# Robustness measures in the nonlinear domain.

## Investigating frozen-time robustness of structured $\mathcal{H}_\infty$ controllers in nonlinear realizations.

Y.J. Naäman

Delft University of Technology

TU Delft

# Robustness measures in the nonlinear domain.

## Investigating frozen-time robustness of structured $\mathcal{H}_\infty$ controllers in nonlinear realizations.

by

Y.J. Naäman

Instructor: S. Theodoulis
Faculty: Faculty of Aerospace Engineering, Delft

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

# Acknowledgements

Looking back on my time spent writing the thesis reminds me of a quote often attributed to the Brazilian writer Fernando Sabino: *“In the end, everything will be okay. If it’s not okay, it’s not yet the end.”* Like many students before me, I often had to remind myself that it was indeed not the end yet and that it would turn out okay. However, as of now, the end has finally been reached and the product is indeed (at least I believe) okay.

I experienced the writing as an intense effort with many ups and downs and I am unsure the end would have been reached without the great help of my supervisor, prof. dr. ir. S. Theodoulis. He exposed me to the field of robust control and provided me with many technical lessons during our many meetings, encompassed by his most famous mantra ”respect the unstable”. Furthermore, he also provided me with many personal life lessons and I will be forever grateful for his guidance.

Furthermore, I want to thank my fellow students of the AEROCON research group. The shared stories about the thesis helped my morale and encouraged me to work harder. A special thank you to Daam, with whom I worked on the F-16 model together and Fredrik, who helped me with the final versions of my report.

Moreover, I am grateful to my beautiful and eternally supportive girlfriend, Ezgi, who helped me during the times I needed it. Be it through mental support, taking care of tasks to take them out of my hands, being excited about my progress or encouraging me before important meetings.

I’d also like to thank my family. My parents, Marianne and Eliezer, for their constant confidence and unconditional support. My sister, Noa, for her advice, help and constant ideas for new food to try together. My cousin, Lennar, for the many long conversations and the activities that we did to allow me a break from the everlasting thesis. My aunt and uncle, Pien and Roel, for their continual kindness and hospitality.

Lastly, a thank you to my roommates, Gert-Jan and Jamil, with whom I often relaxed at home. A special shoutout to Jamil who never failed to provide me with tea in the late evening hours, when I was still working on my thesis. And lastly, I’d like to thank many more of my friends, with whom I enjoyed many moments. From the many football matches played to the evenings in the pub, the dinner parties and the activities away.

*Y.J. Naäman*
*Delft, February 2024*

# Summary

The field of robust control has seen major progress and use in engineering practices after it was first popularized in the 1980s. This coincides with the development of the $\mathcal{H}_\infty$ control synthesis techniques, starting with the algorithm to calculate unstructured controllers. Now modern techniques allow for structured (possibly multi-model and gain-scheduled) $\mathcal{H}_\infty$ control synthesis. The application of this method in aerospace is still an active area of research while practical use in industry also has become popular.

The aforementioned theory on structured $\mathcal{H}_\infty$ synthesis is applied to two examples. A gain-scheduled pitch-rate control system is synthesized and a direct trajectory longitudinal control system is designed using multi-model synthesis. The pitch-rate controller is used as a relatively easy benchmark problem while the direct trajectory longitudinal controller is included to apply the theory to a challenging coupled MIMO system. The linear controllers are then implemented in the nonlinear domain. This transition from the linear to the nonlinear domain can introduce problems for both the gain scheduling and the multi-model approach. The goal of the thesis is to synthesize these two controllers and propose a method to analyze how far the robustness degrades due to the pitfalls encountered when moving from the linear to the nonlinear domain.

To tackle this goal, let us discuss gain scheduling and multi-model synthesis. Starting with gain scheduling, the conventional gain scheduling approach was, and still is, popular in order to enable linear $\mathcal{H}_\infty$ synthesis to control nonlinear systems. A flight envelope consisting of many equilibria points is defined using scheduling parameters. At each equilibrium point, the system is linearized and a controller is synthesized on the resulting linear time-invariant systems. Subsequently, in the nonlinear application, the family of controllers is interpolated in some way. Often using the measurement of the scheduling variables to calculate the gain values in real time. While the technique existed for a longer time, only in the late 1990s and early 2000s were the mathematical rigorous definitions defined and the exact disadvantages of gain scheduling mathematically formulated.

These disadvantages are caused by three pitfalls of gain scheduling. First, gain scheduling can cause the introduction of hidden coupling terms. The introduction of the scheduling parameters in the calculation of the controller gains can excite a hidden feedback loop. Secondly, conventional Jacobian linearization can only be performed at equilibria/trim points, while real flight conditions often diverge from these points, causing trim point uncertainty. Thirdly, the linearization process fails to capture the inherent time-varying properties of the nonlinear system. The thesis proposes a method to assess the first two problems. The third problem, however, is outside the scope of this research.

The multi-model approach also uses linearized systems at equilibria points. However, in contrast to gain scheduling, one controller is synthesized using all linearized systems simultaneously. As a result, no a posteriori interpolation using the scheduling parameters is needed and the resulting linear controller can simply be implemented in the nonlinear domain. This approach contains the same pitfalls as gain scheduling except that there are no hidden-coupling terms as there is no interpolation using the scheduling parameters and hence no hidden feedback loop will appear in the nonlinear domain.

Now that the pitfalls are introduced, they need to be quantified. The hidden coupling terms (if applicable) can be derived by linearizing the controller while taking the scheduling into account as parameters in the system. This can be compared to the ideal linear controller without these hidden coupling terms. These two systems should ideally be equivalent but due to the scheduling parameters inducing the extra feedback loop in the nonlinear domain, there can be a difference. The difference between the systems can be expressed in terms of the gap metric.

To assess trim point uncertainty at any operating point, the system has to be linearized at off-equilibrium points. This can be done using velocity-based linearization. This linearization technique does not rely

on series expansions to establish a linearized system but takes the derivative of the nonlinear system with respect to time. This will result in a system that is linear in the derivatives of the original states. If the initial conditions are correctly chosen, the formulation of a system using the derivatives of the original states is dynamically equivalent to the original system. Hence, by looking at the plant at any two equilibrium and off-equilibrium points, the similarity can be quantified in terms of the gap metric.

Thus, the gap metric at any operating point with respect to both the hidden coupling terms and trim point uncertainty can be calculated. This indicated the distance between the systems and it can be used to identify the aforementioned two problems in itself.

Next to the gap metric, the normalized coprime stability margin can be used to assess the robustness with respect to the hidden coupling terms and trim point uncertainty. This is referred to as frozen-time robustness. The normalized coprime stability margin is a value between 0 and 1 that specifies robustness with respect to unstructured perturbations. If the absolute value of the robustness is of interest and not the contribution of each component, the coprime stability margin is more appropriate than the gap metric and can be used to assess the frozen-time robustness of any operating point. Ultimately, this application of the gap metric and normalized coprime stability margin allows us to quantify the degradation when moving from the linear to the nonlinear domain; at least concerning the hidden coupling terms and trim point uncertainty.

# Contents

# 1 Introduction

## 1.1. Background

In the late 1960s and early 1970s, control research predominantly concentrated on mathematically-framed optimization problems. The primary focus was on state-space concepts, such as controllability, observability, state estimation, and Linear Quadratic Gaussian (LGQ) control [9]. Despite their success in addressing diverse engineering challenges, these optimal control methods were not without controversy, particularly concerning the gain and phase margins of Linear Quadratic Gaussian controllers [10].

Simultaneously, the foundations of what is now known as robust control were being laid, marked by concepts like the small-gain theorem [51]. However, these contributions were not immediately embraced, leading to a widening gap between theoretical optimization and engineering practice. Frequency-based methods, e.g. root-locus and Bode plots, were perceived as simplistic, and optimal control theory seemed preferable due to its mathematical rigidity. In spite of that, optimal control methods often simply assumed sufficient model accuracy without explicit consideration of robustness, especially when dealing with achievable model accuracy. Early attempts to apply these optimal multivariable feedback controllers were not as successful as expected, such as during the LQG control design study for the F-8C Crusader aircraft, revealing issues related to their robustness. This prompted a shift in research focus, leading researchers to revisit earlier work on uncertainty in feedback loop design and the robustness properties of frequency-based methods [39].

During this shift in focus, $\mathcal{H}_\infty$ methods for controller synthesis became increasingly popular. While the foundation of the concept was developed earlier, the first rigorous mathematical formulation was presented in 1981 [50]. In this mathematical formulation, explicit model uncertainty was taken into account and the influence of disturbances on the output was considered. This was all framed through the $\mathcal{H}_\infty$-norm, i.e. the maximum gain in sinusoidal amplitude over the entire frequency range [10].

Developments regarding $\mathcal{H}_\infty$ did not stop there. The proposed $\mathcal{H}_\infty$ synthesis method could only be applied to full-order controllers. In many practical control engineering applications, a simple structured controller, like a PID controller, was preferred because of other considerations. Nevertheless, it took a long time until the structured $\mathcal{H}_\infty$ problem was solved, which only happened in 2006 [2]. Subsequent developments allowed $\mathcal{H}_\infty$ control with possible useful extensions, such as incorporating multiple models.

These advancements extend beyond theoretical realms, finding many practical applications. One example of a noteworthy application was the synthesis of the controller on board the Rosetta space probe following a loss of efficiency in one of the thrusters[3]. Here, uncertainty in thruster behavior was directly taken into account in the synthesis. This prompts the exploration of integrating robust structured $\mathcal{H}_\infty$ methods into complex aerospace and control applications, considering not only linear systems but also nonlinear systems.

One very popular, conventional method to design a controller for a nonlinear system is by using gain scheduling. The design of the controller then consists of four steps. First, a linearized version of the plant has to be created at a set of equilibria points. Careful consideration has to be put into choosing suitable scheduling parameters that capture the nonlinearity of the plant. Subsequently, linear design methods are used to synthesize a controller at each point. Then, in the nonlinear domain, interpolation of the controllers can be done using the measured scheduling parameters. This allows real-time calculation of the controller values throughout operation [37]. Lastly, the controller needs to be analyzed a posteriori, both in the linear (if necessary) and nonlinear domain.

Another approach to control a nonlinear system is by making use of multi-model $\mathcal{H}_\infty$ synthesis. This approach also uses the linearized plants at the equilibria points. However, in this case, one controller is synthesized using all plants simultaneously. This allows the designer to create one unique controller to control all linearized systems. This technique can be used to ensure robustness against parameter variations (even failure modes) [3] or to control a larger part of the operating envelope of a nonlinear system using one linear controller.

The incorporation of linear design methods to achieve nonlinear control is a major advantage [43]. For example, it is possible to use linear $\mathcal{H}_\infty$-synthesis in combination with gain scheduling to design a nonlinear gain-scheduled scheduled $\mathcal{H}_\infty$ controller. Or the multi-model approach can be used to control a large envelope of the nonlinear system using a single controller. However, there are also limitations attached to these methods.

## 1.2. Problem statement

The $\mathcal{H}_\infty$ gain-scheduled approach has several limitations to consider, especially in the transition from the linear to the nonlinear domain. In practical applications, these limitations used to be avoided by engineering heuristics that were summarized by for example ”the system should vary slowly”, ”the scheduling parameters should capture the nonlinearity of the system” and ”the system should stay close to the equilibria conditions” [42]. While enjoying much practical success, only in the 90s there were multiple efforts to mathematically formalize the limitations. While not exactly equal, similar limitations hold for multi-model $\mathcal{H}_\infty$ synthesis as will become clear later.

The purpose of the thesis is to combine the efforts on gain scheduling, multi-model synthesis, nonlinear realizations and modern structured $\mathcal{H}_\infty$ robust control methods to design a controller and analyze, as far as possible, the robustness property of this controller in the nonlinear domain. This brings us to the main problem statement:

*To what extent is it possible to assess the frozen-time robustness of the nonlinear realization of a structured $\mathcal{H}_\infty$ controller?*

Normally, the main problem of a controller that is synthesized using linearized systems at equilibria points is that it only guarantees local stability at these equilibria points. This is caused by three reasons but the two main causes that are investigated in this thesis are hidden coupling terms [27] and trim-point uncertainty [24].

The hidden coupling is caused by a feedback loop between the scheduling parameters and the plant that exists in the nonlinear domain. This is only a problem for a gain-scheduled design as the multi-model approach does not use scheduling parameters in the nonlinear implementation.

The trim-point uncertainty appears because the controller design is only considered at equilibria points. In operation, however, the system will traverse operating points that are not necessarily equilibria points and this is the case for both gain scheduling and multi-model controller synthesis. Ensuring robustness regarding these two limitations (hidden coupling terms and trim-point uncertainty) is referred to in this thesis as frozen-time robustness. The frozen-time robustness extends the local stability guarantees to account for these limitations. Furthermore, the concept of frozen-time robustness could also help nonlinear analysis, as it assesses the aforementioned nonlinear effects.

The third reason why controller design using linearization at operating points can only guarantee local stability is because there is an inherent uncaptured discrepancy between a family of linearized systems and a nonlinear system caused by time variations of the dynamics in the true system. This limitation cannot be tackled by linearization at specific operating points as that can only capture a frozen-form of the dynamics and hence a multi-model or gain scheduling approach inherits this problem. Thus, if time-variation needs to be taken into account other methods than linearization at frozen operating points would need to be used, however, this is deemed as outside the scope of this thesis.

## 1.3. Approach

To answer the research question, an F-16 aircraft model [33] is used. This aircraft is highly maneuverable, which allows the aircraft to sufficiently distance itself from equilibria points and possibly excite hidden coupling loops due to fast changes in scheduling variables. An additional challenge of the thesis was to investigate flight dynamics from a tensor-based point of view. Tensors describe intrinsic physical properties without falling back on an arbitrary representation such as coordinate systems [54]. This allows for a more fundamental formulation of the dynamics.

Subsequently, the pitfalls of the gain scheduling and multi-model design procedures are investigated by looking at research done on gain scheduling and linearization in the 90s [24] [37]. The aim is to extend the local stability guarantees and assess robustness with respect to hidden coupling terms and trim point uncertainty when implementing a multi-model or gain-scheduled controller in the nonlinear domain.

Thereafter, the theory is implemented by looking at two types of controllers. A pitch-rate tracker is designed using the short-period approximation and by making use of gain-scheduling. Thereafter, a direct trajectory longitudinal controller is also synthesized, this time using the multi-model approach. The first controller establishes a relatively easy benchmark problem that allows for easy comparison across methods and allows easier investigation into the theoretical process. The latter controller is used to apply the theory to a more challenging MIMO problem. In both cases, structured $\mathcal{H}_\infty$ methods developed and improved upon from around 2006 [2] are used to synthesize the controller in the linear domain. Note, that synthesizing the controllers (especially the full longitudinal controller) is already a challenge on its own and one of the main goals of the thesis.

## 1.4. Report Outline

The report has been divided into the following sections:

- Chapter 2 discusses the intricacies of the used F-16 model. Furthermore, the implementation of the model in MATLAB/Simulink using tensor form is elaborated upon. This MATLAB/Simulink model is also validated using earlier-developed versions.
- Chapter 3 covers the pitfalls of the gain scheduling and multi-model controller synthesis methods. Furthermore, the analysis method to be able to extend the local stability guarantees is introduced.
- Chapter 4 covers the linear $\mathcal{H}_\infty$ controller design for a pitch-rate controller. Furthermore, the theory of gain scheduling is applied to analyze the controller on a nonlinear system.
- Chapter 5 applies multi-model $\mathcal{H}_\infty$ to synthesize on a more complicated controller. Namely, a full direct trajectory longitudinal control system is designed and analyzed.
- Chapter 6 presents the conclusion of the thesis, summarizes the pitfalls encountered in the study and issues recommendations for further research.

# 2

# F-16 Model

To design a flight control system, some kind of suitable model is necessary. In the case of this research, it is no different. As the problem statement in Chapter 1 mentioned, relatively fast dynamics are most likely to introduce problems in the nonlinear domain. Thus some kind of agile aircraft is needed, which is why the F-16 model originally introduced by Nguyen [33] is deemed suitable. This chapter discusses the model in detail.

Before the model can be introduced, however, the mathematical tools that are used for its development have to be understood. Thus, Section 2.1 gives a brief summary of the required tensor mechanics to understand the implementation. Thereafter, Section 2.2 introduces the model and provides an overview. Subsequently, Section 2.3 investigates the aerodynamic modelling, after which Section 2.4 discusses how the equations of motion are formulated using tensors. Section 2.5 and Section 2.6 cover the sensor dynamics and actuator dynamics respectively. Lastly, Section 2.7 compares the implementation with other versions to ensure the validity of the model. Note that the model is part of a larger project. The author mostly focused on making corrections, the aerodynamic model of Section 2.3, the translational dynamics and the verification of the model.

## 2.1. Introduction to tensors

The whole model is formulated by making use of tensor-based flight dynamics. Thus it is necessary to briefly establish this mathematical tool before the model can be introduced. This section covers a summary of the theory established by Zipfel [54] [55] [56].

From a physical point of view, tensors can be interpreted as follows [55, p. 27]:

> "*Tensors describe properties of intrinsic geometrical or physical objects, i.e., objects that do not depend on the form of presentation (coordinate system).*"

Physical properties are intrinsic to the world and are independent of the arbitrary selection of a coordinate system. Coordinate systems are only introduced to facilitate mathematical manipulations and enable the interpretation of results but do not entail any absolute property of the world around us.

The mathematical definition yields: "*A first-order tensor (vector)* $x$ *is the aggregate of ordered triples, any two of which satisfy the transformation law:*" [55, p. 27]

$$[x]^B = [T]^{BA}[x]^A \tag{2.1}$$

Here, the term $[T]^{BA}$ represents a transformation matrix from coordinate system $]^A$ to coordinate system $]^B$. The notation $]^A$ is employed to indicate that the tensors are already projected onto a coordinate system, specifically coordinate system $A$ in this instance.

Subsequently, "*A second-order tensor $X$ is the aggregate of ordered 9-tuples, any two of which satisfy the transformation law:*" [55, p. 27]

$$[X]^B = [T]^{BA}[X]^A[\overline{T}]^{BA} \tag{2.2}$$

where, again, $[T]^{BA}$ denotes a transformation matrix and, $]^A$ and $]^B$, indicate arbitrary coordinate systems. As evident from the equations, the tensor itself, denoted as $x$ or $X$, remains invariant across coordinate systems. It is only after projection on a coordinate system that a transformation matrix is required to establish equivalence between the two expressions.

This concept strengthens the connection between frames and objects, reducing it to a fundamental physical certainty without being influenced by an arbitrary coordinate system for expressing physical meaning. It serves as a powerful tool to generalize equations of motion, allowing for subsequent projection onto any coordinate system.

### 2.1.1. Rotation tensors

An essential tool for constructing the equations of motion is the rotation tensor. Denoted as $R^{BA}$, the rotation tensor signifies the orientation from frame B in relation to frame A. Rotation is not contingent on specific reference points and can seamlessly relate to both frames. The frames are depicted in Figure 2.1. The orientation between the frames can be determined using the rotation tensor [55, p. 88]:

$$b_i = R^{BA}a_i, \quad i = 1, 2, 3 \tag{2.3}$$

To show that the rotation tensor is indeed a tensor, it is possible to project Equation 2.3 onto two coordinates systems, namely $]^A$ and $]^B$, and show that the rotation tensor is invariant to this projection. Coordinated rotation matrices are linked to transformation matrices, that is:

$$[R^{BA}]^A = [R^{BA}]^B = [\overline{T}]^{BA} \tag{2.4}$$

It is worth noting that the rotation tensor maintains identical coordinates in both coordinate systems.

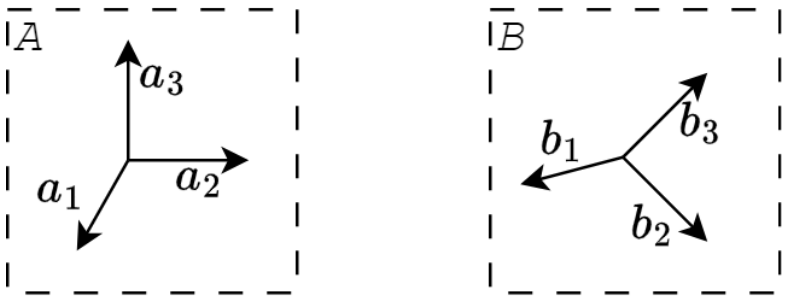



**Figure 2.1:** Frames A and B and their base triads.

### 2.1.2. Rotational derivative, velocity and acceleration

Another tool that needs to be established is a type of derivative that remains invariant across coordinate systems, namely the rotational derivative. Before delving into this concept, it is useful to examine why the ordinary time derivative does not adhere to that property. As an example, take the derivative of vector $s$ transformed from the $A$ coordinate system to the $B$ coordinate system, with $\omega$ representing the angular velocity between $B$ and $A$:

$$\frac{ds}{dt}|_A = \frac{ds}{dt}|_B + \omega \times s \tag{2.5}$$

The introduction of the time derivative of vector $s$ brought about an additional term that negates the tensor property. Now, let's derive the same equation in alignment with a tensor formulation, as outlined in [55, p. 104]:

$$[s]^A = [T]^{AB}[s]^B \tag{2.6}$$

Computing the time derivative by utilizing the chain rule results in:

$$\left[\frac{ds}{dt}\right]^A = [T]^{AB}\left[\frac{ds}{dt}\right]^B + \left[\frac{dT}{dt}\right]^{AB}[s]^B \tag{2.7}$$

This can be manipulated to the following expression:

$$\left[\frac{ds}{dt}\right]^A = [T]^{AB}\left(\left[\frac{ds}{dt}\right]^B + [T]^{BA}\left[\overline{\frac{dT}{dt}}\right]^{BA}[s]^B\right) \tag{2.8}$$

Upon comparison with Equation 2.5, it becomes apparent that $\omega$ is equivalent to $[T]^{BA}\overline{[dT/dt]}^{BA}$. This realization underscores that the classical interpretation of the time derivative of a transformed vector does not adhere to the tensor property.

Instead, a time derivative that adheres is invariant to coordinate systems is introduced: "*The rotational time derivative of a first-order tensor $x$ with respect to any frame $A$, $D^A x$, and expressed in any allowable coordinate system $]^B$ is defined by*" [55, p. 103]:

$$\left[D^A x\right]^B \equiv \left[\frac{dx}{dt}\right]^B + [T]^{BA}\left[\overline{\frac{dT}{dt}}\right]^{BA}[x]^B \tag{2.9}$$

The frames and coordinate systems in the formula can be chosen arbitrarily, but the left side and right side need to align. When the projected coordinate system matches the frame selected for the derivative, the expression simplifies to:

$$\left[D^A x\right]^A \equiv \left[\frac{dx}{dt}\right]^A + [T]^{AA}\left[\overline{\frac{dT}{dt}}\right]^{AA}[x]^B = \left[\frac{dx}{dt}\right]^A \tag{2.10}$$

The term $\overline{[dT/dt]}^{AA}$ represents the time derivative of the unity matrix, which equals zero. When this result is combined with Equation 2.8, one can infer:

$$\left[D^A x\right]^A = [T]^{AB}\left[D^A x\right]^B \tag{2.11}$$

which aligns with the definition of a tensor as per Equation 2.1.

Using the rotational time derivative, it is possible to articulate linear velocity and acceleration. In Figure 2.2, frame $A$, point $A$, point $B$, and the displacement vector $s_{BA}$ are illustrated. $s_{BA}$ represents the displacement between point $B$ and any arbitrary reference point associated with frame $A$. The linear velocity of point $B$ with respect to frame $A$ can then be defined as [55, p. 106]:

$$v_B^A = D^A s_{BA} \tag{2.12}$$

The linear acceleration can then simply be defined as the rotational derivative of the velocity:

$$a_B^A = D^A v_B^A = D^A D^A s_{BA} \tag{2.13}$$

The rotation tensor introduced in Section 2.1.1 can be used to define the angular velocity of frame $B$ with respect to frame $A$, namely as the change in the rotation tensor:

$$\Omega^{BA} = D^A R^{BA}\overline{R^{BA}} \tag{2.14}$$

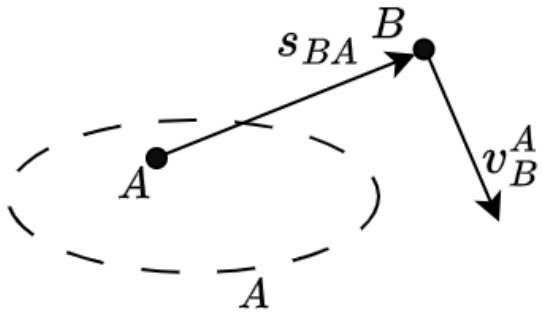



**Figure 2.2:** Linear velocity

### 2.1.3. Euler transformation

The final concept to address before delving into the actual model is the Euler transformation. This can be used to transform the rotational time derivative to another frame:

$$D^A = D^B x + \Omega^{BA} x \tag{2.15}$$

Here, $\Omega^{BA}$ represents the angular velocity matrix between frame $B$ and $A$ as defined earlier. This property is frequently employed later to transition from one frame to another.

## 2.2. Model introduction and overview

Now that the main mathematical tools are discussed, the model can be introduced. Initially, the F-16 model was established in 1979 by Nguyen [33]. This section gives a very brief overview of the model and outlines subsequent developments that have taken place since 1979.

The model uses six actuators, namely the engine, the elevator, the ailerons, the rudder, the speed brakes and the leading-edge flap. The first four are used on nearly every conventional aircraft configuration. The primary function of the leading-edge flap is to postpone stall in case of higher angles of attack flight and the speed brakes are used (as the name suggests) for additional braking. Moreover, it featured comprehensive longitudinal and laterally coupled (subsonic) aerodynamics.

The main coordinate systems that are used in the model are the aerodynamic/wind coordinate system and the body coordinate system, which can be viewed in Figure 2.3. Another often-used coordinate system is the local-level coordinate system, associated with the earth frame. This coordinate system aligns the z-axis downward, the x-axis north and the y-axis east.

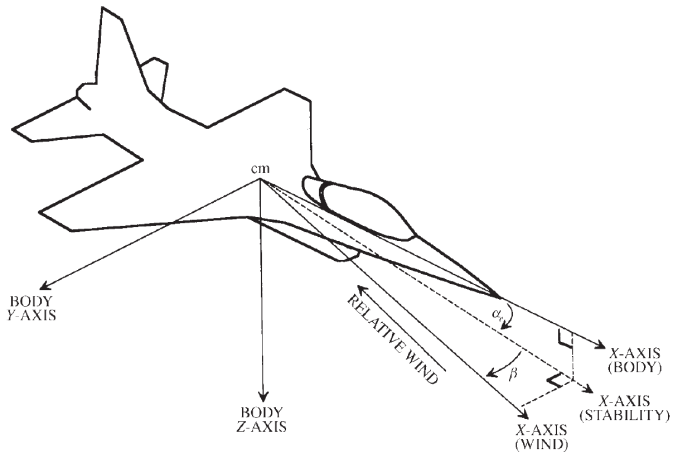



**Figure 2.3:** Definitions of wind and body coordinate systems [46, p. 76].

The description of the model in Nguyen [33] is relatively extensive. This allows for the analysis of interesting behavior such as pitch-out-departure at a higher angle of attack caused by inertial coupling and a non-zero $I_{XZ}$ or recovering from a deep stall trim point. For the development of the aircraft itself and its control system, the detail is useful. However, for some academic/didactic use cases with regard to control systems in general, such an extensive model often is not necessary. Thus Stevens, Lewis, and Johnson [46] simplified the model. The simpler model discards the speed brakes and leading-edge flap and completely decouples the longitudinal and lateral dynamics.

Russell [38] compiles both the original and simplified versions into a C++/Matlab version. Sadly, it can be difficult to leverage powerful MATLAB tools, such as batch linearization, as a significant part of the model is encoded in C++. This model of Russell [38] is therefore purposed for verification but another model, completely written in MATLAB/Simulink is developed and subsequently used for controller synthesis. The new model for the thesis copies the description of Nguyen [33] but removes the speed brakes as done by Russell [38].

To further introduce the new MATLAB/Simulink model, Figure 2.4 shows the outer loop. The inputs are the deflections of the control surfaces and the engine thrust, while the sensor outputs (logically) provide the output of the model. Importantly, the leading-edge flap position is given by a fixed control law, depending on the angle of attack, Mach number and altitude. This was done to align with the model of Russell [38], as a result, the leading edge flap position and the controller contribute two extra states to the model.

Figure 2.5 shows one sublevel of the model, namely the aiframe block. Here the environment, aerodynamics and equations of motion are computed. The environment and aerodynamics are discussed in Section 2.3 and the equations of motion in Section 2.4. Thereafter, the sensor dynamics, that can be seen in the outer loop of Figure 2.4, are discussed in Section 2.5. The actuator dynamics, except for the leading edge flap, are not included in the outer loop but are linearized separately and added later during the synthesis. Section 2.6 discusses the actuator dynamics in detail.

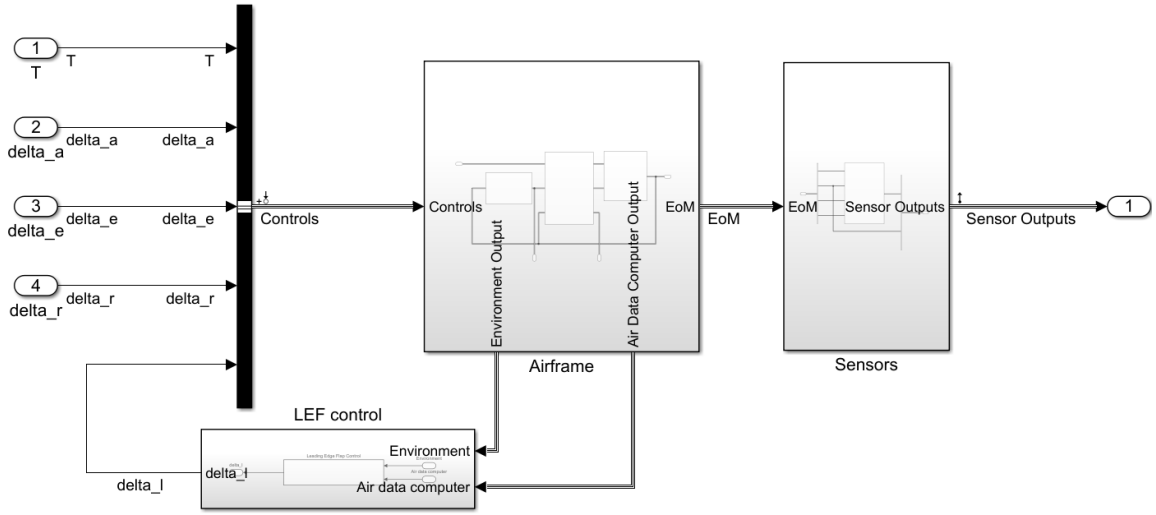



**Figure 2.4:** Overview of aircraft model (without actuators).

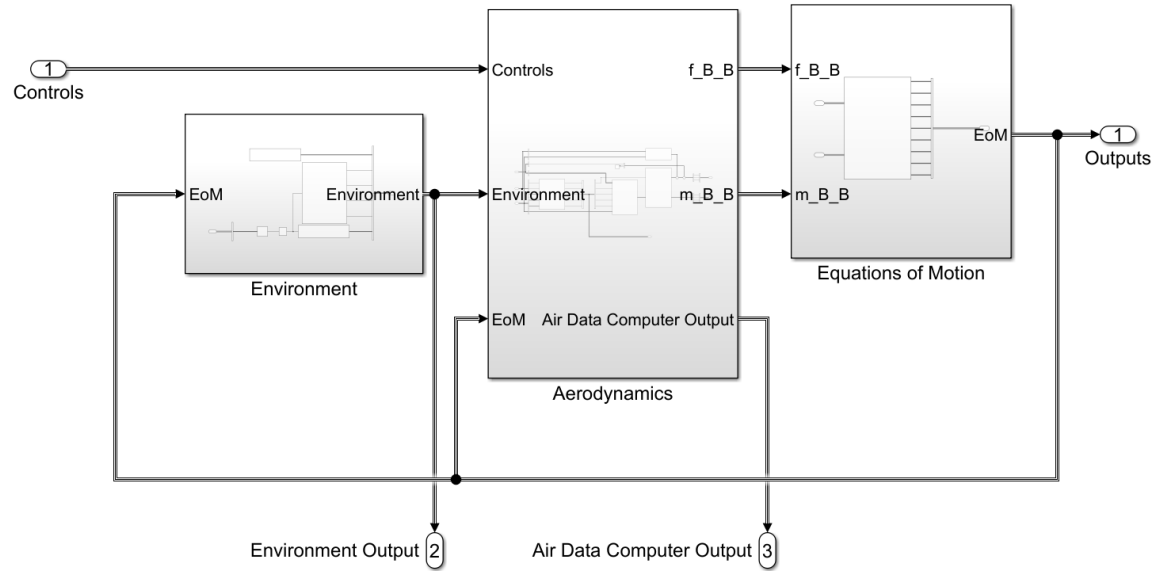



**Figure 2.5:** Overview of aircraft airframe

## 2.3. Aerodynamic model

The aerodynamic data was uncovered with low-speed wind tunnel tests using subscale models and without compressibility effects. This indicates that the aerodynamic data is only valid for Mach numbers smaller than $0.6$. This data was tested in ranges of $-20\,° \leq \alpha \leq 90\,°$ and $-30\,° \leq \beta \leq 30\,°$. Furthermore, it is not possible to assess the landing characteristics of the aircraft as only the clean configuration is available.

The purpose of this section is to calculate the forces and moments acting on the airframe given the actuator positions, the current altitude, attitude, velocity and angular velocity of the aircraft. Figure 2.6 shows the schematic of how that is done in Simulink. Normally, to calculate the forces/moments, the aerodynamic model is divided into the conventional six coefficients describing the forces and moments in each direction. These forces and moments are subsequently governed by their aerodynamic coefficients [46]:

$$\begin{aligned} \text{Axialforce}, \quad & X = \overline{q}\, S\, C_X \\ \text{Sideforce}, \quad & Y = \overline{q}\, S\, C_Y \\ \text{Normalforce}, \quad & Z = \overline{q}\, S\, C_Z \\ \text{Rolling moment}, \quad & l = \overline{q}\, S\, C_l \\ \text{Pitching moment}, \quad & m = \overline{q}\, S\, C_m \\ \text{Yawing moment}, \quad & n = \overline{q}\, S\, C_n \end{aligned} \tag{2.16}$$

Please note that these coefficients are defined with respect to the body axis shown in Figure 2.3. Furthermore, note that gravity is added separately at the end of this block.

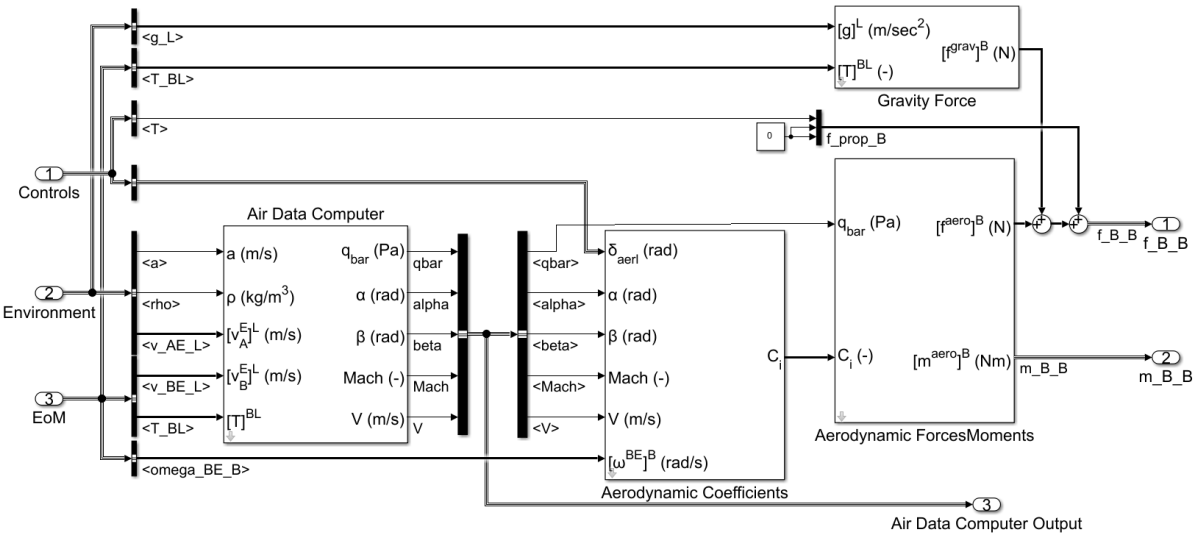



**Figure 2.6:** Aerodynamic model in Simulink.

Thus the calculation of the forces/moments requires the value of the aerodynamic coefficients. In summary, the aerodynamic coefficients are a function of the following parameters, as also shown in Figure 2.6:

$$C(V, \alpha, \beta, p, q, r, \delta_a, \delta_e, \delta_r, \delta_l) \tag{2.17}$$

Not every aerodynamic coefficient depends on all these parameters but details on the full aerodynamic model that has been specified by Nguyen [33], including its MATLAB implementation, is provided in Appendix A.

Lastly, the input for the aerodynamic coefficients is partly the actuator deflections but also includes values provided by the air data computer. Importantly, the air data computer can only read the velocity of the aircraft with respect to the air around the aircraft, not with respect to the earth. Given the velocity of the air with respect to the earth in the local-level coordinate system $[v_A^E]^L$ and the aircraft velocity with respect to the earth in the same coordinate system $[v_B^E]^L$, this produces:

$$[v_B^A]^L = [v_B^E]^L - [v_A^E]^L \tag{2.18}$$

This is the velocity of the body with respect to the air moving past the airframe and hence what the air data sensors encounter. To calculate the velocity, angle of attack and angle of sideslip, this is converted to the body coordinate system.

$$[v_B^A]^B = [T]^{BL}[v_B^A]^L \tag{2.19}$$

with:

$$[v_B^A]^B = \begin{bmatrix} u_A \\ v_A \\ w_A \end{bmatrix}$$

From this vector, the aerodynamic angles can be deduced:

$$\begin{aligned} V &= |||[v_B^A]^B||_2 \\ \alpha &= \arctan(\frac{w_A}{u_A}) \\ \beta &= \arcsin(\frac{v_A}{V}) \end{aligned} \tag{2.20}$$

Lastly, the speed of sound $a$ and the atmospheric density $\rho$ are environmental properties that are, in this model, purely determined by altitude as proscribed by the international standard atmosphere (ISA) model. The dynamic pressure ($\bar{q}$) can simply be calculated by $\bar{q} = 0.5\rho V^2$.

## 2.4. Equations of motion in tensor formulation

After the brief introduction to tensors of Section 2.1, it is possible to derive the equations of motion in tensor form. This section discusses the derivation and shows the application in MATLAB/Simulink of these differential equations. Figure 2.7 shows an overview of the Simulink implementation.

