2024

Robust Flight Control

COURSE ASSIGNMENT

THEODOULIS SPILIOS (ISL-GNC)

TU DELFT | [Company address]

Introduction

The goal of this homework is to understand through a worked example the basic principles of robust control design using the tools of MATLAB/Simulink and relevant toolboxes. This document should be used in parallel with the course slides to prepare the assignment report to be delivered by the students.

The system considered in this homework is a linearized model of a highly agile air-air missile for a given flight condition, similarly to what is described in the course slides. It must be noted here that this type of model is very common in aerospace applications and has the same structure and physics as an aircraft or UAV model. This particular model was chosen because of it being open-source[1]. This model approximates the pitch axis, short period flight dynamics of the missile for a fixed operating point of its flight envelope (i.e. fixed altitude, speed and angle-of attack). The missile model is accompanied by corresponding uncertainty levels as well as by a dynamic model of the control-fin actuator.

Deliverables

The deliverables are both a written report and the various MATLAB/Simulink files that need to be delivered NOT by e-mail to S.Theodoulis@tudelft.nl but through Brightspace. Create a zipped file named by the surnames of the people involved in the project and the year (for example Armstrong_Gagarin_2021.zip). Inside the zipped file you need to put:

I. A **pdf document** named Report_Armstrong_Gagarin_2021.pdf which analytically describes the answers and various plots/figures to each of the questions of the following pages of this assignment written in Word or LaTeX. Create a separate section for each of the 16 questions of this assignment following exactly the same format as here. For example:

Question 1.1: *Flight dynamics (10%)*

Your text & figures

…

Question 3E.3: *Controller analysis & simulation (5%)*

Your text & figures

Keep in mind that you are not only evaluated on the figures but also on the text that you put and your reasoning behind your answers. Make the figures readable, the simulations accurate and your text neither too long nor too short.

II. A **folder** which contains a single Main.m MATLAB file implementing the following standalone (i.e. independently executable) cells corresponding to each part of the current assignment:

- %% Part #1 – System modeling (20%)
- %% Part #2 – Loop shaping (20%)
- %% Part #3a – Weighting filters (5%)
- %% Part #3b – Reference model (5%)
- %% Part #3c – Feedback controller design (*hinfsyn* case) (20%)
- %% Part #3d – Feedback controller design (*hinfstruct* case) (15%)
- %% Part #3e – Feedforward controller design (15%)
- %% Part #4 – Feedback controller redesign *(systune* case) (20%) **OPTIONAL**

[1] Rugh, R. A. (1993). Gain Scheduling for H-infinity Controllers: A Flight Control Example. IEEE Transactions on Control Systems Technology, 1(2), 69-79.

The folder should additionally contain a total of seven Simulink files that are used throughout the assignment which need to be named as follows (the layouts are given in the figures of this document further down):

- Airframe.slx
- ClosedLoop_Cq.slx
- ClosedLoop_CqCsc.slx
- ClosedLoop_CqCscCi.slx
- Design.slx
- OpenLoop_Test.slx
- ClosedLoop_Test.slx

Clean up the folder from any other .asv, .slxc files that are generated when you run your MATLAB code as well as from .mat files that store your results for every cell that you execute.

Prerequisites

In order to fully understand and excel in your work you will need to study the various slides given in the course as well as have a basic understanding of MATLAB/Simulink. Various books, hints and references are given in the present document to greatly facilitate your work. Be aware that at a least a full week of work may be required to cover all the material required for this assignment. Follow this document closely recalling the advice and guidelines given during the course. Given the plethora of tools and concepts you are required to manipulate, this assignment will permit you to master the state of the art relevant to (flight) control design techniques currently used in industry and will accompany you throughout your career.

Enjoy the ride!

1. System modeling (20%)

An overview of the system to control is given in Figure 1 and is the same used throughout the course. The state variables are the angle of attack (AoA) $x_1 = \alpha$ (in rad) which is the angle between the airspeed ($V = \mathcal{M} \cdot a$, in m/s) and the missile axis 1^b, and the pitch rate $x_2 = q$ (in rad/s) which is the second component of the missile angular velocity projected in the missile body coordinate system.

The control input is the fin deflection $u_\mathrm{m} = \delta_q$ (in rad), whereas the controlled output variable is the aerodynamic acceleration along the vertical axis 3^B projected also into the missile coordinate system $y_1 = a_z$ (in g's). The pitch rate $y_2 = x_2 = q$ is considered also as an additional measure used by the control system used to improve performance.

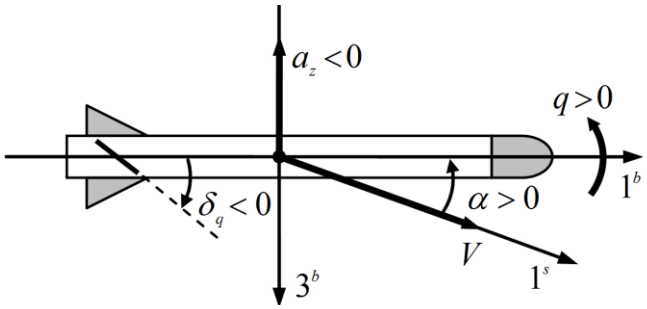

Figure 1 – Missile profile view

The linear model of the missile airframe short period pitch axis is computed at the flight condition: $[\alpha \quad \mathcal{M} \quad h] = [20° \quad 3 \quad 6096\mathrm{m}]$ and corresponds to an increased maneuverability situation. It is given by the 2nd order state space model:

$$\begin{bmatrix}\dot{\alpha}\\ \dot{q}\end{bmatrix} = \begin{bmatrix}-Z_\alpha/V & 1\\ M_\alpha & M_q\end{bmatrix}\begin{bmatrix}\alpha\\ q\end{bmatrix} + \begin{bmatrix}-Z_\delta/V\\ M_\delta\end{bmatrix}\delta_q = \begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}$$
$$\begin{bmatrix}n_z\\ q\end{bmatrix} = \begin{bmatrix}-A_\alpha/g & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\alpha\\ q\end{bmatrix} + \begin{bmatrix}-A_\delta/g\\ 0\end{bmatrix}\delta_q = \begin{bmatrix}y_1\\ y_2\end{bmatrix}$$

The actuator is modeled also by a 2nd order system with input $u_{cmd} = \delta_{q,cmd}$ (in rad) and output (fed to the input of the missile) $u_m = \delta_q$ (in rad). The two states of the actuator are the fin deflection $x_3 = \delta_q$ (in rad), and the fin deflection rate $x_4 = \dot{\delta}_q$ (in rad/s) and are considered to be measurable[2]. The corresponding state space model is then:

$$\begin{bmatrix}\dot{\delta}_q\\ \ddot{\delta}_q\end{bmatrix} = \begin{bmatrix}0 & 1\\ -\omega_a^2 & -2\zeta_a\omega_a\end{bmatrix}\begin{bmatrix}\delta_q\\ \dot{\delta}_q\end{bmatrix} + \begin{bmatrix}0\\ \omega_a^2\end{bmatrix}\delta_{q,cmd}$$
$$\begin{bmatrix}\delta_q\\ \dot{\delta}_q\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\delta_q\\ \dot{\delta}_q\end{bmatrix} + \begin{bmatrix}0\\ 0\end{bmatrix}\delta_{q,cmd} = \begin{bmatrix}x_3\\ x_4\end{bmatrix}$$

where ω_a, ζ_a are respectively the natural frequency and damping ratio of the actuator. Values for the various variables and uncertainty levels are given in Table 1 below. For simplicity, the sensors that would be used to measure the normal acceleration, pitch rate and actuator position are considered as ideal and do not intervene in the assignment.

