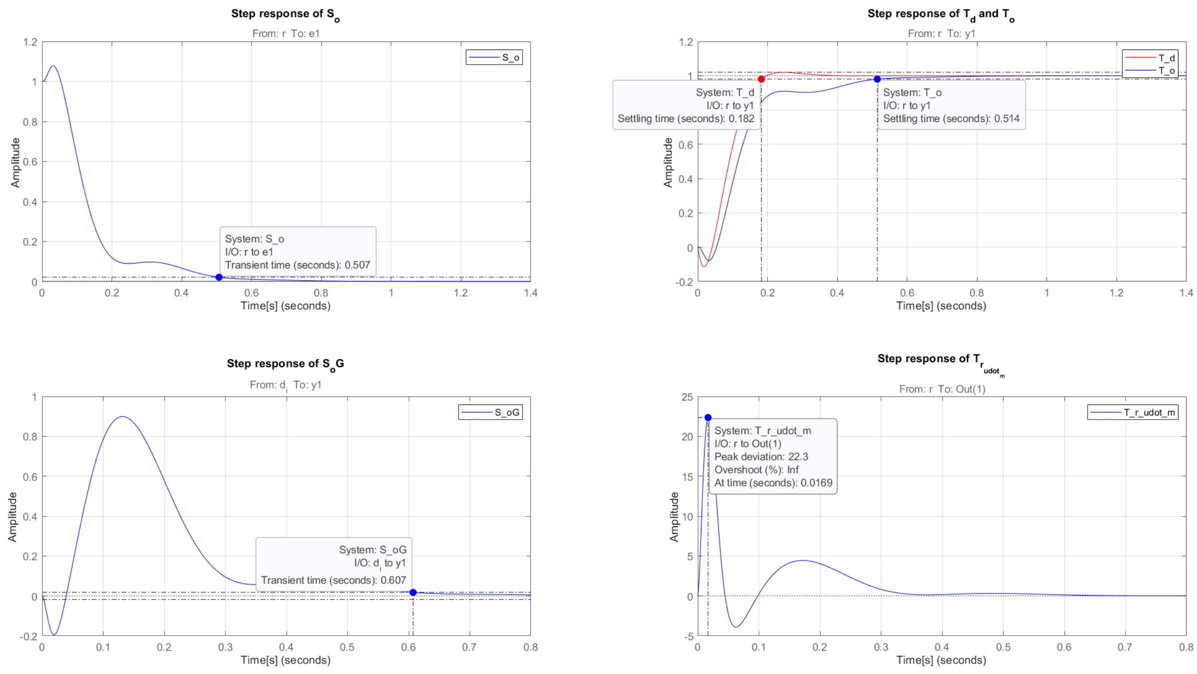


Figure 6.12: Step responses of relevant transfer functions of the feedback control Airframe model for disturbance rejection and model matching

The most important information is shown in the top right plot, where the step of response of the system and reference model are compared. This corresponds to the system's model tracking performance. As can be seen from the figure, although the general shape of the step responses is similar, the settling time of the system is approximately 3 times that of the reference model. This indicates that the performance of the system when it comes to reference model tracking is not optimal and, therefore, further tightening of the model following constraint should be implemented. However, should further tightening for such a constraint occur, it would lead to a negative impact on the other constraints, which is undesired. Another important consideration is the actuator rate limit ($\leq 25°$). This is shown by the bottom right figure of Figure 6.12, which plots actuator rate over time. As can be seen, the peak response of the system is below the actuator limit, which shows good system performance, although it could be further enhanced so as to give a wider margin to the limit. Finally, in terms of the disturbance rejection and reference tracking performance, as can be seen by the top and bottom left plots of Figure 6.12, the controller seems to perform adequately as the amplitude of both the error and the normal acceleration decrease to 0 as $t \to \infty$.

7 Mixed Sensitivity Design (Feedback controller design using hinfstruct)

7.1. Controller Design

7.2. Controller Analysis & Simulation

8
Mixed Sensitivity Design (Feedforward controller design)

8.1. Controller Design

8.2. Controller Order Reduction

8.3. Controller Analysis & Simulation

9 Conclusion

Bibliography

[1] S. Theodoulis, Robust Flight Control: Course Assignment, Delft University of Technology, jun 2024.

[2] MathWorks, "Difference between fixed-structure tuning and traditional h-infinity synthesis," 2024, accessed: 2024-07-05. [Online]. Available: https://nl.mathworks.com/help/robust/gs/difference-between-fixed-structure-tuning-and-traditional-h-infinity-synthesis.html