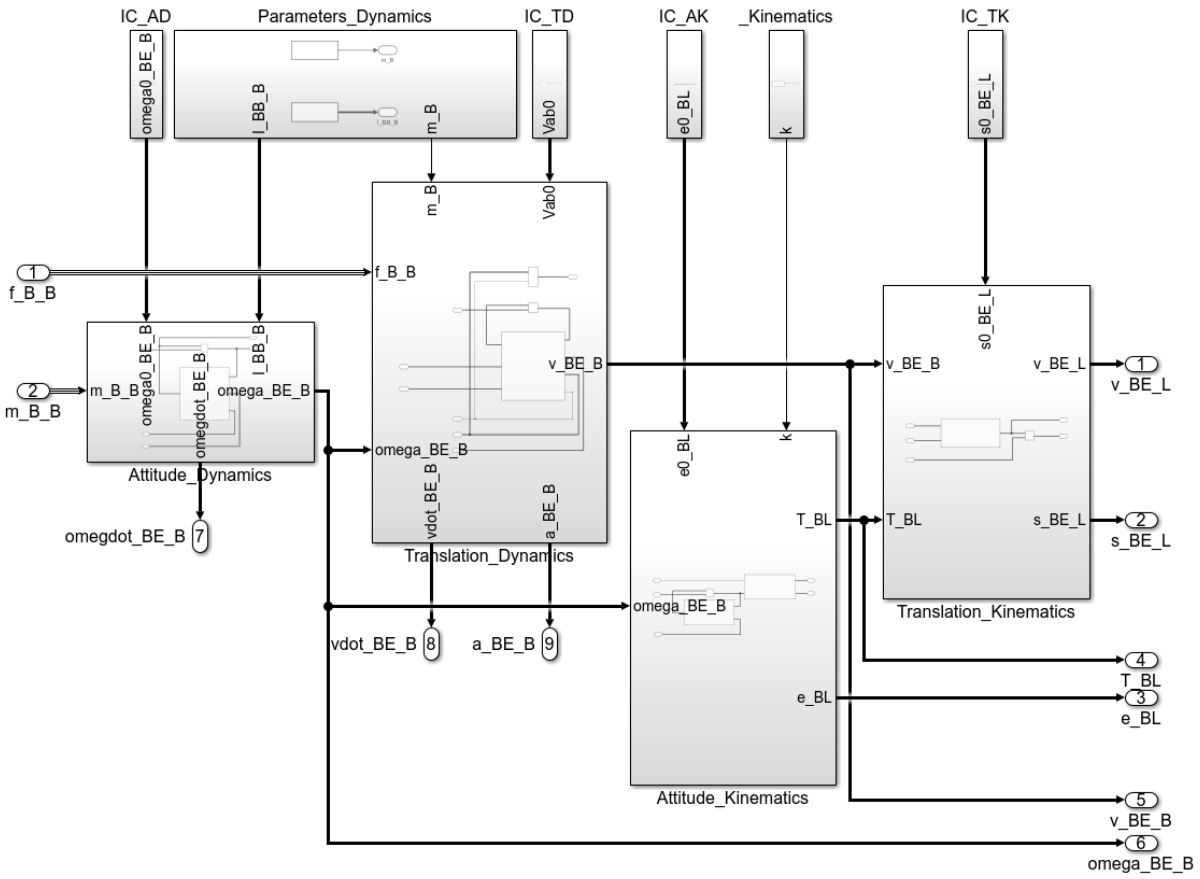



**Figure 2.7:** Equations of motion Simulink implementation

The purpose of the equations of motion is to calculate the translational and angular accelerations, velocities and positions given the acting forces and moments. The Simulink implementation of Figure 2.7 splits this into four main parts, the attitude and translational dynamics and the attitude and translational kinematics, containing three differential equations each. The other parts in Figure 2.7 are simply initial conditions or constants, such as the mass moment of inertial. This results in twelve distinct states defined in the equations of motion

$$\overline{x} = \left[ \; V, \alpha, \beta, p, q, r, x_L, y_L, z_L, \phi, \theta, \psi \; \right] \tag{2.21}$$

The upcoming sections will delve into the discussion of each of the four main parts and their respective implementations. Note that most of the mathematical derivation is retrieved from Zipfel [55], but the derivation of the translational dynamics in the wind axis from a tensor point of view is a novelty.

### 2.4.1. Translational dynamics

As customary, Newton's second law serves as the foundation for translational dynamics. In this instance, the equations are approached from a tensor-based perspective, leading to [55, p. 369]:

$$D^I(mv_B^I) = f_{a,b} + mg \tag{2.22}$$

Here, $m$ denotes the vehicle mass, $f_{a,b}$ represents the aerodynamic and propulsion forces acting on the vehicle, and $v_B^I$ signifies the velocity of the body with respect to the inertial frame $I$. Assuming that the Earth frame $E$ serves as an inertial reference frame and the mass $m$ remains constant over time, this expression can be rephrased as:

$$mD^E v_B^E = f_{a,b} + mg \tag{2.23}$$

Now, employing the Euler transformation, it becomes possible to express the derivative in terms of the body frame and the vehicle incident angles, which are utilized in the aerodynamic model:

$$mD^B v_B^E + m\Omega^{BE} v_B^E = f_{a,b} + mg \tag{2.24}$$

To arrive at the familiar coordinated equations of translational dynamics, everything can be projected onto the body coordinate system $]^B$:

$$m[D^B v_B^E]^B + m[\Omega^{BE}]^B [v_B^E]^B = [f_{a,b}]^B + m[g]^B \tag{2.25}$$

Working out the rotational derivative (see Equation 2.10) and transforming the gravity term $g$ to the local-level coordinate system yields:

$$m\left[\frac{dv}{dt}\right]^B + m[\Omega^{BE}]^B [v_B^E]^B = [f_{a,b}]^B + m[T]^{BL}[g]^L \tag{2.26}$$

The matrix formulation of the coordinated equations of motion becomes:

$$m\left\{ \begin{bmatrix} du/dt \\ dv/dt \\ dw/dt \end{bmatrix}^B + \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}^B \begin{bmatrix} u \\ v \\ w \end{bmatrix}^B \right\} = \begin{bmatrix} f_{a,p_1} \\ f_{a,p_2} \\ f_{a,p_3} \end{bmatrix}^B + \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix}^{BL} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}^L \tag{2.27}$$

If desirable, these can even be worked out to scalar form.

The above equation yields the values using the body coordinate system. In the application of the thesis, the wind frame and the wind coordinate system are used as this enables immediate linearization of the relevant states. In the derivation, it is assumed that there are no gusts and/or wind present and that the two frames thus coincide (in that case the frame is sometimes referred to as the kinetic frame[7, p. 46]). To start the derivation, Equation 2.24 is rewritten to the wind frame ($W$) using the Euler transformation:

$$mD^W v_B^E + m\Omega^{WB} v_B^E = -m\Omega^{BE} v_B^E + f_{a,b} + mg \tag{2.28}$$

Coordinating to the associated wind coordinate system results in:

$$m\left[D^W v_B^E\right]^W + m\left[\Omega^{WB}\right]^W \left[v_B^E\right]^W = -m\left[\Omega^{BE}\right]^W \left[v_B^E\right]^W + [f_{a,b}]^W + m[g]^W \tag{2.29}$$

The body rates and forces with respect to the inertial/earth frame in the wind coordinate system ($\left[\Omega^{BE}\right]^W$ and $[f_{a,b}]^W$) are not known hence transformation matrices are necessary to coordinate these expressions in the body coordinate system:

$$m\left[D^W v_B^E\right]^W + m\left[\Omega^{WB}\right]^W \left[v_B^E\right]^W = -m\,[T]^{WB}\left[\Omega^{BE}\right]^B [\overline{T}]^{WB}\left[v_B^E\right]^W + [T]^{WB}[f_{a,b}]^B + m\,[T]^{WB}[T]^{BL}[g]^L \tag{2.30}$$

To implement the equation, each unknown term in the expression is tackled separately. Starting with the leftmost term, $m\left[D^W v_B^E\right]^W$. As there are no gusts, $W$ coincides with frame $B$, which means that $\left[D^W v_B^E\right]^W$ is simply the change in velocity of the aircraft body with respect to the earth expressed in the wind axis. Since the first component of the wind axis is defined along the direction of the velocity this simply results in:

$$m\left[D^W v_B^E\right]^W = m\begin{bmatrix}\dot{V}\\0\\0\end{bmatrix}^W \tag{2.31}$$

To calculate $[T]^{WB}$, the transformation from the body axis to the wind axis needs to be investigated. This is defined by first a positive rotation around the y-axis, of the angle of attack, which yields an intermediate (also known as the stability) coordinate system. Subsequently, a negative (right-handed rotation) of the sideslip angle, from the intermediate axis, around the body z-axis produces the wind axis. This is displayed in Figure 2.8, yielding us the following transformation matrices from the body axis to the wind axis:

$$\begin{aligned}[T]^{BS} &= \begin{bmatrix}\cos(\alpha) & 0 & -\sin(\alpha)\\ 0 & 1 & 0\\ \sin(\alpha) & 0 & \cos(\alpha)\end{bmatrix}^{BS}\\ [T]^{WS} &= \begin{bmatrix}\cos(\beta) & \sin(\beta) & 0\\ -\sin(\beta) & \cos(\beta) & 0\\ 0 & 0 & 1\end{bmatrix}^{WS}\end{aligned} \tag{2.32}$$

$$[T]^{WB} = [T]^{WS}[\bar{T}]^{BS} = \begin{bmatrix}\cos(\alpha)\cos(\beta) & \sin(\beta) & \sin(\alpha)\cos(\beta)\\ -\cos(\alpha)\sin(\beta) & \cos(\beta) & -\sin(\alpha)\sin(\beta)\\ -\sin(\alpha) & 0 & \cos(\alpha)\end{bmatrix}^{WB} \tag{2.33}$$

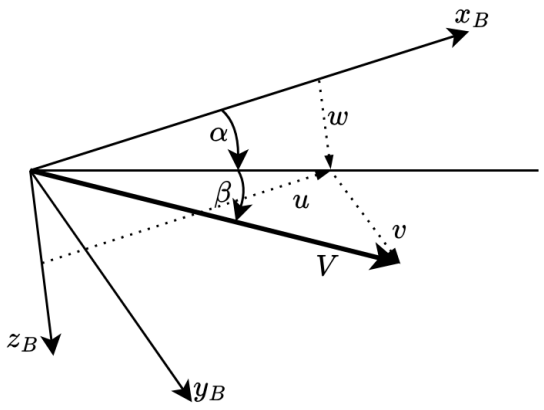



**Figure 2.8:** Body coordinate system to wind coordinate system.

The last novel expression to derive is the angular rates of the wind frame to the body frame expressed in the wind coordinate system, that is $\left[\Omega^{WB}\right]^W$. The angular rates between these frames are related

by a transformation dictated by the derivatives of $\alpha$ and $\beta$. Let us consider the non-skew-symmetric rates $\left[\omega^{WB}\right]^W$ derived similarly as done in [45, p. 179]. If the velocity vector as defined in the wind frame changes in position, it rotates by $\dot{\beta}$ around the $z_B$ axis from the perspective of the body frame. Subsequently a rotation of $-\dot{\alpha}$ around the new $y_B$ axis accounts for the rotation along the other plane. In equations, this is expressed as:

$$\left[\omega^{WB}\right]^W = \begin{bmatrix} 0 \\ 0 \\ \dot{\beta} \end{bmatrix}^W + [T]^{WS} \begin{bmatrix} 0 \\ -\dot{\alpha} \\ 0 \end{bmatrix}^S = \begin{bmatrix} -\dot{\alpha}\ \sin(\beta) \\ -\dot{\alpha}\ \cos(\beta) \\ \dot{\beta} \end{bmatrix}^W \tag{2.34}$$

This can be transformed to skew-symmetric form ($\left[\Omega^{WB}\right]^W = \left[\omega^{WB}\right]^W \times I$).

Now, all the tools are present to work out the left side of equation Equation 2.30:

$$\begin{aligned} m\left[D^W v_B^E\right]^W + m\left[\Omega^{WB}\right]^W \left[v_B^E\right]^W &= m \begin{bmatrix} \dot{V} \\ 0 \\ 0 \end{bmatrix}^W + m \begin{bmatrix} 0 & -\dot{\beta} & -\dot{\alpha}\cos(\beta) \\ \dot{\beta} & 0 & \dot{\alpha}\sin(\beta) \\ -\dot{\alpha}\cos(\beta) & -\dot{\alpha}\sin(\beta) & 0 \end{bmatrix}^W \begin{bmatrix} V \\ 0 \\ 0 \end{bmatrix}^W \\ &= m \begin{bmatrix} V \\ V\dot{\beta} \\ V\dot{\alpha}\cos(\beta) \end{bmatrix} \end{aligned} \tag{2.35}$$

To solve for $\dot{V}, \dot{\beta}, \dot{\alpha}$ (note the order), both sides of the equations are multiplied from the left by:

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & V^{-1} & 0 \\ 0 & 0 & V^{-1}\arccos(\beta) \end{bmatrix}$$

In the end, we have arrived at the final equations describing the translational dynamics derived from the wind axes:

$$m \begin{bmatrix} \dot{V} \\ \dot{\beta} \\ \dot{\alpha} \end{bmatrix} = X\left(-m[T]^{WB}\left[\Omega^{BE}\right]^B [\overline{T}]^{WB}\left[v_B^E\right]^W + [T]^{WB}[f_{a,b}]^B + m[T]^{WB}[T]^{BL}[g]^L\right) \tag{2.36}$$

Each component is known and the differential equation as a whole has been implemented in Simulink. Figure 2.9 shows how this has been done. The MATLAB function can be found in Appendix B.1.

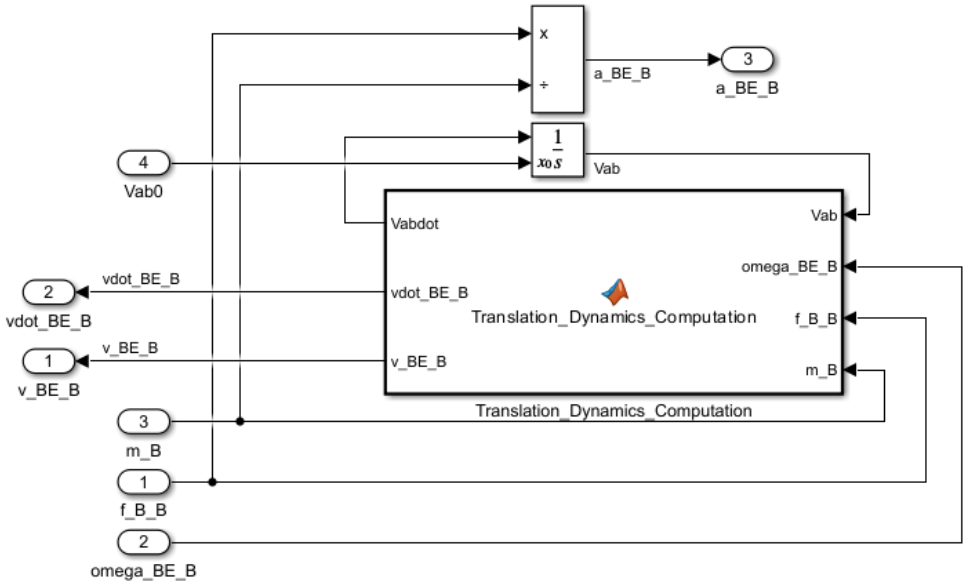



**Figure 2.9:** Translational dynamics implementation in Simulink.

### 2.4.2. Translational kinematics

The translational kinematics cover the transformation of the coordinated body velocities to a coordinate system that is easy to display on the inertial frame. If the definition of the rotational derivative is employed, this results in [55, p. 371]:

$$D^E s_{BE} = v_B^E \tag{2.37}$$

This can, for example, be worked out in the local-level coordinate system:

$$[D^E s_{BE}]^L = [\overline{T}]^{BL} [v_B^E]^B \tag{2.38}$$

This is implemented in Simulink, as shown in Figure 2.10 and the MATLAB function can be found in Appendix B.2.

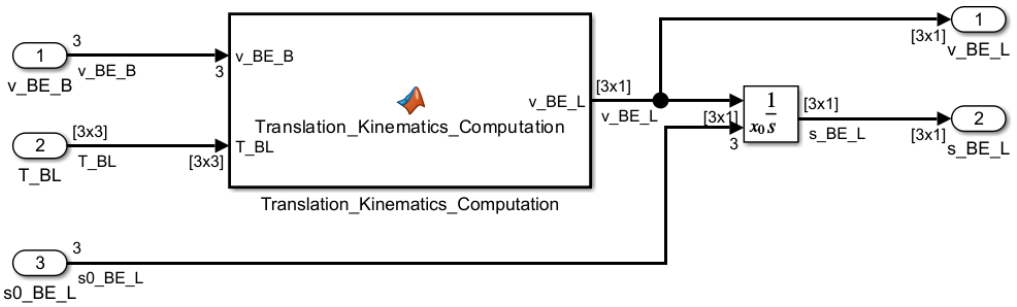



**Figure 2.10:** Translational kinematics implementation in Simulink.

### 2.4.3. Attitude dynamics

For attitude dynamics, it is possible to formulate the equations of motion in angular form using Newton's Law:

$$D^E(I_B^B \omega^{BE}) = m_B \tag{2.39}$$

with the Earth serving as the inertial frame. Here, $I_B^B$ represents the vehicle inertia, $\omega^{BE}$ denotes the angular velocity, and $m_B$ signifies the aerodynamic and propulsion moment. Once again, taking the derivative with respect to the vehicle body is preferred to unveil the vehicle dynamics, resulting in:

$$D^B(I_B^B \omega^{BE}) + \Omega^{BE} I_B^B \omega^{BE} = m_B \tag{2.40}$$

The rigid body assumption dictates that $D^B I_B^B$ is equal to zero and this results in:

$$I_B^B D^B \omega^{BE} + \Omega^{BE} I_B^B \omega^{BE} = m_B \tag{2.41}$$

This can be projected on the body coordinate system:

$$\left[\frac{d\omega^{BE}}{dt}\right]^B = \left([I_B^B]^B\right)^{-1} \left(-[\Omega^{BE}]^B [I_B^B]^B [\omega^{BE}]^B + [m_B]^B\right) \tag{2.42}$$

Expanding the conservation of angular momentum with the presence of other rotary devices $B_R$ introduces additional terms on the left side of the equation, for example:

$$I_B^B D^B \omega^{BE} + \Omega^{BE} I_B^B \omega^{BE} + I_{B_R}^{B_R} D^B \omega^{B_r E} + \Omega^{BE} I_{B_R}^{B_R} \omega^{B_R E} = m_B \tag{2.43}$$

where C is the joint center of mass of the body and the rotary device. The coordinated form yields:

$$\left[\frac{d\omega^{BE}}{dt}\right]^B = \left([I_B^B]^B\right)^{-1} \left(-[\Omega^{BE}]^B [I_B^B]^B [\omega^{BE}]^B + [\Omega^{BE}]^B [I_{B_R}^{B_R}]^B + [m_B]^B\right) \tag{2.44}$$

The differential equation implementation in Simulink is shown in Figure 2.11 and the MATLAB implementation can be found in Appendix B.3.

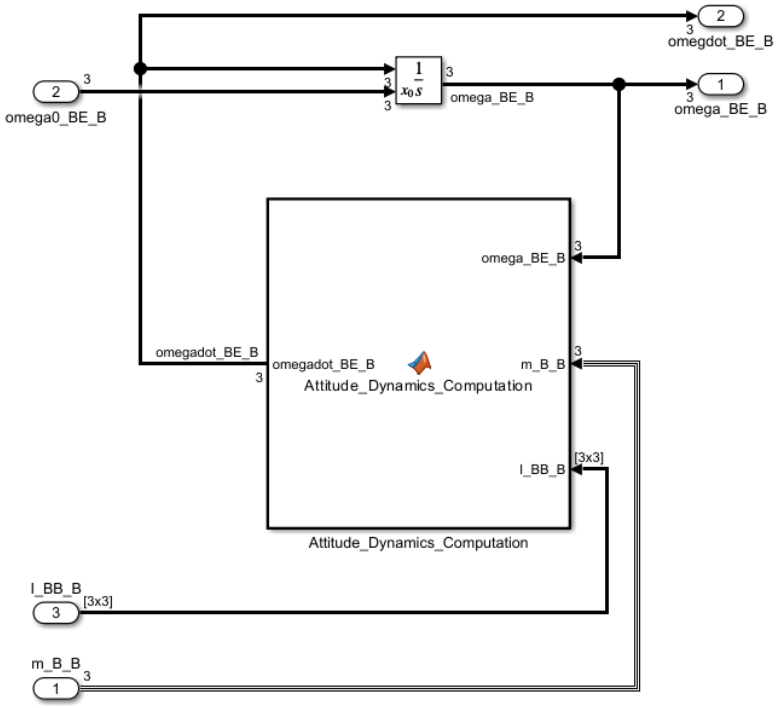



**Figure 2.11:** Attitude dynamics implemented in Simulink.

### 2.4.4. Attitude kinematics

Three additional kinematic equations can be introduced depending on the frame in use. For instance, in the body frame projection, $\overline{[\omega^{BE}]}^{B} = [p \quad q \quad r]$, and from here, a second set of differential equations can be deduced, whether in Euler angles or quaternions.

In this application, the direct Euler angle differential equations are used [55, p 120], computed by using a direct transformation matrix between the Euler angles and the body frame:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)/\cos(\theta) & \cos(\phi)/\cos(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.45}$$

This formulation is how it was implemented in the MATLAB function can be seen in Section B.4.

The Euler angles are used to construct the transformation matrix $[T]^{BL}$ that is used in the translational dynamics and kinematics. This is given by the equation:

$$[T]^{BL} = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\theta\sin\phi \\ \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{2.46}$$

Note that when integrating the differential equations can produce errors due to numerical inaccuracies when performing this on a computer [55, p 119]. To enforce orthonormality at each time, an extra orthogonalization step on the $[T]^{BL}$ is performed to ensure this property is not lost. That is done by applying a correction term from integration step $(n)$ to $(n+1)$:

$$[T(n+1)]^{BL} = [T(n)]^{BL} + k(1/2)(I - [T(n)]^{BL}\overline{[T(n)]}^{BL})[T(n)]^{BL}; \tag{2.47}$$

with $k$ being an arbitrary constant and $I$ the identity matrix. For the model, $k$ was set to $1$. The MATLAB function implementation can also be found in Appendix B.4. The Simulink implementation of the differential equation and the computation of the direction cosine matrix is given in Figure 2.12.

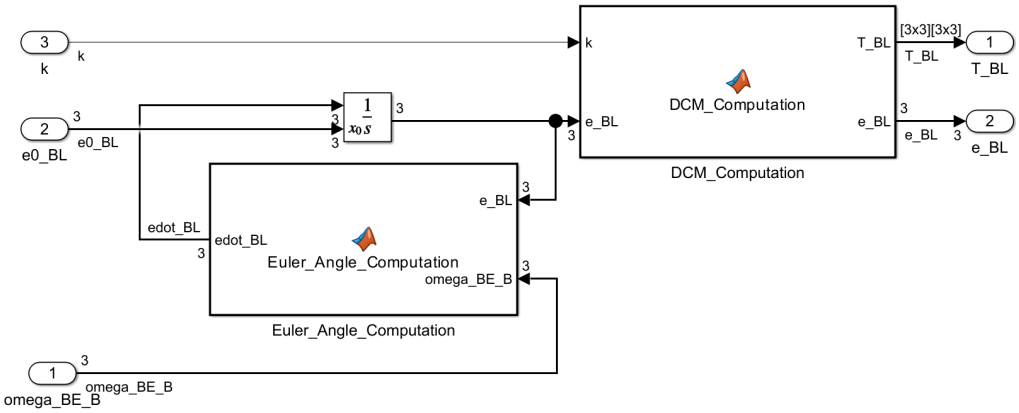



**Figure 2.12:** Attitude kinematics implemented in Simulink.

The aforementioned equations have not been derived from a tensor point of view but simply have been provided. If these equations were derived using tensor differential equations, you start with:

$$\left[\frac{dR^{IB}}{dt}\right]^{B} = \left[\Omega^{IB}\right]^{B}\left[R^{EB}\right]^{B} \tag{2.48}$$

That is, the change in rotation tensor between the inertial (earth) frame and the body frame is defined by the angular rates between the inertial and body frame multiplied by the current rotation tensor, in this case, all expressed in the body frame. Reversing the frame sequence to express the formula in the known body rates $\left[\Omega^{BE}\right]^{B}$ using the transpose:

$$\overline{\left[\frac{dR^{BI}}{dt}\right]}^{B} = \overline{\left[\Omega^{BI}\right]}^{B}\overline{\left[R^{BI}\right]}^{B} \tag{2.49}$$

If the connection between the rotation tensor and a transformation matrix is used ($\overline{\left[R^{BI}\right]}^{B} = [T]^{BI}$), that produces the differential equations of the direct cosine matrix:

$$\left[\frac{dT}{dt}\right]^{BI} = \overline{\left[\Omega^{BI}\right]}^{B}[T]^{BI} \tag{2.50}$$

Normally, these are computed using the Earth as inertial frame and the local-level system. These yield one differential equation for each element, hence nine equations, however only 3 of those are independent. These can be explicitly computed by taking orthogonality conditions between the three base vectors. From the elements of the direct cosine matrices, the Euler angles can be recovered.

## 2.5. Sensor dynamics

While many of the states are also directly the output, there is one exception in the model, namely the normal accelerations. This is measured by an accelerometer that captures the accelerations of the airframe itself. However, if it is not aligned with the center of gravity, the angular velocities of the sensor with respect to the center of gravity have to be taken into account. Starting from the fact that the position vector of the sensor/accelerometer ($S$) with respect to a point on the inertial frame $(I)$ is equal to the position of the body c.o.g ($B$), with respect to a point on the inertial frame and the position between the sensor and the body [55, p 153]:

$$s_{SI} = s_{BI} + s_{BS} \tag{2.51}$$

The acceleration (second derivative) of the position is what is of interest:

$$D^I D^I s_{SI} = D^I D^I s_{BI} + D^I D^I s_{BS} \tag{2.52}$$

$D^I D^I s_{BS}$ can be expressed in body rates after using successive Euler transformations [55]:

$$\begin{aligned} D^I D^I s_{BS} &= D^I \left( D^B s_{BS} + \Omega^{BI} s_{BS} \right) = D^I \left( \Omega^{BI} s_{BS} \right) \\ &= D^I \Omega^{BI} s_{BS} + \Omega^{BI} D^I s_{BS} \\ &= D^I \Omega^{BI} s_{BS} + \Omega^{BI} \left( D^B s_{BS} + \Omega^{BI} s_{SB} \right) \\ &= D^I \Omega^{BI} s_{BS} + \Omega^{BI} \Omega^{BI} s_{BS} \end{aligned} \tag{2.53}$$

It was assumed that the sensor position sensor is fixed ($D^B s_{BS} = 0$). Inserting this value back into Equation 2.52 and replacing the second derivatives with accelerations:

$$a_S^I = a_B^I + D^I \Omega^{BI} s_{BS} + \Omega^{BI} \Omega^{BI} s_{BS} \tag{2.54}$$

This relates the acceleration of the body at the center of gravity with the acceleration of the sensor. However, the accelerometer outputs the proper acceleration, that is the acceleration with respect to an observer in free fall [46, p. 33]. This means that the accelerometer measurement in steady wing-level flight ($[a_S^I]^B = 0$) produces $\overline{[n_S^I]}^B = [0, 0, -1]$ (body axis z points down, while the normal acceleration is $1\,$g up). In order to correct this gravity has to be subtracted from the accelerometer measurement Lastly, the output of the accelerometer is normally given in terms of gravitational acceleration, and not in $\mathrm{m/s}$. This conversion is also performed:

$$n_S^I = \frac{a_S^I - g}{g_0} = \frac{1}{g_0} \left( a_B^I + D^I \Omega^{BI} s_{BS} + \Omega^{BI} \Omega^{BI} s_{BS} - g \right) \tag{2.55}$$

Where $n_S^I$ denotes the normal accelerations in g acting on the sensor. This equation can be projected on any coordinate system. In this case, the body coordinate system is suitable for all components, except for the gravitational term which has to be converted from the local tangent coordinate system:

$$\left[ n_S^I \right]^B = \frac{1}{g_0} \left( \left[ a_B^I \right]^B + \frac{d}{dt} (\left[ \Omega^{BI} \right]^B) s_{BS} + \left[ \Omega^{BI} \right]^B \left[ \Omega^{BI} \right]^B \left[ s_{BS} \right]^B - [T]^{BL} [g]^L \right) \tag{2.56}$$

The Earth can be used as an inertial frame. If the conventional definition that steady-state flight results in an acceleration of (positive) $1\,$g is necessary, the sign of the z-axis can simply be flipped. The resulting MATLAB function can be found in Appendix B.5.

## 2.6. Actuator dynamics

Every control surface is simply modeled as a first-order lag with a static time constant. Additionally, both their rate of change and absolute values are saturated. Table 2.1 shows these properties.

**Table 2.1:** Summary of control surfaces properties.

| | $\delta_a$ | $\delta_e$ | $\delta_r$ | $\delta_l$ |
|---|---|---|---|---|
| Maximum deflection [°] | 21.5 | 25 | 30 | 25 |
| Minimum deflection [°] | -21.5 | -25 | -30 | 0 |
| Maximum deflection rate [°/s] | 80 | 60 | 120 | 80 |
| Bandwidth [rad/s] | 20.2 | 20.2 | 20.2 | 7.35 |

The ailerons, elevator, and rudder are used as system inputs but the leading-edge flap is not, in order to conform to the models of Nguyen [33] and Russell [38]. Instead, it is already included in the loop with a fixed control law, as could be seen in Figure 2.4. The control law is given by:

$$\delta_{l,cmd} = 1.38 \frac{2s + 7.25}{s + 7.25} \alpha - 9.05 \frac{\bar{q}}{P} + 1.45 \tag{2.57}$$

where $P$ is the static pressure. This control law and the actuator position introduce two new states.

Lastly, the turbofan jet engine with after-burning capabilities is a bit more complex than the other actuators and hence also needs additional complexity to model. Namely, the thrust response to throttle

input is modeled as a variable first-order lag. Additionally, the engine contains variations in thrust values corresponding to the idle, military, and maximum thrust levels. Moreover, the simulation incorporates engine gyroscopic effects by representing the engine angular momentum at a constant value of $216.9\,\text{kg} - \text{m}^2$. The detailed implementation is discussed in Appendix C.

## 2.7. Model verification

To verify the new model, it is compared to other versions of the F-16. As was mentioned, Russell [38] provided an open-source model that can be used as a benchmark. It is used to verify the trimming (Section 2.7.1) and the linearized state and input matrices (Section 2.7.2). Lastly, results directly from the book of Stevens, Lewis, and Johnson [46] are used to verify the output equations in Section 2.7.3.

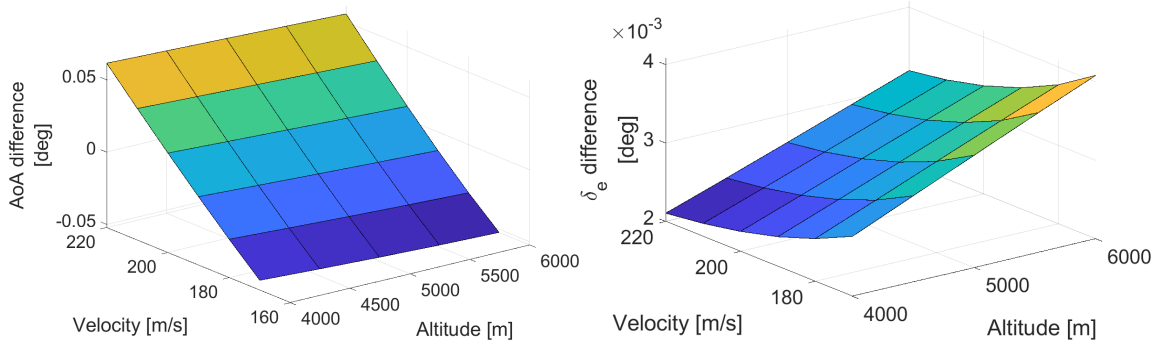
### 2.7.1. Trimming

For trimming, steady-state wing-level flight points are considered. Each equilibria point is defined by a certain $\alpha_{eq}$ and, $\delta_{e,eq}$ and these values should correspond to the benchmark. The trim point is retrieved numerically by establishing the settings as shown in Table 2.2. For steady-state flight, all known states are zero except for $V$ and $z_L$, which are specified to provide the desired airspeed and altitude.

**Table 2.2:** Trim settings.

| State | Known | Steady-state |
|---|---|---|
| $V$ | ✓ | ✓ |
| $\alpha$ | | ✓ |
| $\beta$ | ✓ | ✓ |
| $p$ | ✓ | ✓ |
| $q$ | ✓ | ✓ |
| $r$ | ✓ | ✓ |
| $x_L$ | | |
| $y_L$ | | ✓ |
| $z_L$ | ✓ | ✓ |
| $\phi$ | ✓ | ✓ |
| $\theta$ | | ✓ |
| $\psi$ | ✓ | ✓ |
| $\delta_{l,cmd}$ | | ✓ |
| $\delta_l$ | | ✓ |

The inputs are not known but simply in steady state and are an output of the trimming procedure done in MATLAB/Simulink by making use of the *findop* method. The trimming is performed across the flight envelope and compared to the model of Russell [38]. The resulting difference in steady-state angle of attack and steady-state elevator deflection is plotted in Figure 2.13. The error reaches $0.06°$ at max for the angle of attack and $4 \times 10^{-3}°$ for the elevator deflection.




**(a)** Difference in the angle of attack (Russell [38] - new model) in the flight envelope.

**(b)** Difference in the elevator angle (Russell [38] - new model) in the flight envelope.

**Figure 2.13:** Difference in trimming results.

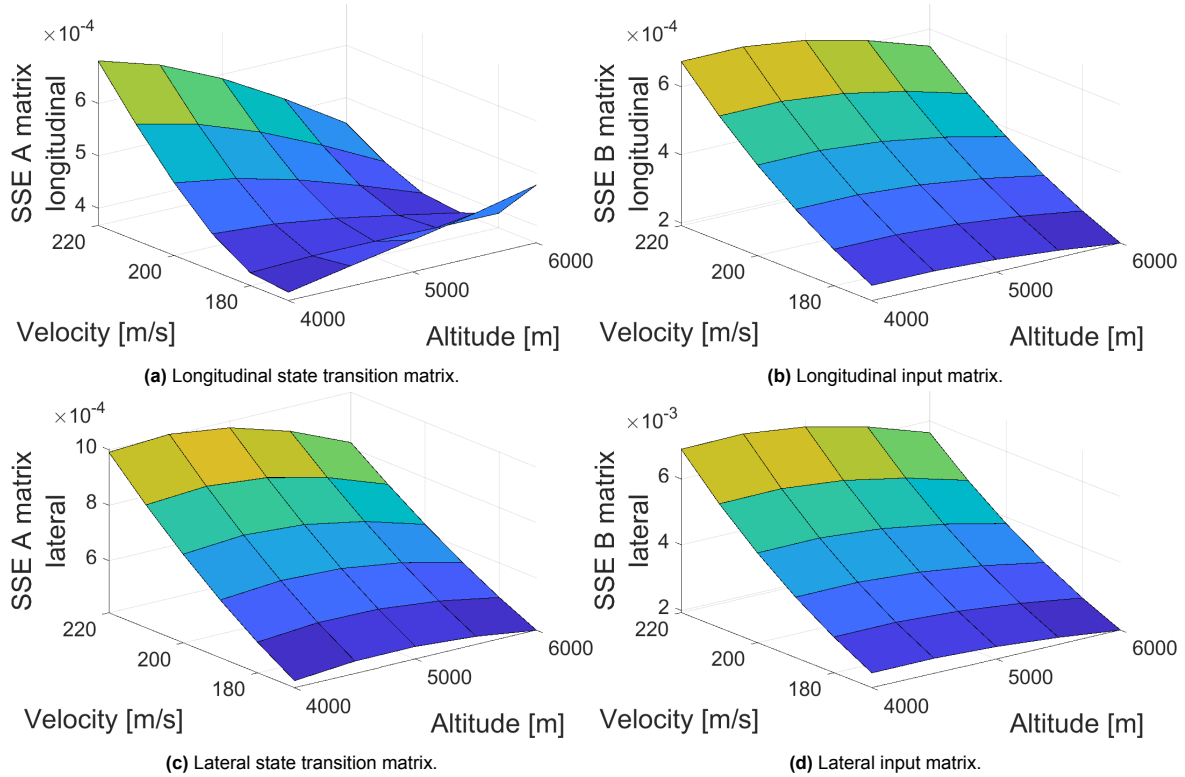
### 2.7.2. Linearized state and input matrices

To further verify if the model is configured correctly, the linearized state space system across the flight envelope is also compared to the model of Russell [38]. The resulting state ($A$) and input ($B$) matrices, should be similar. The model of Russell [38] does not include sensor dynamics, hence the output equations are separately verified in Section 2.7.3. The linearization was performed numerically using the *linearize* function of MATLAB/Simulink but the theory is further investigated in Section 3.1.

To assess the similarity of the $A$ and $B$ matrices, the sum of square errors is used. $A_{Russ}$ and $B_{Russ}$ denote the matrices of the model provided by Russell [38] and $A_{new}$ and $B_{new}$ denote the state and output matrices of the new model. Furthermore, the matrices are split in the longitudinal ($_{long}$) and lateral ($_{lat}$) dynamics. Subsequently, the equations that define the sum of squared errors become:

$$\begin{aligned} SSE_{A,long} &= trace((A_{Russ,long} - A_{new,long})'(A_{Russ,long} - A_{new,long})) \\ SSE_{B,long} &= trace((B_{Russ,long} - B_{new,long})'(B_{Russ,long} - B_{new,long})) \\ SSE_{A,lat} &= trace((A_{Russ,lat} - A_{new,lat})'(A_{Russ,lat} - A_{new,lat})) \\ SSE_{B,lat} &= trace((B_{Russ,lat} - B_{new,lat})'(B_{Russ,lat} - B_{new,lat})) \end{aligned} \tag{2.58}$$

Figure 2.14 shows that the SSE is generally very low across the flight envelope. It is smaller than $10 \times 10^{-4}$ for every matrix except for the lateral $B$ matrix which reaches $8 \times 10^{-3}$.




**(a)** Longitudinal state transition matrix.

**(b)** Longitudinal input matrix.

**(c)** Lateral state transition matrix.

**(d)** Lateral input matrix.

**Figure 2.14:** Sum of squared errors of components of the matrices construction the state space.

### 2.7.3. Linearized output equations

The output equations cannot be verified using the model of Russell [38] as the accelerometer is not modeled. However, Stevens, Lewis, and Johnson [46] provides the longitudinal linearized accelerometer output equation in case of a distance of $0$ m and $4.572$ m ( = $15$ ft) between the accelerometer and the center of gravity. Furthermore, they used a simplified aerodynamic model, which is thus also temporarily used in the new model, solely to verify the output equations. Lastly, these output equations were valid at sea-level altitude with a velocity of $153$ m/s. These provided equations [46, p 311], if converted to SI units, become:

$$\begin{aligned} n_{z,0} &= 0.00398V + 15.88\alpha + 1.481q + 1.908\delta_e \\ n_{z,15} &= 0.00398V + 16.26\alpha + 0.978q - 2.748\delta_e \end{aligned} \tag{2.59}$$

The equations of the new model at the same conditions are:

$$\begin{aligned} n_{z,0} &= 0.00398V + 15.87\alpha + 1.480q + 1.909\delta_e \\ n_{z,15} &= 0.00398V + 16.25\alpha + 0.978q - 2.774\delta_e \end{aligned} \tag{2.60}$$

As can be seen, the output equations are nearly identical, indicating that the (longitudinal) sensor dynamics are implemented correctly.