[2] The first $x_3 = \delta_q = u_\mathrm{m}$ is fed to the missile input whereas the second $x_4 = \dot{\delta}_q$ is only used for plotting.

Variable	Value	Unit	Description
g	9.80665	m/s^2	standard gravity acceleration
a	316.0561	m/s	speed of sound at $6096m$
$\mathcal{M}$	3	-	Mach number
Z_α	1236.8918	m/s^2	normal force derivative
M_α	-300.4211	$1/s^2$	pitch moment derivative
M_q	$\simeq 0$	$1/s$	damping derivative
Z_δ	108.1144	m/s^2	control force derivative
M_δ	-131.3944	$1/s^2$	control moment derivative
A_α	1434.7783	$m/s^2/rad$	normal acceleration derivative
A_δ	115.0529	$m/s^2/rad$	control acceleration derivative
ω_a	150	rad/s	actuator natural frequency
ζ_a	0.7	-	actuator damping ratio

Table 1 - Data values

Figure 2 shows the Simulink diagram of the airframe comprising the actuator and missile blocks as they should be connected in subsequent questions. Notice the single input to the actuator u_{cmd} as well as the two outputs y_1, y_2 corresponding to the controlled variable (normal acceleration) and additional measure (pitch rate) respectively.

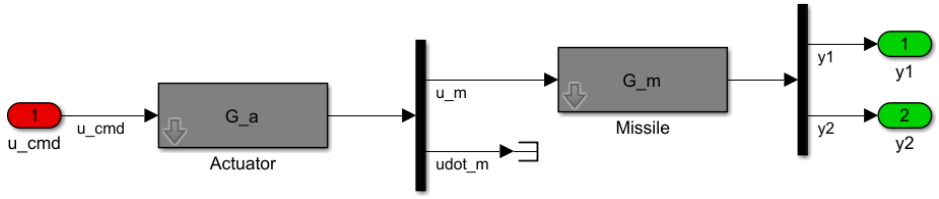

Figure 2 - Airframe Simulink model (Airframe.slx)

Question 1.1: *Flight dynamics (10%)*

The nonlinear missile model was extensively discussed in the lectures. Here you are required to discuss on the process of how to obtain the nominal (no uncertainty) linear missile pitch dynamics state space model given in Page 4 of the current document.

You may explain the equilibrium point computation and linearization procedures as well as the various software tools in MATLAB/Simulink and hypotheses made. You may give insight and discuss on the domain of potential validity of the model related also to its flight envelope as explained in the lecture and in the lecture slides.

What is expected is for example a one page discussion, carefully written so that your reasoning and understanding capacities may be assessed. No extensive mathematical treatment is required and no MATLAB code needed.

Hints: additional sources include Zipfel's book for flight dynamics (Modeling and Simulation of Aerospace Vehicle Dynamics) as well as the user guide of the Simulink Control Design blockset for tools concerning trimming and linearization.

Question 1.2: *Model construction & analysis (10%)*

The first part of the question concerns the definition in MATLAB of the state space models of the missile and actuator as given in Page 3, using the data from Table 1. Create the missile and actuator models G_m, G_a and rename the default input, state and output names to match the variables given in Page 3 (you may use u_m, (x1, x2), (y1, y2) for the missile and u_cmd, (x3, x4), (u_m, udot_m) for the actuator). Use the .InputName, .StateName, .OutputName properties of the systems to achieve this. Save the state space systems for later use using the save command (e.g. save G_a G_a). Put a Simulink file named Airframe.slx as in Figure 2 (pay attention on the numbering of the inputs and outputs!) and use function linearize to obtain the state space model of the airframe G_am (actuator plus missile). You should get a state space model with one input, four states and two outputs. In the report write down the numerical values of the corresponding state space matrices A, B, C, D (use print screen) and the zpk form of each transfer function from u_cmd to y1, y2 respectively (e.g. G_am_nz, G_am_q). Pay attention to the ordering of the states (i.e. x1, x2, x3, x4); use the function ss2ss to rearrange the state vector if necessary.

The second part of the question concerns the analysis of the open loop airframe model G_am in terms of poles and zeros. Plot the input/output poles and zeros of the system using the iopzmap function of MATLAB and click on the various poles and zeros to visualize the annotation rectangles with the information for each one (totally 5 rectangles: 3 for the zeros of the acceleration/pitch rate channel, plus 2 for the low frequency (missile) and high frequency (actuator) poles). Comment on the stability of the system, on the frequency separation between the missile and actuator poles as well as on the damping of both (this info is available directly from the annotation rectangles). Finally comment on the zeros of the acceleration channel (minimum or non-minimum phase). Verify all this using the damp and zero functions but be careful to select only the individual transfer functions of each of the two channels each time (e.g. G_am(1,1)). Convince yourself that your data is right by verifying also the step response of the actuator and of the missile (no need to put them in the report).

Hints: You can use cell mode (Preferences -> Editor/Debugger -> Code Folding, tick all boxes) in MATLAB for your programming, with individual code cells for each part of the assignment that saves and loads all necessary data (use %% to start a new cell and Ctrl+Enter to run it) only for the current assignment part.

2. Loop shaping (20%)

Starting from the airframe model G_am and the Simulink file Airframe.slx built in the previous question, the goal of this second part, is to design two controller gains C_q and C_sc as illustrated in Figure 3. The resulting closed loop system G with input u_p and output y1 is used in later parts for design.

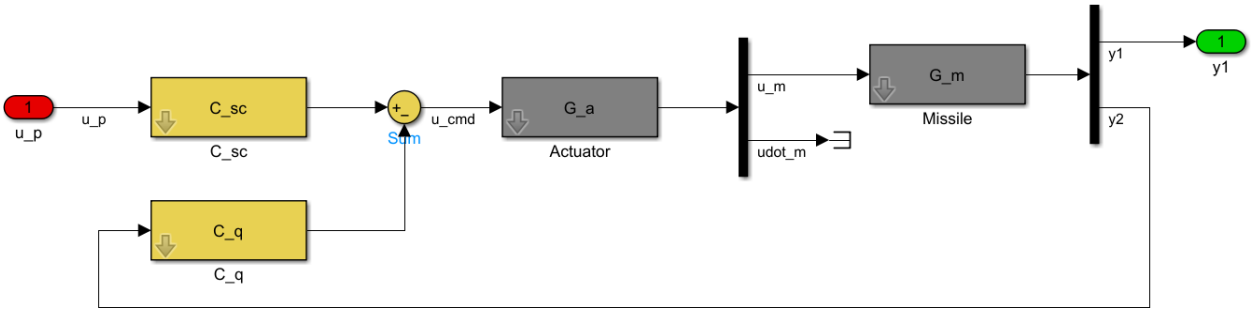

Figure 3 - Inner loop controllers (ClosedLoop_CqCsc.slx)

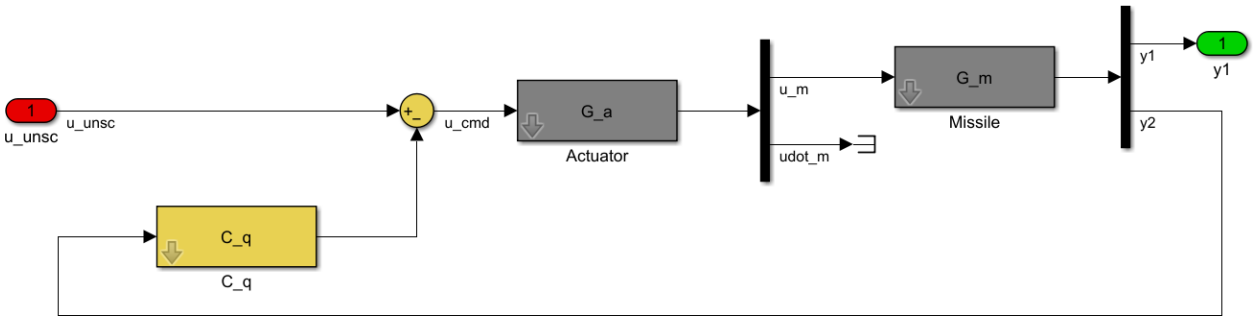

Figure 4 - Inner loop damping controller (ClosedLoop_Cq.slx)

Question 2.1: *Damping gain design (5%)*

The damping gain C_q is first designed in this question using simple root locus techniques. This gain is applied on the pitch rate $y_2 = q$ as illustrated in Figure 3 using negative feedback in order to increase the damping of the airframe dominant (i.e. missile poles). In order to do this you will need the second component of the airframe model G_ol_q = G_am(2,1) and the rlocus (or rlocusplot) function of MATLAB. You can select the appropriate gain by zooming on the dominant pole locus and clicking on the plot in order to obtain a closed loop damping (for example approximately 0.7 as a typical value). What is then the computed control gain C_q corresponding to Figure 3?

Construct a Simulink file named ClosedLoop_Cq.slx as in Figure 4 below and implement your control gain C_q you computed previously. Use the function linearize in order to compute the transfer function G_cl_q_unsc from the input u_unsc to the output y1. Write the output of the transfer function zpk(G_cl_q_unsc) in your report as well as the open loop one G_am(1,1). Comment on the poles and zeros of each one, what does C_q achieve?

Hints: The rlocus function needs to be called with a negative argument (i.e. –G_am(2,1)) in order to yield a stabilizing gain. Be careful since the rlocus command will yield a positive gain! The final C_q is negative. You may also need to increase the resolution of the rlocus (or rlocusplot) commands to compute C_q with a higher precision. Alternatively you can use sisotool which has very similar functionality.

Question 2.2: *Scaling gain design (5%)*

The scaling gain C_sc now is used to make the inner loop transfer function dcgain of Figure 3 equal to 1. Use function dcgain to compute the gain of G_cl_q_unsc = G and invert it to compute C_sc. What is its value? Construct a Simulink file named ClosedLoop_CqCsc.slx corresponding to Figure 3 and implement your gains C_q and C_sc computed previously. Use again the function linearize to obtain the total pre-compensated inner loop transfer function G and write down the output of zpk(G). Plot the step response of G and verify that it converges to 1. Right click on the figure (Characteristics -> Settling Time) and compute the 5% settling time. Motivating the need for an external loop acceleration loop applied on y1, try and insert a step disturbance on y2, is this disturbance rejected on y1? Why not? What would it require for the control law to permit disturbance rejection?

Hints: To do the simulation you can modify the same Simulink file using a constant $u_p = 0$, and an additive step disturbance on y_2, say at $t = 1s$. Use then the Simulink data inspector to plot u_p, y_1 superimposed. Be careful to adjust the simulation tolerance at least to 1e-6 with a variable step solver for example.

Question 2.3: *Integral gain design (10%)*

In order to be able to reject disturbances on the plant output and also to ensure proper reference tracking of the output y_1, an integrator as well as an integral gain C_i is added in the forward path of the system as it is illustrated in Figure 5. Now, if the acceleration $y_1 = n_z$ is fed back and subtracted from a reference signal r_f (see also Figure 6 in the next page), the overall acceleration (integral) controller:

$$C_{i,sc} = \frac{C_{sc}C_i}{s}$$

will be able to provide the desirable characteristics. The goal of this section is to design a first value for the control gain C_i, even though in the subsequent sections this controller may be allowed to be of more complicated structure and hence yield more performance.

Construct a Simulink file named ClosedLoop_CqCscCi.slx as in Figure 5 below and implement your control gains C_q, C_sc you computed previously and select a unitary value for the moment for C_i. Use the function linearize in order to compute the open loop transfer function G_ol_nz from the tracking error input e1_f to the output y1. Write down the output of the transfer function zpk(G_ol_nz) in your report.

In order to compute the integral gain C_i, the easiest way is to adjust it using standard open loop shaping techniques. Increasing or decreasing the loop gain via C_i will increase or decrease the gain crossover frequency ω_{gc} (the frequency for which the gain crosses the $0dB$ line and hence for which the phase margin is evaluated) and hence the bandwidth of the system.

Use MATLAB's utility sisotool with G_ol_nz as argument in order to adjust C_i either to guarantee a 60° phase margin or otherwise a 5% settling time which is the smallest possible. Give in your report the value of the gain for the two tunings as well as the 2x2 plot of sisotool which illustrates the loop gain on the left and the root locus and step response on the right. Write down also the output of the y_1 closed loop zpk(T) for the second case. Make sure you use right-click options on all four plots to show the grid, the stability margins, the 5% settling time etc. Finally, comment on the results you obtained: is this loop at-a-time tuning (C_q, C_sc, C_i) optimal in some way or not, is there any other feedback type we may use, etc. You may need to complete this part after having done the next sections.

*Hints: In order to adjust the gain of the controller C_i, you may either drag the gain curve directly up and down with your mouse on the sisotool Bode plot, or edit directly the value of the controller C on the Controllers and Fixed Blocks interface on the far left of the interface. Be careful that to compute your acceleration closed loop T, you need to close the loop of Figure 5 as in Figure 6 (no feedforward or reference model)! You can use the function feedback(G*C_i/s,1,-1) directly in MATLAB for simplicity.*

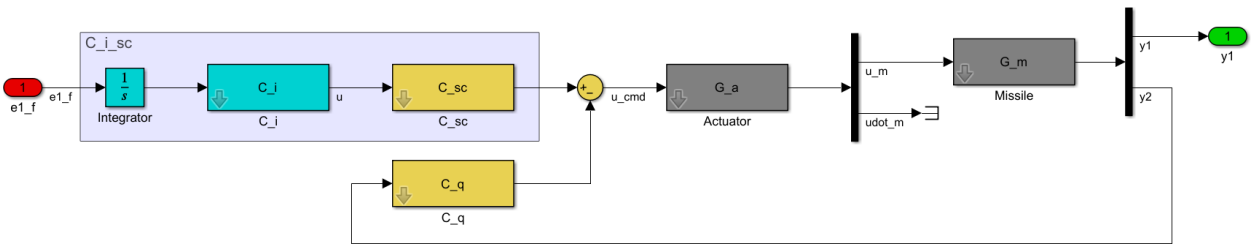

Figure 5 - Outer loop tracking controller (ClosedLoop_CqCscCi.slx)

3. Mixed sensitivity design (60%)

The goal of this third part is to design two controllers for the single-input, single-output (SISO) pre-compensated system G of the previous second part. The first controller is a feedback disturbance rejection controller C_e applied on the filtered acceleration tracking error $e_{1,f} = r_f - y_1$ whereas the second controller F_f is a feedforward, model following controller applied on the reference acceleration $r = y_{1,cmd}$. Note that the C_e needs to have integral action, hence it is made up from an integrator in series with a phase lead controller C_i.

Take a look at Figure 6 below: you will recognize the actuator and missile models G_a, G_m (in grey), the already designed inner loop gains C_sc, C_q (in yellow) as well as the controllers C_i and F_f (in cyan) to be designed in this section. There are also two inputs (**pay attention at the numbering of the input/output ports!**): reference signal r and input disturbance d_i (in red) and six outputs: tracking error $e_1 = r - y_1$, feedback controller output u, controlled output y_1, model following error $e_{1,d} = y_{1,d} - y_1$, disturbed controller output u_p and actuator output rate $\dot{u}_m$ (in green). The reference model T_d (in blue) will be discussed in later sections. Finally, the tiny input and output arrows on the above signals will be discussed in Part 4 of this assignment since they will be used in order to design the controllers in a signal-based approach using systune.