# 3

# Quantifying the pitfalls of gain scheduling and multi-model design

This chapter introduces the pitfalls of the gain scheduling and multi-model design techniques that are used in the upcoming flight control problems. Let us first consider gain scheduling. This technique remains widespread for nonlinear control because of the simple and intuitive framework it provides, despite the advances of other methods in control system theory [27]. However, it does not offer a priori global stability and performance guarantees, instead relying on local stability and performance [25] [37]. It accomplishes this through a four-step divide-and-conquer approach.

First, the nonlinear model is linearized on a set of equilibria points creating a family of linear time-invariant (LTI) systems.

Secondly, in the linear domain, many techniques can be used to synthesize a controller for each of the LTI systems to satisfy desired performance and robustness requirements.

Thirdly, the nonlinear realization can be created by interpolating the gains a posteriori with respect to the scheduling variables that define the set of equilibria points. In the implementation, this entails updating the gains in real-time based on the measured values of the scheduling variables [36] [42].

Lastly, the performance of the controller has to be assessed both in the linear domain and in the nonlinear domain. The linear analysis can be simple if the local stability and performance analysis has been included in the synthesis, however, if necessary this analysis can be extended and points that were not included on the grid used for design can also be analyzed. In the nonlinear domain, the local stability and performance are often analyzed by simulation. It can be of interest if the a posteriori nonlinear analysis can be quantified in some way instead of only relying on simulations. This is where the concept of frozen-time robustness might be useful.

To achieve this quantification, the pitfalls that can be encountered with gain-scheduled controllers in nonlinear control are investigated. As far as the author is aware, the three main problems are:

- Trim point uncertainty [23].
- Hidden coupling terms [21].
- Stability of time-varying systems [49].

For the multi-model approach, the main problems are similar. It does suffer the same trim point uncertainty and the problem of a discrepancy in the case of a time-varying system, as it also uses linearized plants at equilibria. However, a multi-model approach in itself does not cause any hidden coupling terms. This is the case as it does not calculate the gains using interpolation with respect to the scheduling parameters that cause these terms.

The goal of this chapter is to find a method to analyze if a gain-scheduled or multi-model controller remains robust despite the problems of trim point uncertainty and hidden coupling terms; that is to ensure frozen-time robustness. This is useful to extend the local stability of the controller to a larger part

of the nonlinear implementation and, as mentioned earlier, can be of use when assessing the nonlinear performance of the controller. The third problem, that of stability in the case of a time-varying system, is not taken into account in this analysis which causes the stability guarantees to still be local in nature. This means that the stability guarantee is only valid at a certain point and frozen at that point. Nevertheless, the increase of valid local stability points is an advantage compared to conventional design that does not take these factors into account.

The structure of the chapter is as follows. First, Section 3.1 explains why conventional Jacobian linearization introduces trim point uncertainty and how that can be omitted using velocity-based linearization. Secondly, the cause of hidden coupling terms and the effects on the controller are discussed in Section 3.2. Thereafter, the problem of loss of stability in fast-varying systems is briefly covered in Section 3.3. The quantification of the effects using the gap metric and the coprime stability margin, as far as that is possible, is discussed in Section 3.4. Lastly, an example of the quantification of these factors and its use in posterior analysis is shown in Section 3.5.

# 3.1. Linearization and trim point uncertainty

Linearization concerns the process of converting a nonlinear system to a Linear Time-Invariant (LTI) system. The conventional approach using Jacobian linearization is considered in Section 3.1.1, while a different method called velocity-based linearization (VBL) is covered in Section 3.1.2. The main advantage of using VBL is that it does not limit itself to equilibria points and can thus be used to quantify trim point uncertainty.

## 3.1.1. Jacobian linearization

As a starting point, a general nonlinear system is considered,

$$\begin{aligned} \dot{x} &= F(x,u) \\ y &= G(x,u) \end{aligned} \tag{3.1}$$

where $x$ denotes the states, $u$ the inputs and $y$ the outputs. $F(.)$ and $G(.)$ are general formulations of nonlinear functions.

Let us define a specific trajectory denoted by $\tilde{x}(t)$, $\tilde{u}(t)$, and $\tilde{y}(t)$, which represents the solution to the following system of differential equations:

$$\begin{aligned} \dot{\tilde{x}} &= F(\tilde{x},\tilde{u}) \\ \tilde{y} &= G(\tilde{x},\tilde{u}) \end{aligned} \tag{3.2}$$

This allows for the formulation of a time-varying system with respect to the trajectory through a first-order series expansion [22].

$$\begin{aligned} \delta\dot{x} &= F(\tilde{x},\tilde{u}) + \nabla_x F(\tilde{x},\tilde{u})\delta x + \nabla_u F(\tilde{x},\tilde{u})\delta u + \varepsilon_{\mathsf{F}} \\ \delta y &= G(\tilde{x},\tilde{u}) + \nabla_x G(\tilde{x},\tilde{u})\delta x + \nabla_u G(\tilde{x},\tilde{u})\delta u + \varepsilon_{\mathsf{G}} \end{aligned} \tag{3.3}$$

where

$$\delta u = u - \tilde{u}, \quad \delta y = y - \tilde{y}, \quad \delta x = x - \tilde{x}, \quad \delta\dot{x} = \dot{x} \tag{3.4}$$

And the errors are simply defined by the difference between the original equation for $\dot{x}$ and the approximated equation $\delta\dot{x}$, and the same thing can be done for $y$ and $\delta y$ [23]:

$$\begin{aligned} \varepsilon_{\mathsf{F}} &= F(x,u) - F(\tilde{x},\tilde{u}) - \nabla_x F(\tilde{x},\tilde{u})\delta x - \nabla_u F(\tilde{x},\tilde{u})\delta u \\ \varepsilon_{\mathsf{G}} &= G(x,u) - G(\tilde{x},\tilde{u}) - \nabla_x G(\tilde{x},\tilde{u})\delta x - \nabla_u G(\tilde{x},\tilde{u})\delta u \end{aligned} \tag{3.5}$$

This serves as an exact (local to the trajectory) reformulation of the initial nonlinear system but transforms into an approximation when the errors are assumed to be zero. This is the same as neglecting the higher-order Taylor approximation terms. Additionally, to make the system linear, the series expansion should be conducted in proximity to equilibria points, ensuring that $F(\tilde{x},\tilde{u}) \approx 0$. Hence, the nonlinear system can be locally approximated by a Linear Time-Varying (LTV) system as stated in [25],

$$\begin{aligned} \delta\dot{\hat{x}} &= \nabla_x F(\tilde{x},\tilde{u})\delta\hat{x} + \nabla_u F(\tilde{x},\tilde{u})\delta u \\ \delta\hat{y} &= \nabla_x G(\tilde{x},\tilde{u})\delta\hat{x} + \nabla_u G(\tilde{x},\tilde{u})\delta u \end{aligned} \tag{3.6}$$

Using this relationship, one can evaluate the stability of the nonlinear system. Specifically, the local bounded input bounded output (BIBO) stability of the nonlinear system in Equation 3.1 can be confirmed if the expression in Equation 3.6 exhibits exponential stability when $\delta u$ is zero. It's important to note that this assertion holds true only under the condition that the system in Equation 3.6 is robustly stable concerning the errors $\varepsilon_{\mathsf{F}}$ and $\varepsilon_{\mathsf{G}}$ [23].

As was mentioned, the system outlined by Equation 3.6 conforms to an LTV system. LTV systems exhibit different behavior at different times. In the realm of LTV systems, frozen-time theory proves useful for establishing requirements for exponential stability. Using frozen-time approximation, the trajectory is approximated by a single point that is reached during the trajectory. At this single point, a simple LTI system can be established information on the time-varying properties of the LTV system is lost during this process. The slower the LTV system changes, the smaller the effects of time variations, and the more it resembles the LTI approximation. If the trajectory reaches a singular equilibrium point ($x_1$, $u_1$) at time $t_1$, Equation 3.6 can be assessed and assumes the following form:

$$\begin{aligned} \delta\dot{\hat{x}} &= \nabla_x F(x_1,u_1)\delta\hat{x} + \nabla_u F(x_1,u_1)\delta u \\ \delta\hat{y} &= \nabla_x G(x_1,u_1)\delta\hat{x} + \nabla_u G(x_1,u_1)\delta u \end{aligned} \tag{3.7}$$

If all eigenvalues of $\nabla_x F(\tilde{x},\tilde{u})$ reside in the right half plane $\mathbb{C}^{-\epsilon} = s \in \mathbb{C}|Re(s) \leq -\epsilon$ for some $\epsilon < 0$, and the time-variations in the elements of $\nabla_x F(\tilde{x},\tilde{u})$ are limited, it can be demonstrated that the system achieves exponential stability. The values for $\nabla_x F(\tilde{x},\tilde{u})$ can be assessed by using $\nabla_x F(x_1,u_1)$ For precise definitions regarding the limitations on the state matrix elements, Ilchmann, Owens, and Prätzel-Wolters [15] can be consulted.

As of now, the connection between a single LTI system and the LTV system was elaborated upon, along with the applicable sufficient conditions for investigating this stability. This analysis can be broadened to encompass a family of LTI systems instead of a single LTI system [23]. The system can be evaluated at multiple equilibria points reached at any time $\tau$ that can be combined to express the LTV system as multiple frozen LTI systems:

$$\begin{aligned} \delta\dot{\hat{x}} &= \nabla_x F(\tilde{x}_\tau,\tilde{u}_\tau)\delta\hat{x} + \nabla_u F(\tilde{x}_\tau,\tilde{u}_\tau)\delta u \\ \delta\hat{y} &= \nabla_x G(\tilde{x}_\tau,\tilde{u}_\tau)\delta\hat{x} + \nabla_u G(\tilde{x}_\tau,\tilde{u}_\tau)\delta u \end{aligned} \tag{3.8}$$

It is important to note that the family of LTI systems itself is an LTV/LPV system but is different from the original LTV system it approximates. One is a collection of dynamical systems while the other is a distinct dynamical system. The LTV system of Equation 3.6, however, does inherit the stability and robustness properties of the family of LTI systems of Equation 3.8 given some conditions. That is, the LTV system is: "*locally BIBO stable in the vicinity of equilibrium operation provided that the members of its family of equilibrium linearizations are uniformly stable and the rate of variation is sufficiently slow.*" [25]. The slow variation condition ensures that the time variations of the elements of the state matrix $\nabla_x F(\tilde{x},\tilde{u})$ are limited (LTI to LTV congruence) but simultaneously also restricts the speed with which the system moves from one equilibrium point to the next equilibrium point to enforce $F(\tilde{x},\tilde{u}) \approx 0$ (LTV to nonlinear model congruence).

The assessment of the stability criteria of the nonlinear model thus contains 2 steps. First, the system is approximated by an LTV system that describes one or multiple trajectories using a Taylor approximation. The stability of the LTV system is subsequently evaluated by one or multiple LTI systems that are found by linearizing around the equilibria points.

### 3.1.2. Velocity-based linearization

Velocity-based linearization aims to relax the equilibrium conditions associated with conventional linearization. The system was proposed by Leith and Leithead [23][24] [26], The theoretical foundation is summarized in this section.

Let us first derive the problems with using Jacobian linearization by, again starting from any arbitrary nonlinear system:

$$\begin{aligned} \dot{x} &= F(x,u) \\ y &= G(x,u) \end{aligned} \tag{3.9}$$

Additionally, consider a trajectory $x(t), u(t)$ that attains the point $x_1, u_1$ at time $t_1$. When employing standard frozen-time series expansion, this yields [16] [24]:

$$\begin{aligned} \delta\dot{\hat{x}} &= F(x_1,u_1) + \nabla_x F(x_1,u_1)\delta\hat{x} + \nabla_u F(x_1,u_1)\delta u \\ \delta\hat{y} &= G(x_1,u_1) + \nabla_x G(x_1,u_1)\delta\hat{x} + \nabla_u G(x_1,u_1)\delta u \end{aligned} \tag{3.10}$$

where

$$\delta u = u - u_1, \quad \delta\hat{y} = \hat{y} - y_1, \quad \delta\hat{x} = \hat{x} - x_1, \quad \delta\dot{\hat{x}} = \dot{\hat{x}} \tag{3.11}$$

For this approximation to be accurate, it is necessary for $\delta\hat{x}$ and $\delta\hat{y}$ to remain small, ensuring that the approximation stays within the local region where it is valid. By substituting Equation 3.11 into Equation 3.10, the following equation can be derived:

$$\begin{aligned} \dot{\hat{x}} &= F(x_1,u_1) - \nabla_x F(x_1,u_1)x_1 - \nabla_u F(x_1,u_1)u_1 + \nabla_x F(x_1,u_1)\hat{x} + \nabla_u F(x_1,u_1)u \\ \hat{y} &= G(x_1,u_1) - \nabla_x G(x_1,u_1)x_1 - \nabla_u G(x_1,u_1)u_1 + \nabla_x G(x_1,u_1)\hat{x} + \nabla_u G(x_1,u_1)u \end{aligned} \tag{3.12}$$

Let us examine Equation 3.12 when approximating the derivative at the point $x_1$ used for linearization. In this case, $\dot{\hat{x}}_1$ becomes [23]:

$$\begin{aligned} \dot{\hat{x}}_1 &= F(x_1,u_1) - \nabla_x F(x_1,u_1)x_1 - \nabla_u F(x_1,u_1)u_1 + \nabla_x F(x_1,u_1)\hat{x}_1 + \nabla_u F(x_1,u_1)u_1 \\ &= F(x_1,u_1) + \nabla_x F(x_1,u_1)(\hat{x}_1 - x_1) + \nabla_u F(x_1,u_1)(u_1 - u_1) \\ &= F(x_1,u_1) + \nabla_x F(x_1,u_1)\delta\hat{x} \\ &= F(x_1,u_1) = \dot{x}_1 \\ \ddot{\hat{x}}_1 &= \nabla_x F(x_1,u_1)\,\dot{\hat{x}} + \nabla_u F(x_1,u_1)\,\dot{u}_1 = \ddot{x}_1 \end{aligned} \tag{3.13}$$

Therefore, it offers a first-order estimate of $\dot{x}$ and a second-order estimate of $x$ at each operational point. However, when solely equilibrium points are applied within the LTI systems family, it becomes evident that this characteristic may not consistently be true since $F(x_1,u_1)$ is assumed to be 0, a condition true only at an equilibrium point. Consequently, conventional Jacobian linearization essentially forms a zeroth-order approximation for points deviating from the equilibria.

As previously mentioned, the practice of employing series expansion around multiple operating points is common to encompass the entire operational range with a family of series expansions approximating the nonlinear model. It has been demonstrated [23, p. 309] that the approximation errors of a family of series expansions around equilibrium points (zeroth-order approximation) may not converge to zero, whereas first-order series do when increasing the gridding. This is intuitively logical since, regardless of how many points are added to the grid, no information about off-equilibrium points is incorporated, hence, the error at these points may not necessarily decrease. This presents a clear advantage over Jacobian linearization, particularly noticeable at initial conditions distant from equilibrium points or during rapid transients. It would thus be preferable to not assume $F(x_1,u_1) \approx 0$. However, the system, as depicted in Equation 3.10, then loses a critical property, namely linearity.

This can be resolved, however, by differentiating Equation 3.12 with respect to time, resulting in the following system, called the velocity-based linearization:

$$\begin{aligned} \dot{\hat{x}} &= \hat{w} \\ \dot{\hat{w}} &= \nabla_x F(x_1,u_1)\,\hat{w} + \nabla_u F(x_1,u_1)\,\dot{u} \\ \dot{\hat{y}} &= \nabla_x G(x_1,u_1)\,\hat{w} + \nabla_u G(x_1,u_1)\,\dot{u} \end{aligned} \tag{3.14}$$

With the appropriate initial conditions:

$$\hat{x}(t_1) = x(t_1), \quad \hat{w}(t_1) = \dot{\hat{x}}(t_1) = \dot{x}(t_1) = F(x_1,r_1), \quad \dot{\hat{y}}(t_1) = y(t_1) = G(x_1,r_1) \tag{3.15}$$

this system is dynamically equivalent to the system of Equation 3.12 but is linear. This system is of the same form as when differentiating the nonlinear system of Equation 3.9 directly which yields:

$$\begin{aligned} \dot{x} &= w \\ \dot{w} &= \nabla_x F\left(x,u\right) w + \nabla_u F\left(x,u\right) \dot{u} \\ \dot{y} &= \nabla_x G\left(x,u\right) w + \nabla_u G\left(x,u\right) \dot{u} \end{aligned} \tag{3.16}$$

The connection between Equation 3.16 and Equation 3.14 is straightforward. Equation 3.14 represents the frozen-form of Equation 3.16 at the specific point $x_1, u_1$. Figure 3.1 provides a summary illustrating how the velocity-based linearization family can be directly obtained from taking the derivative of the nonlinear model, or through the utilization of a first-order series expansion, followed by taking the derivative of this expression.

Velocity-based linearization can thus be a useful approach to linearize a nonlinear system at any arbitrary operating point. In conventional gain-scheduled or multi-model design, only Jacobian linearization is used and hence the plant can only be linearized at equilibria/trim points. The nonlinear plant is, of course, not limited to these trim points, which introduces the trim point uncertainty. This uncertainty can be quantified by linearizing the nonlinear system at the off-equilibrium points and comparing this to the equilibrium points, but this is further discussed in Section 3.4.

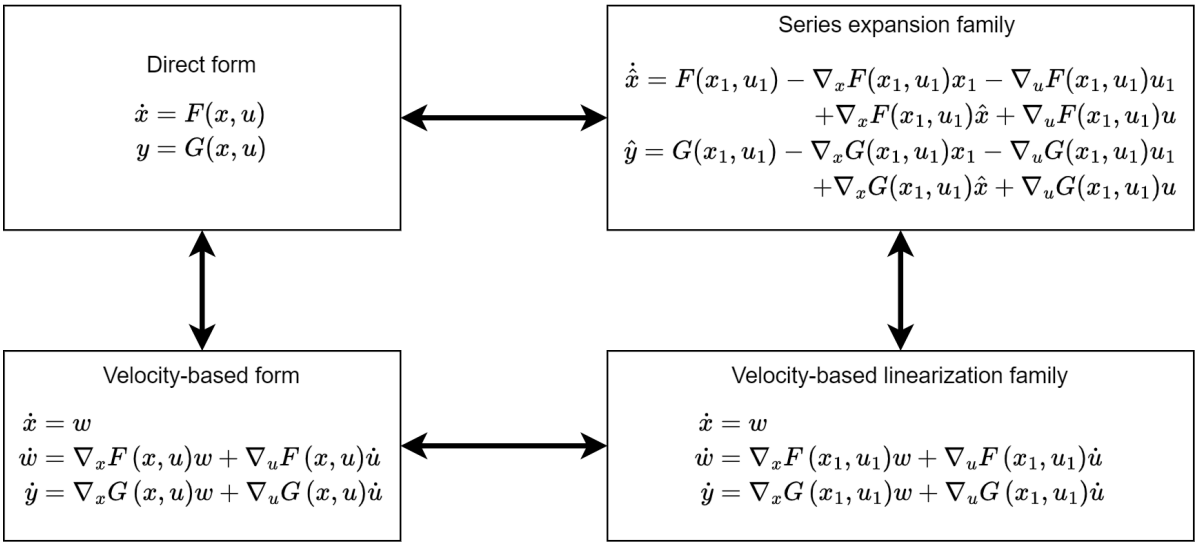



**Figure 3.1:** Velocity-based representations of nonlinear system [26].

## 3.2. Hidden coupling terms

The issue of hidden coupling terms arises when endogenous variables are used as scheduling variables, i.e. the scheduling variables are a state or output of the system. If this is the case, the gains vary in tandem with the system variables and cause the emergence of the hidden coupling terms [34] [37]. In essence, a feedback loop is introduced between the scheduled variables of the controller and the nonlinear system [21]. This extra feedback loop violates the *linearization property* [17]. This property is achieved if the closed-loop system of the controlled nonlinear plant exhibits the same input-output (linearization) properties as the closed-loop system of the linear plant and linear controller, at any given operating point.

To further understand the nature of the problem, the derivation of the coupling terms is summarized following the method of Lhachemi, Saussié, and Zhu [27] [28]. Let us start with any nonlinear system:

$$\begin{aligned} \dot{x} &= f(x,u,w) \\ y &= h(x,u,w) \end{aligned} \tag{3.17}$$

$x$ is the state vector, $u$ is the input vector, $w$ are the *exogonous* input signals that can be used for scheduling and $y$ contains the output signals available for control and/or scheduling.

Then, let us consider a gain-scheduled controller containing an integrator to regulate this plant. This controller has the form:

$$\begin{aligned} \dot{x}_i &= r - y \\ \dot{x}_c &= A_i(\theta)x_i + A_c(\theta)x_c + B_r(\theta)r + B_y(\theta)y \\ u &= C_i(\theta)x_i + C_c(\theta)x_c + D_r(\theta)r + D_y(\theta)y \\ \theta &= v(y, w) \end{aligned} \tag{3.18}$$

Here, the integrator term of the error is captured by $\dot{x}_i$. The states of the controller itself are contained by $x_c$ and the input signal is denoted by $r$. The state matrix $A(\theta)$ and output matrix $C(\theta)$ are split into two parts. $A_i(\theta)$ and $C_i(\theta)$ contain the state/output dynamics corresponding to the integrated error state and $A_c(\theta)$ and $C_c(\theta)$ contain the controller state/output equations. A similar partition is done for the input matrix $B(\theta)$ and the feedthrough matrix $D(\theta)$, albeit this time it is split with respect to the reference signals $r$ and the output signals $y$. Lastly, it is assumed that each point of the scheduling variables $\theta = v(y_{eq}, w_{eq})$ corresponds to a single equilibrium point and vice versa.

Note that when the closed-loop system is at an equilibrium point, the following holds for the controller:

$$\begin{aligned} 0 &= r - y \\ 0 &= A_i(\theta_{eq})x_{i,eq} + A_c(\theta_{eq})x_{c,eq} + B_r(\theta_{eq})r_{eq} + B_y(\theta_{eq})y_{eq} \\ u_{eq} &= C_i(\theta_{eq})x_{i,eq} + C_c(\theta_{eq})x_{c,eq} + D_r(\theta_{eq})r_{eq} + D_y(\theta_{eq})y_{eq} \\ \theta_{eq} &= v(y_{eq}, w_{eq}) \end{aligned} \tag{3.19}$$

These relations will be used later to construct the linearized controller at the equilibria points.

To derive the hidden coupling terms, first the controller from Equation 3.18 has to be linearized by employing a first-order series expansion at an equilibrium point. The steps of Jacobian linearization as mentioned in Section 3.1.1 are followed. Each individual term ($A_i(\theta)x_i$, $A_c(\theta)x_c$, ..., $D_y(\theta)y_e$) of the equation can be approximated using a first-order series expansion. As an example, let us start with the series expansion of the term $A_i(\theta)x_i$ at the equilibrium point $\theta_{eq}$ and $x_{i,eq}$:

$$A_i(\theta)x_i \approx A_i(\theta_{eq})x_{i,eq} + A_i(\theta_{eq})\delta x_i + \frac{\partial A_i(\theta)}{\partial \theta}\Big|_{(\theta_{eq})} \delta\theta \; x_{i,eq} \tag{3.20}$$

This equation is valid close the the equilibrium point $\theta_{eq}$ and $x_{i,eq}$. $\partial x_i$ denotes the perturbations from the equilibrium point, $\partial x_i = x_i - x_{i,eq}$ and this definition can be applied to all states, inputs, outputs and reference values:

$$\begin{aligned} \partial x_c &= x_c - x_{c,eq} \\ \partial r &= r - r_{eq} \\ \partial y &= y - y_{eq} \\ \partial u &= u - u_{eq} \\ \partial \theta &= \theta - \theta_{eq} \end{aligned}$$

Three terms appear in Equation 3.20, namely:

1. The equilibrium term: $A_i(\theta_{eq})x_{i,eq}$.
2. The perturbation term with respect to $x_i$, that is: $A_i(\theta_{eq})\delta x_i$.
3. The hidden coupling term: $\frac{\partial A_i}{\partial \theta}\big|_{(\theta_{eq})} \delta\theta \; x_{i,eq}$.

The equilibrium and perturbation terms appear as expected to achieve Jacobian linearization. However, the hidden coupling term appears because the series expansion has to be treated as multivariate, with respect to $\theta$ and $x_i$, not only with respect to $x_i$. The reason why this term is 'hidden' is that this term only appears in the closed-loop system. To elaborate, in the linear controller design, the scheduling

parameters are fixed, but once it is implemented as a controller in the closed-loop system, the scheduling parameters vary. If this value is large during operation, the behavior of the controller can behave differently than expected in the nonlinear domain.

This procedure can be applied to all other parts of the equation ($A_c(\theta)x_c$, $B_r(\theta)r$, ..., $D_y(\theta)y$) and added together. For the state equations, the sum of the equilibrium terms is equal to zero according to Equation 3.19, and for the output equations, they are equal to $u_{eq}$. The perturbation terms form the new linear system and the hidden coupling terms can be collected resulting in the expression of Equation 3.21:

$$\begin{aligned}
\delta\dot{x}_i &= \delta r - \delta\dot{y} \\
\delta\dot{x}_c &= A_i(\theta_{eq})\delta x_i + A_c(\theta_{eq})\delta x_c + B_c(\theta_{eq})\delta r + B_c(\theta_{eq})\delta y + B_\theta(\theta_{eq})\delta\theta \\
\delta u &= C_i(\theta_{eq})\delta x_i + C_c(\theta_{eq})\delta x_c + D_r(\theta_{eq})\delta r + D_y(\theta_{eq})\delta y + D_\theta(\theta_{eq})\delta\theta \\
\delta\theta &= \frac{\partial\nu}{\delta y}\Big|_{v^{-1}(\theta_{eq})}\delta y + \frac{\partial\nu}{\partial w}\Big|_{v^{-1}(\theta_{eq})}\delta w
\end{aligned} \tag{3.21}$$

With $\delta\dot{x}_i = \dot{x}_i - \dot{x}_{i,eq} = \dot{x}_i - 0 = \dot{x}_i$ and similarly $\delta\dot{x}_c = \dot{x}_c$. Furthermore, the hidden coupling terms are given by the matrices $B_\theta(\theta_{eq})$ and $D_\theta(\theta_{eq})$. These matrices collect the hidden coupling terms that came to appear as shown in the example of Equation 3.20. To simplify notation, if $\theta$ is a vector containing the scheduling variables, then $\theta_l$ refers to the $l$th term of the vector $\theta$. Subsequently the column $l$ of $B_\theta$ and $D_\theta$ can be defined as:

$$\begin{aligned}
B_{\theta,l}(\theta_{eq}) &= \frac{\partial A_i}{\partial\theta_l}\Big|_{(\theta_{eq})}x_{i,eq} + \frac{\partial A_i}{\partial\theta_l}\Big|_{(\theta_{eq})}x_{c,eq} + \frac{\partial B_i}{\partial\theta_l}\Big|_{(\theta_{eq})}r_{eq} + \frac{\partial B_i}{\partial\theta_l}\Big|_{(\theta_{eq})}y_{eq} \\
D_{\theta,l}(\theta_{eq}) &= \frac{\partial C_i}{\partial\theta_l}\Big|_{(\theta_{eq})}x_{i,eq} + \frac{\partial C_c}{\partial\theta_l}\Big|_{(\theta_{eq})}x_{c,eq} + \frac{\partial D_r}{\partial\theta_l}\Big|_{(\theta_{eq})}r_{eq} + \frac{\partial D_y}{\partial\theta_l}\Big|_{(\theta_{eq})}y_{eq}
\end{aligned} \tag{3.22}$$

This means that the $l$th column of the matrices $B_\theta(\theta_{eq})$ and $D_\theta(\theta_{eq})$ correspond to the hidden coupling term caused by scheduling variable $\theta_l$.

The extra hidden coupling terms violate the aforementioned *linearization property* that needs to hold at each equilibrium point to establish an agreement between the nonlinear and linear closed-loop systems. This thesis tries to assess the impact of the hidden coupling terms on a gain-scheduled controller. This is further discussed in Section 3.4.

In addition to the equilibria points, however, the off-equilibria points are also of interest to find the hidden coupling terms at any arbitrary operating point. To do this, the velocity-based linearization technique of Section 3.1.2 is used. Taking the derivative of Equation 3.18, then produces:

$$\begin{aligned}
\ddot{x}_i &= \dot{r} - \dot{y} \\
\ddot{x}_c &= A_i(\theta)\dot{x}_i + A_c(\theta)\dot{x}_c + B_r(\theta)\dot{r} + B_y(\theta)\dot{y} + B_\theta(\theta)\dot{\theta} \\
\ddot{u} &= C_i(\theta)\dot{x}_i + C_c(\theta)\dot{x}_c + D_r(\theta)\dot{r} + D_y(\theta)\dot{y} + D_\theta(\theta)\dot{\theta} \\
\dot{\theta} &= \frac{\partial\nu}{\delta y}\dot{y} + \frac{\partial\nu}{\partial w}\dot{w}_m
\end{aligned} \tag{3.23}$$

Here, the hidden coupling terms are defined as:

$$\begin{aligned}
B_{\theta,l}(\theta) &= \frac{\partial A_i}{\partial\theta_l}x_i + \frac{\partial A_i}{\partial\theta_l}x_c + \frac{\partial B_i}{\partial\theta_l}r + \frac{\partial B_i}{\partial\theta_l}y \\
D_{\theta,l}(\theta) &= \frac{\partial C_i}{\partial\theta_l}x_i + \frac{\partial C_c}{\partial\theta_l}x_c + \frac{\partial D_r}{\partial\theta_l}r + \frac{\partial D_y}{\partial\theta_l}y
\end{aligned} \tag{3.24}$$

Now the hidden coupling terms appear because of the product rule when taking a derivative. To elaborate on how this happens and simultaneously explain how Equation 3.23 and Equation 3.24 were

constructed, let us again take an example by taking the derivative of the term $A_i(\theta)x_i$, resulting in:

$$\begin{aligned}\frac{d(A_i(\theta)x_i)}{dt} &= A_i(\theta)\frac{dx_i}{dt} + \frac{d(A_i(\theta))}{dt}x_i \\ &= A_i(\theta)\frac{dx_i}{dt} + \frac{\partial(A_i(\theta))}{\partial\theta}\frac{d\theta}{dt}x_i \\ &= A_i(\theta)\dot{x}_i + \frac{\partial A_i(\theta)}{\partial\theta}\dot{\theta}\ x_i\end{aligned} \tag{3.25}$$

The same procedure can be applied to the other terms resulting in Equation 3.23 with the hidden coupling terms being collected in Equation 3.24. This procedure is similar compared to applying the multivariate series expansion as in Equation 3.20.

This is not yet a linear approximation at a point but a reformulation of the original LPV controller of Equation 3.18. According to the theory of velocity-based linearization of Section 3.1.2, Equation 3.23 is still equivalent to Equation 3.18 if the initial conditions are the same as it is simply a derivative of the original system [24]. It becomes an LTI system when frozen-time theory is applied and values of a specific operating point (including the values of the scheduling parameters) are inserted:

$$\begin{aligned}\ddot{x}_i &= \dot{r} - \dot{y} \\ \ddot{x}_c &= A_i(\theta_{op})\dot{x}_i + A_c(\theta_{op})\dot{x}_c + B_r(\theta_{op})\dot{r} + B_y(\theta_{op})\dot{y} + B_\theta(\theta_{op})\dot{\theta} \\ \ddot{u} &= C_i(\theta_{op})\dot{x}_i + C_c(\theta_{op})\dot{x}_c + D_r(\theta_{op})\dot{r} + D_y(\theta_{op})\dot{y} + D_\theta(\theta_{op})\dot{\theta} \\ \dot{\theta} &= \frac{\partial\nu}{\delta y}\Big|_{v^{-1}(\theta_{op})}\dot{y} + \frac{\partial\nu}{\partial w}\Big|_{v^{-1}(\theta_{op})}\dot{w}_m\end{aligned} \tag{3.26}$$

with,

$$\begin{aligned}B_{\theta,l}(\theta_{op}) &= \frac{\partial A_i}{\partial\theta_l}\Big|_{(\theta_{op})}x_{i,op} + \frac{\partial A_i}{\partial\theta_l}\Big|_{(\theta_{op})}x_{c,op} + \frac{\partial B_i}{\partial\theta_l}\Big|_{(\theta_{op})}r_{op} + \frac{\partial B_i}{\partial\theta_l}\Big|_{(\theta_{op})}y_{op} \\ D_{\theta,l}(\theta_{op}) &= \frac{\partial C_i}{\partial\theta_l}\Big|_{(\theta_{op})}x_{i,op} + \frac{\partial C_c}{\partial\theta_l}\Big|_{(\theta_{op})}x_{c,op} + \frac{\partial D_r}{\partial\theta_l}\Big|_{(\theta_{op})}r_{op} + \frac{\partial D_y}{\partial\theta_l}\Big|_{(\theta_{op})}y_{op}\end{aligned} \tag{3.27}$$

As can be seen, the values of the state space matrices are now constant as they are only dependent on the (now fixed) values of the scheduling parameters. Furthermore, the hidden coupling terms are also defined as they are dependent on the values of the scheduling parameters and the operating points, which now are both fixed. This means that the system is now a collection of linear differential equations in terms of the derivatives of the original states ($\dot{x}_i$ and $\dot{x}_c$), inputs ($\dot{u}$), reference values ($\dot{r}$) and measurement values ($\dot{y}$). This linearization method has the advantage that it also allows quantification of the hidden coupling terms at off-equilibria points. The quantification method is further discussed in Section 3.4.

## 3.3. Stability of time-varying systems

Both linearization methods of Section 3.1 use frozen-time theory to obtain a linear time-invariant system. However, the original approximated system is often time-varying or parameter-varying instead. That is, it has the form:

$$\begin{aligned}\dot{x}(t) &= A(\theta(t))\ x(t) + B(\theta(t))\ u(t) \\ y(t) &= C(\theta(t))x(t)\end{aligned} \tag{3.28}$$

where $\theta(t)$ is a parameter varying trough time. This discrepancy between the frozen form used to approximate the LTI system and the overall time-varying closed-loop system can destabilize the system [43][49], in even the simplest case of nominal stability. This problem was also referred to when discussing linearization in Section 3.1.1.

To establish this problem an example [1][42] is looked at in more detail. Consider the following system:

$$\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} -a\sin(2t) & -a\cos(2t) \\ -a\cos(2t) & a\sin(2t) \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \tag{3.29}$$

The eigenvalues of the system are the following:

$$\lambda_{1,2} = \frac{a - 2 \pm \sqrt{a^2 - 4}}{2} \tag{3.30}$$

The eigenvalues are constant and lie in the left-half complex plane given $a < 2$, implying that the family of frozen form systems derived from the dynamical system is stable for these values. However, if the state transition matrix is investigated:

$$\Phi(t,0) = \begin{bmatrix} e^{(a-1)t}\sin(t) & e^{-t}\sin(t) \\ -e^{(a-1)t}\sin(t) & e^{-t}\cos(t) \end{bmatrix} \tag{3.31}$$

It can be seen that for $1 < a < 2$, the dynamics are unstable, contradicting the frozen-form result.

It is possible to guarantee exponential stability of the time-varying system using frozen-time stability, given that the rate of variations is sufficiently slow [42]. Calculating the maximum rate of variation is quite complex (see [15]) for the F-16 aircraft model and is deemed out of the scope of this thesis. Nonetheless, incorporating the other two effects in the frozen-time robustness already extends the local stability guarantees and can be used for nonlinear analysis in order to rule out the other two problems.

## 3.4. Quantifying frozen-time robustness using the normalized coprime stability margin

This section looks at quantifying the robustness penalty introduced by trim point uncertainty and the hidden coupling terms. This is achieved using the coprime stability margin and the gap metric [52].

To understand the coprime stability margin, first coprime uncertainty has to be understood. This type of uncertainty is particularly useful for feedback system analysis [11]. Subsequently, to understand coprime uncertainty, first, the coprime factorization of the plant has to be analyzed. In the SISO case, it is possible to decompose the plant $p$ into $p = nm^{-1}$. Here, $n$ and $m$ are said to be stable coprime transfer functions, which means that there exists a stable transfer function $x$ and $y$ such that:

$$xm + yn = 1 \tag{3.32}$$

This decomposition ensures that $n$ contains all the right half-plane zeros of the plant $p$ while $m$ contains all the right half-plane poles [4, p. 51]. To extend this definition to the MIMO case, it must be taken into account that multiplication from the left is different than multiplication from the right. Thus two transfer function matrices $M$ and $N$ are said to be *right* coprime if there exist stable transfer function matrices $X_r$ and $Y_r$ such that [53, p. 72]:

$$\begin{bmatrix} X_r & Y_r \end{bmatrix} \begin{bmatrix} M \\ N \end{bmatrix} = X_r M + Y_r N = I \tag{3.33}$$

On the other hand, they are said to be *left* coprime if:

$$\begin{bmatrix} \tilde{M} & \tilde{N} \end{bmatrix} \begin{bmatrix} X_l \\ Y_l \end{bmatrix} = \tilde{M} X_l + \tilde{N} Y_l = I \tag{3.34}$$

With these definitions, the *right* coprime factorization of $P$ is achieved by $P = NM^{-1}$, while the left coprime factorization is $P = \tilde{M}^{-1}\tilde{N}$. There are infinitely many solutions to Equation 3.33 and Equation 3.34, thus often the normalized coprime factorization is used to additionally constrain $N$ and $M$ or $\tilde{N}$ and $\tilde{M}$ such that:

$$\begin{aligned} \text{Right coprime: } & N^T N + M^T M = I \\ \text{Left coprime: } & \tilde{N}^T \tilde{N} + \tilde{M}^T \tilde{M} = I \end{aligned} \tag{3.35}$$

The (normalized) coprime factorization of the plant can be used to impose perturbations on the separate matrices $\tilde{M}^{-1}$ and $\tilde{N}$. The left coprime factor perturbed plant is displayed in Figure 3.2, where $\tilde{\Delta}_N$ denotes the perturbations on $\tilde{N}$ and $\tilde{\Delta}_M$ the perturbations on $\tilde{M}$. Furthermore, $\Delta$ can be defined as:

$\Delta := [\tilde{\Delta}_N \ \tilde{\Delta}_M]$. With these definitions, the closed-loop system is well-posed and internally stable for all $\|\Delta\|_\infty \leq \epsilon$ if and only if [53, p. 315]:

$$\left\| \begin{bmatrix} K \\ I \end{bmatrix} (I + PK)^{-1} \tilde{M}^{-1}] \right\|_\infty \leq 1/\epsilon \tag{3.36}$$

One advantage of using coprime factor uncertainty is that it can describe uncertainty in the locations of the poles and zeros well. For example, it can be used to represent a situation where uncertainty can cause the system to become unstable. This is in contrast to additive and multiplicative uncertainty which cannot change the number of closed right half plane poles compared to the nominal plant [4].