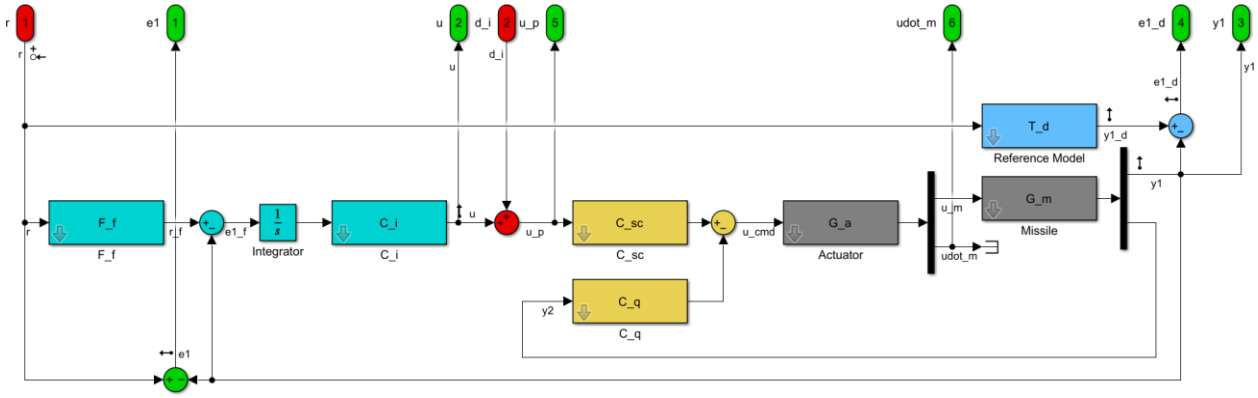

Figure 6 - Overall control setup (ClosedLoop_Test.slx)

A. Weighting filters (5%)

The first, and probably most important step, towards designing robust control systems, is the adjusting of representative templates or weighting filters, as a function of the design requirements (i.e. bandwidth, overshoot, roll-off rate, etc.) for a given system; in our case the missile airframe.

As it will be shown in the following parts of the assignment, the two most important filters are the ones applied on the output sensitivity function (named for convenience here W_1), on the control signal (W_2) and on the model following error (W_3). Additional filters may be used if needed in the input disturbance d_i (or if present) in the output disturbance output d_o.

Question 3A.1: *Weighting filter discussion (2.5%)*

What is required of you first is a brief discussion on the general characteristics of the filters W_1, W_2 (skip W_3 for the moment) mentioned above focusing on their role to shape the various loop transfer functions as discussed in the course slides. Which ones need to be low-pass or high-pass and what are their asymptotic properties in low or how frequencies? What are the tools to design them?

Hints: You may refer to the MATLAB Robust Control Toolbox documentation and especially to the functions makeweight, and to Multivariable Feedback Control, S. Skogestad, I. Postlethwaite, Ch. 2-3 and Essentials of Robust Control, K. Zhou, J.C. Doyle, Ch. 6. You may describe the characteristics of the filter inverses W_1^{-1}, W_2^{-1} since they are linked directly to the desired S, KS.

Question 3A.2: *Weighting filter computation (2.5%)*

A first crude approximation of the weighting filters W_1, W_2 is required for this question. Make a new MATLAB cell (e.g. Part #3a – Weighting filters) in order to compute and save these filters for use during the control design phases later in this assignment. As you saw in the makeweight function, in order to construct these filters, you will need 4 parameters: their low frequency (LF) and high frequency (HF) gains **dcgain**, **hfgain** respectively as well as their desired gain **mag** at a given frequency **freq**. Concerning the first two parameters, for the weighting filters they need to be low (respectively high) for W_1 and the opposite for W_2. As far as the last two parameters are concerned, we require that the $-3.01dB$ bandwidth of the sensitivity function to be $4\,rad/s$, whereas for the control signal we wish at least $-15dB$ attenuation at the actuator $-3.01dB$ bandwidth frequency. For the peak value of the desired sensitivity function $M_S = M_1$ (at HF), you can use a minimum $PM \geq 30°$ via the approximate formula:

$$PM \geq 2\arcsin\left(\frac{1}{2M_S}\right) \text{ (in rad)}$$

Using for both filters W_1, W_2 a continuous time filter of order 1 of the form:

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

recompute the parameters A_i, M_i, ω_i for each filter, based on the dcgain, hfgain, mag, freq you used to obtain the filters with the makeweight function previously. Plot the filters **inverse** gain wrt frequency (use function sigma in red/blue for the first/second filter putting annotations using the mouse at critical frequencies) in one single plot. What do the parameters M_i, ω_i, A_i represent in the frequency plots?

Remark: Note that the parameters of the filters will be further adjusted later when designing the controllers in order to yield satisfactory performance. For the moment we focus only on the general properties of the filters. Note also that you may choose $M_1 = A_2, A_1 = M_2$ for simplicity!

B. Reference model (5%)

As discussed previously, the goal of the closed loop controllers to be designed in the next parts, is to oblige the system output y_1 to behave as the output y_d of the reference model T_d. This was illustrated in Figure 6 by creating a model following error signal $e_{1,d} = y_{1,d} - y_1 = T_d r - y_1$.

Question 3B.1: *Reference model computation (5%)*

The form of the transfer function for the reference model is selected as a 2nd order system with natural frequency ω_d and damping ratio ζ_d, augmented by the open loop *acceleration-channel non-minimum phase zero* of the missile airframe z_m:

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

You are requested to adjust the variables ω_d, ζ_d of the reference model in order to obtain a 5% step response settling time for the filter $t_{s,d} = 0.18s$, with an overshoot $M_d = 5\%$. Plot the step response of the filter, making sure to show the settling time and peak time by right clicking on the figure and using the characteristics tab. Write down the output of zpk(T_d) and save your reference model. Again, make a separate cell for this computation (i.e. %% Part #3b – Reference model).

Hints: To compute ω_d, ζ_d, you may use either the brute force method with one for loop per variable to compute, or a more sophisticated optimization algorithm such as fmincon in MATLAB. Otherwise, it is well known from robust control theory, that the maximum achievable bandwidth is approximately equal to half of that of the NMP zero of the open loop acceleration channel.

C. Feedback controller design (hinfsyn) (20%)

In this part you are required to design the disturbance rejection/tracking controller C_i of Figure 6, assuming that the feedforward controller F_f is unitary and the gains C_q, C_sc are given by the results of Part 2. The general idea is to start designing a full order dynamic controller and then try to simplify it using model order reduction techniques in order to investigate potential loss of performance.

Question 3C.1: *Controller design (10%)*

The first step in order to design a $\mathcal{H}_\infty$ controller is to construct the Simulink design file (named Design.slx) that will be used for the controller design. The approach that we will be using is a mixed sensitivity S/KS constraint formalism, plus a third model following constraint. Take a look at Figure 7: apart from the inner loop controllers (yellow), and airframe (grey) blocks constituting the SISO G system as explained above, there are three green blocks modeling the sensitivity, control sensitivity and model followings constraints respectively (W_1, W_2, W_3). The first is applied on the tracking error $e_1 = r - y_1$, the second on the outer loop control signal u whereas the third on the model following error signal $e_{1,d} = y_{1,d} - y_1 = T_d r - y_1$.

Globally there is one disturbance signal coinciding with the reference input $w = r$ (red), one controller input (measurement) signal $v = e_1$ (dark green), one controller output signal (orange), and three performance signals as described before z_1, z_2, z_3 (green). The controller to be designed $C_e(s) = C_i(s)/s$ (see also Figure 6) is to be placed between v and u in order to close the loop.