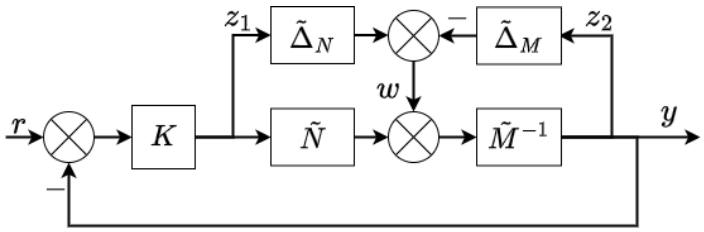



**Figure 3.2:** Left coprime factorization with indicated uncertainty blocks [53, p 316].

Subsequently, it is possible to define the normalized coprime stability margin $b(P, K)$ as $\epsilon$ of Equation 3.36. This results in an expression between zero and one that indicates robustness with respect to the $\Delta$ perturbations, the higher the normalized coprime stability margin the higher the gain bounded $\Delta$ perturbations can become before stability is lost. For the calculation of the coprime stability margin the following equivalency can be used [53, p. 321]:

$$b(P,K) = \left\| \begin{bmatrix} K \\ I \end{bmatrix} (I + PK)^{-1} \tilde{M}^{-1}] \right\|_\infty^{-1} = \left\| \begin{bmatrix} I \\ K \end{bmatrix} (I + PK)^{-1} \begin{bmatrix} I & P \end{bmatrix} \right\|_\infty^{-1} \tag{3.37}$$

The normalized coprime stability margin thus indicates stability with respect to these gain-bounded frequency-dependent uncertainties of $\Delta$. It turns out that any calculated normalized coprime stability margin using the left coprime factorization is equivalent to when using the right coprime factorization [53, p. 323].

The reason why the coprime stability margin is used in the thesis, is twofold. First, the normalized coprime stability margin in itself is a measure of robustness against the aforementioned uncertainties [31] and is related to the multi-loop simultaneous stability margins [12]. A reduction of the coprime stability margin thus indicates a reduction in robustness. This concept can be applied to an off-equilibrium plant with a controller containing hidden coupling terms.

To elaborate, let us consider a closed-loop system for which a gain-scheduled controller $K(\theta)$, which uses real-time interpolation of $K_1, K_2, ...K_n$, was designed using multiple linearized plants ($P_1, P_2, ..., P_n$). The controller $K_1$ was designed at plant $P_1$, controller $K_2$ at plant $P_2$, etc. Once operating in the nonlinear domain, the actual controller at a (possibly off-equilibrium) operating point will be slightly different than any of $K_1, K_2, ...K_n$. namely of the form $K_{off}$. This is caused by the effects of the hidden coupling terms and because at the off-equilibrium point, the controller is constructed using the interpolation of the closest controllers. $K_{off}$ can be found using velocity-based linearization on the controller $K(\theta)$ as seen in Section 3.2.

In addition, the closest linearized plant that was used in the design is not the same as the one that is encountered at an operating point due to the trim point uncertainty. At this operating point, the off-equilibrium plant can be calculated using velocity-based linearization established in Section 3.1.2, resulting in plant $P_{off}$. Subsequently, it is possible to calculate the normalized coprime stability margin at this operating point $b(P_{off}, K_{off})$ by making use of Equation 3.37. Furthermore, it can be compared to the coprime stability margin of the linear controller plant combination at its closest equilibrium point. The distance of a plant to another plant is calculated using the gap metric (the gap metric is elaborated

upon in the next paragraph). If for example, plant $3$ is the closest equilibrium plant used in the linear design, the coprime stability margin $b(P_3, K_3)$ can be calculated. To generalize the notation, if the closest equilibrium plant $P_{eq}$ and its corresponding controller $K_{eq}$ are found, the stability margin $b(P_{eq}, K_{eq})$ can be calculated. This value can be compared and the measured degradation is caused by the two factors. Note, that as long as $b(P_{off}, K_{off}) > 0$, the system is (frozen-time) stable [53] at this point.

The second reason for the suitability of the normalized coprime stability margin is its relation to the gap metric. The gap metric measures the distance between two systems, in terms of their closed-loop behavior on a scale from $0$ to $1$ (for the precise definitions and computation see [53, p 350]). This will allow us to capture the effects of the hidden coupling terms and trim point uncertainty in itself. Let us consider the gap metric between an off-equilibrium plant, $P_{off}$, and its closest plant used during linear design, denoted as $P_{eq}$, resulting in $\delta(P_{eq}, P_{off})$. Furthermore, let us consider the controller corresponding to the off-equilibrium plant including hidden coupling terms, denoted by $K_{off}$. Thereafter, $K_{off}$ can be compared to the controller corresponding to the closest plant $K_{eq}$, with the hidden-coupling terms of the controller set to $0$. This comparison can be performed using the gap metric resulting in $\delta(K_{eq}, K_{off})$. Then, these gap metrics are related to the minimum coprime stability margins that take both factors into account at any (possibly off-equilibrium) operating point.

The general formula that relates the coprime stability margin of a controller plant combination $(P_1, K_1)$ to another controller plant combination $(P_2, K_2)$ and the gap metrics between the controllers and the plants is given by [53, p 367]:

$$\arcsin(b(P_2, K_2)) \geq \arcsin(b(P_1, K_1)) - \arcsin(\delta(P_1, P_2)) - \arcsin(\delta(K_1, K_2)) \tag{3.38}$$

This equation can be applied to our situation. If $b(P_{eq}, K_{eq})$ denotes the coprime stability margin corresponding to the closest equilibrium point to the off-equilibrium operating point, the gap metrics can be used to compute the minimum coprime stability margin found at the off-equilibrium point, $b(P_{off}, K_{off})$:

$$\arcsin(b(P_{off}, K_{off})) \geq \arcsin(b(P_{eq}, K_{eq})) - \arcsin(\delta(P_{off}, P_{eq})) - \arcsin(\delta(K_{off}, K_{eq})) \tag{3.39}$$

Thus, the gap metric offers a rather straightforward method to assess the effects of the trim point uncertainty and hidden coupling in itself and the inequality enables interpretation of the gap metric with respect to the coprime stability margin. If the main interest is the absolute value of the normalized coprime stability margin, Equation 3.37 could directly be used to calculate $b(P_{off}, K_{off})$.

Using the normalized coprime stability margin, it is possible to guarantee stability and assess robustness against unstructured uncertainty at any arbitrary operating point in the presence of hidden coupling terms and trim point uncertainty. This stability is referred to in this thesis as frozen-time stability as the inherent time variations of a system mentioned in Section 3.3 are not included.

The upcoming design procedures of the thesis will incorporate the aforementioned theory and thus look as follows. First, The plant is linearized at multiple operating points. Thereafter, linear control design is employed to either construct a controller at each of the points and interpolate later (gain scheduling), or the controller is synthesized using all plants simultaneously (multi-model synthesis). However, thereafter, the a posteriori nonlinear analysis can be done more rigorously. The hidden coupling terms of the controller are to be derived and the velocity-form of the plant is constructed. Subsequently, it is possible to look at typical flight maneuvers and investigate the coprime stability margin and/or the gap metrics along the trajectory of these maneuvers. Furthermore, in contrast to conventional design, local stability can be guaranteed at each operating point. Before this is applied to the flight control systems, Section 3.5 gives an example of the a posteriori analysis using a controller design and analysis example.

## 3.5. Controller analysis example

To show an application of the analysis of trim point uncertainty and hidden coupling terms, a straightforward example using gain scheduling is shown in this section. Let us consider the following plant:

$$\begin{aligned} \dot{x} &= -x + 2\sin(u) \\ y &= \tanh(x) \end{aligned} \tag{3.40}$$

The first step is to linearize the plant around a set of operating points. To achieve this, the plant is transformed to the velocity-form:

$$\begin{aligned} \ddot{x} &= -\dot{x} + 2\cos(u)\dot{u} \\ \dot{y} &= \frac{1}{\cosh(x)^2}\dot{x} \end{aligned} \tag{3.41}$$

To complete the linearization, any operating point $(u_{op},\ x_{op})$ can be inserted:

$$\begin{aligned} \ddot{x} &= -\dot{x} + 2\cos(u_{op})\dot{u} \\ \dot{y} &= \frac{1}{\cosh(x_{op})^2}\dot{x} \end{aligned} \tag{3.42}$$

As can be seen, this system is now linear if the new state is defined as $\dot{x}$ and the new input as $\dot{u}$. These are the derivatives of the original state and input.

The set of operating points used for the linear controller synthesis is the set equilibrium points given by $y_{eq} = \{-0.95,\ -0.9,\ -0.85,\ \ldots,\ 0.9,\ 0.95\}$. From Equation 3.40 it can be inferred that to achieve an equilibrium condition, it needs to hold that:

$$\begin{aligned} x_{eq} &= \operatorname{arctanh}(y_{eq}) \\ -x_{eq} + 2\sin(u_{eq}) = 0 \Rightarrow u_{eq} &= \frac{\arcsin(x_{eq})}{2} \end{aligned} \tag{3.43}$$

Then for the second step, at these equilibria points, a linear controller can be designed. For the sake of this example, a simple PI controller is used. The structure can be seen in Figure 3.3.

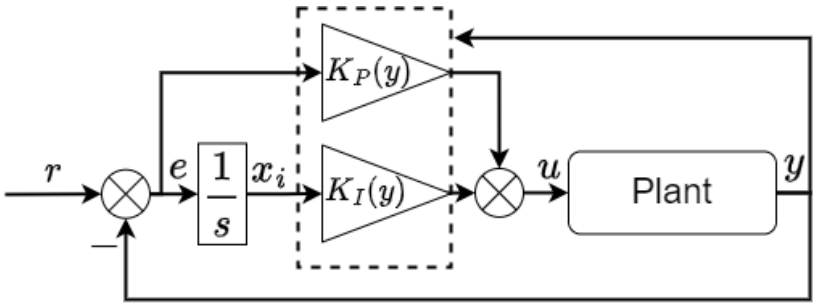
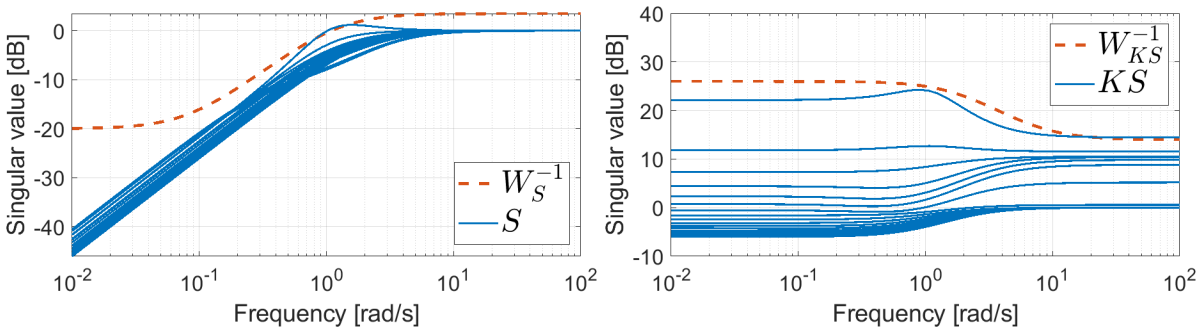



**Figure 3.3:** Controller structure and realization used for the example.

For the controller synthesis, a simple mixed-sensitivity $\mathcal{H}_\infty$ design is used to calculate the gains. Figure 3.4 shows the closed-loop sensitivity and control effort transfer functions at all linearized equilibria points. Furthermore, the imposed weighting functions for this optimization function are indicated. The corresponding linear step responses can be seen in Figure 3.5. The sensitivity function is constrained at low frequencies while the controller signal is constrained (relatively little) at high frequencies. This has been done arbitrarily as synthesizing the controller in itself, is not the goal of the example.




**(a)** Sensitivity function with its weighting function.

**(b)** Controller signal attenuation function with its weighting function.

**Figure 3.4:** S/KS linear design results.

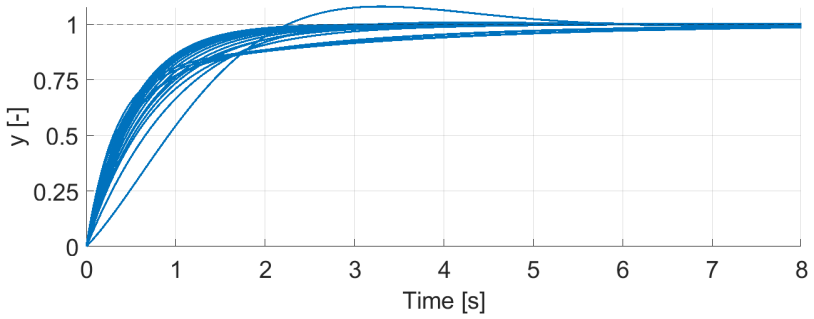



**Figure 3.5:** Linear step response

Thereafter, for the third step, the controllers need to be interpolated in some manner. In the example, simple linear interpolation is used. The $P$ and $I$ values of the controller across the plant envelope can be seen in Figure 3.6.

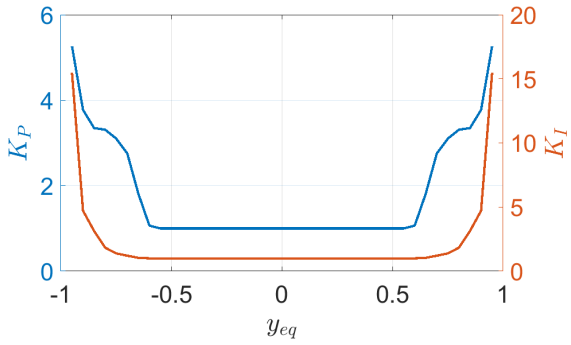



**Figure 3.6:** PI controller values versus equilibrium point.

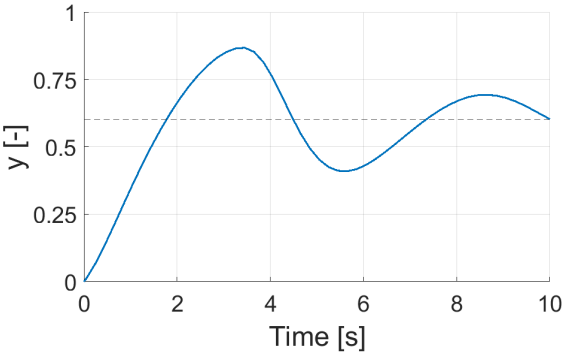



**Figure 3.7:** Nonlinear step response.

Now, the last step concerns the (nonlinear) analysis. To start with, the nonlinear step response (with a magnitude of $0.6$) is considered. This is shown in Figure 3.7. As can be inferred this does not correspond well to the linear simulation as shown in Figure 3.5.

To investigate the problem encountered in the nonlinear domain, the trim-point uncertainty and the hidden coupling terms are quantified using the gap metric. The linearized plant at any off-equilibrium point encountered in the trajectory can be calculated using the velocity-based linearization of the plant that was shown in Equation 3.42, resulting in $P_{off}$. Subsequently, the closest equilibrium point can be found and the linearized plant at that equilibrium point is calculated, resulting in $P_{eq}$. Finally, the gap metric between the two plants can be computed, resulting in $\delta(P_{eq}, P_{off})$. This can be done for every discrete point in the trajectory.

Next, the hidden coupling terms of the controller can be derived. First, consider the controller diagram of Figure 3.3. The system of equations describing this diagram equals:

$$\begin{aligned} \dot{x_i} &= e \\ u &= K_P(y)e + K_I(y)x_i \end{aligned} \tag{3.44}$$

The velocity-form can be calculated by taking the time derivative of Equation 3.44

$$\begin{aligned} \ddot{x_i} &= \dot{e} \\ \dot{u} &= K_P(y)\dot{e} + K_I(y)\dot{x}_i + \frac{\partial K_P(y)}{\partial y}\dot{y}e + \frac{\partial K_I(y)}{\partial y}x_i\dot{y} \end{aligned} \tag{3.45}$$

The latter two terms of the output equation are the hidden coupling terms. To complete the linearization, the values of an operating point $(e_{op}, x_{i,op}, y_{op})$ can be inserted. Furthermore, numerical derivatives of

the gain surfaces at an operating point are known, resulting in:

$$\begin{aligned} \ddot{x}_i &= \dot{e} \\ \dot{u} &= K_P(y)\dot{e} + K_I(y)\dot{x}_i + \left.\frac{\partial K_P(y)}{\partial y}\right|_{y_{op}} \dot{y}e_{op} + \left.\frac{\partial K_I(y)}{\partial y}\right|_{y_{op}} x_{i,op}\dot{y} \end{aligned} \tag{3.46}$$

Hence the linearized controller including hidden coupling terms at an off-equilibrium operating point (denoted by $K_{off}$) can be compared to the controller designed at the closest equilibrium point with the hidden coupling terms set to $0$ (denoted by $K_{eq}$) by using the gap metric. Note that an extra input has to be added to perform this comparison, namely the derivative of the scheduling parameter $\dot{y}$. This results in $\delta(K_{eq}, K_{off})$.

Now, the gap metrics of both factors can be calculated at every discretized point of the trajectory. The result can be seen in Figure 3.8. Furthermore, using Equation 3.39 the minimum coprime stability margin along the trajectory can also be calculated, the result of which is shown in Figure 3.9. It can be deduced that frozen-time stability cannot be guaranteed as the minimum coprime stability margin becomes smaller than $0$. Furthermore, from the gap metrics, it can be inferred that the main cause of the degradation of the frozen-time stability is due to the hidden coupling terms as its gap metric reaches far higher values compared to the trim-point uncertainty. This could prompt a closer investigation into the controller realization. The varying gain was placed after the integrator which causes the extreme hidden coupling action and the problem can be alleviated by putting the varying gain in front of the integrator.

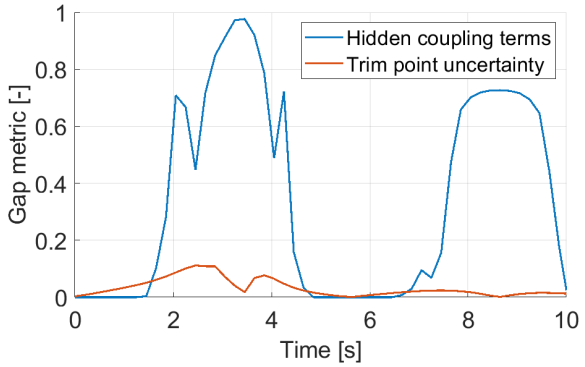



**Figure 3.8:** Gap metrics due to hidden coupling and trim-point uncertainty.

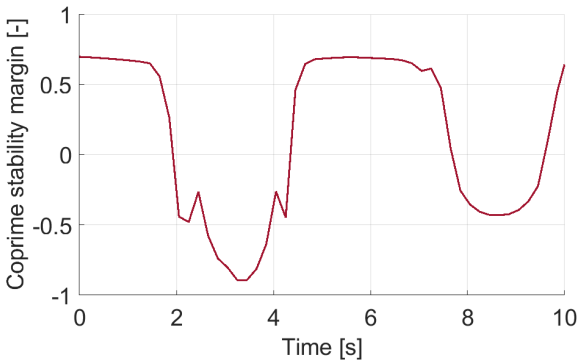



**Figure 3.9:** Minimum coprime stability margin along discretized trajectory.

Nonetheless, the purpose of the example is not to construct the perfect controller for this problem but to clearly show that the gap metric and/or (minimum) coprime stability margin can be used for a posteriori nonlinear analysis. Furthermore, it can extend the local stability guarantees of the linear controller design to include the two effects. In the following two chapters, the theory is applied to two practical flight control systems.

# 4

# Pitch-rate control

The first application to be considered covers a pitch-rate control system using gain-scheduling and $\mathcal{H}_\infty$ synthesis. To simplify the design process the short-period approximation is used for the synthesis. The inputs of the controller are the pitch-rate error and the angle of attack. The elevator is used as the actuator to control this signal, as it is the only longitudinal actuator that has sufficient bandwidth to control this relatively fast control problem. To enable the linear design methodology, the nonlinear plant has been linearized using conventional Jacobian linearization from Section 3.1.1. Only equilibria points were considered. Figure 4.1 shows the points in the flight envelope that are used for the design.

The structure of this chapter is as follows. First, the short-period approximation is briefly discussed in Section 4.1. Subsequently, the controller structure and design requirements are formalized and the resulting gain surfaces are shown in Section 4.2. Moreover, additional linear and robust analysis is included in Section 4.3. Lastly, the nonlinear analysis is performed in Section 4.4.

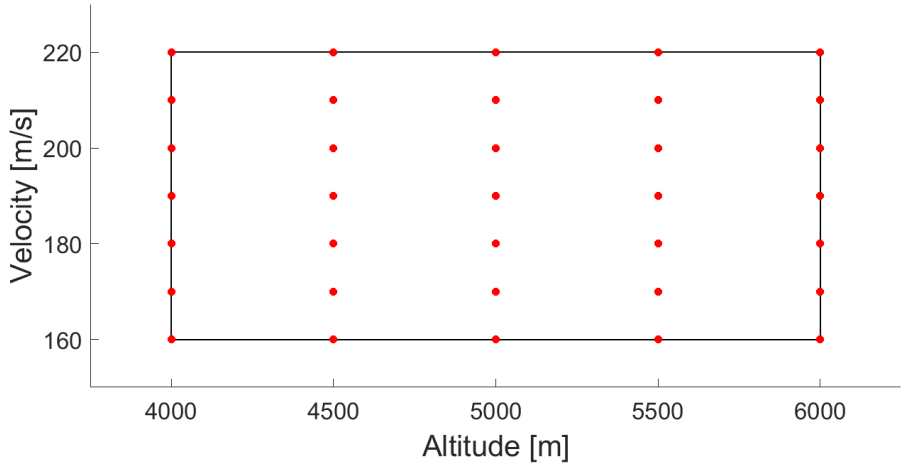
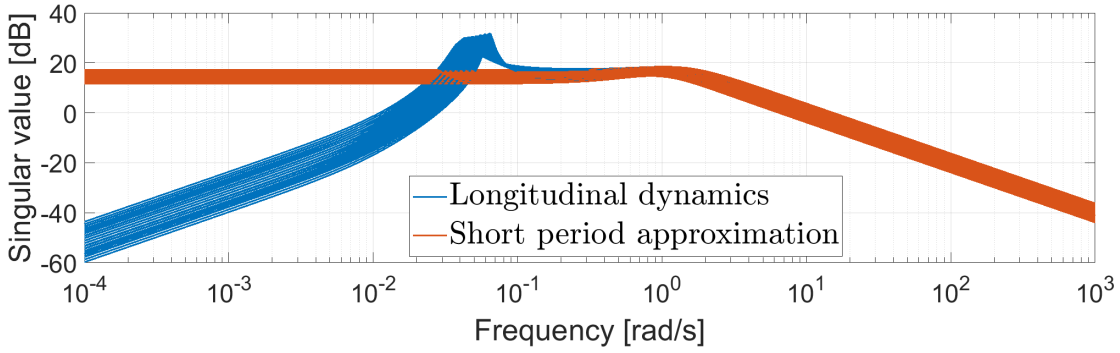



**Figure 4.1:** Linearized points in flight envelope

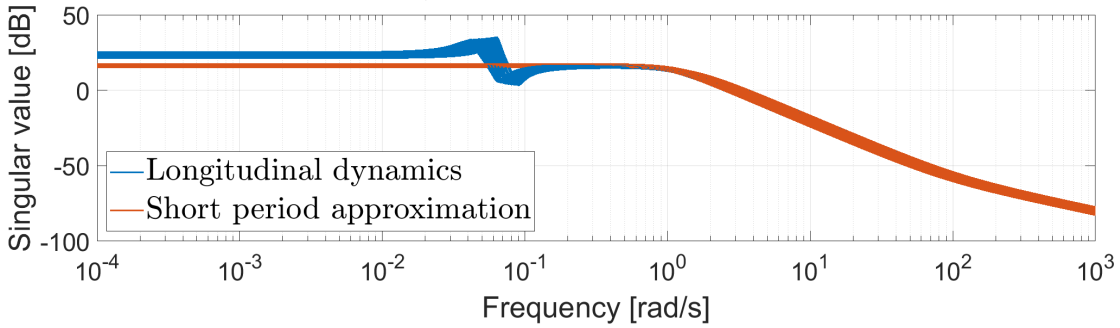
## 4.1. Short-period approximation

To simplify the design of the pitch-rate controller an approximated form of the true dynamics is used. The first reduction of the system is performed by only considering the longitudinal modes ($V$, $\theta$, $\alpha$, and $q$). Subsequently, a second simplification is used to obtain a single input system. For a pitch-rate controller, the fast states are dominant and can be approximated by dropping the low-frequency states associated with the phugoid ($V$ and $\theta$) while retaining the fast states ($\alpha$ and $q$). The elevator is the preferred actuator to control the short period as it is a relatively fast actuator. This yields the well-known short-period approximation [46]. This is compared to the full longitudinal system in Figure 4.2. As can be seen, the approximation is accurate in the high-frequency range but loses accuracy at the

lower frequencies. This approximation can thus be used for the pitch-rate controller as that controller is only concerned with short timescales. A full longitudinal controller, also including longer timescales, is designed in Chapter 5.




**(a)** Singular values of transfer function $\delta_e \rightarrow q$.




**(b)** Singular values of transfer function $\delta_e \rightarrow \alpha$.

**Figure 4.2:** Validity of short period approximation.

## 4.2. Structure, constraints and synthesis

This section discusses the structure of the controller, the imposed constraint with the resulting closed-loop characteristics at each flight point, and ultimately the final obtained gain surfaces. Similar design constraints as done by Theodoulis and Proff [48] are used.

### 4.2.1. Controller structure

The structure of the controller and I/O layout can be seen in Figure 4.3. The goal is to drive the error in pitch-rate $q_e$ to zero. Moreover, the controller has to be robust to input disturbance $d_{\delta_e}$ and output disturbances $d_q$ and $d_\alpha$. The sensors are assumed to be perfect but rejection to high-frequency measurement noise, introduced by $n_q$ and $n_\alpha$, is checked. This is mostly important for the angle of attack measurement as this is known to be a noisy signal.

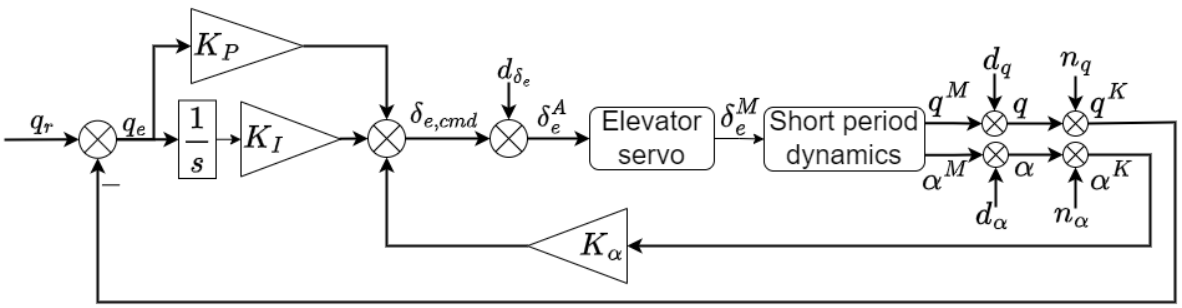



**Figure 4.3:** Pitch-rate control design I/O layout.

The tracking is performed by a simple PI controller to ensure no steady-state error. Furthermore, an extra loop is introduced by feedback on the angle of attack to improve the damping and transient response. This is a classical structure for pitch-rate controllers and has been used many times [13] [46]. This structure gives us three degrees of freedom in the parameter space, namely $K_P$, $K_I$, and $K_\alpha$.

### 4.2.2. Hard constraints

The hard constraints are inspired by classical mixed-sensitivity [4] design but incorporate an additional tracking constraint in the time domain and a constraint on the pole locations on the imaginary plane, also known as $\mathcal{D}$-stability. Four hard constraints are considered:

1. Reference tracking
2. Output disturbance rejection
3. Control signal attenuation
4. $\mathcal{D}$-stability

Sensor noise attenuation is additionally considered as a soft constraint.

#### Reference tracking

This constraint is added to ensure a reference trajectory is followed. It is possible to enforce this constraint by limiting the $\mathcal{H}_\infty$ norm of the weighted reference error signal to less than $1$ [48]. The inverted weighting function could then ensure low gain attenuation in the low-frequency region, which translates to a low steady-state error.

However, a more natural method to ensure reference tracking can be done in the time domain. In that case, the $\mathcal{H}_2$ norm (root mean squared error between signals) is to be a constraint by a reference model [48]. Given a reference model $T_{ref}$ and the actual closed-loop transfer function $T_m(K)$, the constraint is as follows:

$$H_{T_{ref}} = \frac{\left|\left|\frac{1}{s}T_m(K) - T_{ref}\right|\right|_2}{e_{rel}\left|\left|\frac{1}{s}\left(T_{ref} - 1\right)\right|\right|_2} \leq 1 \tag{4.1}$$

where $e_{rel}$ can be specified to enforce a maximum percentual matching error between the signals. Here the default value of $e_{rel} = 0.1$ was used.

The result of this constraint is shown in Figure 4.4. The rise time and maximum overshoot are sufficient. There is an additional underdamped pair of complex poles that causes oscillating behavior before complete settling is achieved, however, the deviation from the value to be tracked is limited.

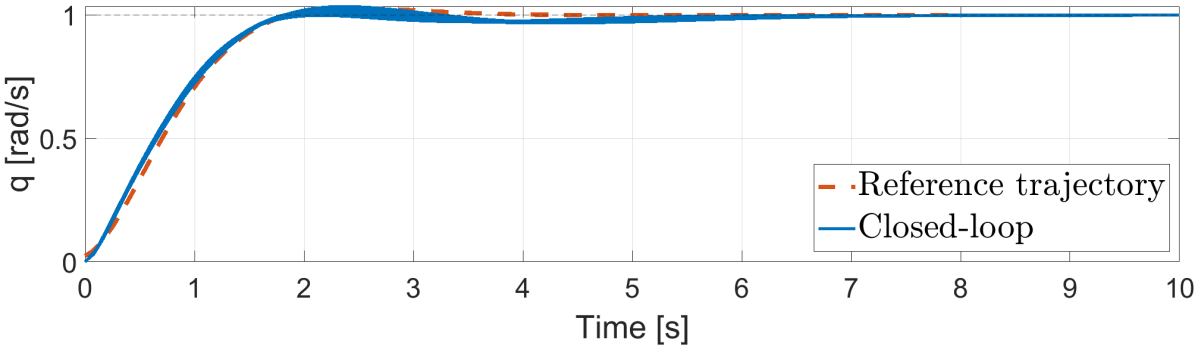



**Figure 4.4:** Output and reference step response.

#### Output disturbance rejection

This constraint concerns the rejection of plant output with respect to $q$. That is $S_o$ is the transfer function from $d_q$ to $q$. The low-frequency gain must be reduced such that disturbances in this range are rejected. The constraint is denoted in Equation 4.2 and makes use of the weighting function $W_{S_o}$:

$$H_{S_o} = ||W_{S_o}S_o(K)||_\infty \leq 1 \tag{4.2}$$

The inverted weighting function $W_{S_o}$ is chosen such that the low-frequency attenuation converges to $-60$ dB, reaches the value of $0$ dB at $1.5$ rad/s and subsequently constraints the high-frequency gain at $6$ dB. The bode plot of the inverted weighting function can be viewed in Figure 4.5. Furthermore, the figure shows the resulting output sensitivity function of the closed-loop systems.

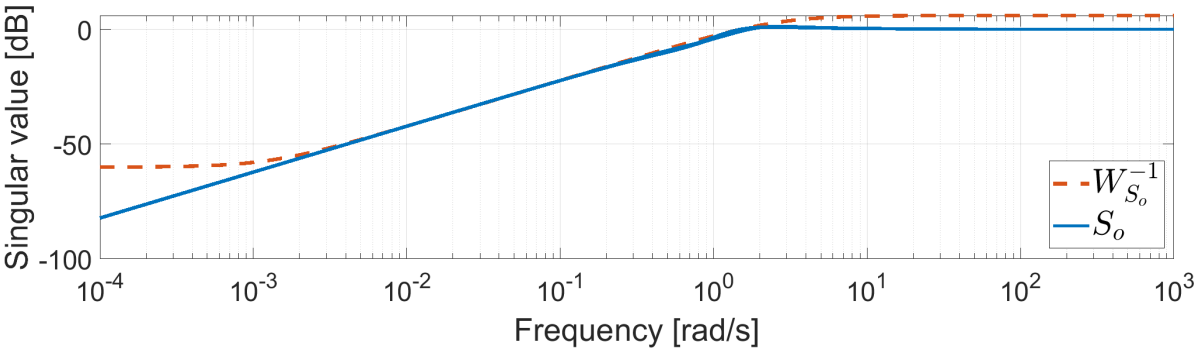



**Figure 4.5:** Output sensitivity and inverted weighting function.

### Control signal attenuation

To ensure the actuator is within its bandwidth limits, the transfer function from the reference command to the actuator command signal is restricted. This is mathematically formulated by restricting $KS$ (which is defined as $q_r$ to $\delta_{e,cmd}$) using the weighting function $W_{KS}$:

$$H_{KS} = ||W_{KS}K_S(K)||_\infty \leq 1 \tag{4.3}$$

The elevator is modeled as a low-pass filter with a cut-off frequency of $20.2$ rad/s. Thus the inverted weighting function is defined to ensure $-18$ dB gain attenuation at that frequency. Furthermore, the high-frequency signals are limited to the aforementioned value of $-18$ dB to limit resonance in the actuator, while the low-frequency signals are in principle free but arbitrarily limited here to $-10$ dB. The weighting function and the resulting closed-loop transfer functions can be seen in Figure 4.6.

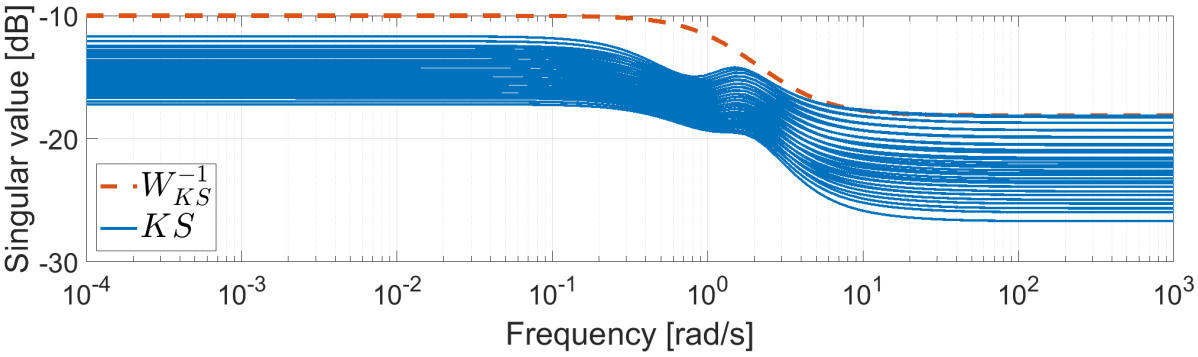



**Figure 4.6:** Control signal attenuation and inverted weighting function.

The resulting command signal attenuation satisfies the requirements enforced by the weighting function at low frequencies. However, at high frequencies, the requirement is very tight at some equilibrium points and a trade-off needs to be considered between high-frequency control signal attenuation and transient time. This is a classical trade-off during control system design and the value of $-18$ dB was used as a compromise.

### $\mathcal{D}$-Stability

This requirement ensures the location of the closed poles of the system resides within a specific subset of the complex plane, denoted as $\mathcal{D}$, across all design points. This ensures the fulfillment of certain criteria, including rapid decay and damping [48]. This region is defined as [8]:

$$\mathcal{D} = \left\{ z \in \mathbb{C} : f_D(z) = L + zM + \overline{z}M^\mathsf{T} < 0 \right\} \tag{4.4}$$

where $L = L^T$, $M$ and $L$ are real matrices that define a region on the complex plane. In this case, a minimum decay rate of $0.4$ and a damping ratio of $0.6$ were used to define a conic section on the complex plane. The conic section and the resulting locations of the poles can be seen in Figure 4.7.

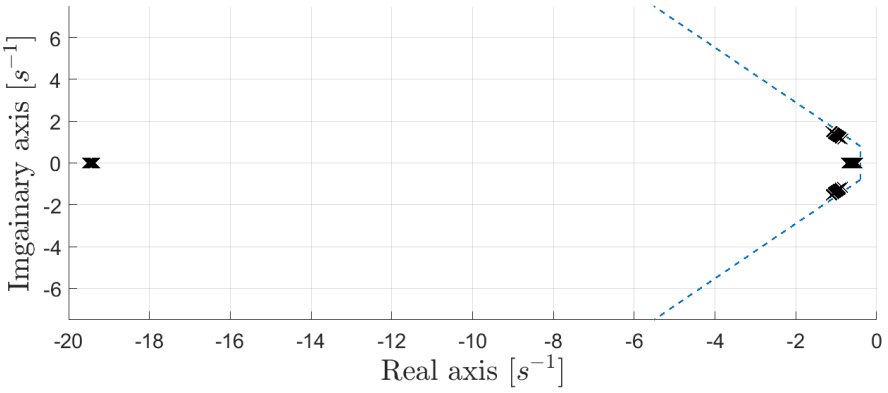



**Figure 4.7:** Pole location requirement.

### 4.2.3. Soft constraints

The soft constraints are of less importance than the hard constraints and are only optimized after the hard constraints are satisfied. For the pitch-rate controller, only one soft constraint is used, namely to enforce sensor noise attenuation with respect to the noisy angle of attack sensor.

#### Sensor noise attenuation

In this case, the sensor noise attenuation requirement covers the rejection of noise from $n_\alpha$ to $q$, of which the transfer is denoted by $T_\alpha$. The sensor dynamics were assumed to be 1 and this transfer function can be recognized as the off-diagonal element of the output complementary sensitivity matrix function $T_o$. Even though this is an off-diagonal component, only this element of the transfer function matrix is considered because the angle of attack measurement is most susceptible to noise and the aim is to limit its effect on the pitch-rate output. If the weighting function is defined as $W_{T_\alpha}$, the constraint becomes:

$$\min S_{T_\alpha} = ||W_{T_\alpha} T_\alpha(K)||_\infty \tag{4.5}$$

The inverted weighting function and resulting closed-loop value can be seen in Figure 4.8. The max gain of the inverted weighting function is set to $-15$ dB (equal to a magnitude of $0.178$). However, a more stringent requirement is needed for higher-frequency noise. To establish this noise rejection requirement, a simple model of the angle of attack filter is considered as defined by Nguyen [33]. Here, the filter is defined as a simple lag filter with a bandwidth of $\omega = 10$. Thus, at that frequency, the gain is set to be $-30$ dB. Furthermore, the very high-frequency values are to be attenuated by at least $-40$ dB, and a second-order transfer function is used for the inverted weighting function to ensure a sharper cutoff. The results can be seen in Figure 4.8.