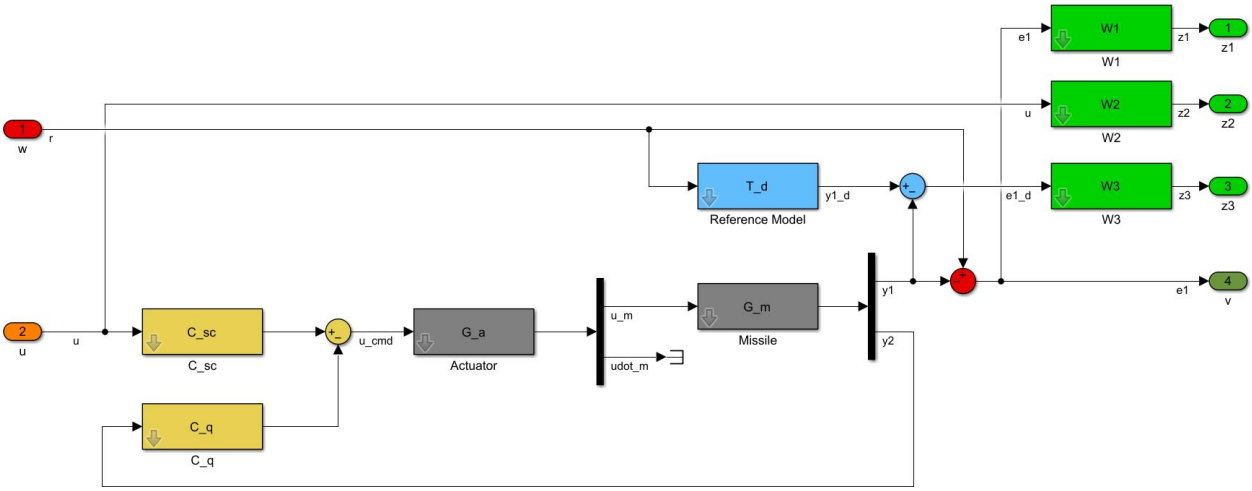

Figure 7 - Controller design model (Design.slx)

As a first exercise, prove that the **weighted closed loop** matrix transfer function $T_{wz}(s)$ is given by the following formula:

$$T_{wz} = \begin{bmatrix} W_1 S_o \\ W_2 C_e S_o \\ W_3 (T_d - T_0) \end{bmatrix}$$

where $\boldsymbol{z} = [z_1 \quad z_2 \quad z_3]^T$ is the weighted performance vector, $S_o = (1 + GC_e)^{-1}$ is the output sensitivity function and $T_o = 1 - S_o = GC_eS_o$ the output cosensitivity function.

Hints: Just use Figure 7, making the abstraction that the yellow and grey blocks form a SISO plant G and that the controller C_e is connected in feedback between the signals v and u.

As a second exercise, prove also mathematically as before that the **weighted open loop** matrix transfer function $P(s)$ is given by:

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

and then use the function linearize in Figure 7 to numerically compute this 4×2 system $P(s)$. Show in your report the output of zpk(P) and verify that the individual transfer functions of the above equation coincide with this result.

As a third exercise, design the controller C_e (and hence C_i) using full order $\mathcal{H}_\infty$ synthesis and MATLAB's hinfsyn function. This function needs P (computed previously), the number of measurements & controls (both equal to 1, see Figure 7) and some optimization options (increase the relative tolerance to 1e-6). Start the tuning using the weighting filters you computed in Question 3A.2 by also setting $W_3 = W_1$, which is a quite relaxed model following constraint. The output of hinfsyn will give you the controller C_e, the weighted closed loop T_{wz} (its size being 3×1, see above), and the global performance level γ. Write in your report the value of γ and plot the singular values $\sigma(T_{wz})$ (use amplitude not dB!) and also these of the transfer functions $T_{wz_i}(s), i = 1,2,3$ on the same plot and show their maximums. Verify that $\|T_{wz}(s)\|_\infty \leq \gamma$ according to theory (maximum of the singular value plot) using norm(T_wz,'inf'). Give also the values of the individual γ_i's (for each $T_{wz_i}(s)$). Which one is the smallest (less violated constraint)?

As a fourth exercise, optimize the constraints! As you saw previously, the model matching constraint is the less violated and hence can be tightened. Go back to your weighting filters, keep your initial W_1, W_2 computed in Question 3A.2, and then gradually tighten the model following constraint by reducing only the mag parameter of the W_3 filter from $-3.01dB$ to the value for which either of the three gamma_i goes beyond one. You should be able to get well more than $-15dB$ model following attenuation at $\omega_{S_o} = 4rad/s$ without violating the control sensitivity constraint. Give in a table the final values of the three filters you used as well as the resulting individual and global performance levels γ_i, γ. Plot again the singular value plots exactly as above.

Hints: It is recommended to refer to the documentation of the hinfsyn and hinfsynOptions functions of MATLAB in order to grasp their functioning.

Question 3C.2: *Controller order reduction (5%)*

The goal of this question is to form the controller C_i to be implemented in your Simulink file ClosedLoop_Test.slx of Figure 6 from the computed controller C_e of the previous question. In order to do this, we will proceed in two stages: first controller simplification in order to take out unwanted dynamics, and second model-order reduction in order to further reduce its complexity.

The initial full order, not simplified yet, controller (let us call it C0_e) you designed in Question 3C.1 should be of order equal to 9 (why?), and is unnecessarily complicated to be implemented. As a first question write down in your report the output of zpk(C0_e) and of pole(C0_e), zero(C0_e). You should be then able to identify two very LF poles, one very LF zero and a pair of very HF pole and zero. Identify in your report which ones they are. Use then zpkdata(C0_e,'v') to obtain its zpk description and reform C_e in a minimal form C_e_min selecting only the appropriate zeros and poles (totally 6 zeros and 7 poles). Give the value of zpk(C_e_min) in your report.

Hints: You will need to adjust the gain of C_e_min due to the cancelling of the HF pole/zero pair. Remember that a HF pole $1/(s+\omega_{HF})$ *contributes with a dcgain of* $1/\omega_{HF}$ *(the inverse holds for a HF zero), hence you must adjust the gain accordingly. Verify from the Bode plots the gain and the phase of the controllers.*

The second question concerns model order reduction: first obtain from C_e_min you computed above the integral controller C_i_min by eliminating the remaining LF pole of C_e_min. Recall that:

$$C_e = C_i/s$$

Write down in your report the output of zpk(C_i_min). How many poles and zeros remain? Use model order reduction (for example via the MATLAB function balred) to obtain a reduced order controller C_i_red keeping only the dominant poles of C_i_min and preserving its dcgain. Write down on your report the output of zpk(C_i_red). Give also the Bode plots of C0_e, C_e_min (they should be almost identical) and of C_i_min, C_i_red. How close are the latter in terms of gain and phase and how much is the phase advance provided by the controller? Give also the pole-zero maps of C_i_min and of C_i_red. Use annotations in your plots to highlight information and to distinguish between the various controllers.

Hints: It is recommended to refer to the documentation of the minreal and balred functions of MATLAB in order to grasp their functioning. You can also use the model reducer app and the balanced truncation functionality. Can the former be used for the zero-pole cancellations?

Question 3C.3: *Controller analysis & simulation (5%)*

After having computed the integral controller C_i (C_i_red in this case), we will need to analyze more deeply its impact on the frequency response of some common loop transfer functions. Additionally we will need to compare these loop transfers with the filter inverses you used for the controller tuning in order to evaluate any violations.

Refer to Figure 6 (make sure you constructed correctly the Simulink file ClosedLoop_Test.slx) which can be used to compute these transfer functions as well as simulate your controller in Simulink directly. Use the function linearize again to obtain the closed loop matrix transfer matrix $T(s)$ from the two inputs (in red) to the six outputs (in green) using the feedback controller $C_i = C_{i,red}$ you computed previously. For the moment consider only a unitary feedforward controller $F_f = 1$ in order to evaluate only the feedback loop properties.

From this 6×2 matrix transfer function $T(s)$, isolate only the following transfers in order to plot them further down:

$$
\begin{aligned}
T_{r\to e_1} &= T(1,1) = S_o \\
T_{r\to u} &= T(2,1) = C_e S_o \\
T_{r\to y_1} &= T(3,1) = T_o \\
T_{r\to e_{1,d}} &= T(4,1) = T_m \\
T_{r\to \dot{u}_m} &= T(6,1) \\
T_{d_i\to u} &= T(2,2) = -T_i \\
T_{d_i\to y_1} &= T(3,2) = S_o G \\
T_{d_i\to u_p} &= T(5,2) = S_i
\end{aligned}
$$

As a first exercise, plot in a 2×3 figure the singular values (use function sigma) of these transfers (use red for the weighting filters, blue for the transfers) as in the table below and add annotations for the first three by clicking on the curves at the various critical frequency points dictated from the parameters of the weighting filters (low, middle and high frequencies). Comment about the various frequency plots and whether the controller computed violates or not the weighting filters. Given that your tuning in **Question 3C.1** was done so that the individual performance levels of $S_o, C_e S_o, T_m$ are smaller than unity, you should be able to observe how the filter inverses limit the loop transfer functions.

W_1^{-1}, S_o, S_i	$W_2^{-1}, C_e S_o$	W_3^{-1}, T_m
T_i, T_o	$S_o G$	$C_e, C_{e,red}$

As a second exercise, plot the gain and phase of the open loop by breaking the loop **at the input of the actuator** as shown in Figure 8 (make an OpenLoop_Test.slx file). To do this, again use the function linearize to obtain the open loop transfer function, and the MATLAB function margin to compute the gain and phase margins. Give their values at your report, additionally compute the delay margin using the following formula (convert from seconds to milliseconds):

$$\mathrm{DM} = \mathrm{PM}/\omega_{gc}$$

where PM is your phase margin (in rad!) and ω_{gc} (in rad/s) the gain crossover frequency (the frequency at which your gain crosses $0dB$). Finally make an annotation at your margin figure with your mouse at $300rad/s$ to show the gain of the system at this frequency. Is it less than $-30dB$?

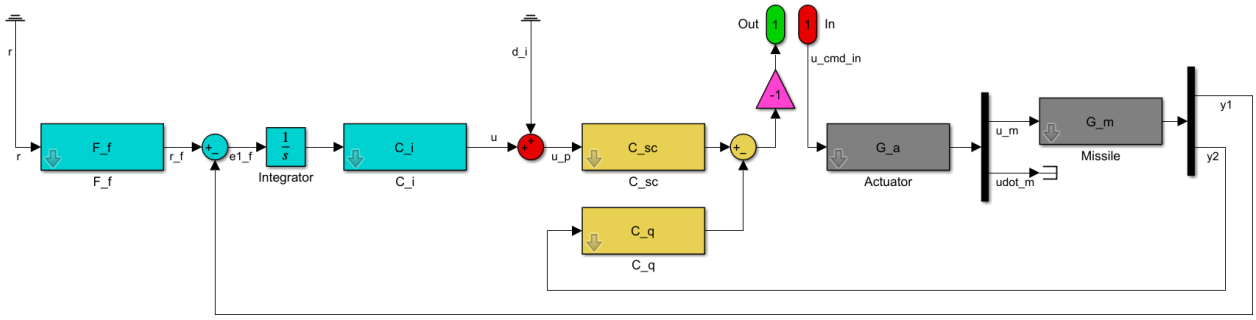

Figure 8 - Open loop analysis diagram (OpenLoop_Test.slx)

As a third exercise, plot the step responses of the following transfers (blue for the curves, red for the reference model):

S_o	T_d, T_o
$S_o G$	$T_{r\to \dot{u}_m}$

and create annotations on the step responses using your mouse in order to illustrate the transient time for $S_o, S_o G$, settling time for T_d, T_o and peak response for $T_{r\to \dot{u}_m}$ (convert to deg/s from rad/s for this one, maximum value should be $\leq 25^\circ/sec$). Make a table with the values that you obtain and comment especially on the settling time you obtained with respect to the one of the reference model. Is the model matching constraint, impacting on the closeness of T_o to T_d, tight enough? What happens if you try to reduce the gain of W_3 even more?

D. Feedback controller design (hinfstruct) (15%)

In this part you are required to design <u>again</u> the disturbance rejection/tracking controller C_i of Figure 6, assuming that the feedforward controller F_f is unitary and the gains C_q, C_sc are given by the results of Part 2 as before. However, the idea this time is not to design a full order dynamic controller and then try to simplify it using model order reduction techniques as in Part 3.C, but to design directly a controller of **given dimension and complexity**.

Question 3D.1: *Controller design (10%)*

It was illustrated in Question 3.C.2 that a reduced-order controller $C_{i,red}$ with two poles and two zeros (or equivalently $C_{e,red}$ containing an integrator in addition) can approximate very well the initial controller C_i. Your task here is to design a new controller $C^*_{e,red}$ (ultimately the integral controller $C^*_{i,red}$) with the same complexity as $C_{e,red}$, but this time directly without using model-order reduction techniques. The controller will have the following form (use MATLAB function tunableTF to define a genss system with two poles and zeros times an integrator):

$$C^*_{e,red} = \frac{1}{s}\cdot\frac{K^*(s^2 + n_1^* s + n_2^*)}{s^2 + d_1^* s + d_2^*} = \frac{1}{s}\cdot C^*_{i,red}$$

where $K^*, n_1^*, n_2^*, d_1^*, d_2^*$ are the five parameters to be tuned. For comparison, suppose that your previously computed controller $C_{e,red}$ parameters are denoted as K, n_1, n_2, d_1, d_2. In order to perform the tuning, use exactly the same Simulink file (Design.slx) you used for Question 3C.1 and the same weights W_1, W_2, W_3 in order to be able to compare the two cases. The MATLAB function that permits to solve the same H_∞ problem as before <u>but additionally imposing the structure of the controller</u> as is the case here is hinfstruct (not hinfsyn as previously), taking as inputs the open loop weighted plant $P(s)$, an initial controller $C^*_{e,red-init}$ and the function options. The output is the tuned controller genss object $C^*_{e,red}$, the performance level γ^* and some tuning information.

As a first question, make a 5×2 table and give in the first columns the numerical values of $C_{i,red}$ and on the second column the re-tuned ones of $C^*_{i,red}$ (use function getValue to obtain the tuned value of the controller from hinfstruct). How do they compare? What is the value of γ^* that you obtain this time compared to the one you obtained with the function hinfsyn? Why do you think it is slightly bigger (hence slightly worse)? Remember always that with hinfsyn you were initially designing a 9th order controller.

Hints: It is recommended to refer to the documentation of the hinfstruct and hinfstructOptions functions of MATLAB in order to grasp their functioning. You may need to use parallel computing to speed up computations since using more than 20 random starts is recommended. You may also need to also decrease the gain tolerance to obtain tighter results. A way to obtain $C^*_{i,red}$ from the tuned $C^*_{e,red}$ is to multiply by a pure differentiator and apply the mineral function to delete the integrator. Otherwise you can use zpkdata to select the appropriate poles and zeros.