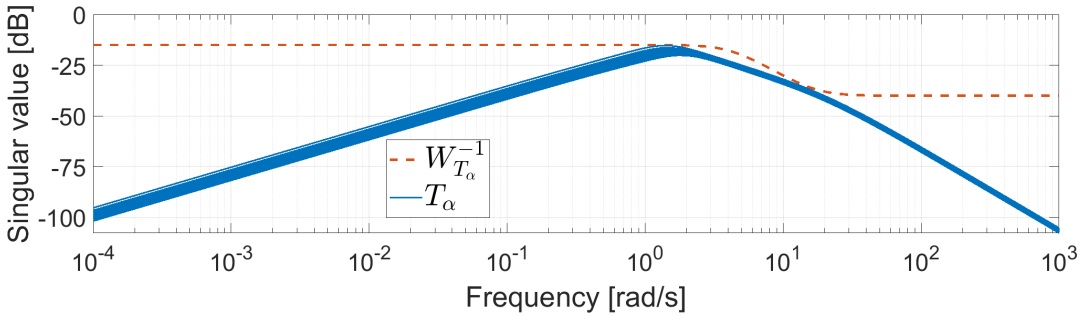


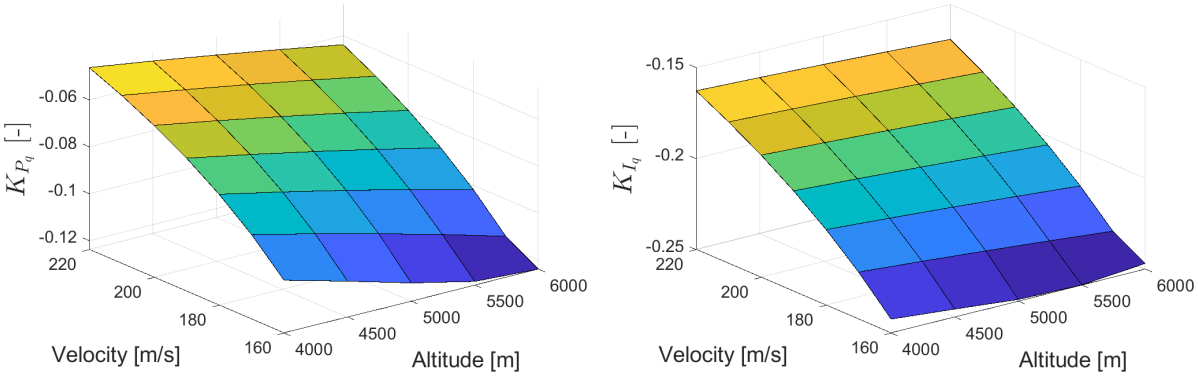

**Figure 4.8:** $n_\alpha$ to $q$ constraint and closed-loop transfer function.

Note that since the sensor is modeled as an ideal sensor, i.e. the transfer function is equal to one, the transfer function from $n_\alpha$ to $q$ is equal to the transfer function from $d_\alpha$ to $q$. The latter governs the output

disturbance rejection which requires low-frequency attenuation. As can be seen in Figure 4.8 this is also the case.

### 4.2.4. Controller gain surfaces

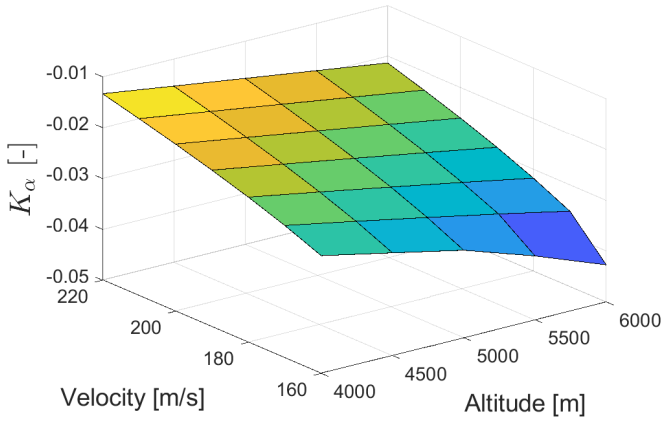
The resulting controller surfaces can be seen in Figure 4.9. These surfaces are constructed by linear interpolation between the chosen design points. The surfaces are very smooth, which is beneficial for the nonlinear implementation later. An interesting note is that at the highest altitude and lowest velocity equilibrium point, the gain values seem a little bit off compared to the others. This is also the most difficult point to satisfy which explains this (very minor) anomaly.




**(a)** Gain surface describing proportional value $K_{P_q}$.

**(b)** Gain surface describing integral value $K_{I_q}$.

**Figure 4.9:** Gain surfaces describing the pitch-rate controller.




**(c)** Gain surface describing inner alpha loop $K_\alpha$.

**Figure 4.9:** Gain surfaces describing the pitch-rate controller (continued).

## 4.3. Linear analysis

This section covers additional analysis of the system concerning parts that were not explicitly taken into account in the design procedure. That is, these criteria are only checked a posteriori. The two criteria that are considered here are:

1. Input disturbance rejection
2. Stability margins

#### Input disturbance rejection

To ensure rejection of the plant disturbance $d_{\delta_e}$ to $q$, this transfer function also called the load sensitivity function ($PS_i$), is checked. The low-frequency gain should be low to ensure robustness against

inverse multiplicative uncertainties at the actuator input. The frequency response can be seen in Figure 4.10. As can be seen, low-frequency attenuation is achieved, without explicitly constraining the transfer function.

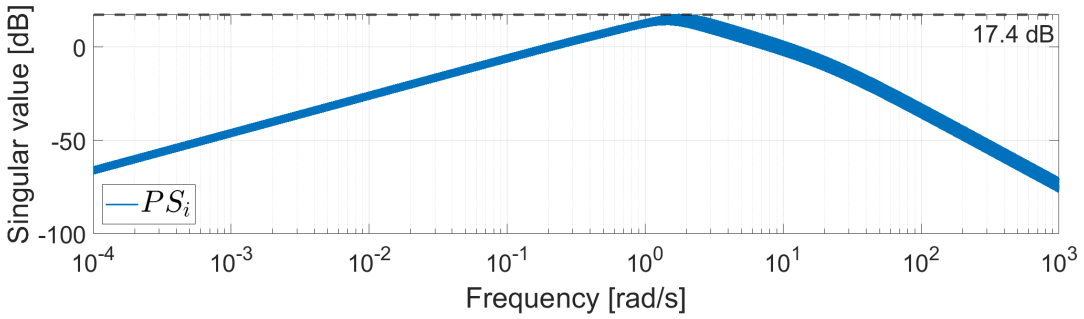



**Figure 4.10:** Closed loop value of input sensitivity times plant, that is the transfer function from $d_{\delta_e}$ to $q$.

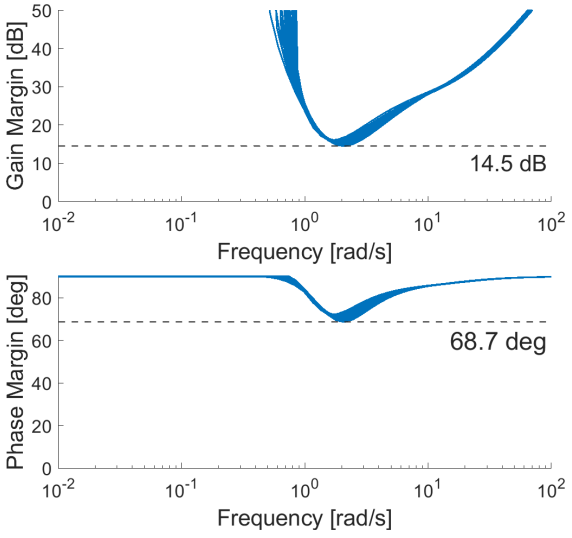
### Stability margins

An important property that needs to be inspected is the stability margins of the system. There are many ways to look at the margin criteria. Here the symmetric, disk phase and gain margins are investigated. These can be expressed using the $S - T$ functions [41]:

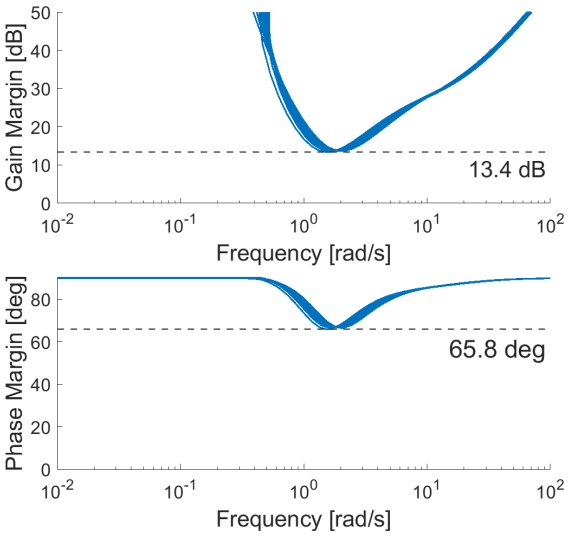
$$\alpha_{\text{max}} = \left|\left| S - \frac{1}{2} \right|\right|_{\infty}^{-1} = \left|\left| \frac{S - T}{2} \right|\right|_{\infty}^{-1} \tag{4.6}$$

The disk margins can consider each loop separately but can also be used to look at the simultaneous input/output disk margins. In this case, all permutations are looked at but there is only one input thus the simultaneous input margin is equivalent to the loop-at-a-time input disk margin.

The $6$ dB gain margin and $45°$ phase margin minimum prescribed by the DoD Mil-Spec [32] requirements can be used as guidelines for the requirements on the margins. Even though this requirement refers to the classical gain and phase margin, all disk margins except for one satisfy the requirement as well. The disk margins are significantly more conservative than the classical margins therefore the requirement is more than sufficiently satisfied. The only disk margin that does not in itself adhere to the $6$ dB, $45°$ guideline is the simultaneous input/output stability margin. This is the most conservative measure possible and only falls short with $1$ dB on the gain margin. However, this is still a very robust value given that it is such a conservative margin. Thus it can be concluded that the controller as a whole also displays very robust stability margins.

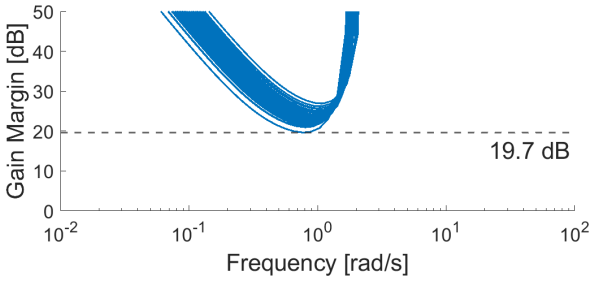


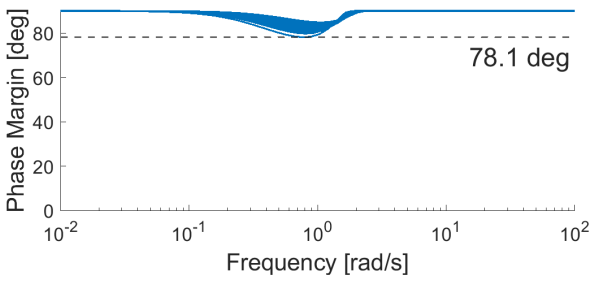

**(a)** Loop at a time input ($\delta_e^C$) stability margin.



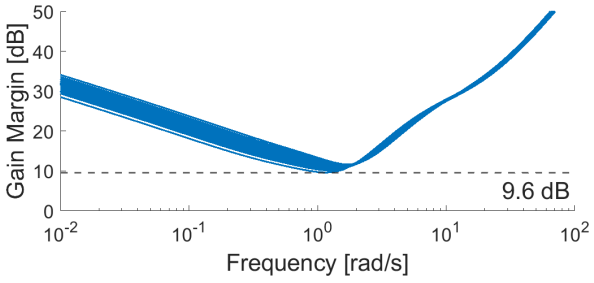

**(b)** Loop at a time output ($q$) stability margin.

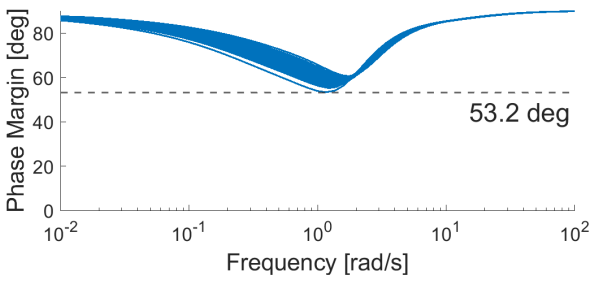
**Figure 4.11:** Disk based stability margins.



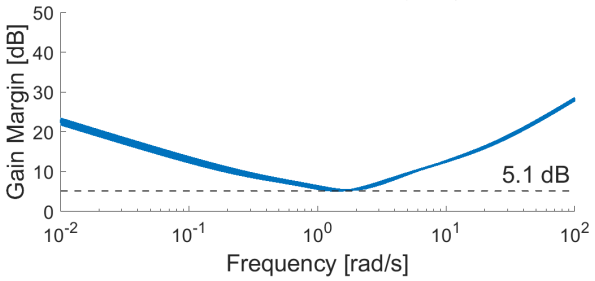



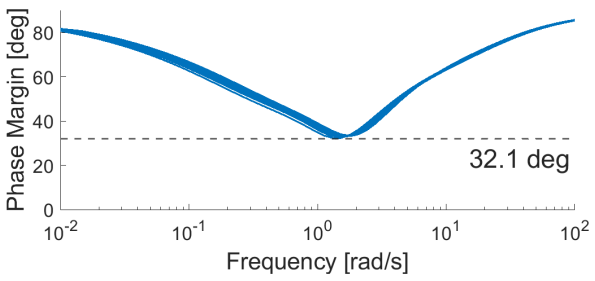

**(c)** Loop at a time output ($\alpha$) stability margin.




**(d)** Simultaneous output stability margin.




**(e)** Simultaneous input and output stability margin.

**Figure 4.11:** Disk based stability margins (continued).

## 4.4. Nonlinear implementation and analysis

This chapter uses the theory from Chapter 3 to establish frozen-time robustness of the closed-loop system using the gap metric and minimum normalized coprime stability margin. First, the baseline normalized coprime stability margin is calculated in Section 4.4.1. Subsequently, in Section 4.4.2, the hidden coupling terms are derived for the used nonlinear realization and it is discussed if the chosen realization limits the hidden coupling sufficiently. Then the trim point uncertainty is explored in Section 4.4.3. Lastly, the influence of the two factors on the normalized coprime stability margin is looked at in Section 4.4.4. The goal of this section is to discover if the controller shows frozen-time robustness in the nonlinear domain and what the possible factors of concern are.

### 4.4.1. Normalized coprime stability margin

By making use of Equation 3.37 for $b(P, K)$, the normalized coprime stability margin can be calculated for the linear controller-plant combination at each equilibrium point in the flight envelope. The result can be seen in Figure 4.12.

This normalized coprime stability margin is only valid in the linear domain and at the equilibria points. It will degrade in the nonlinear domain because of trim point uncertainty and hidden coupling. In the upcoming sections, precisely this degradation of the normalized stability margin, because of these phenomena, is calculated and investigated.

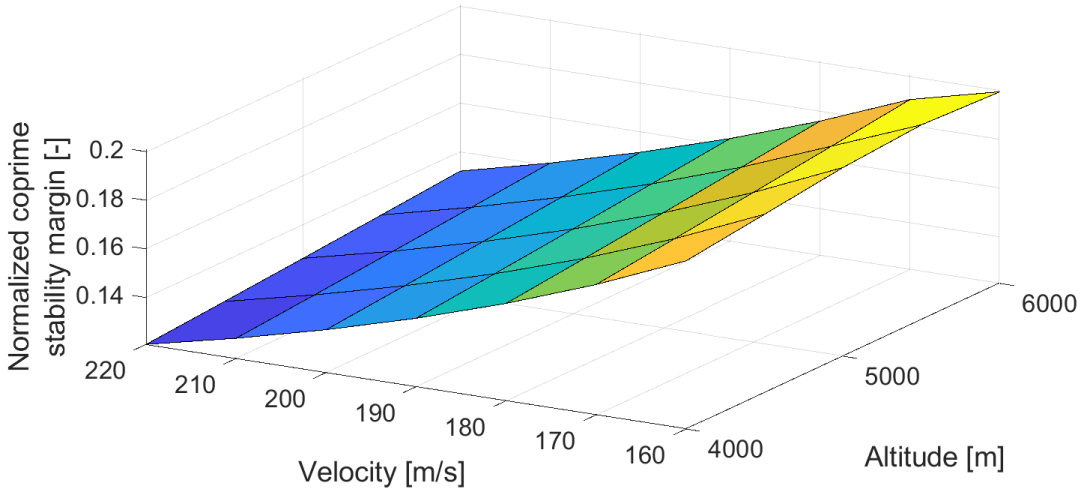



**Figure 4.12:** Normalized coprime stability margin of pitch-rate controller at the equilibria points.

### 4.4.2. Hidden coupling

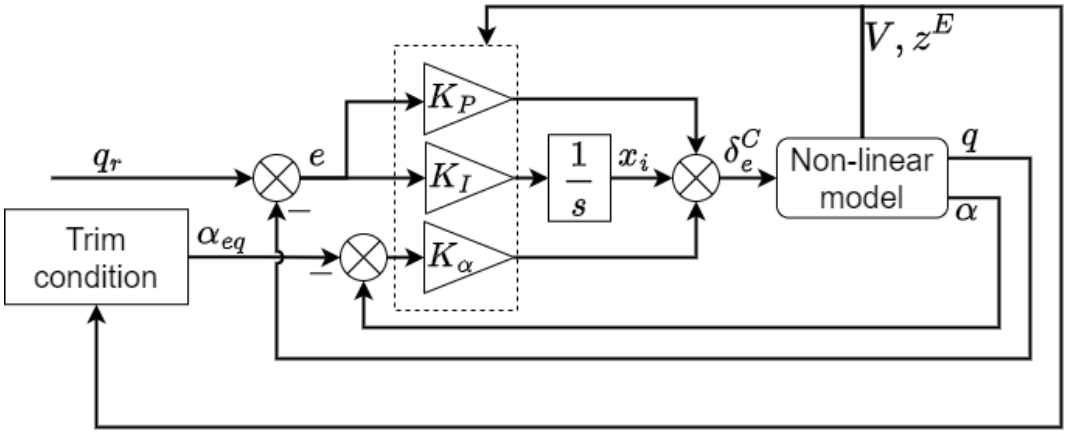
The extent to which hidden coupling has an effect greatly depends on the specific controller realization in the nonlinear domain. Even the location of the scheduled controller gains with respect to the integrator has an effect. The two realizations that are considered are as follows:

1. Subtract the equilibrium values and varying these using the real-time scheduling parameters (Figure 4.13a).
2. Subtract a fixed nominal condition (Figure 4.13b).

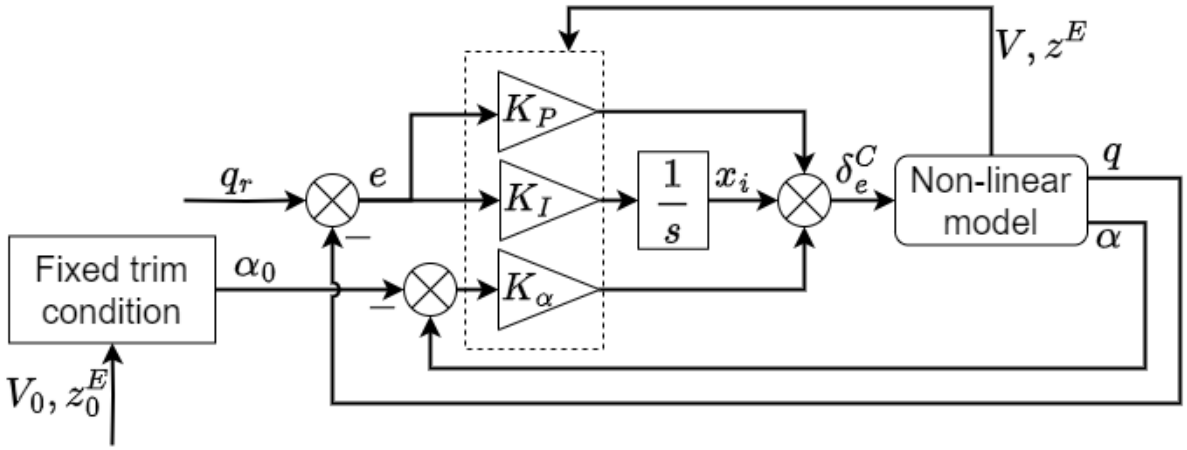
The suitability of the realizations is assessed at the equilibria points. The effect of hidden coupling at off-equilibrium points is not explicitly calculated in this section, but only when specific trajectories/maneuvers are considered.

As can be seen in Figure 4.13a, this implementation uses the scheduling parameters to interpolate the corresponding trim condition ($\alpha_{eq}$) at any operating point in the flight envelope. In theory, this achieves equivalence with the linearized system, however, it introduces an extra feedback loop based on the scheduling parameters. This causes a kind of hidden coupling effect which is difficult to calculate as derivatives of the trim conditions appear in the expression. In practice, this may be a hard value to precisely estimate on a physical system [28]. Thus this implementation is deemed unsuitable for the pitch-rate controller.

In contrast, the second method chooses, *a priori*, a fixed trim condition to establish a nominal value $\alpha_0$. The nominal value can for example correspond to the equilibrium point in the middle of the flight envelope. This formulation induces coupling effects that can be estimated well, now only depending on the derivatives of the controller gain functions. The disadvantage is that this formulation loses direct equivalency with the linearized system due to the nominal condition not being valid for every controller-plant combination. This effect is largest at points away from the nominal condition. However, it will be shown later that this realization does retain frozen-time robustness very well. Thus the fixed trim realization is appropriate for our pitch-rate controller.




**(a)** Realization using varying trim conditions.




**(b)** Realization using fixed trim condition.

**Figure 4.13:** Different controller realizations.

To calculate the exact effect of the hidden coupling terms, the theory exposed in Section 3.2 can be used. Consider the fixed trim realization of the controller as shown in Figure 4.13b:

$$\begin{aligned} \dot{x_i} &= K_I(V, z^E)e \\ \delta_e &= x_i + K_P(V, z^E)e + K_\alpha(V, z^E)(\alpha - \alpha_0) \end{aligned} \tag{4.7}$$

Linearization using the velocity-linearization framework yields:

$$\begin{aligned} \ddot{x_i} &= K_I(V, z^E)\dot{e} + \left.\frac{\partial K_I(V, z^E)}{\partial V}\right|_{op} e_{op}\dot{V} + \left.\frac{\partial K_I(V, z^E)}{\partial z^E}\right|_{op} e_{op}\dot{z}^E \\ \dot{\delta_e} &= \dot{x_i} + K_P(V, z^E)\dot{e} + \left.\frac{\partial K_P(V, z^E)}{\partial V}\right|_{op} e_{op}\dot{V} + \left.\frac{\partial K_P(V, z^E)}{\partial z^E}\right|_{op} e_{op}\dot{z}^E + \\ &\quad K_\alpha(V, z^E)\dot{\alpha} + \left.\frac{\partial K_\alpha(V, z^E)}{\partial V}\right|_{op} (\alpha_{op} - \alpha_0)\dot{V} + \left.\frac{\partial K_\alpha(V, z^E)}{\partial z^E}\right|_{op} (\alpha_{op} - \alpha_0)\dot{z}^E \end{aligned} \tag{4.8}$$

Note that at equilibrium $e_{op} = 0$ causing many terms disappear and the equation simplifies to:

$$\begin{aligned} \ddot{x_i} &= K_I(V, z^E)\dot{e} \\ \dot{\delta_e} &= \dot{x_i} + K_P(V, z^E)\dot{e} + K_\alpha(V, z^E)\dot{\alpha} + \left.\frac{\partial K_\alpha(V, z^E)}{\partial V}\right|_{eq} (\alpha_{eq} - \alpha_0)\dot{V} + \left.\frac{\partial K_\alpha(V, z^E)}{\partial z^E}\right|_{eq} (\alpha_{eq} - \alpha_0)\dot{z}^E \end{aligned} \tag{4.9}$$

Two hidden coupling terms remain, namely:

- $\frac{\partial K_\alpha(V,z^E)}{\partial V}\Big|_{eq}(\alpha_{eq}-\alpha_0)\dot{V}$, which is caused by the coupling of the controller scheduling variable $V$ and the plant.
- $\frac{\partial K_\alpha(V,z^E)}{\partial z^E}\Big|_{eq}(\alpha_{eq}-\alpha_0)\dot{z}^E$, which is caused by the coupling of the controller scheduling variable $z^E$ and the plant.

Now, the gap metric can be used to quantify the robustness degradation because of the hidden coupling terms. The gain surfaces are known and the partial derivatives of these surfaces can thus be numerically approximated. The resulting controller is compared to the ideal case when no hidden coupling terms were present, for which it was designed. This ideal controller is of the velocity-form:

$$\begin{aligned}\ddot{x_i} &= K_I(V,z^E)\dot{e}\\ \dot{\delta_e} &= \dot{x_i} + K_P(V,z^E)\dot{e} + K_\alpha(V,z^E)\dot{\alpha} + 0\dot{V} + 0\dot{z}^E\end{aligned} \tag{4.10}$$

If the controller of Equation 4.9 is denoted as $K_{hct}$ and the controller of Equation 4.10 denoted as $K_{eq}$, the gap metric $\delta(K_{hct}, K_{eq})$ can be calculated. The resulting gap metric at each equilibrium point is shown in Figure 4.14. It can be seen, that the gap metric increases at higher altitudes and lower velocities, that is regions away from the nominal point. Upon comparison with Equation 4.9, it becomes clear that the contribution of the derivatives of $K_\alpha(V,z^E)$ and the distance between the equilibrium point $\alpha_{eq}$ and the chosen fixed $\alpha_0$ value determine the hidden coupling terms close to equilibria. As the derivatives of $K_\alpha(V,z^E)$ are fairly constant, due to the smooth gain surfaces, the magnitude of the effect thus mostly depends on the distance between the nominal angle of attack $\alpha_0$ and the actual angle of attack at equilibrium $\alpha_{eq}$. Hence, the error is largest at the edge of the flight envelope which corresponds to what can be seen in Figure 4.14.

All in all, the resulting values of the gap metric (about $1 \times 10^{-5}$) are very small in comparison to the coprime stability margins (about $0.12$ to $0.20$) of the combined linearized plant and linearized controller at equilibria points. Thus, it can be concluded the realization is suitable to limit the effect of hidden-coupling terms on frozen-time robustness.

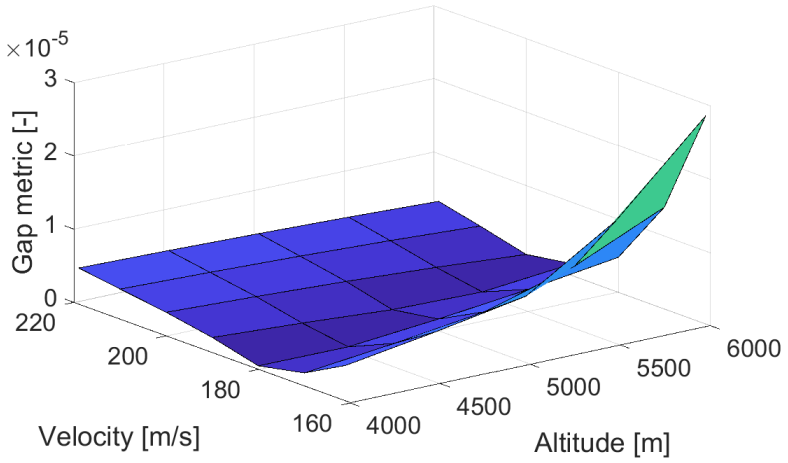



**Figure 4.14:** Gap metric between the ideal controller and the controller containing hidden coupling terms.

### 4.4.3. Trim point uncertainty

Now let us consider the trim point uncertainty by looking at a flight maneuver that traverses through off-equilibrium points. Chapter 2 derived the equation of motions in tensor form and showed how to coordinate the equations for implementation. If it is considered that all the equations are coordinated and the relevant parameters are available then the nonlinear system can be summarized by:

$$\begin{aligned}\dot{x} &= F(x,u)\\ y &= G(x,u)\end{aligned} \tag{4.11}$$

With the states ($x$), inputs ($u$), and outputs (for longitudinal motion) being defined as:

$$\begin{aligned} x &= \left(V, \alpha, \beta, p, q, r, x_L, y_L, z_L, \phi, \theta, \psi, \delta_{l,cmd}, \delta_l\right)^T \\ u &= \left(\delta_a, \delta_e, \delta_r, \delta_t\right)^T \\ y &= \left(V, \alpha, q, \theta, z_L, n_z\right)^T \end{aligned} \tag{4.12}$$

In the end, this is a purely symbolic expression, except for the aerodynamic coefficients. As was established in Section 2.3, and in more detail in Appendix A, the dependencies of the aerodynamic coefficients are captured by (a part of) the states and inputs through the use of look-up tables. To perform velocity-based linearization also the derivatives of the aerodynamic coefficients are necessary. By use of the chain and product rule, it is possible to isolate these coefficients whereafter these can be numerically approximated. To show this, let us take an example using the equation for $\dot{q}$. In Section 2.4.3 the matrix form of the attitude dynamics was derived and if you convert these to scalar form and insert the values of the inertia, the atmospheric density at (fixed) sea level, and the wing surface area it results in the equation:

$$\dot{q} = 0.9604p \cdot r + 4.878 \times 10^{-4} V^2 C_m(V, \alpha, \beta, q, \delta_e, \delta_l) - 0.0176p^2 + 0.0176r^2 \tag{4.13}$$

To perform the velocity-based linearization the equation has to be differentiated with respect to time:

$$\begin{aligned} \ddot{q} =& 0.9604\dot{p} \cdot r + 0.9604p \cdot \dot{r} \\ &+ 4.878 \times 10^{-4} \cdot 2V\dot{V} C_m(V, \alpha, \beta, q, \delta_e, \delta_l) \\ &+ 4.878 \times 10^{-4} V^2 \frac{\partial C_m(V, \alpha, \beta, q, \delta_e, \delta_l)}{\partial t} \\ &- 0.0352p \cdot \dot{p} + 0.0352r \cdot \dot{r} \end{aligned} \tag{4.14}$$

with

$$\frac{\partial C_m(V, \alpha, \beta, q, \delta_e, \delta_l)}{\partial t} = \frac{\partial C_m(V, \alpha, \beta, q, \delta_e, \delta_l)}{\partial V}\dot{V} + \frac{\partial C_m(V, \alpha, \beta, q, \delta_e, \delta_l)}{\partial \alpha}\dot{\alpha} + ... + \frac{\partial C_m(V, \alpha, \beta, q, \delta_e, \delta_l)}{\partial \delta_l}\dot{\delta}_l$$

This is simply a linear combination of the derivatives of the states and inputs. To simplify the notation, let $\Gamma$ denote the set of all variables used in the function of the aerodynamic coefficients, such that $C_m(\Gamma)$. This results in the expression:

$$\frac{\partial C_m(\Gamma)}{\partial t} = \sum_{n \in \Gamma} \frac{\partial C_m(\Gamma)}{\partial n}\dot{n}$$

inserting this back into Equation 4.14 yields:

$$\begin{aligned} \ddot{q} =& 0.9604\dot{p}r + 0.9604p\dot{r} + 4.878 \times 10^{-4} \cdot 2V\dot{V} C_m(\Gamma) \\ &+ 4.878 \times 10^{-4} V^2 \sum_{n \in \Gamma} \frac{\partial C_m(\Gamma)}{\partial n}\dot{n} - 0.0352p\dot{p} + 0.0352r\dot{r} \end{aligned} \tag{4.15}$$

As mentioned before, this is simply a linear combination of the derivatives of the states and inputs. This might become even more clear if an operating point ($_{op}$) is considered and the derivatives of the states and inputs are collected:

$$\begin{aligned} \ddot{q} =& (0.9604r_{op} - 0.0352p_{op})\dot{p} + (0.9604p_{op} + 0.0352r_{op})\dot{r} \\ &+ 4.878 \times 10^{-4} \cdot 2V_{op} C_m(\Gamma_{op})\dot{V} + 4.878 \times 10^{-4} \cdot V_{op}^2 \sum_{n \in \Gamma} \left.\frac{\partial C_m(\Gamma)}{\partial n}\right|_{op} \dot{n} \end{aligned} \tag{4.16}$$

The only term that is not readily available at an operating point is the derivatives of the aerodynamic coefficients with respect to the states and inputs ($\sum_{n \in \Gamma} \left.\frac{\partial C_m(\Gamma)}{\partial n}\right|_{op}$). However, these terms term can simply be found by numerical approximation using the look-up tables. This velocity-form can be derived for every state and output equation of the model to complete the velocity-based linearization.

It can be seen that the expressions quickly become too large to handle. Hence, the differentiation is all done using the *symbolic math toolbox* from MATLAB. The method can be verified, however, by comparing it to conventional linearization at equilibria values. A similar method using the sum of square differences of the elements as was used in the verification of the tensor model (see Section 2.7) is applied. The sum of square errors of all elements on the state space matrices (A,B,C,D) acquired with velocity-based linearization versus Jacobian linearization can be calculated. When adding all of these together, the resulting sum of square errors is smaller than $2 \times 10^{-8}$ at each point of the flight envelope, as shown in Figure 4.15. This indicates extremely good congruence between the two methods of linearization.

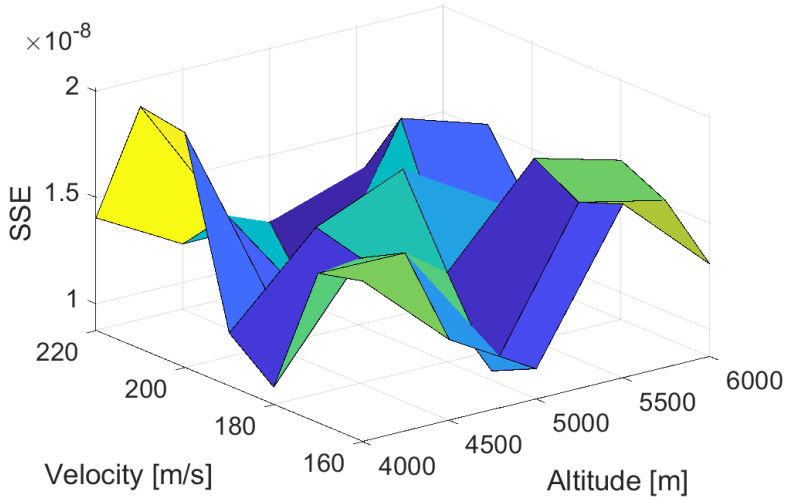
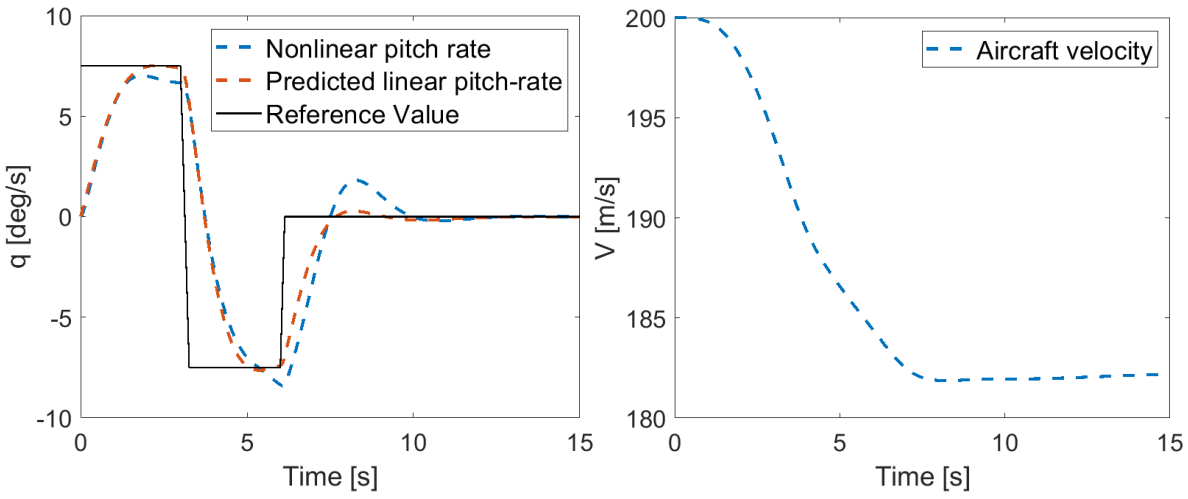



**Figure 4.15:** SSE of all state-space matrices using Jacobian linearization vs velocity-based linearization.

Subsequently, let us apply the velocity-based linearization to quantify the trim point uncertainty along a typical maneuver. Figure 4.16a compares a (full longitudinal) linear and a nonlinear simulation, given the fixed-trim controller implementation, when a large pitch-rate reference value is imposed. The nonlinear simulation does not converge precisely to the reference value during both the pitch-up and pitch-down maneuvers. Figure 4.16b shows the corresponding aircraft velocity. It can be investigated more thoroughly if this is caused by the effect of the trim point uncertainty.



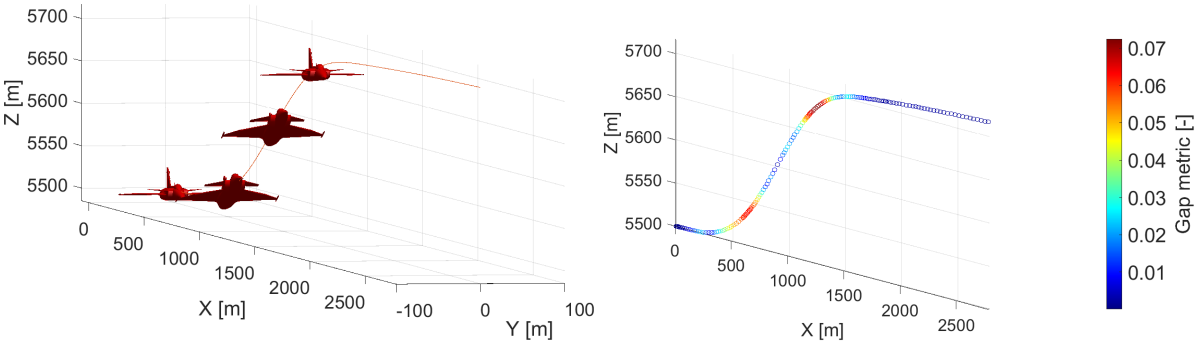

**(a)** Pitch-rate response.

**(b)** Velocity response.

**Figure 4.16:** Nonlinear simulation of the pitch-rate controller with a large reference input.

At each operating point, the plant can be linearized using velocity-based linearization. This results in the plant $P_{off}$ corresponding to a specific operating point. Subsequently, the plant $P_{eq}$ denotes the linearized plant corresponding to the *closest* equilibrium point from the operating point. The distance between these plants in terms of the gap metric can be calculated resulting in $\delta(P_{eq}, P_{off})$. This can be applied to the previously shown maneuver of Figure 4.16. The attitude and location of the aircraft during this trajectory are shown in Figure 4.17a. This trajectory is discretized at each point, and the gap metric is calculated indicating the trim point uncertainty. The results are shown in Figure 4.17b. Along this trajectory, the maximum gap metric found is equal to $0.078$ and occurs during the initial pitch-up maneuver and subsequent pitch-down maneuver.

The value of $0.078$ is significant compared to the normalized coprime stability margins. Hence the effect of trim point uncertainty does seem to degrade the controller robustness in the nonlinear domain. This also seems reflected in tracking performance, as the simulation shown in Figure 4.16 does not completely follow the predicted linear response once the aircraft has sustained the pitch-up command after about $2$ seconds. This is in agreement with the regions where the trim point uncertainty is largest. However, even when looking at these maneuvers with large step inputs, the frozen-time stability can still be guaranteed. This will be seen in the next section.




**(a)** Trajectory and attitude of aircraft (plotted by using [40]).

**(b)** Gap metric to equilibrium conditions along the trajectory.

**Figure 4.17:** Trajectory and trim point uncertainty.

### 4.4.4. Frozen-time stability

To conclude the nonlinear analysis, the two effects are combined by making use of the inequality relating the gap metrics to the minimum coprime stability margin. The gap metric, between the linearized plant at an operating point and the linearized plant at the closest equilibrium point, was already calculated in the previous section and denoted by $\delta(P_{eq}, P_{off})$. Subsequently, let $K_{off}$ denote the controller at the same operating point including the hidden coupling terms. This can be calculated by using Equation 4.8. Then let $K_{eq}$ denote the controller at the closest equilibrium point, but this time with the hidden coupling terms set to zero. The distance between these systems in terms of gap metric can also be calculated at the operating point. This results in the expression $\delta(K_{off}, K_{eq})$. Now that all the pieces are in place, it is possible to relate these gap metrics to the minimum coprime stability margin at the operating point by:

$$\arcsin(b(P_{off}, K_{off})) \geq \arcsin(b(P_{eq}, K_{eq})) - \arcsin(\delta(K_{off}, K_{eq})) - \arcsin(\delta(P_{eq}, P_{off})) \tag{4.17}$$

Here $(b(P_{eq}, K_{eq})$ is the coprime stability margin at the closest equilibrium point from the operating point.

This equation is applied to the same maneuver as used for the trim point uncertainty. Figure 4.18 shows the resulting minimum coprime stability margin along the discretized trajectory. It can be seen that the minimum guaranteed margin degrades from its equilibrium point, however, frozen-time stability is still enforced as the minimum coprime stability margin remains larger than $0$. Hence it can be concluded that the local stability guarantees of gain scheduling are extended to also include the points along this trajectory.

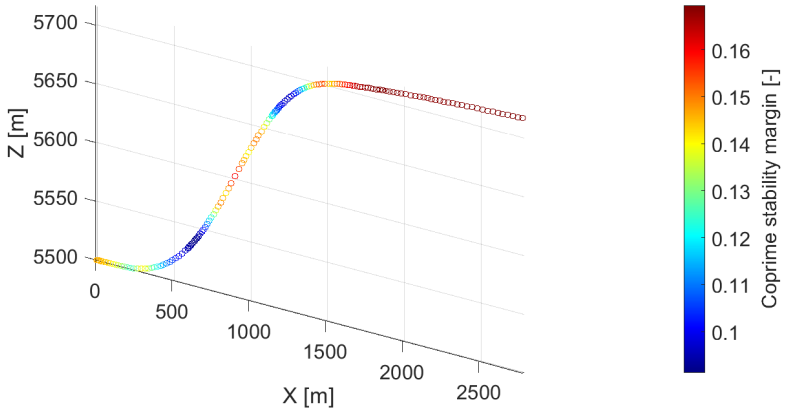



**Figure 4.18:** Minimum coprime stability margin along the trajectory.

Note, that the biggest contribution to the degradation is mostly caused by the effect of trim point uncertainty as a suitable realization has been chosen to limit the effects of hidden coupling. This degrades the normalized coprime stability margin by a maximum of about $45\%$. The controller had very robust disk margins (which are related to the coprime stability margin), thus it is argued that the controller still maintains sufficient frozen-time robustness. Nevertheless, it might be desirable to reduce the degradation.

However, since the trim point uncertainty dominates, it is difficult to improve the frozen-time stability. Attempts were made to include additional scheduling parameters (such as $q$ and $\alpha$) in order to capture the off-equilibrium dynamics and improve the controller performance. It turned out to be difficult to set up a scheduling scheme that properly captures these dynamics, while also having sufficiently slowly varying scheduling parameters. Nonetheless, ideally, the knowledge of the degradation of the controller because of the trim point uncertainty would be taken into account in the design procedure, and this could be of interest in future research.

Moreover, in this analysis, only the example of one doublet input where $q_{ref} = 7.5^\circ$ is considered. The pitch-rate controller is guaranteed to be frozen-time robust during this maneuver. In practical application, this analysis can be extended to many maneuvers and cover a larger part of the flight envelope to extend the local stability guarantees. This highlights how this technique framework can be used to analyze the nonlinear domain and ensure frozen-time robustness.

# 5

# Direct trajectory longitudinal control

The automatic flight control system has seen an increase in functionality over the years. The success of the pitch-rate controllers was followed by the pitch attitude hold controller and altitude hold controller. Thereafter, maneuvering modes such as vertical speed and altitude selection were introduced followed by automatic landing control systems. Later, around the 1980s, the first full flight regime autothrottle entered the, partly to ensure minimum fuel use after the increase in fuel prices [19]. Historically, these control systems were designed using a single-input, single-output control strategy, where each loop was closed one at a time. For a full direct trajectory longitudinal control system, the airspeed and direction have to be controlled simultaneously and, while the loop-at-a-time approach can work, the performance is far from optimal as there is no harmony between the loops [20]. $\mathcal{H}_\infty$ synthesis methods allow for simultaneous loop closure in MIMO systems and improve this harmony. To enable the controller to control a part of the nonlinear flight envelope (the same as used in Chapter 4), the multi-model approach is used.

That also means, that while the pitch-rate controller can be designed using the short-period approximation, this is impossible when looking at direct trajectory longitudinal control. The longitudinal controller has to be able to control the direction and speed simultaneously, i.e. the complete longitudinal velocity vector, both on short and long timescales. Thus, the full longitudinal dynamics of the aircraft have to be used for the controller synthesis and a MIMO controller is necessary.

Another complication of this problem is that it is prone to oversaturation of the engine. Both an increase in airspeed and vertical direction require a throttle increase. In order to tackle this problem, it has often been approached from a total energy point of view to limit the thrust and load factor/ pitch-rate demands to the inner loop to ensure proper envelope protection of the controller [14] [18] [19]. However, in this thesis, it is assumed that the aircraft can provide enough energy to increase the speed and altitude and the total available energy does not have to be distributed.

The last matter to discuss before the structure and constraints can be introduced is the choice of regulated variables. The airspeed is a natural choice in order to control the magnitude of the velocity vector [5] but the direction can be controlled using multiple variables. One option is to use the flight-path angle ($\gamma$) [29] [44] but it is also possible to use the vertical speed ($V_z$) [5] [6] [47] as a second variable to completely constrain the longitudinal velocity vector. In this thesis, the vertical speed ($V_z$) is used to control the direction as the used $\mathcal{H}_\infty$ synthesis methods have been observed to be more commonly used in conjunction with $V_z$, however, both options are possible.

## 5.1. Structure, constraints and synthesis

This section is set up similarly as in the pitch-rate controller of Section 4.2 and the same flight envelope is used. It shows the structure of the longitudinal controller, the requirements with the resulting closed-loop transfer functions, and the obtained controller.

### 5.1.1. Controller structure

Figure 5.1 shows the inputs and outputs of the controller $K$. The controller has two outputs, six inputs and has two degrees of freedom since it both contains the reference and measured airspeed/vertical speed. The inputs and outputs are the same as used in several earlier proposed designs [5] [6] [47].

The used actuators are the engine ($\delta_t$) and elevator ($\delta_e$) and the used measurements are the airspeed ($V$), the vertical speed ($V_z$), the normal acceleration ($n_z$), and the pitch-rate ($q$). The first two measurement values are necessary for tracking and the inclusion of the latter two measurement values significantly improves the damping and transient response of the closed-loop system. Furthermore, the two-degrees-of-freedom controller form was necessary to satisfy all the upcoming constraints.

Moreover, Figure 5.1 also shows the disturbance and noise channels that will be used to define the constraints in the design synthesis. The output disturbances $d$ and sensor noise channels $n$ consists of:

$$\begin{aligned} d &= [d_V, d_{V_z}, d_{n_z}, d_q]^T \\ n &= [n_V, n_{V_z}, n_{n_z}, n_q]^T \end{aligned} \tag{5.1}$$

These are both $4 \times 1$ vectors that are added to the $4 \times 1$ signals containing the outputs of the system.

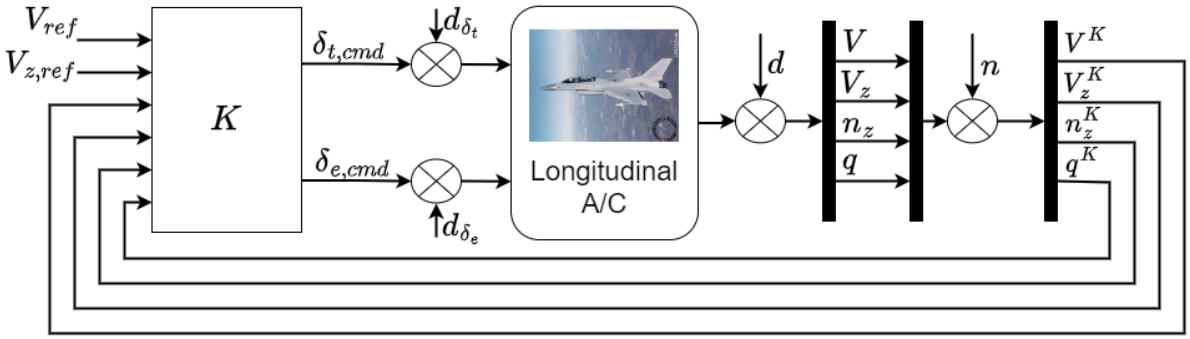



**Figure 5.1:** Longitudinal controller I/O layout.

This brings us to the structure of the controller $K$. It has been structured such that the main components are decentralized and Figure 5.2 shows this controller structure. It can be described by five basic components:

1. $K_{V_{ref}}(s)$ and $K_{V_{z,ref}}$ are first-order transfer functions that adjust the command signal to match a certain response.
2. Two integrators are acting on the error signals of the variables to be tracked, namely $e_V$ and $e_{V_z}$, to ensure no steady-state errors.
3. $K_1$ is a $2 \times 2$ static matrix that ensures the integral action on both errors is propagated to both outputs. This is necessary for the decoupling of the system.
4. $K_2$ is a $2 \times 4$ static matrix that also ensures decoupling of the system, this time directly using the output measurements except for $q$, which is filtered using $K_q(s)$
5. That brings us to $K_q(s)$. This is a first-order transfer function acting on the measurement $q^K$ It was necessary to include a dynamic component to the pitch-rate measurement to improve the short-period damping and reduce overshoot.

The controller is synthesized using a multi-model approach. In this part of the flight envelope, no scheduling was required to ensure the constraints were met. Now that the controller structure has been shown, the constraints can be introduced.

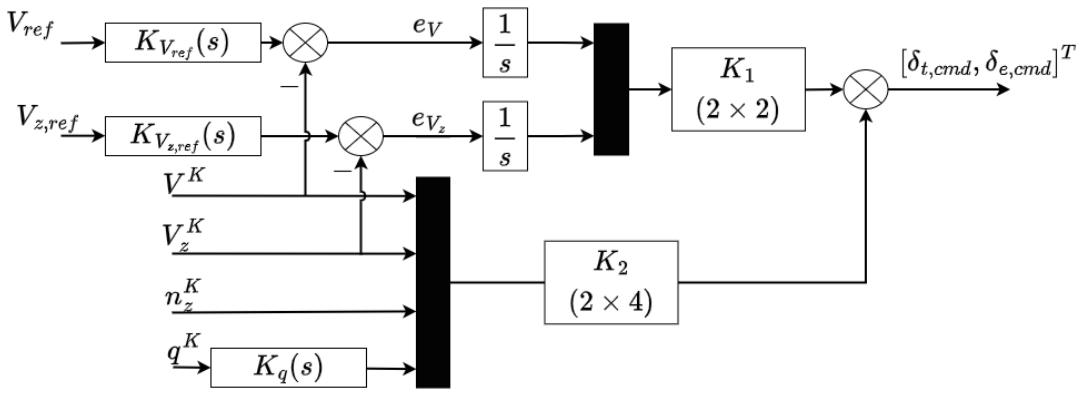



**Figure 5.2:** Longitudinal controller structure.

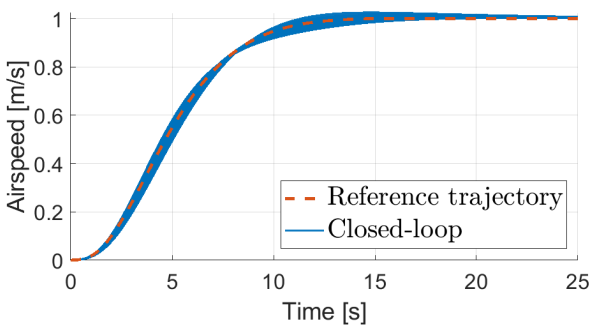
### 5.1.2. Soft constraints

A similar approach as done in Section 4.2 is used, however, now only soft constraints are used. Since the coupling is relevant the complete transfer function matrices are considered. The soft constraints were initially inspired by the four-block problem but ultimately the following factors were incorporated:

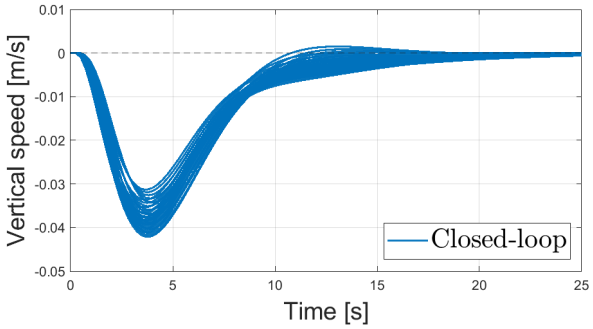
1. Reference tracking
2. Output disturbance rejection
3. Control signal attenuation
4. Input disturbance rejection
5. $\mathcal{D}$-stability

#### Reference tracking

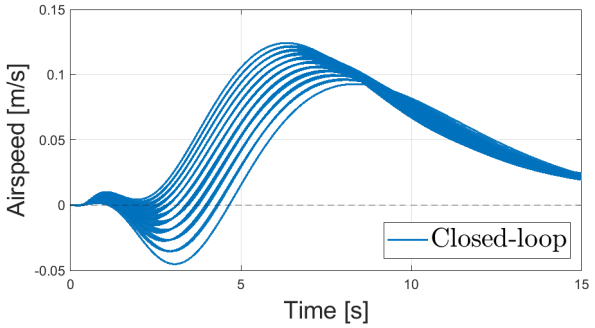
The tracking requirements are formulated using the $\mathcal{H}_2$ norm. Only the reference inputs to their corresponding output (that is $V_{ref}$ to $V$ and $V_{z,ref}$ to $V_z$), contained a reference tracking requirement. This is a similar type of constraint as was used for the pitch-rate controller of Section 4.2. The reference trajectories and the closed-loop values can be seen in Figure 5.3.



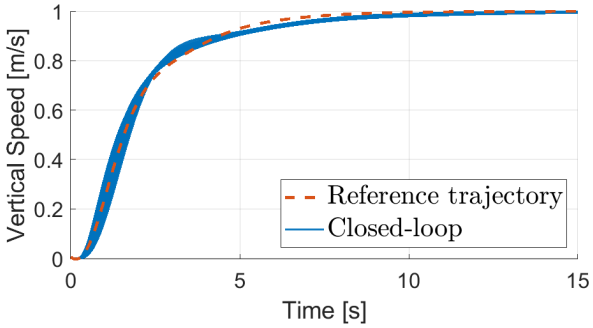

**(a)** Step response $V_{ref} \rightarrow V$ and reference model.




**(b)** Step response $V_{ref} \rightarrow V_z$.



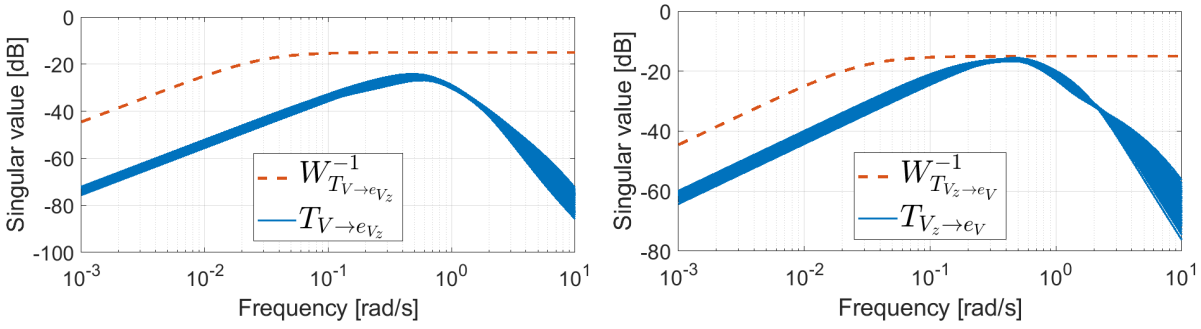

**(c)** Step response $V_{z,ref} \rightarrow V$.




**(d)** Step response $V_{z,ref} \rightarrow V_z$ and reference model.

**Figure 5.3:** $H_2$ tracking requirement.

The off-diagonal (cross-coupling) components are constrained through a maximum gain in the frequency domain of the transfer functions from the reference channel to the opposite error channel. These are the transfer functions from $V_{ref}$ to $e_{V_z}$ and from $V_{z,ref}$ to $e_V$. This is done differently than the used reference tracking as there is no specific preferred response. The only imposed constraint is to limit the cross-coupling error to a maximum of $15\%$ (about $-16$ dB) and ensure low gain at low frequencies such that the steady-state error reduces to zero. These constraints and resulting closed-loop values can be seen in Figure 5.4. Note that the controller has two degrees of freedom which means that the transfer function from the input reference values to the output values is not the same as the output sensitivity function which is the case with a one-degree-of-freedom controller.



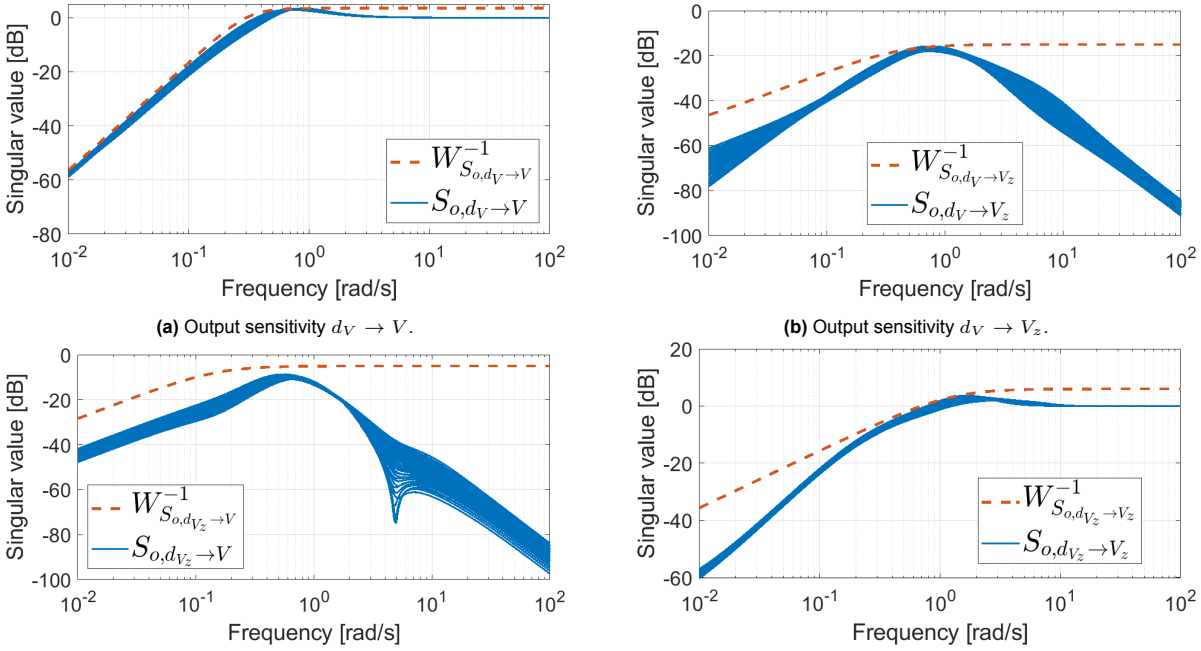

**(a)** Frequency response of transfer function $V_{ref} \to e_{V_z}$.

**(b)** Frequency response of transfer function $V_{z,ref} \to e_V$.

**Figure 5.4:** Frequency response and constraints of the transfer functions from $V_{ref}$ to $e_{V_z}$ and from $V_{z,ref}$ to $e_V$.

### Output disturbance rejection

The output disturbances considered contain the transfer functions from $d_V$, $d_{V_z}$, $d_{nz}$ and $d_q$ to $V$ and $V_z$. The complete output sensitivity matrix is of dimensions $4 \times 4$ but the effect of the disturbances on the vertical acceleration $n_z$ and pitch-rate $q$ are not included in the synthesis. This is done as these variables are not regulated and hence not necessarily constrained. What is left is thus a $2 \times 4$ matrix. First, let us consider the $2 \times 2$ transfer functions matrix containing the transfer functions of the disturbances $d_V$ and $d_{V_z}$ to the tracked outputs $V$ and $V_z$. The constraints and resulting closed-loop transfer functions can be seen in Figure 5.5.




**(a)** Output sensitivity $d_V \to V$.

**(b)** Output sensitivity $d_V \to V_z$.

**(c)** Output sensitivity $d_{V_z} \to V$.

**(d)** Output sensitivity $d_{V_z} \to V_z$.

**Figure 5.5:** Output sensitivity constraints and closed-loop values.

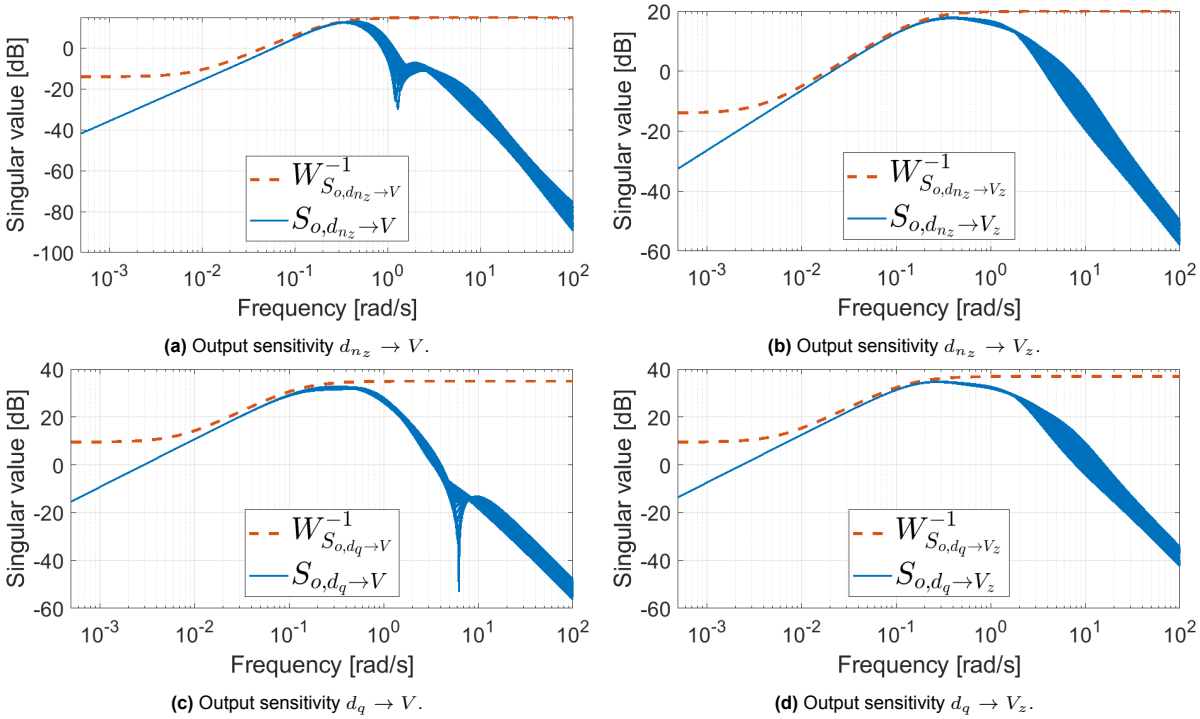
The diagonal components, which are the transfer functions $d_V \to V$ and $d_{V_z} \to V_z$ need to reject the low-frequency disturbance signals. A high pass filter is thus required for the weighting filters on the diagonal components. The low-frequency region is limited to $-70$ dB while the high-frequency limit is set to $3.5$ dB and $6$ dB respectively. Furthermore, the $0$ dB crossover frequency is chosen a bit differently, with $d_V \to V$ requiring a crossover frequency of $\omega_c = 0.2$ rad while $d_V \to V$ is a slightly faster transfer function with $\omega_c = 0.75$ rad.

The off-diagonal components are also limited to ensure maximum peak error and a maximum steady state error given a disturbance in the other channel. The $d_V \to V_z$ low-frequency gains are limited to $-80$ dB with the high-frequency gain being limited to $-15$ dB. The other off-diagonal channel $d_{V_z} \to V$ shows worse, but still sufficient, disturbance rejection with a low-frequency attenuation gain of $-50$ dB and high-frequency gain limit of $-5$ dB.

Subsequently, let us consider the influence of the disturbances $d_{n_z}$ and $d_q$ on $V$ and $V_z$. The frequency domain specifications and their respective constraints can be seen in Figure 5.6. The effect of the disturbance $d_{n_z}$ was constrained to $20$ dB at a maximum, which translates to a magnitude of $10\,(\text{ms}^{-1})/\text{g}$, to both outputs and it was ensured that the low-frequencies disturbances are not propagated to the outputs of interest.

The amplification of disturbance $d_q$ to both outputs is constrained by $37.5$ dB at max, which is equal to $1.3\,(\text{ms}^{-1})/(°\text{s}^{-1})$ and also ensures low-frequency attenuation. Note that the singular values in Figure 5.6 are computed using the values of $q$ in radians, while in the text they were converted to degrees for easier interpretation.

Lastly, it can be pointed out that the transfer functions from $d_{n_z}$ and $d_q$ to $V$ and $V_z$ are equal to the transfer functions from $n_{n_z}$ and $n_q$ on $V$ and $V_z$ since the sensors are modeled as ideal sensors. That indicates high-frequency noise attenuation should be present as well. As can be seen in Figure 5.6, this is indeed the case. Thus not only is low frequency disturbance rejection achieved, high frequency noise attenuation is also attained.




**(a)** Output sensitivity $d_{n_z} \to V$.

**(b)** Output sensitivity $d_{n_z} \to V_z$.

**(c)** Output sensitivity $d_q \to V$.

**(d)** Output sensitivity $d_q \to V_z$.

**Figure 5.6:** Output sensitivity constraints and closed-loop values.

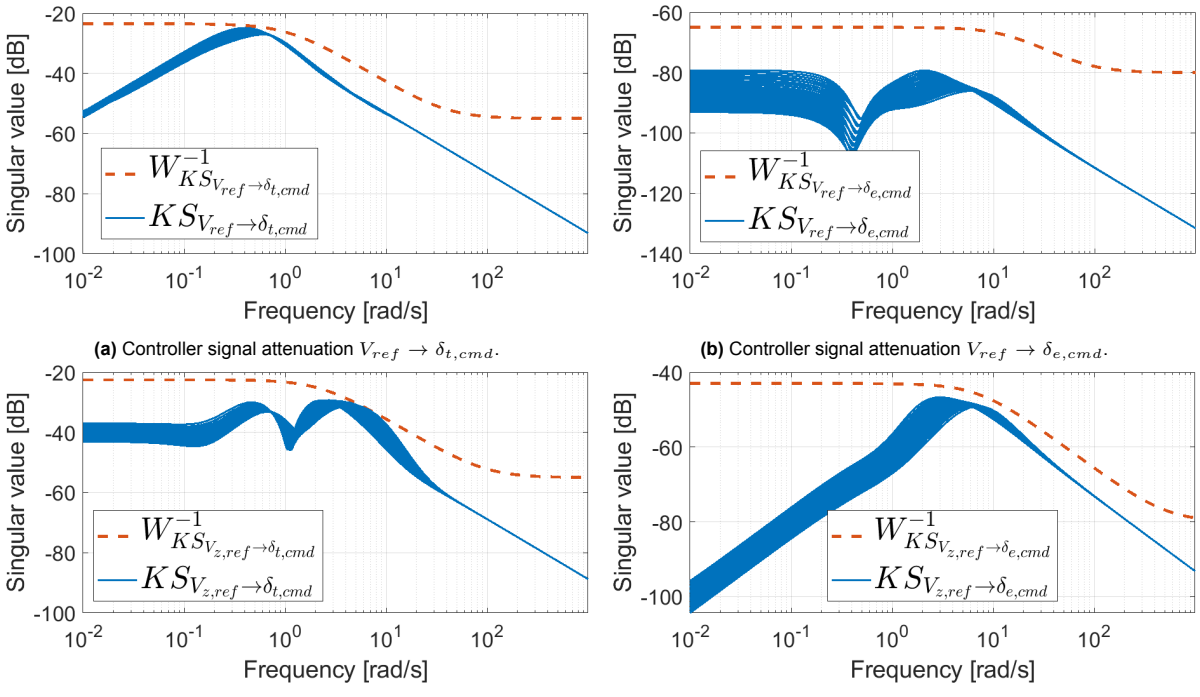
### Control signal attenuation

The transfer functions from $V_{ref}$ and $V_{z,ref}$ to $\delta_{t,cmd}$ and $\delta_{e,cmd}$ are to be restricted. This is needed to ensure that the actuator command signals are within the bandwidth of the actuator. The weighting functions in Figure 5.7 are derived from the physical actuator limits.

Starting with the engine. The engine requires a normalized input between zero and one, which is denoted by $\delta_{t,cmd}$. As a guideline, it is enforced that a change of $15\,\mathrm{m\,s^{-1}}$ in airspeed maximally causes a change in $\delta_{t,cmd}$ command of 1, that is $\Delta\delta_{t,cmd} \leq 1$ in case $\Delta V_{ref} = 15$. Thus the $KS$ function of this channel at a maximum is constrained to the value of $\frac{1}{15} = -23.5\,\mathrm{dB}$. Furthermore, a high-frequency limit of $-55\,\mathrm{dB}$ is enforced for high-frequency noise attenuation. This transfer function from $V_{ref}$ to $\delta_{t,cmd}$ and the corresponding inverted weighting function enforcing the constraint can be seen in Figure 5.7a.

Furthermore, let us consider the transfer function from $V_{z,ref}$ to $\delta_{t,cmd}$. Here, the maximum command signal was not the main concern. Instead, the attenuation of high-frequency noise was the limiting factor. It has been imposed that at $3\,\mathrm{rad\,s^{-1}}$, the maximum gain would be $-30\,\mathrm{dB}$ to achieve this goal. This constraint can be seen in Figure 5.7c.

Now let us consider the elevator command signals. As a guideline, it is proposed that a speed input of $1\,\mathrm{m\,s^{-1}}$ maximally induces a commanded elevator deflection signal of $0.4\,^\circ/\mathrm{s}$ which equals $-43\,\mathrm{dB}$ in radians. Furthermore, at the bandwidth cutoff frequency of the elevator servo ($22\,\mathrm{rads^{-1}}$), the amplification is enforced to be $10\,\mathrm{dB}$ less, namely $-53\,\mathrm{dB}$. An increase in airspeed does not require a lot of elevator adjustment as in this case, it is the engine that is limiting. To account for that the weighting function has been arbitrarily tightened, as can be seen in Figure 5.7b. On the other hand, the constraint on the elevator is much more relevant when a vertical speed command is given as much more available elevator authority is needed to ensure the aircraft pitches up or down. The weighting function and frequency response of the reference vertical speed to the elevator command signal can be seen in Figure 5.7d.




**(a)** Controller signal attenuation $V_{ref} \to \delta_{t,cmd}$.

**(b)** Controller signal attenuation $V_{ref} \to \delta_{e,cmd}$.

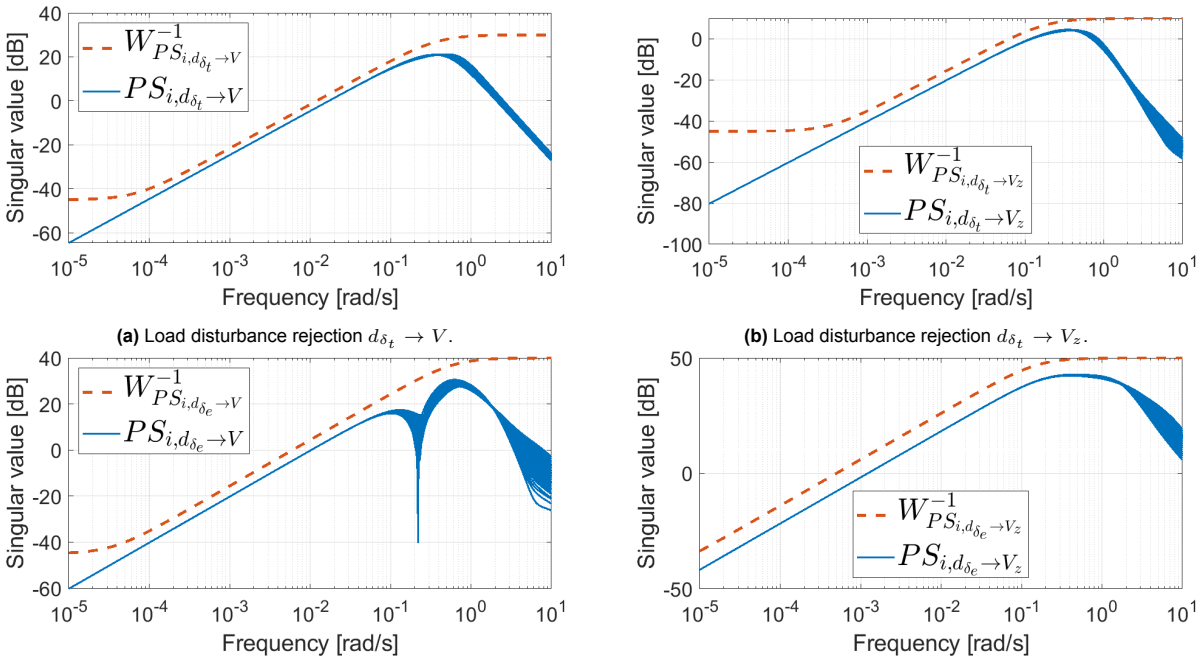
**(c)** Controller signal attenuation $V_{z,ref} \to \delta_{t,cmd}$.

**(d)** Controller signal attenuation $V_{z,ref} \to \delta_{e,cmd}$.

**Figure 5.7:** Controller signal attenuation constraints and closed-loop values.

### Input disturbance rejection

The effect of disturbances at the input to the output can be capitulated by the transfer functions of $d_{\delta_t}$ and $d_{\delta_e}$ to $V$ and $V_z$. The reason this requirement had to be included was to enforce enough integral action to ensure that these transfer functions contain proper attenuation of low-frequency disturbance signals. The requirements are not very tightly formulated as only the low-frequency attenuation was of concern. The low-frequency gain was constrained to be less than $-45$ dB and Figure 5.8 shows that this is achieved.




**(a)** Load disturbance rejection $d_{\delta_t} \to V$.

**(b)** Load disturbance rejection $d_{\delta_t} \to V_z$.

**(c)** Load disturbance rejection $d_{\delta_e} \to V$.

**(d)** Load disturbance rejection $d_{\delta_e} \to V_z$.

**Figure 5.8:** Load disturbance rejection.

### $\mathcal{D}$-Stability

Both the damping ratio and the minimum decay rate are constrained. The damping ratio has been set to $0.4$ and the minimum decay rate to $0.2$. This results in the closed-loop pole locations as shown in Figure 5.9. It can be seen that these requirements are satisfied, which ensures the desired dynamic behavior of the linear closed-loop system.

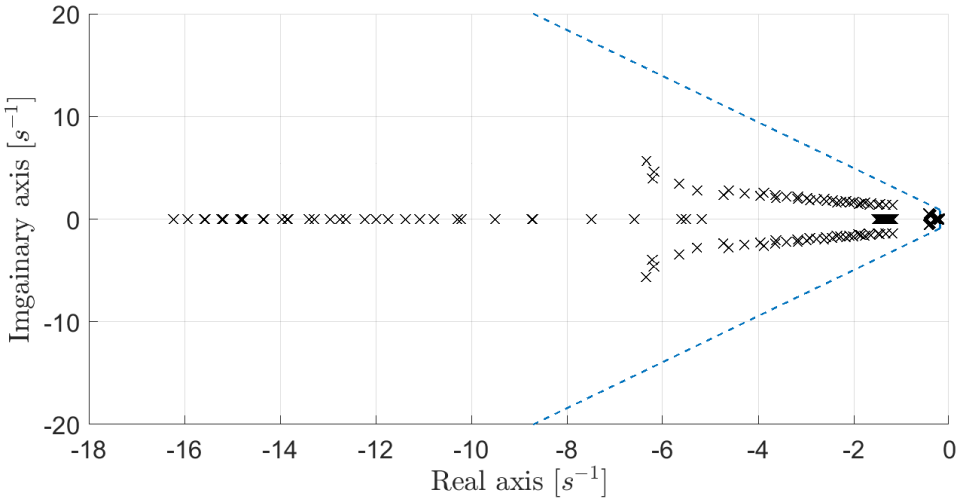



**Figure 5.9:** Pole location requirement.

### 5.1.3. Synthesized controller

All in all, the resulting controller values can be seen in Equation 5.2.

$$\begin{aligned} K_{V_{ref}}(s) &= \frac{1.3142(s+0.1603)}{(s+0.2104)} \\ K_{V_{z,ref}}(s) &= \frac{18.219(s+0.2296)}{(s+4.184)} \\ K_q(s) &= \frac{-0.0729(s+0.5185)}{(s+0.2876)} \\ K_1 &= \begin{bmatrix} -0.0168 & -0.0020 \\ -0.0002 & 0.0012 \end{bmatrix} \\ K_2 &= \begin{bmatrix} -0.1099 & 0.0114 & -0.3696 & -37.274 \\ -0.0010 & 0.0078 & 0.0538 & -4.384 \end{bmatrix} \end{aligned} \tag{5.2}$$

While the values of the first two columns in $K_1$ and $K_2$ might appear to be small, keep in mind that they translate a $\mathrm{m}/\mathrm{s}$ value into a radian value. Thus, they are definitely not equal to zero, and trying to impose that condition does reduce the performance and robustness. Furthermore, the controller has five poles. Two of these were imposed at the origin because of the integrators and the other three correspond to the first-order filters on the input.