As a second question, use the function lft(P,C_e,1,1) to obtain the weighted closed loop transfer function T_{wz} as in **Question 3C.1**. Again, isolate its three transfer functions, compute the individual $\gamma_{S_o^*}, \gamma_{C_e S_o^*}, \gamma_{T_m^*}$ and plot the singular values of $T^*_{wz}(s)$ and of its three individual transfers. Finally, give the Bode diagrams of $C_{i,min}, C_{i,red}$ (blue, green) computed with hinfsyn and $C^*_{i,red}$ (magenta) you computed with hinfstruct. Comment on the closeness between the controllers.

Question 3D.2: *Controller analysis & simulation (5%)*

In this question you will need to do the same exact analysis and plot the same results as in Question 3C.3 when you used the reduced order controller $C_{e,red}$, but this time using the controller $C^*_{e,red}$ you just tuned. This means frequency-domain and time-domain results for the closed loop transfer functions and also the open loop transfer function with the loop opened before the actuator. Plot your results now in magenta and also include in the same plot the hinfsyn results in blue. How do the various results compare? You may again use annotations on the figures to compare the results.

Hints: As proposed previously, use a cell Part #3d for your coding in this part and a cell Part #3c for the hinfsyn case. Save your transfer functions, controllers, stability margins, performance levels etc. in one single structure, for example Results_hinfsyn.gamma=gamma, Results_hinfsyn.C_i=C_i, etc. in order to call them every time you need to compare. Hence you will have a structure Results_hinfsyn, another one Results_hinfstruct, another Results_feedforward for the next part, etc.

E. Feedforward controller design (15%)

In this question you are required to design the feedforward controller F_f of Figure 6 in order that your system output y_1 matches more closely the output of the reference model $y_{1,d}$, or otherwise so that your model following error $e_{1,d} = y_{1,d} - y_1 = T_d r - y_1$ becomes smaller than previously. Recall that even though we have included a model following constraint T_m through a weighting filter W_3 in our design process, the feedback controller C_e may not be adequate by itself to minimize this error if we do not want to overly increase the performance level γ by making W_3 too severe.

Question 3E.1: *Controller design (5%)*

You are now asked to design the feedforward controller using system inversion. Suppose that your closed loop transfer function from r to y_1, assuming $F_f(s) = 1$, was previously computed as T_o (normally of order 7, i.e. 4 states for the missile and the actuator, plus 3 states for the controller itself) using your hinfstruct controller $C^*_{e,red}$. If you want your total closed loop, including your feedforward applied at the reference input, to behave as your reference model T_d, then you may set:

$$T_d = F_f T_o \Leftrightarrow F_f = T_d \cdot T_o^{-1}$$

and name this first feedforward controller as $F_{f,init}$. Write in your report the output of zpk(F_f_init). What can you observe on the singular value plot of the controller? Is it possible to implement it?

Question 3E.2: *Controller order reduction (5%)*

As a first exercise you are required to simplify the controller $F_{f,init}$ by reducing its order. Observe first the existence of a RHP (right-hand plane) pole-zero pair that can be cancelled as well as a pair of high frequency zeros that can be truncated. To remove these three zeros and the pole, use the function zpkdata to obtain the zeros, poles and gain of the controller (use the option 'v' of the function) and then select only the poles and zero you want to retain. Form the truncated controller $F_{f,lf}$ retain only the low frequency ones, totally 5 poles and 5 zeros, using the function zpk. Be careful that the dc gains of $F_{f,init}, F_{f,lf}$ match, for this you can use the function dcgain of MATLAB to compute the dc gains and do the adjustment. Write in your report the output of zpk(F_f_lf), plot the singular values of both controllers and compare the results.

As a second exercise, you are ready now to reduce the order of F_f_lf further using model-order reduction techniques. To achieve this use the function balred of MATLAB that performs a balanced reduction of a system and use the function balredOptions to adjust the reduction properties. Adjust the method to 'absolute' in order to be able to select the frequency band in which you want the reduction to focus. Down to what order you may reduce the order without significant degradation of the controller frequency content at low-mid frequencies? Give the output of zpk(F_f), where F_f is your final reduced controller as well as the pole zero map and singular values of $F_f, F_{f,init}$.

Question 3E.3: *Controller analysis & simulation (5%)*

As a first exercise concerning the frequency domain, and using again the function linearize on your Simulink analysis diagram ClosedLoop_Test.slx of Figure 6 and $C^*_{e,red}, F_f$ as controllers (i.e. not unitary feedforward now), plot the singular values of the model matching error $T_{r\to e_{1,d}} = T_m$ for all four cases: weighting filter inverse W_3^{-1} (red), result from hinfsyn (blue), result from hinfstruct (magenta), result from feedforward (green). What is the attenuation you get at the bandwidth ω_1 now and how many dB's have you gained with respect to the no feedforward cases? Additionally plot the singular values of all the four controllers you designed, namely the hinfsyn controller C_i, the hinfstruct controller C_i^* and the feedforward controller F_f. What can it be said concerning the shape of the controllers?

As a second exercise concerning the time domain, plot the step response of your closed loop $T_{r\to y_1} = T_o$ for all cases using the same color code, and comment on the closeness of the response to the reference model T_d (in red). How much have we gained in terms of settling time? Why is not the response matching exactly the one of the reference model despite the fact that you used model inversion? Finally, plot the actuator deflection rate (i.e. transfer function $T_{r\to \dot{u}_m}$ from the reference r to the actuator angle derivative $\dot{u}_m$ as shown in Figure 6) in degrees for all cases, how far are you from the saturation level of $25°/sec$?

Plot the above 4 figures in a (2,2) subplot with the frequency-domain results on the left column and the time-domain results on the right column. Use annotations to show important properties such as settling times, peaks, etc.

Hints: It is heavily recommended to refer again to the documentation of the balred and zpkdata functions of MATLAB in order to grasp their functioning. Be careful when you select the poles and zeros of the feedforward controller as their ordering may change from one run to the other!

4. Signal-based redesign (20%) (OPTIONAL)

Even though the functions hinfsyn and hinfstruct are very powerful and can compute a full order or even a reduced order controller that respects the desired constraints, their drawback is that they minimize all the H_∞ constraints **simultaneously** (here we have disturbance rejection, control signal attenuation and model following) and not **individually**. The former optimization leads to a single performance level γ that reflects the global level of success of the solution provided considering all the constraints. The result is that if the designer does want to respect $\gamma \leq 1$, the feedback controller is not the tightest possible and a feedforward controller may also be needed especially in order to satisfy the model following constraint.

In this last part we show that by individually optimizing the H_∞ constraints it is possible to obtain better and tighter results than hinfstruct or hinfsyn and that a feedforward controller is not mandatory.

This is achieved through the Control System Toolbox functions systune/slTuner and the associated Control System Tuner (**CST**) user interface of the Simulink Control Design toolset, that represent the state of the art in control system design. We proceed in two phases: first, reproduce the tuning constraints, select the tunable parts of the system and set the tuning options in the Control System Tuner interface and second generate a script for MATLAB and systune/slTuner in order to tune and analyze the results.

A. Tuning interface (10%)

In order to create the tuning interface we will use the Control System Tuner app linked to your existing ClosedLoop_Test.slx Simulink file of Figure 6. Use the following steps to create it following also the procedure that was used in the classroom.

Question 4A.1: *Tuning interface generation (10%)*

Step 1: Create a cell Part #4 – Feedback controller (systune) to load all models necessary for ClosedLoop_Test.slx, i.e. actuator and plant models G_a, G_m, pitch rate and scaling controllers K_q, K_{sc} and weighting filters W_1, W_2, W_3. Additionally initialize the feedback controller C_i with the one obtained from Question 3D.1 via hinfstruct, and finally the feedforward controller F_f to unity.