## 5.2. Linear analysis

This section covers additional a posteriori analysis of the system. The most important linear factors were already taken into account in the optimization leading to a rather short analysis. However, the stability margins were not included, while these are naturally important robustness properties. In this case, this is the only factor being looked at in the a posteriori linear analysis.

Without imposing any constraints on the margins, the resulting values are sufficient. The loop-at-a-time $S-T$ based disk margins at the input and output are summarized in Table 5.1. At all inputs and outputs, the (disk-based) gain margins are larger than $6\,\mathrm{dB}$, and the (disk-based) phase margins are larger than $45\,°$. As these are more conservative than the classical gain and phase margins, it can be deduced that the latter also comply with the inequalities. The values of the disk margins across the whole critical frequency range can be found in Appendix D.

**Table 5.1:** Minimum loop-at-a-time disk margins at inputs and outputs.

| Location | Gain Margin [dB] | Phase Margin [°] |
|---|---|---|
| $\delta_{t,cmd}$ | 7.9 | 46.0 |
| $delta_{e,cmd}$ | 7.7 | 45.4 |
| $V$ | 7.8 | 45.5 |
| $V_z$ | 8.5 | 48.9 |
| $n_z$ | 10 | 54.8 |
| $q$ | 14.1 | 67.8 |

## 5.3. Nonlinear implementation and analysis

This section discusses the nonlinear implementation and analysis of the direct trajectory controller. As the controller does not use scheduling parameters, there are no hidden coupling terms. First, the normalized coprime stability margin is calculated, and a scaling procedure to improve its interpretability is proposed. Subsequently, the trim point uncertainty (which in this case is the only factor degrading the frozen-time robustness) during two maneuvers is assessed. The goal of this chapter is to investigate how much the frozen-time robustness is degraded and find out what the implications of this degradation are for the controller.

### 5.3.1. Normalized coprime stability margin

The normalized coprime stability margin of the linear controller-plant combination at each point in the flight envelope can be found in Figure 5.10. As can be seen, the values are very low (in the order of

$1.5 \times 10^{-3}$ ). This is partly caused by the large discrepancy in units. For normalized coprime uncertainty it is assumed that the stable uncertainty blocks are constrained by: $\| \begin{bmatrix} \tilde{\Delta}_N & \tilde{\Delta}_M \end{bmatrix} \|_\infty \leq \epsilon$ [4] [35] and this $\epsilon$ is constant across all channels. However, a perturbation of $1\,\mathrm{rad/s}$ at $q$ is not equivalent in likelihood compared to a perturbation of $1\,\mathrm{m/s}$ applied at $V$. The system is not scaled well resulting in normalized coprime uncertainty values close to zero.

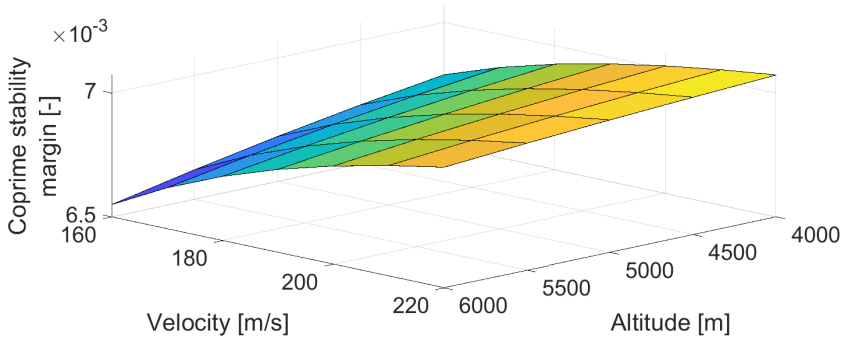



**Figure 5.10:** Normalized coprime stability margin of direct trajectory longitudinal controller at equilibria.

To remedy this problem a slightly different formulation of the plant is used. Let us define a diagonal $n \times n$ scaling matrix $N$, where $n$ denotes the number of outputs of the system. If this scaling matrix is introduced to the system as shown in Figure 5.11, this does not change the system. By manipulating the block diagram, it is possible to get the following equivalent system

$$\frac{PK}{1+PK} = N^{-1}\frac{NPKN^{-1}}{1+NPKN^{-1}}N = N^{-1}\frac{\hat{P}\hat{K}}{1+\hat{P}\hat{K}}N \tag{5.3}$$

with:

$$\hat{P} = NP, \text{ and } \hat{K} = KN^{-1}$$

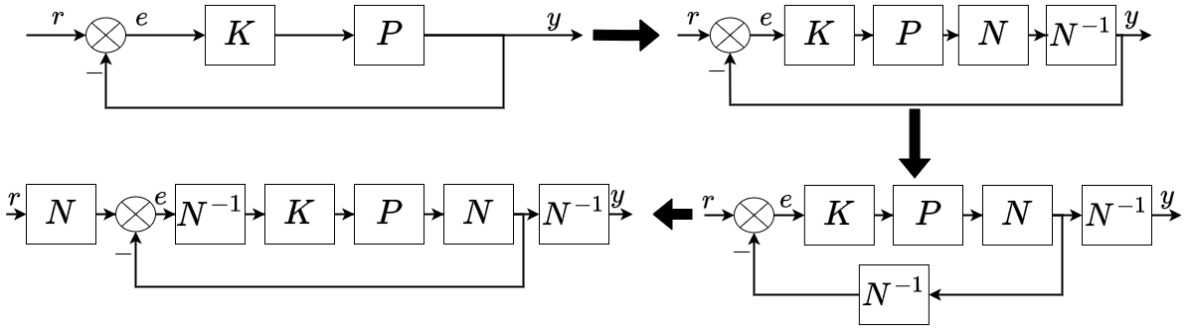



**Figure 5.11:** Block diagram manipulations for proposed scaling procedure.

As the two systems are equivalent, the poles need to be equivalent to the original system. The multiplication from the left with $N^{-1}$ and from the right with $N$ does not change the location of the poles as long as the matrix $N$ does not contain any dynamics. To prove this consider two state space systems $P_1$ and $P_2$:

$$P_1 = \left[\begin{array}{c|c} A_1 & B_1 \\ \hline C_1 & D_1 \end{array}\right], \quad P_2 = \left[\begin{array}{c|c} A_2 & B_2 \\ \hline C_2 & D_2 \end{array}\right]. \tag{5.4}$$

If the two systems are cascaded, where the output of $P_2$ is the input of $P_1$. This results in [53, p 34]:

$$\begin{aligned} P_1P_2 &= \left[\begin{array}{c|c} A_1 & B_1 \\ \hline C_1 & D_1 \end{array}\right]\left[\begin{array}{c|c} A_2 & B_2 \\ \hline C_2 & D_2 \end{array}\right] \\ &= \left[\begin{array}{cc|c} A_1 & B_1C_2 & B_1D_2 \\ 0 & A_2 & B_2 \\ \hline C_1 & D_1C_2 & D_1D_2 \end{array}\right] \end{aligned} \tag{5.5}$$

The stability of the cascaded state space system is purely reflected in the eigenvalues of the new state matrix ($A$). If $P_1$ is a static system this indicates that $A_1 = B_1 = C_1 = 0$ and only $D_1 \neq 0$. The state space realization of the cascading system reduces to:

$$P_1P_2 = \left[\begin{array}{cc|c} 0 & 0 & 0 \\ 0 & A_2 & B_2 \\ \hline 0 & D_1C_2 & D_1D_2 \end{array}\right] = \left[\begin{array}{c|c} A_2 & B_2 \\ \hline D_1C_2 & D_1D_2 \end{array}\right] \tag{5.6}$$

From here it can be deduced that the state matrix of $P_1P_2$ is equal to $A_2$ and thus that they have the same eigenvalues. The same proof can be used in case $P_2$ is the static system instead of $P_1$, with the eigenvalues of the $A_1$ matrix then being equal to the eigenvalues of the state matrix of the cascaded system.

Hence if we can guarantee the stability of the scaled system $(\hat{P}, \hat{K})$, it also guarantees the stability of the original $(P, K)$ system, given matrix $N$ does not contain any dynamics. The matrix $N$ can thus be chosen such that the scaling of the system is improved. This reduces the conservativeness of the normalized coprime stability margin and improves the intuitive interpretation of the values.

Applying this to the coprime stability margin at every point yields the values of Figure 5.12. These are indeed higher values compared to Figure 5.11. The values used for $N$ are:

$$N = diag\{[0.1\ 0.1\ 0.1\ 0.1\ 0.5\ 3]\} \tag{5.7}$$

These values were found by numerical optimization using *fmincon* of the coprime stability margin. This leads to an increase of $10$ times the original margin values and facilitates better interpretation of the normalized coprime stability margin. The scaled plants and controllers are used in the upcoming section.

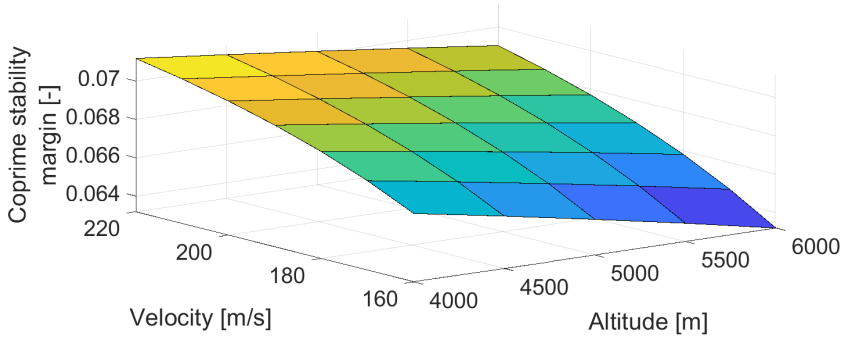



**Figure 5.12:** Scaled normalized coprime stability margin.

### 5.3.2. Trim point uncertainty and frozen-time robustness

Since the multi-model approach was used to design one unique controller for this part of the flight envelope, there are no scheduling parameters and hence also no hidden coupling terms. Thus, the only factor of importance for the frozen-time robustness is the trim point uncertainty. Subsequently, to assess the trim point uncertainty a similar analysis compared to Section 4.4.3 is included, and the same velocity-form of the plant is used to find the off-equilibrium plant linearizations.

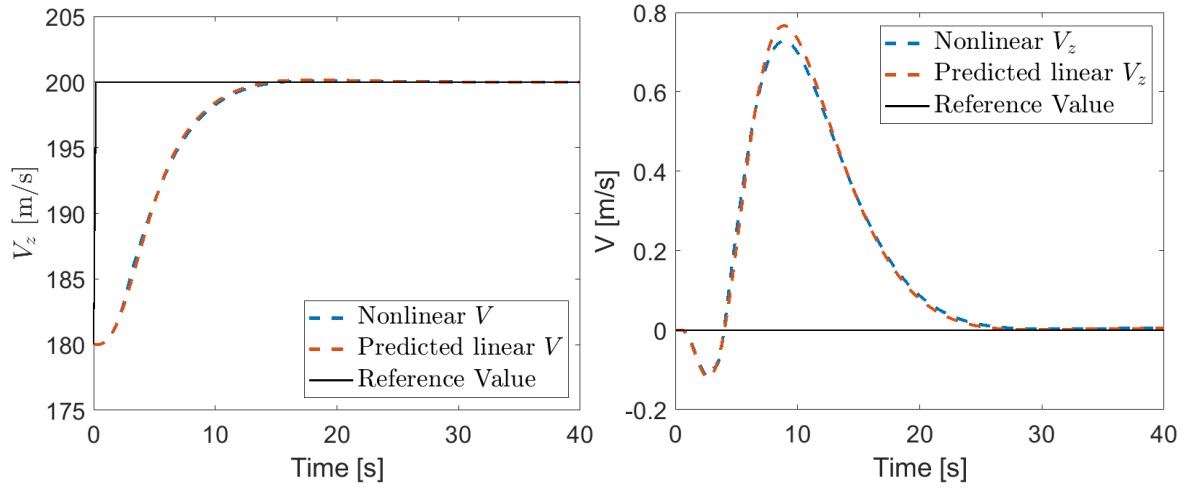
In this case, two typical maneuvers are considered. First, an airspeed step response is investigated and subsequently, the response to a vertical speed square signal is looked at. Let us commence with investigating the step response of an airspeed input. The resulting response in airspeed and vertical air speed can be seen in Figure 5.13. The reference input is tracked very similarly to the predicted linear response.

Now we are interested if the robustness is preserved during this maneuver. At an operating point, the velocity-form can be used to find the linearized plant $P_{off}$. This plant can be scaled to $\hat{P}_{off}$ using

the matrix $N$ of the previous section. The scaled controller ($\hat{K}$) is fixed and does not change during operation. Subsequently, it is possible to calculate the absolute coprime stability margin at the operating point, by adapting the equation for the normalized stability margin (see Equation 3.37):

$$b(\hat{P}_{off}, \hat{K}) = \left\| \begin{bmatrix} \hat{K} \\ I \end{bmatrix} (I + \hat{P}_{off}\hat{K})^{-1}(M^T) - 1] \right\|_{\infty}^{-1} = \left\| \begin{bmatrix} I \\ \hat{K} \end{bmatrix} (I + \hat{P}_{off}\hat{K})^{-1} \begin{bmatrix} I & \hat{P}_{off} \end{bmatrix} \right\|_{\infty}^{-1} \quad (5.8)$$

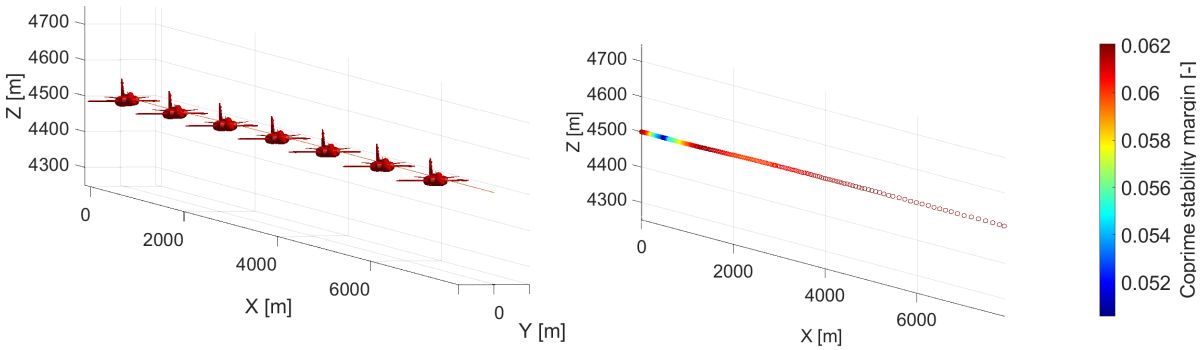
Now this equation can be applied at each discrete point along the trajectory of the manouvre. The trajectory of the aircraft in 3D space is shown in Figure 5.14a and is not very surprising as the aircraft simply retains attitude while moving forward faster. Then Figure 5.14b shows the result of Equation 5.8. Here the absolute scaled normalized coprime stability margin can be seen at the discrete points of the trajectory. It can be inferred that the aircraft loses a bit of coprime stability margin when it moves from the initial equilibrium to the second equilibrium. During the acceleration, the coprime margin becomes a little smaller but only by about $15\%$. It can thus be concluded that during an increase in airspeed, the robustness with respect to the coprime stability margin is maintained. This is the case as the trajectory moves relatively close to the equilibria points, causing the encountered trim point uncertainty during an increase or decrease in airspeed to be limited.




**(a)** Airspeed step response.

**(b)** Vertical speed response.

**Figure 5.13:** Nonlinear simulation of direct trajectory longitudinal controller with reference velocity input.



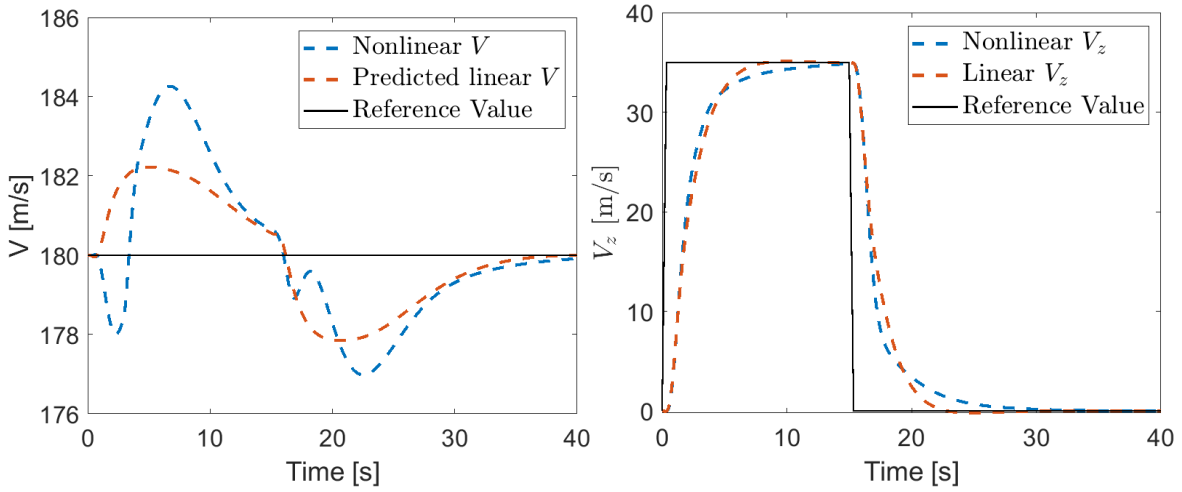

**(a)** Aircraft trajectory and attitude during manoeuvre of Figure 5.14

.

**(b)** Absolute coprime stability margin along the trajectory of airspeed step response.

**Figure 5.14:** Nonlinear simulation of direct trajectory longitudinal controller with reference velocity input.

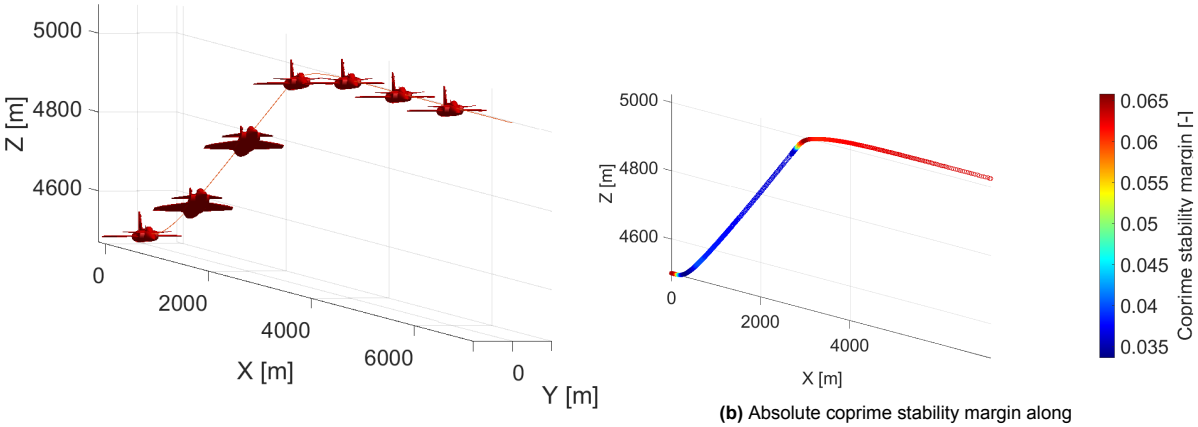
Subsequently, a vertical speed input is considered. Intuitively, the robustness is supposed to degrade a little more during such a maneuver as it requires further distance from the equilibrium points. The nonlinear response is shown in Figure 5.15. It can already be seen that the agreement with the linear simulation is slightly worse compared to the previous maneuver and, as will be seen, this is also supported by looking at the off-equilibria conditions. The trajectory in 3D space including the aircraft attitude is shown in Figure 5.16a. The corresponding scaled normalized coprime stability margin along discrete points of the trajectory can be seen in Figure 5.16b. During this maneuver, the value of the coprime stability margin becomes much smaller compared to the previous maneuver. The degradation that is encountered is around $50\%$. This is caused by the aircraft having to traverse a state further from the equilibria points for which it was not designed as it needs to pitch up to reach the desired flight path.




**(a)** Airspeed response.

**(b)** Vertical speed step response response.

**Figure 5.15:** Nonlinear simulation of trajectory with reference vertical speed.




**(a)** Attitude along the trajectory.

**(b)** Absolute coprime stability margin along the trajectory of the vertical speed step response.

**Figure 5.16:** Nonlinear simulation of longitudinal trajectory controller in 3D space following $35\,\mathrm{m\,s^{-1}}$ vertical speed reference.

Hence what can be concluded regarding the frozen-time robustness of the controller given a vertical speed input? First of all, it is clear that there is a significant penalty with respect to the normalized coprime stability margin when performing a vertical speed maneuver with a significant amplitude (which is not the case when increasing the airspeed). This is also reflected in the performance of the controller in the nonlinear simulation compared to the linear simulation of Figure 5.15. Since the controller showed very robust properties on the equilibrium points this points toward sufficient robustness still being maintained for manoeuvres using a similar vertical speed profile.

Nevertheless, while the controller shows reasonable performance and frozen-time robustness, it could still be an interesting topic for future research to incorporate the trim point uncertainty in the design process. This could also ensure robustness given even larger vertical speed input signals (although saturation of the command signal would also need to be taken into account). The multi-model framework seems like a natural way to incorporate off-equilibrium plants in the design procedure.

However, while this was attempted, it turned out to greatly increase the difficulty of synthesizing a controller that fulfills all the constraints of Section 5.1, while including off-equilibrium plants. For example, the off-equilibrium plant that contained a coprime stability margin of around $0.035$ turned out to be very different from the closest equilibrium plant making it hard to find a single controller that can control both plants well. Subsequently, it might be possible to use constraints that differ per operating point, i.e. varying goals for each plant. However, this increases the complexity of the synthesis significantly because of the many possible off-equilibrium plants that could be taken into account. Furthermore, there are also no clear guidelines for off-equilibria design constraints which will have to be created. However, this topic could be of interest in further research with the aim of reducing the degradation of the frozen-time robustness due to trim-point uncertainty.

# 6

# Conclusion & Recommendations

This final chapter concludes the research. The problem statement is examined and the extent to which this question could be addressed is summarized. Thereafter, topics that were touched upon during this research and could be of interest in further research are discussed, even though, as of now they were not able to come to full fruition.

## 6.1. Conclusion

To establish the conclusion of the thesis, let us return to the overall problem statement posed in the introduction:

> *To what extent is it possible to* *assess the frozen-time robustness* (6.1.2) *of the nonlinear realization of a* *structured $\mathcal{H}_\infty$ controller?* (6.1.1)

The conclusion is split into two parts. First, the design and synthesis of robust scheduled structured $\mathcal{H}_\infty$ controllers are analyzed. The achieved goals are discussed in Section 6.1.1. Secondly, the nonlinear controller implementation and how it relates to (frozen-time) robustness is concluded upon in Section 6.1.2.

### 6.1.1. Structured $\mathcal{H}_\infty$ controller synthesis

Chapter 4 and Chapter 5 discussed the procedure of synthesizing a pitch-rate and a direct trajectory longitudinal controller using gain-scheduled $\mathcal{H}_\infty$. The design procedure was performed by optimizing and/or analyzing the following properties:

- Reference input tracking
- Output disturbance rejection
- Controller signal attenuation
- Input disturbance rejection
- Stability margins
- Closed-loop pole locations
- Sensor noise attenuation

In the case of the pitch-rate controller, the design was straightforward because of the simpler structure of the problem, as the plant only consisted of 2 outputs and 1 input. For the flight-path controller, the situation was more complicated. The plant consisted of 2 inputs and 4 outputs, two of which needed to be tracked. Furthermore, the coupling of the system enforced constraints on the off-diagonal components of the weighting matrices. These were individually considered for maximum control of the result.

### 6.1.2. Frozen-time robustness

Chapter 3 considered the problems encountered when using gain scheduling and multi-model synthesis for nonlinear control. It was investigated how at each point in the operating domain, the effects of hidden coupling (if applicable) and trim point uncertainty could be quantified. Ensuring robustness regarding these two phenomena was referred to as frozen-time robustness and this was assessed using the coprime stability margin.

The velocity-based linearization framework of Leith and Leithead [24] allowed us to linearize at every operating point. This could be applied to the hidden coupling terms, which were investigated using the framework of Lhachemi, Saussié, and Zhu [27]. Furthermore, this framework was also directly applied to the plant itself, yielding a look into the trim point uncertainty. By making use of velocity-based linearization, the coprime stability margin along a trajectory could be quantified, with respect to degradation because of hidden coupling and trim point uncertainty. This can be extended to the whole operating envelope of the aircraft to ensure frozen-time stability and robustness against coprime factor uncertainty at each point.

The theory was applied to the synthesized pitch-rate and direct trajectory longitudinal controller. By making use of the (minimum) coprime stability margin the local stability guarantees given by linear design synthesis could be extended to include hidden coupling and trim point uncertainty. Furthermore, the coprime stability margin could be used to identify possible problems regarding these two factors, improving the possible analysis in the nonlinear domain.

## 6.2. Recommendations

During the research for the thesis, many topics and/or problems were encountered that could presently not be solved. This section discusses the main points that seem promising for further research.

### 6.2.1. Increased frozen-time analysis and synthesis

The thesis only considered the coprime stability margin to assess the frozen time robustness of the controlled system. This was a convenient method to assess the robustness against unstructured coprime uncertainty, which is of course a very fundamental form of uncertainty. However in theory it is possible to use far more extensive analysis to analyze the frozen-time systems. Possibly loop transfer function analysis, as done in the design procedure, could be performed. Robustness against different kinds of uncertainties could also be investigated or more emphasis could be put on guaranteeing performance instead of only guaranteeing stability.

Furthermore, it might be possible to directly incorporate the frozen-time systems into the design procedure. An example of accomplishing this could be by using additional scheduling parameters such as $\alpha$, $\delta_e$ and $q$ to capture off-equilibrium dynamics in the synthesis. However, three main problems were encountered when trying to apply this to the F-16 control systems.

First, once additional parameters are considered, it is not necessarily true anymore that each point in the scheduling set maps to a single point in the nonlinear dynamics. To give an example, one could envision a controller that uses $V$, $z^E$ and $\alpha$ as scheduling parameters. Now let us schedule on a point where these are equal to $200\,\mathrm{m/s}$, $1000\,\mathrm{m}$ and $3^\circ$ respectively. In off-equilibria situations, this can be possible at a moment during which the aircraft is pitching up with a pitch-rate of $2^\circ$/s. However, the aircraft could just as well be pitching down by -$10^\circ$/s. Both situations can result in different aerodynamic coefficients and thus different dynamics. It is therefore hard to establish a scheduling scheme that properly captures the off-equilibria dynamics while using a subset of the outputs as scheduling parameters.

Secondly, the parameters that capture off-equilibrium conditions are often by nature also fast-varying. Off-equilibrium dynamics are inherently only temporary as the aircraft constantly moves between equilibria points. In the case of the F-16 aircraft, these are parameters such as $\alpha$ and $q$. However, this can cause the problem that the fast variation of scheduling parameters runs into the pitfall of LTV/LPV discrepancy with a linearized system as described by Section 3.3. This complicates the scheduling approach and in itself can, if not careful, destabilize the closed-loop system.

Lastly, the dimension of the controller increases exponentially with each additional scheduling parameter. Hence, the computational complexity of the controller also increases exponentially.

Another approach could be by incorporating off-equilibrium plants in a multi-model approach. However, this was met with another set of problems. In the application of the direct trajectory longitudinal controller, the off-equilibria plants of interest were so different in comparison to the equilibria plants that the same constraints for both types of plants could not be satisfied simultaneously. Thus, some kind of varying goal would need to be used but this increases the complexity of the controller synthesis significantly. There is no real guideline on choosing which off-equilibrium plants to include and which constraints to enforce and there are many permutations possible. However, this could be of interest in further research.

All in all, these encountered problems are not easy to overcome and turned out to be too complex for the scope of this thesis. However, it seems that there could be improvements possible in this area, either from the nonlinear analysis or from trying to improve the synthesis itself.

### 6.2.2. Velocity-based linearization using tensors

During the research, it was investigated if the tensor formulations could be unified with the velocity-based linearization concept. To investigate the topic let us try to derive the velocity-form, from a tensor point of view.

Let us start with the attitude dynamics. Equation 2.41 is used as starting point. This equation was:

$$I_B^B D^B \omega^{BE} + \Omega^{BE} I_B^B \omega^{BE} = m_B \tag{6.1}$$

Multiplying both sides with $D^B$ yields:

$$D^B I_B^B D^B \omega^{BE} + D^B(\Omega^{BE} I_B^B \omega^{BE}) = D^B m_B \tag{6.2}$$

Applying the chain rule, assuming a rigid body and solving for the acceleration derivatives results in:

$$D^B D^B \omega^{BE} = (I_B^B)^{-1}\left(-(D^B \Omega^{BE}) I_B^B \omega^{BE} - \Omega^{BE} I_B^B (D^B \omega^{BE}) + D^B m_B\right) \tag{6.3}$$

Now projecting on the body frame:

$$\left[D^B D^B \omega^{BE}\right]^B = (\left[I_B^B\right]^B)^{-1} \cdot \Big(-\left[D^B \Omega^{BE}\right]^B \left[I_B^B\right]^B \left[\omega^{BE}\right]^B - \left[\Omega^{BE}\right]^B \left[I_B^B\right]^B \left[D^B \omega^{BE}\right]^B + \left[D^B m_B\right]^B\Big) \tag{6.4}$$

Working this out in a coordinated form produces:

$$\begin{aligned}\begin{bmatrix} \ddot{p} \\ \ddot{q} \\ \ddot{r} \end{bmatrix}^B = (\left[I_B^B\right]^B)^{-1}\Bigg( &- \begin{bmatrix} 0 & -\dot{r} & \dot{q} \\ \dot{r} & 0 & -\dot{p} \\ -\dot{q} & \dot{p} & 0 \end{bmatrix}^B (\left[I_B^B\right])^B \begin{bmatrix} p \\ q \\ r \end{bmatrix}^B \\ &- \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}^B (\left[I_B^B\right]^B)^B \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix}^B + \left[D^B m_B\right]^B \Bigg)\end{aligned} \tag{6.5}$$

With:

$$\left[I_B^B\right]^B = \begin{bmatrix} I_{11} & I_{12} & I_{13} \\ I_{21} & I_{22} & I_{23} \\ I_{31} & I_{32} & I_{33} \end{bmatrix}^B \tag{6.6}$$

Every term is familiar except for $\left[D^B m_B\right]^B$, which is the derivative of the aerodynamic forces. To compute this, let $[m_B]^B$ be a function, denoted by $f$ that depends on all variables governing the aerodynamic forces, namely $V$, $\alpha$, $\beta$, $p$, $q$, $r$, $\delta_a$, $\delta_e$, $\delta_r$, $\delta_l$ and let $\Gamma$ denote the set of all these variables on which $[m_B]^B$ depends; mathematically this means that:

$$[m_B]^B = [f(V, \alpha, \beta, p, q, r, \delta_a, \delta_e, \delta_r, \delta_l)]^B = [f(\Gamma)]^B \tag{6.7}$$

Then the derivative can be found by simply employing the product rule, with respect to all the dependent variables (as was also shown in Section 4.4.3):

$$\left[D^B m_B\right]^B = \begin{bmatrix} \frac{df_1(\Gamma)}{dt} \\ \frac{df_2(\Gamma)}{dt} \\ \frac{df_3(\Gamma)}{dt} \end{bmatrix}^B = \begin{bmatrix} \sum_{n\in\Gamma} \frac{\partial f_1(\Gamma)}{\partial n} \cdot \frac{dn}{dt} \\ \sum_{n\in\Gamma} \frac{\partial f_2(\Gamma)}{\partial n} \cdot \frac{dn}{dt} \\ \sum_{n\in\Gamma} \frac{\partial f_3(\Gamma)}{\partial n} \cdot \frac{dn}{dt} \end{bmatrix}^B \tag{6.8}$$

That is, the derivative can thus be computed by multiplying the numerical approximated derivative of the look-up tables with respect to the dependent variables with the partial derivatives of the dependent variables with respect to time, and subsequently summing these. The former is simply a value depending on the interpolation of the look-up tables while the latter produces states and inputs for the velocity-form ($\dot{V}$, $\dot{\alpha}$, $\dot{\beta}$, $\dot{p}$, $\dot{q}$, $\dot{r}$, $\dot{\delta_a}$, $\dot{\delta_e}$, $\dot{\delta_r}$, $\dot{\delta_l}$).

Combining these results gives us directly the velocity-based linearization of the system. For each operating point, the system is linear in $\dot{p}, \dot{q}$ and $\dot{r}$. The velocity-based linearized attitude dynamics can be directly derived from the tensor equation.

However, let us now move to the derivation of the velocity-form of the translational dynamics. Equation 2.24 is used as starting point:

$$mD^B v_B^E + m\Omega^{BE} v_B^E = f_{a,b} + mg \tag{6.9}$$

Subsequently, it is possible to multiply both sides with the rotational derivative in the body frame $D^B$:

$$D^B(mD^B v_B^E) + D^B(m\Omega^{BE} v_B^E) = D^B f_{a,b} + D^B(mg) \tag{6.10}$$

$D^B(mg)$ is assumed to be $0$, that is there are no changes in the gravitational field and the body is assumed to be rigid. Furthermore, the equation can be solved for the derivatives of the linear acceleration by making use of the chain rule (which applies to the rotational time derivative [55, p 105]):

$$mD^B D^B v_B^E = -m(D^B\Omega^{BE})v_B^E - m\Omega^{BE}(D^B v_B^E) + D^B f_{a,b} \tag{6.11}$$

Now coordinating to the body frame implies:

$$\left[D^B D^B v_B^E\right]^B = -\left[D^B\Omega^{BE}\right]^B \left[v_B^E\right]^B - \left[\Omega^{BE}\right]^B \left[D^B v_B^E\right]^B + m^{-1}\left[D^B f_{a,b}\right]^B \tag{6.12}$$

Solving this in matrix form constructs:

$$\begin{aligned} \begin{bmatrix} \ddot{u} \\ \ddot{v} \\ \ddot{w} \end{bmatrix}^B = &- \begin{bmatrix} 0 & -\dot{r} & \dot{q} \\ \dot{r} & 0 & -\dot{p} \\ -\dot{q} & \dot{p} & 0 \end{bmatrix}^B \begin{bmatrix} u \\ v \\ w \end{bmatrix}^B \\ &- \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}^B \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix}^B + m^{-1}\left[D^B f_{a,b}\right]^B \end{aligned} \tag{6.13}$$

This results in the velocity-form of the system in the body coordinate system. There is, however, still a problem with this expression. For our synthesis, the linearized equations were expressed in the wind coordinate system. The expression of Equation 6.13 is in the body coordinate system. As of now, it is unclear to the author how to directly use tensors to express the differential equation in the wind coordinate system. The procedure used to get the differential equations in the wind coordinate system was to first coordinate the system and subsequently solve for the relevant transformation angles derivatives that appeared in the angular velocity vectors. This is only possible after coordination as the elements of the coordinate system are part of the result and inherently not a tensor.

A similar problem is encountered when exploring the attitude kinematics. Also here, the elements in the transformation matrices are used to get the expression for the differential equations. It seems possible that there could be a method to directly get an expression for both the attitude kinematics and translational dynamics in the wind frame using tensors but further investigation would be necessary. However, as of now, to perform the velocity-based linearization, the equations of Chapter 2 were simply used in the coordinated form.

### 6.2.3. Including uncertainty in linear synthesis

In many applications of robust control design, the controller is also made robust to parametric uncertainty. That is uncertainty analysis is performed regarding the parameters used in the model. Each parameter always includes an uncertainty margin. For example, the actual center of gravity varies and is not the same as the one used in the model. The same can be said for the mass, moments of inertia, aerodynamic coefficients, and many more parameters.

In the conventional robust design procedure, this uncertainty analysis is included to ensure the controller still performs well during actual flight. This analysis was omitted in the current thesis because of time constraints.

Furthermore, a particularly interesting application of using parametric uncertainty to increase robustness is Passive Fault Tolerance. There are system faults encountered in flight that can be modeled. Examples are an increase of sensor noise/bias or complete sensor failure, actuators jamming or losing control effectiveness and many other scenarios could be envisioned. Including these uncertainties in the robust design procedure yields an interesting application of fault-tolerant design. This is currently an active field of research [30] and the application on the F-16 model could be an interesting application of the research.

### 6.2.4. Flight-path controller allocation problem and over-actuation

In Chapter 5, it was mentioned that the direct trajectory longitudinal control problem is inherently also an allocation problem, in case of insufficient available thrust. In this thesis, it was assumed that this situation would not be encountered but that is of course unrealistic. Energy allocation methods or other control allocation methods can be interesting to apply to the F-16 longitudinal trajectory control.

This also brings us to the attention of over-actuation and integral windup which was not considered in this design. A disadvantage of the $\mathcal{H}_\infty$ design procedure is that it cannot directly include this nonlinear phenomenon in the optimization procedure. To fully complete the robust flight-path angle controller these issues would have to be taken into account and could be a topic for further research.

# Bibliography


[1] J Aggarwal and E Infante. “Some remarks on the stability of time-varying systems”. In: *IEEE Transactions on Automatic Control* 13.6 (1968), pp. 722–723.

[2] Pierre Apkarian and Dominikus Noll. “Nonsmooth H∞ synthesis”. In: *IEEE Transactions on Automatic Control* 51.1 (2006), pp. 71–86.

[3] Pierre Apkarian and Dominikus Noll. “The H∞ control problem is solved”. In: *Aerospace Lab* 13 (2017), pages–1.

[4] Declan Bates and Ian Postlethwaite. *Robust multivariable control of aerospace systems*. Vol. 8. IOS Press, 2002.

[5] Caroline Bérard, Jean-Marc Biannic, and David Saussié. *La commande multivariable: Application au pilotage d’un avion*. Dunod, 2012.

[6] Jean-Marc Biannic and Clément Roos. “Robust Autoland Design by Multi-Model H∞ Synthesis with a Focus on the Flare Phase”. In: *Aerospace* 5.1 (2018), p. 18.

[7] Jean-Luc Boiffier. “Dynamics of flight: the equations”. In: *(No Title)* (1998).

[8] Mahmoud Chilali, Pascal Gahinet, and Pierre Apkarian. “Robust pole placement in LMI regions”. In: *IEEE transactions on Automatic Control* 44.12 (1999), pp. 2257–2270.

[9] Peter Dorato. “A historical review of robust control”. In: *IEEE Control Systems Magazine* 7.2 (1987), pp. 44–47. ISSN: 0272-1708.

[10] John Doyle. “Robust and optimal control”. In: *Proceedings of 35th IEEE Conference on Decision and Control*. Vol. 2. IEEE. 1996, pp. 1595–1598.

[11] Keith Glover and Duncan McFarlane. “Robust stabilization of normalized coprime factor plant descriptions with H/sub infinity/-bounded uncertainty”. In: *IEEE transactions on automatic control* 34.8 (1989), pp. 821–830.