Step 2: Open the ClosedLoop_Test.slx file (see Figure 6) and verify it can correctly load all the necessary blocks (Ctrl+D). Right click on the reference signal r and define it as an input perturbation from the Linear Analysis Points submenu. Right click on the error signal e_1, control signal u, and model following error signal e_{1d} and define them as output measurement. This will enable CST later to define the closed loop transfer functions that you want to shape. Add also y_1, y_{1d} as the same type for visualization purposes. Open the CST (Apps->Control System Tuner) from your Simulink ClosedLoop_Test.slx file.

Step 3: In the tuning tab hit: Select Blocks->Add Blocks and click on the C_i block to make it tunable. Close the dialog box and then return to the main CST window. On the left pane (Data Browser), double click on the ClosedLoop_Test_C_i object in the Tuned Blocks, change its parameterization to 'transfer function' and ensure that the number of poles and zeros is both set to 2. This will permit you to tune the same type of controller as in the hinfstruct case. Hit OK to close the dialog.

Step 4: Add now the three tuning goals we have used in the previous parts of the exercise: namely a sensitivity, control times sensitivity and model following goal. All three can be entered via the menu New Goal -> Gain Limits in the Tuning tab of the CST. For the first, when the dialog box opens use So as name,

select r, e_1 as I/O (Input/Output) signals respectively and put $1/W_1$ as gain limitation. For the second use CeSo as name, r, u as I/O signals respectively and $1/W_2$ as gain limitation whereas for the third use Tm as name, r, e_{1d} as I/O signals respectively and $1/W_3$ as gain limitation. Hit apply multiple times to ensure all data was entered. Make the three tuning goals Hard from the Manage Goals menu of the Tuning tab.

Finally add a visualization of the output step response by selecting New Plot -> New Step in the Control System Tab and then use r as input signal and y_1, y_{1d} as output signals. Hit OK and then go to the step response main window, right click and select I/O Grouping->All to but both your system output and reference model output in one plot. Enable also the grid and visualize the 5% settling time also by right clicking on the plot and enabling it from the Characteristics menu. Ensure that when you right click on the plot, in the Properties menu and then in the Options tab the settling time is set to 5%. Rearrange your 4 plots, clicking on the View tab, then Custom, then 2x2. Your CST should now look like in Figure 9, assuming you entered correctly all the data and initialized the controllers accordingly.

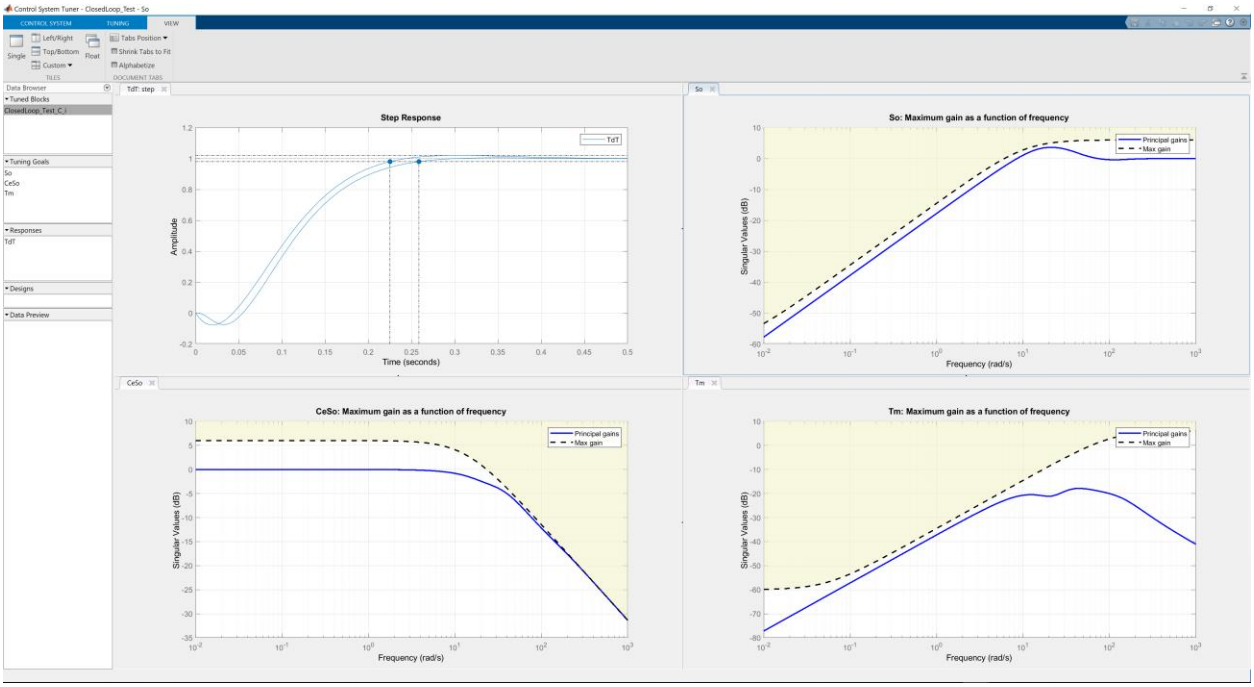

Figure 9 - CST interface

Step 5: Adjust the Tuning Options in the Tuning tab as follows: enable multiple starting points and use at least 20 randomized starts, enable parallel processing (assuming you have the Parallel Computing Toolbox) and enable the show report with final summary of the tuning. Even though you can press on the tune button to tune the controller within the CST interface, we prefer here to do it by generating MATLAB code and use systune instead. In order to do this, simply click on the small arrow below the Tune button and hit generate script with current values.

The deliverable here is the generated code in your report, a figure like Figure 9 above showing the various objectives and results as well as written comments on the various parts and functions used.

Hints: It is heavily recommended to refer to the documentation of the systune and slTuner functions of MATLAB in order to grasp their functioning. Additionally refer to the Control System Tuner interface documentation for additional details.

B. Feedback controller redesign (systune) (10%)

The goal of this final part is to re-design the controller C_i using systune in order to tighten the model following constraint without violating the performance level γ and hence without needing a feedforward controller. A comparison with the hinfstruct case can be also interesting.

Question 4B.1: *Controller design (5%)*

In order to perform the design of the controller C_i (assuming always a unitary feedforward controller F_f) we keep using the same values of the weighting filters as previously and integrate the code obtained in Question 4.A.1 in cell Part #4. Reduce the value of the mag parameter of the W_3 filter further than the one of Question 3.C.1 and observe how the individual performance levels $\gamma_{S_0}, \gamma_{C_eS_o}, \gamma_{T_m}$ gradually increase approaching unity.

This will consistently improve your model following performance making the settling time of your output y_1 to match the one of the reference model output y_{1d}. Globally, you should be able to obtain an attenuation well beyond $-30dB$ at the bandwidth frequency ω_1. In your report, comment on the procedure you used and give the output of zpk for both the hinfstruct controller $C^*_{i,red}$ and the systune controller $C^{\#}_{i,red}$. Additionally, plot the plot-zero map of both controllers and comment on their closeness.

Question 4B.2: *Controller analysis & simulation (5%)*

Similarly to what you have previously, compare only the feedforward design results (green) with the systune design results (in black) both in the time and frequency domain. How do the results compare now? Use annotations with your mouse to illustrate important results (i.e. settling times, maximum control rates, attenuation gains and frequencies, etc.). If you did the design correctly you should be able to completely match the reference model settling time while respecting the sensitivity and control time sensitivity constraints without needing a feedforward controller.