[12] Keith Glover, Glenn Vinnicombe, and George Papageorgiou. “Guaranteed multi-loop stability margins and the gap metric”. In: *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)*. Vol. 4. IEEE. 2000, pp. 4084–4085.

[13] Ilkay Gumusboga. “Robust Pitch Rate Control Augmentation System for an F-16 Aircraft”. In: *Journal of Aeronautics and Space Technologies* 14.2 (2021), pp. 243–249.

[14] Florian Holzapfel et al. “Flight dynamics aspects of path control”. In: *AIAA Guidance, Navigation and Control Conference and Exhibit*. 2007, p. 6772.

[15] Achim Ilchmann, Dieter H Owens, and Dieter Prätzel-Wolters. “Sufficient conditions for stability of linear time-varying systems”. In: *Systems & control letters* 9.2 (1987), pp. 157–163.

[16] Tor Arne Johansen et al. “Off-equilibrium linearisation and design of gain-scheduled control with application to vehicle speed control”. In: *Control Engineering Practice* 6.2 (1998), pp. 167–180.

[17] Isaac Kaminer et al. “A velocity algorithm for the implementation of gain-scheduled controllers”. In: *Automatica* 31.8 (1995), pp. 1185–1191.

[18] Erik Karlsson et al. “Dynamic flight path control coupling for energy and maneuvering integrity”. In: *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE. 2016, pp. 1–6.

[19] A Lambregts. “Vertical flight path and speed control autopilot design using total energy principles”. In: *Guidance and Control Conference*. 1983, p. 2239.

[20] AA Lambregts. “Functional integration of vertical flight path and speed control using energy principles”. In: *NASA. Langley Research Center NASA Aircraft Controls Research, 1983* (1984).

[21] Douglas A Lawrence and Wilson J Rugh. “Gain scheduling dynamic linear controllers for a nonlinear plant”. In: *Automatica* 31.3 (1995), pp. 381–390.

[22] Douglas J Leith. “Input-output linearization by velocity-based gain-scheduling”. In: *International Journal of Control* 72.3 (1999), pp. 229–246. ISSN: 0020-7179.

[23] Douglas J Leith and WE Leithead. “Gain-scheduled and nonlinear systems: dynamic analysis by velocity-based linearization families”. In: *International Journal of Control* 70.2 (1998), pp. 289–317.

[24] Douglas J Leith and WE Leithead. “Gain-scheduled controller design: an analytic framework directly incorporating non-equilibrium plant dynamics”. In: *International Journal of Control* 70.2 (1998), pp. 249–269. ISSN: 0020-7179.

[25] Douglas J Leith and William E Leithead. “Survey of gain-scheduling analysis and design”. In: *International journal of control* 73.11 (2000), pp. 1001–1025. ISSN: 0020-7179.

[26] Douglas J. Leith and William E. Leithead. “Gain-Scheduled Control: Relaxing Slow Variation Requirements by Velocity-Based Design”. In: *Journal of Guidance, Control, and Dynamics* 23.6 (2000), pp. 988–1000. ISSN: 0731-5090. DOI: 10.2514/2.4667. URL: https://dx.doi.org/10.2514/2.4667.

[27] Hugo Lhachemi, D Saussié, and G Zhu. “Gain-scheduling control design in the presence of hidden coupling terms”. In: *Journal of guidance, control, and dynamics* 39.8 (2016), pp. 1871–1879.

[28] Hugo Lhachemi, David Saussié, and Guchuan Zhu. “Handling hidden coupling terms in gain-scheduling control design: Application to a pitch-axis missile autopilot”. In: *AIAA guidance, navigation, and control conference*. 2016, p. 0365.

[29] Bei Lu and Fen Wu. “Probabilistic robust control design for an f-16 aircraft”. In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. 2005, p. 6080.

[30] Julien Marzat et al. “Model-based fault diagnosis for aerospace systems: a survey”. In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of aerospace engineering* 226.10 (2012), pp. 1329–1360.

[31] Duncan C McFarlane and Keith Glover. *Robust controller design using normalized coprime factor plant descriptions*. Springer, 1990.

[32] ANON MIL-STD. *Department of Defence interface standard—flying qualities of piloted aircraft*. 1990.

[33] Luat T Nguyen. *Simulator study of stall/post-stall characteristics of a fighter airplane with relaxed longitudinal static stability*. Vol. 12854. National Aeronautics and Space Administration, 1979.

[34] Robert A Nichols, Robert T Reichert, and Wilson J Rugh. “Gain scheduling for H-infinity controllers: A flight control example”. In: *IEEE Transactions on Control systems technology* 1.2 (1993), pp. 69–79.

[35] Ian R Petersen and Roberto Tempo. “Robust control of uncertain systems: Classical results and recent developments”. In: *Automatica* 50.5 (2014), pp. 1315–1335.

[36] Wilson J Rugh. “Analytical framework for gain scheduling”. In: *1990 American Control Conference*. IEEE. 1990, pp. 1688–1694.

[37] Wilson J Rugh and Jeff S Shamma. “Research on gain scheduling”. In: *Automatica* 36.10 (2000), pp. 1401–1425.

[38] Richard S Russell. “Non-linear F-16 simulation using Simulink and Matlab”. In: *University of Minnesota, Tech. paper* (2003).

[39] Michael G Safonov. “Origins of robust control: Early history and future speculations”. In: *Annual Reviews in Control* 36.2 (2012), pp. 173–181.

[40] Scordamaglia, Valerio. *Trajectory and Attitude Plot Version 3*. MATLAB Central File Exchange. Version 3. URL: https://www.mathworks.com/matlabcentral/fileexchange/5656-trajectory-and-attitude-plot-version-3.

[41] Peter Seiler, Andrew Packard, and Pascal Gahinet. “An introduction to disk margins [lecture notes]”. In: *IEEE Control Systems Magazine* 40.5 (2020), pp. 78–95.

[42] Jeff S Shamma, Michael Athans, et al. “Analysis of gain scheduled control for linear parameter-varying plants”. In: (1988).

[43] Jeff S Shamma and Michael Athans. “Gain scheduling: Potential hazards and possible remedies”. In: *IEEE Control Systems Magazine* 12.3 (1992), pp. 101–107.

[44] Lars Sonneveldt et al. “Nonlinear adaptive trajectory control applied to an F-16 model”. In: *Journal of Guidance, control, and Dynamics* 32.1 (2009), pp. 25–39.

[45] Robert F Stengel. *Flight dynamics*. Princeton university press, 2005.

[46] Brian L Stevens, Frank L Lewis, and Eric N Johnson. *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015. ISBN: 1118870980.

[47] Julian Theis et al. “Robust autopilot design for landing a large civil aircraft in crosswind”. In: *Control Engineering Practice* 76 (2018), pp. 54–64.

[48] Spilios Theodoulis and Michael Proff. “Robust flight control tuning for highly agile missiles”. In: *AIAA Scitech 2021 Forum*. 2021, p. 1568.

[49] M Wu. “A note on stability of linear time-varying systems”. In: *IEEE transactions on Automatic Control* 19.2 (1974), pp. 162–162.

[50] George Zames. “Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses”. In: *IEEE Transactions on automatic control* 26.2 (1981), pp. 301–320.

[51] George Zames. “On the input-output stability of nonlinear time-varying feedback systems”. In: *IEEE Transactions on Automatic Control* 11 (1966), pp. 228–238.

[52] George Zames. “Unstable systems and feedback: The gap metric”. In: *Proc. of the Allerton Conference, 1980*. 1980, pp. 380–385.

[53] Kemin Zhou and John Comstock Doyle. *Essentials of robust control*. Vol. 104. Prentice hall Upper Saddle River, NJ, 1998.

[54] Peter Zipfel. “Tensor Flight Dynamics”. In: *AIAA Atmospheric Flight Mechanics Conference*. 2011, p. 6725.

[55] Peter H Zipfel. *Modeling and Simulation of Aerospace Vehicle Dynamics (AIAA Education)*. AIAA (American Institute of Aeronautics & Ast, 2007.

[56] Peter H Zipfel. “Perturbation methods in atmospheric flight mechanics”. In: *AIAA Journal* 11.9 (1973), pp. 1247–1251.

# A

# Appendix: Aerodynamic model equations and implementation

## A.1. Aerodynamic equations

The equations to model the aerodynamics are retrieved from Nguyen [33]. Terms containing the speed brakes have been set to 0. Note that the units of the actuators are in degrees in the upcoming equations. Furthermore, every coefficient that is denoted as a function of specific parameters (e.g. $C_{X,0}\left(\alpha,\beta,\delta_e\right)$) can be found as a look-up-table in [33].

### X-axis

First the X-axis coefficient yields:

$$C_X = C_{X,0}\left(\alpha,\beta,\delta_e\right) + \Delta C_{X,lef}\left(1-\frac{\delta_{lef}}{25}\right) \quad + \frac{cq}{2V}\left[C_{X_q}(\alpha) + \Delta C_{X_{q,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right]$$

where

$$\Delta C_{X,lef} = C_{X,lef}(\alpha,\beta) - C_{X,0}\left(\alpha,\beta,\delta_e = 0^\circ\right)$$

### Y-axis

For the Y-axis force coefficient:

$$\begin{aligned} C_Y =& C_{Y,0}(\alpha,\beta) + \Delta C_{Y,lef}\left(1-\frac{\delta_{lef}}{25}\right) \\ &+ \left[\Delta C_{Y,\delta_{a=20^\circ}} + \Delta C_{Y,\delta_{a=20^\circ,1ef}}\left(1-\frac{\delta_{1ef}}{25}\right)\right]\left(\frac{\delta_a}{20}\right) \\ &+ \Delta C_{Y,\delta_{r=30}}\left(\frac{\delta_r}{30}\right) \\ &+ \frac{b}{2\,V}\left\{\left[C_{Y_r}(\alpha) + \Delta C_{Y_{r,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right] r + \left[C_{Y_p}(\alpha) + \Delta C_{Y_{P,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right] p\right\} \end{aligned}$$

where

$$\begin{aligned} &\Delta C_{Y,lef} = C_{Y,lef}(\alpha,\beta) - C_{Y,0}(\alpha,\beta) \\ &\Delta C_{Y,\delta_{a=20^\circ}} = C_{Y,\delta_{a=20^\circ}}(\alpha,\beta) - C_{Y,0}(\alpha,\beta) \\ &\Delta C_{Y,\delta_{a=20^\circ,lef}} = C_{Y,\delta_{a=20^\circ},lef}(\alpha,\beta) - C_{Y,lef}(\alpha,\beta) - \left[C_{Y,\delta_{a=20^\circ}}(\alpha,\beta) - C_{Y,0}(\alpha,\beta)\right] \\ &\Delta C_{Y,\delta_{r=30^\circ}} = C_{Y,\delta_{r=30^\circ}}(\alpha,\beta) - C_{Y,0}(\alpha,\beta) \end{aligned}$$

Z-axis
For the $z$-axis force coefficient:

$$C_Z = C_{Z,0}\left(\alpha,\beta,\delta_e\right) + \Delta C_{Z,1ef}\left(1-\frac{\delta_{lef}}{25}\right) + \frac{cq}{2V}\left[C_{z_q}(\alpha) + \Delta C_{Z_{q,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right]$$

where

$$\Delta C_{Z,lef} = C_{Z,lef}(\alpha,\beta) - C_{Z,0}\left(\alpha,\beta,\delta_e=0^\circ\right)$$

Rolling moment
For the rolling-moment coefficient:

$$\begin{aligned} C_l =& C_{l,0}\left(\alpha,\beta,\delta_e\right) + \Delta C_{l,lef}\left(1-\frac{\delta_{lef}}{25}\right) \\ &+ \left[\Delta C_{l,\delta_{a=20^\circ}} + \Delta C_{l,\delta_{a=20^\circ,1ef}}\left(1-\frac{\delta_{1ef}}{25}\right)\right]\left(\frac{\delta_a}{20}\right) \\ &+ \Delta C_{l,\delta_{r=30}}\left(\frac{\delta_r}{30}\right) + \frac{b}{2V}\left\{\left[C_{l_r}(\alpha) + \Delta C_{l_{r,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right] r\right. \\ &\left.+ \left[C_{l_p}(\alpha) + \Delta C_{l_{p,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right] p\right\} + \Delta C_{l_\beta}(\alpha)\beta \end{aligned}$$

where

$$\begin{aligned} &\Delta C_{l,lef} = C_{l,lef}(\alpha,\beta) - C_{l,0}\left(\alpha,\beta,\delta_e=0^\circ\right) \\ &\Delta C_{l,\delta_{a=20^\circ}} = C_{l,\delta_{a=20^\circ}}(\alpha,\beta) - C_{l,0}\left(\alpha,\beta,\delta_e=0^\circ\right) \\ &\Delta C_{l,\delta_a=20^\circ,1ef} = C_{l,\delta_{a=20^\circ,1ef}}(\alpha,\beta) - C_{l,1ef}(\alpha,\beta) - \left[C_{l,\delta_{a=20^\circ}}(\alpha,\beta) - C_l\left(\alpha,\beta,\delta_e=0^\circ\right)\right] \\ &\Delta C_{l,\delta_{r=30^\circ}} = C_{l,\delta_{r=30^\circ}}(\alpha,\beta) - C_{l,0}\left(\alpha,\beta,\delta_e=0^\circ\right) \end{aligned}$$

Pitching moment
For the pitching moment coefficient:

$$\begin{aligned} C_m =& C_{m,0}\left(\alpha,\beta,\delta_e\right) n_{\delta_e}\left(\delta_e\right) + C_z\left(x_{cg,ref} - x_{cg}\right) + \Delta C_{m,1ef}\left(1-\frac{\delta_{1ef}}{25}\right) \\ &+ \frac{cq}{2V}\left[C_{m_q}(\alpha) + \Delta C_{m_{q,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right] \\ &+ \Delta C_m(\alpha) \end{aligned}$$

where

$$\Delta C_{m,lef} = C_{m,lef}(\alpha,\beta) - C_{m,0}\left(\alpha,\beta,\delta_e=0^\circ\right)$$

Yawing moment
For the yawing moment coefficient:

$$\begin{aligned} C_n =& C_{n,0}\left(\alpha,\beta,\delta_e\right) + \Delta C_{n,lef}\left(1-\frac{\delta_{lef}}{25}\right) - C_Y\left(x_{cg,ref} - x_{cg}\right)\frac{c}{b} \\ &+ \left[\Delta C_{n,\delta_{a=20}} + \Delta C_{n,\delta_a=20^\circ,lef}\left(1-\frac{\delta_{lef}}{25}\right)\right]\left(\frac{\delta_a}{20}\right) \\ &+ \Delta C_{n,\delta r=30^\circ}\left(\frac{\delta_r}{30}\right) + \frac{b}{2V}\left\{\left[C_{n_r}(\alpha) + \Delta C_{n_{r,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right] r\right. \\ &\left.+ \left[C_{n_p}(\alpha) + \Delta C_{n_{p,lef}}(\alpha)\left(1-\frac{\delta_{lef}}{25}\right)\right] p\right\} + \Delta C_{n_\beta}(\alpha)\beta \end{aligned}$$

where

$$\Delta C_{n,lef} = C_{n,lef}(\alpha,\beta) - C_{n,0}\left(\alpha,\beta,\delta_e = 0^\circ\right)$$
$$\Delta C_{n,\delta a=20^\circ} = C_{n,\delta_{a=20^\circ}}(\alpha,\beta) - C_{n,0}\left(\alpha,\beta,\delta_e = 0^\circ\right)$$
$$\Delta C_n, \delta_{a=20^\circ,lef} = C_{n,\delta_{a=20^\circ,lef}}(\alpha,\beta) - C_{n,lef}(\alpha,\beta) - [C_{n,\delta_{a=20^\circ}}(\alpha,\beta) - C_n\left(\alpha,\beta,\delta_e = 0^\circ\right)]$$
$$\Delta C_{n,\delta_{r=30^\circ}} = C_{n,\delta_{r=30^\circ}}(\alpha,\beta) - C_{n,0}\left(\alpha,\beta,\delta_e = 0^\circ\right)$$

## A.2. MATLAB code implementation

The equations of the section above have been implemented in a MATLAB function. This function can be found below. All the lookup tables are contained in the AeroCoefficients structure that has to be provided to the function and is unpacked.

```
1     function [C_X, C_Y, C_Z, C_l, C_m, C_n] = AeroCoefficients_Computation(delta_aerl, alpha,
          beta, Mach, V, omega_BE_B, AeroCoefficients, b, c, x_cgR, x_cg)
2
3 %% Inputs
4
5 % Conversions
6 deg2rad = pi/180;
7 rad2deg = 180/pi;
8
9 % Control inputs
10 delta_a = delta_aerl(1,1);
11 delta_e = delta_aerl(2,1);
12 delta_r = delta_aerl(3,1);
13 delta_l = delta_aerl(4,1);
14
15 % Angular rate vector
16 p = omega_BE_B(1,1);
17 q = omega_BE_B(2,1);
18 r = omega_BE_B(3,1);
19
20 % Breakpoints
21 alpha1_vec   = AeroCoefficients.alpha1;
22 alpha2_vec   = AeroCoefficients.alpha2;
23 beta1_vec    = AeroCoefficients.beta1;
24 delta_e1_vec = AeroCoefficients.delta_e1;
25 delta_e2_vec = AeroCoefficients.delta_e2;
26
27 %% CX
28 CX0_vec     = AeroCoefficients.CX.CX0;
29 CX_lef_vec  = AeroCoefficients.CX.CX_lef;
30 CX_q_vec    = AeroCoefficients.CX.CX_q;
31 CX_qlef_vec = AeroCoefficients.CX.CX_qlef;
32
33 % Interpolate
34 CX0         = interpn(alpha1_vec, beta1_vec, delta_e1_vec, CX0_vec, alpha, beta, delta_e, '
       linear');
35 CX_lef      = interpn(alpha2_vec, beta1_vec, CX_lef_vec, alpha, beta, 'linear');
36 CX_q        = interp1(alpha1_vec, CX_q_vec, alpha, 'linear');
37 CX_qlef     = interp1(alpha2_vec, CX_qlef_vec, alpha, 'linear');
38 CX0_e0      = interpn(alpha1_vec, beta1_vec, delta_e1_vec, CX0_vec, alpha, beta, 0, 'linear')
       ;
39
40 % Computations
41 DCX_lef = CX_lef - CX0_e0;
42 C_X = CX0 + DCX_lef * (1-delta_l / (deg2rad * 25)) + (c * q) / (2 * V) * (CX_q + CX_qlef *
       (1-delta_l / (deg2rad * 25)));
43
44 %% CY
45 CY0_vec         = AeroCoefficients.CY.CY0;
46 CY_rud30_vec    = AeroCoefficients.CY.CY_rud30;
47 CY_r_vec        = AeroCoefficients.CY.CY_r;
48 CY_p_vec        = AeroCoefficients.CY.CY_p;
49 CY_lef_vec      = AeroCoefficients.CY.CY_lef;
50 CY_ail20lef_vec = AeroCoefficients.CY.CY_ail20lef;
```

```
CY_ail20_vec    = AeroCoefficients.CY.CY_ail20;
DCY_plef_vec    = AeroCoefficients.CY.DCY_plef;
DCY_rlef_vec    = AeroCoefficients.CY.DCY_rlef;

% Interpolate
CY0             = interpn(alpha1_vec, beta1_vec, CY0_vec, alpha, beta, 'linear');
CY_rud30        = interpn(alpha1_vec, beta1_vec, CY_rud30_vec, alpha, beta, 'linear');
CY_r            = interp1(alpha1_vec, CY_r_vec, alpha, 'linear');
CY_p            = interp1(alpha1_vec, CY_p_vec, alpha, 'linear');
CY_lef          = interpn(alpha2_vec, beta1_vec, CY_lef_vec, alpha, beta, 'linear');
CY_ail20        = interpn(alpha1_vec, beta1_vec, CY_ail20_vec, alpha, beta, 'linear');
CY_ail20lef     = interpn(alpha2_vec, beta1_vec, CY_ail20lef_vec, alpha, beta, 'linear');
DCY_plef        = interp1(alpha2_vec, DCY_plef_vec, alpha, 'linear');
DCY_rlef        = interp1(alpha2_vec, DCY_rlef_vec, alpha, 'linear');

% Computations
DCY_lef      = CY_lef - CY0;
DCY_ail20    = CY_ail20 - CY0;
DCY_ail20lef = CY_ail20lef - CY_lef - (CY_ail20 - CY0);
DCY_rud30    = CY_rud30 - CY0;
C_Y = CY0 + DCY_lef*(1 - delta_l / (25 * deg2rad)) + (DCY_ail20 + DCY_ail20lef * (1 - delta_l
     / (25 * deg2rad))) * (delta_a / (20 * deg2rad)) ...
      + DCY_rud30 * (delta_r / (30 * deg2rad)) + b / (2 * V) *((CY_r + DCY_rlef * (1 -
          delta_l / (25 * deg2rad))) * r + (CY_p + DCY_plef * (1 - delta_l / (25 * deg2rad)))
           * p);


%% CZ
CZ0_vec     = AeroCoefficients.CZ.CZ0;
CZ_lef_vec  = AeroCoefficients.CZ.CZ_lef;
CZ_q_vec    = AeroCoefficients.CZ.CZ_q;
CZ_qlef_vec = AeroCoefficients.CZ.CZ_qlef;

% Interpolate
CZ0         = interpn(alpha1_vec, beta1_vec, delta_e1_vec, CZ0_vec, alpha, beta, delta_e, '
    linear');
CZ_lef      = interpn(alpha2_vec, beta1_vec, CZ_lef_vec, alpha, beta, 'linear');
CZ_q        = interp1(alpha1_vec, CZ_q_vec, alpha, 'linear');
CZ_qlef     = interp1(alpha2_vec, CZ_qlef_vec, alpha, 'linear');
CZ0_e0      = interpn(alpha1_vec, beta1_vec, delta_e1_vec, CZ0_vec, alpha, beta, 0, 'linear')
    ;

% Computations
DCZ_lef = CZ_lef - CZ0_e0;

C_Z = CZ0 + DCZ_lef * (1-delta_l / (25 * deg2rad)) + (c * q) / (2 * V) * (CZ_q + CZ_qlef *
    (1-delta_l / (25 * deg2rad)));


%% CL
CL0_vec         = AeroCoefficients.CL.CL0;
CL_rud30_vec    = AeroCoefficients.CL.CL_rud30;
CL_r_vec        = AeroCoefficients.CL.CL_r;
CL_p_vec        = AeroCoefficients.CL.CL_p;
CL_lef_vec      = AeroCoefficients.CL.CL_lef;
CL_ail20lef_vec = AeroCoefficients.CL.CL_ail20lef;
CL_ail20_vec    = AeroCoefficients.CL.CL_ail20;
DCL_plef_vec    = AeroCoefficients.CL.DCL_plef;
DCL_rlef_vec    = AeroCoefficients.CL.DCL_rlef;
DCL_beta_vec    = AeroCoefficients.CL.DCL_beta;

% Interpolate
CL0             = interpn(alpha1_vec, beta1_vec, delta_e2_vec, CL0_vec, alpha, beta, delta_e,
     'linear');
CL_rud30        = interpn(alpha1_vec, beta1_vec, CL_rud30_vec, alpha, beta, 'linear');
CL_r            = interp1(alpha1_vec, CL_r_vec, alpha, 'linear');
CL_p            = interp1(alpha1_vec, CL_p_vec, alpha, 'linear');
CL_lef          = interpn(alpha2_vec, beta1_vec, CL_lef_vec, alpha, beta, 'linear');
CL_ail20        = interpn(alpha1_vec, beta1_vec, CL_ail20_vec, alpha, beta, 'linear');
CL_ail20lef     = interpn(alpha2_vec, beta1_vec, CL_ail20lef_vec, alpha, beta, 'linear');
DCL_plef        = interp1(alpha2_vec, DCL_plef_vec, alpha, 'linear');
```

```
115 DCL_rlef        = interp1(alpha2_vec, DCL_rlef_vec, alpha, 'linear');
116 DCL_beta        = interp1(alpha1_vec, DCL_beta_vec, alpha, 'linear');
117 CL_e0          = interpn(alpha1_vec, beta1_vec, delta_e2_vec, CL0_vec, alpha, beta, 0, '
        linear');
118
119 % Computations
120 DCL_lef         = CL_lef - CL_e0;
121 DCL_ail20       = CL_ail20 - CL_e0;
122 DCL_ail20lef    = CL_ail20lef - CL_lef - (CL_ail20 - CL_e0);
123 DCL_rud30       = CL_rud30 - CL_e0;
124
125 C_l = CL0 + DCL_lef*(1 - delta_l / (25 * deg2rad)) + (DCL_ail20 + DCL_ail20lef * (1 - delta_l
        / (25 * deg2rad))) * (delta_a / (20 * deg2rad)) ...
126       + DCL_rud30 * (delta_r / (30 * deg2rad)) + b / (2 * V) *((CL_r + DCL_rlef * (1 -
              delta_l / (25 * deg2rad))) * r + (CL_p + DCL_plef * (1 - delta_l / (25 * deg2rad)))
               * p) + DCL_beta * beta;
127
128
129 %% CM
130 CM0_vec         = AeroCoefficients.CM.CM0;
131 CM_lef_vec      = AeroCoefficients.CM.CM_lef;
132 CM_q_vec        = AeroCoefficients.CM.CM_q;
133 DCM_qlef_vec     = AeroCoefficients.CM.DCM_qlef;
134 DCM_vec         = AeroCoefficients.CM.DCM;
135 eta_delta_e_vec = AeroCoefficients.CM.eta_delta_e;
136
137 % Interpolate
138 CM0         = interpn(alpha1_vec, beta1_vec, delta_e1_vec, CM0_vec, alpha, beta, delta_e, '
        linear');
139 CM_lef      = interpn(alpha2_vec, beta1_vec, CM_lef_vec, alpha, beta, 'linear');
140 CM_q        = interp1(alpha1_vec, CM_q_vec, alpha, 'linear');
141 DCM_qlef    = interp1(alpha2_vec, DCM_qlef_vec, alpha, 'linear');
142 DCM         = interp1(alpha1_vec, DCM_vec, alpha, 'linear');
143 eta_delta_e = interp1(delta_e1_vec, eta_delta_e_vec, delta_e, 'linear');
144
145 CM0_e0      = interpn(alpha1_vec, beta1_vec, delta_e1_vec, CM0_vec, alpha, beta, 0, 'linear')
        ;
146
147 % Computations
148 DCM_lef = CM_lef - CM0_e0;
149 C_m = CM0 * eta_delta_e + C_Z * (x_cgR - x_cg) + DCM_lef * (1-delta_l / (25 * deg2rad)) + (c
        * q) / (2 * V) * (CM_q + DCM_qlef * (1-delta_l / (25 * deg2rad))) + DCM;
150
151
152 %% CN
153 CN0_vec         = AeroCoefficients.CN.CN0;
154 CN_rud30_vec    = AeroCoefficients.CN.CN_rud30;
155 CN_r_vec        = AeroCoefficients.CN.CN_r;
156 CN_p_vec        = AeroCoefficients.CN.CN_p;
157 CN_lef_vec      = AeroCoefficients.CN.CN_lef;
158 CN_ail20lef_vec = AeroCoefficients.CN.CN_ail20lef;
159 CN_ail20_vec    = AeroCoefficients.CN.CN_ail20;
160 DCN_plef_vec    = AeroCoefficients.CN.DCN_plef;
161 DCN_rlef_vec    = AeroCoefficients.CN.DCN_rlef;
162 DCN_beta_vec    = AeroCoefficients.CN.DCN_beta;
163
164 % Interpolate
165 CN0         = interpn(alpha1_vec, beta1_vec, delta_e2_vec, CN0_vec, alpha, beta, delta_e, '
        linear');
166 CN_rud30    = interpn(alpha1_vec, beta1_vec, CN_rud30_vec, alpha, beta, 'linear');
167 CN_r        = interp1(alpha1_vec, CN_r_vec, alpha, 'linear');
168 CN_p        = interp1(alpha1_vec, CN_p_vec, alpha, 'linear');
169 CN_lef      = interpn(alpha2_vec, beta1_vec, CN_lef_vec, alpha, beta, 'linear');
170 CN_ail20    = interpn(alpha1_vec, beta1_vec, CN_ail20_vec, alpha, beta, 'linear');
171 CN_ail20lef = interpn(alpha2_vec, beta1_vec, CN_ail20lef_vec, alpha, beta, 'linear');
172 DCN_plef    = interp1(alpha2_vec, DCN_plef_vec, alpha, 'linear');
173 DCN_rlef    = interp1(alpha2_vec, DCN_rlef_vec, alpha, 'linear');
174 DCN_beta    = interp1(alpha1_vec, DCN_beta_vec, alpha, 'linear');
175 CN_e0       = interpn(alpha1_vec, beta1_vec, delta_e2_vec, CN0_vec, alpha, beta, 0, 'linear')
        ;
176
```

```
177 % Computations
178 DCN_lef      = CN_lef - CN_e0;
179 DCN_ail20    = CN_ail20 - CN_e0;
180 DCN_ail20lef = CN_ail20lef - CN_lef - (CN_ail20 - CN_e0);
181 DCN_rud30    = CN_rud30 - CN_e0;
182 C_n = CN0 + DCN_lef * (1 - delta_l / (25 * deg2rad)) - C_Y * (x_cgR - x_cg) * c / b + (
        DCN_ail20 + DCN_ail20lef * (1 - delta_l / (25 * deg2rad))) * (delta_a / (20 * deg2rad))
        ...
183           + DCN_rud30 * (delta_r / (30 * deg2rad)) + b / (2 * V) *((CN_r + DCN_rlef * (1 -
                  delta_l / (25 * deg2rad))) * r + (CN_p + DCN_plef * (1 - delta_l / (25 *
                  deg2rad))) * p) + DCN_beta * beta;
184
185
186
187 end
```

# B

# Appendix: Equations of motion and sensor dynamics MATLAB functions

Section 2.4 discussed the derivation of the equations of motion from a tensor point of view. For completeness, the MATLAB implementation is posted here. Furthermore, the sensor dynamics from Section 2.5 can also be found here.

## B.1. Translational dynamics

Equation 2.36 in Section 2.4.1 showed the derivation of the translational equations of motion in the wind frame:

$$m \begin{bmatrix} \dot{V} \\ \dot{\beta} \\ \dot{\alpha} \end{bmatrix} = X \left( -m[T]^{WB} \left[\Omega^{BE}\right]^{B} [\overline{T}]^{WB} \left[v_B^E\right]^{W} + [T]^{WB}[f_{a,b}]^{B} + m[T]^{WB}[T]^{BL}[g]^{L} \right)$$

This is implemented in MATLAB using the following function:

```
function [Vabdot, vdot_BE_B, v_BE_B] = Translation_Dynamics_Computation(Vab, omega_BE_B,
    f_B_B, m_B)

%% Inputs

V     = Vab(1);
alpha = Vab(2);
beta  = Vab(3);

p = omega_BE_B(1);
q = omega_BE_B(2);
r = omega_BE_B(3);

%% Auxiliary variables

X = [1, 0, 0;
     0, 1/V, 0;
     0, 0, 1/(V*cos(beta))];

T_W_B = [ cos(alpha)*cos(beta), sin(beta),  cos(beta)*sin(alpha);
         -cos(alpha)*sin(beta), cos(beta), -sin(alpha)*sin(beta);
                 -sin(alpha),         0,            cos(alpha)];

Omega_BE_B = [+0             -omega_BE_B(3) +omega_BE_B(2); ...
              +omega_BE_B(3) +0             -omega_BE_B(1); ...
              -omega_BE_B(2) +omega_BE_B(1) +0           ];


%% Translational Dynamics (TD)
Vbadot = X*(-m_B*T_W_B*(Omega_BE_B)*(T_W_B.')*[V; 0; 0]+T_W_B*f_B_B)/m_B;
```

```
30
31 Vdot     = Vbadot(1);
32 alphadot = Vbadot(3);
33 betadot  = Vbadot(2);
34 Vabdot = [Vdot; alphadot; betadot];
35
36 % Compute uvw dynamics
37 u = V*cos(alpha)*cos(beta);
38 v = V*sin(beta);
39 w = V*sin(alpha)*cos(beta);
40
41 udot = cos(alpha)*cos(beta)*Vdot - cos(beta)*sin(alpha)*V*alphadot - cos(alpha)*sin(beta)*V*
       betadot;
42 vdot = sin(beta)*Vdot + cos(beta)*V*betadot;
43 wdot = cos(beta)*sin(alpha)*Vdot + cos(alpha)*cos(beta)*V*alphadot - sin(alpha)*sin(beta)*V*
       betadot;
44
45 %% Outputs
46 v_BE_B    = [u; v; w];
47 vdot_BE_B = [udot; vdot; wdot];
```

## B.2. Translational kinematics

The translational kinematics could be found in Section 2.4.2. There the following equation was introduced:

$$[D^E s_{BE}]^L = [\overline{T}]^{BL}[V_B^E]^B$$

Which is implemented in the following function:

```
1 function v_BE_L  = Translation_Kinematics_Computation(v_BE_B, T_BL)
2
3 %% Translational Kinematics (TK)
4
5 % Local velocity vector (body)
6 v_BE_L = T_BL'*v_BE_B;
```

## B.3. Attitude dynamics

For the attitude dynamics, Section 2.4.3 showed the derivation of:

$$\left[\frac{d\omega^{BE}}{dt}\right]^B = \left([I_B^B]^B\right)^{-1}\left(-[\Omega^{BE}]^B[I_B^B]^B[\omega^{BE}]^B + [\Omega^{BE}]^B[I_{B_R}^{B_R}]^B + [m_B]^B\right)$$

And also this equation has a MATLAB implementation:

```
1 function omegadot_BE_B = Attitude_Dynamics_Computation(omega_BE_B, m_B_B, I_BB_B, l_BrE_Br_B)
2
3 %% Auxiliary variables
4
5 % Angular rate SS matrix (body)
6 Omega_BE_B = [+0             -omega_BE_B(3) +omega_BE_B(2); ...
7               +omega_BE_B(3) +0             -omega_BE_B(1); ...
8               -omega_BE_B(2) +omega_BE_B(1) +0           ];
9
10 %% Attitude Dynamics (AD)
11 omegadot_BE_B = I_BB_B\(-Omega_BE_B*(I_BB_B*omega_BE_B + l_BrE_Br_B) + m_B_B);
```

## B.4. Attitude kinematics

The last differential equation of the equations of motion was shown in Section 2.4.4:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi)/cos(\theta) & cos(\phi)/cos(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

```
function edot_BL = Euler_Angle_Computation(e_BL, omega_BE_B)

% Euler angles (BVR)
phi_B   = e_BL(1,1);
theta_B = e_BL(2,1);
psi_B   = e_BL(3,1); %#ok<NASGU>

%% Auxiliary variables

% AK matrix
T_edotomega = [1 +sin(phi_B)*tan(theta_B) +cos(phi_B)*tan(theta_B);...
               0 +cos(phi_B)              -sin(phi_B);...
               0 +sin(phi_B)*sec(theta_B) +cos(phi_B)*sec(theta_B)];

%% Euler angle computation
% Euler angle rate (BVR)
edot_BL = T_edotomega*omega_BE_B;
```

Furthermore, Section 2.4.4 showed the computation of the transformation matrix $[T]^{BL}$:

$$[T]^{BL} = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\theta\sin\phi \\ \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\theta\cos\phi \end{bmatrix}$$

and the organisation step:

$$[T(n+1)]^{BL} = [T(n)]^{BL} + k(1/2)(I - [T(n)]^{BL}[\overline{T(n)}]^{BL})[T(n)]^{BL};$$

This was combined in the following MATLAB function:

```
function [T_BL, e_BL]  = DCM_Computation(k, e_BL)

%% Inputs

% Euler angles (BVR)
phi_B   = e_BL(1,1);
theta_B = e_BL(2,1);
psi_B   = e_BL(3,1);

%% DCM computation

% Euler angles (BVR)
phi_B   = phi_B - 2*pi*floor((phi_B+pi)/(2*pi));    % -pi   <= phi   <= +pi
theta_B = theta_B; %#ok<ASGSL>
psi_B   = psi_B;                                    %#ok<ASGSL> % -pi   <= psi   <= +pi

% DCM matrix (BVR)
T_BL = [+cos(psi_B)*cos(theta_B)                                        +sin(psi_B)*cos(theta_B)
                                    -sin(theta_B);...
        +cos(psi_B)*sin(theta_B)*sin(phi_B)-sin(psi_B)*cos(phi_B) +sin(psi_B)*sin(theta_B)*
            sin(phi_B)+cos(psi_B)*cos(phi_B) +cos(theta_B)*sin(phi_B);...
        +cos(psi_B)*sin(theta_B)*cos(phi_B)+sin(psi_B)*sin(phi_B) +sin(psi_B)*sin(theta_B)*
            cos(phi_B)-cos(psi_B)*sin(phi_B) +cos(theta_B)*cos(phi_B)];

% Orthogonality
T_BL = T_BL + k*(1/2)*(eye(3)-T_BL*T_BL')*T_BL;

%% Outputs

e_BL = [phi_B; theta_B; psi_B];
```

## B.5. Sensor dynamics

Lastly, the equation for the sensor dynamics was given by:

$$\left[n_S^I\right]^B = \frac{1}{g_0}\left(\left[a_B^I\right]^B + \frac{d}{dt}(\left[\Omega^{BI}\right]^B)s_{BS} + \left[\Omega^{BI}\right]^B\left[\Omega^{BI}\right]^B\left[s_{BS}\right]^B - [T]^{BL}[g]^L\right)$$

This was introduced in Section 2.5. To compute this equation, the following function was used:

```
function n_SE_B = fcn(a_BE_B, omega_BE_B, omegadot_BE_B, T_BL, s_SB_B, g0)

%% Inputs


%% Auxiliary variables

% Angular rate SS matrix (body)
Omega_BE_B = [+0              -omega_BE_B(3) +omega_BE_B(2); ...
              +omega_BE_B(3) +0              -omega_BE_B(1); ...
              -omega_BE_B(2) +omega_BE_B(1) +0           ];

% Angular acceleration SS matrix (body)
Omegadot_BE_B = [+0                -omegadot_BE_B(3) +omegadot_BE_B(2); ...
                 +omegadot_BE_B(3) +0                -omegadot_BE_B(1); ...
                 -omegadot_BE_B(2) +omegadot_BE_B(1) +0           ];

g_L = [0; 0; g0];

%% Grubin transformation

% Acceleration
a_SE_B = a_BE_B + Omega_BE_B*Omega_BE_B*s_SB_B + Omegadot_BE_B*s_SB_B - T_BL*g_L;
% Modify sign of normal acceleration
a_SE_B = [1 0 0; 0 1 0; 0 0 -1]*a_SE_B;

% Load factor
n_SE_B = a_SE_B/g0;
```

# C

# Appendix: Engine model

The engine simulation of Nguyen [33] is expanded upon in this section. Furthermore, the MATLAB/Simulink implementation is shown. This can be seen in Figure C.1. It basically consists of 3 steps. Firstly, a conversion from normalized throttle command $\delta_t$ to a commanded power is performed. Secondly, the engine dynamics have to be simulated. Lastly, the conversion from power to thrust has to be performed, that depends on the current operating condition of the engine.

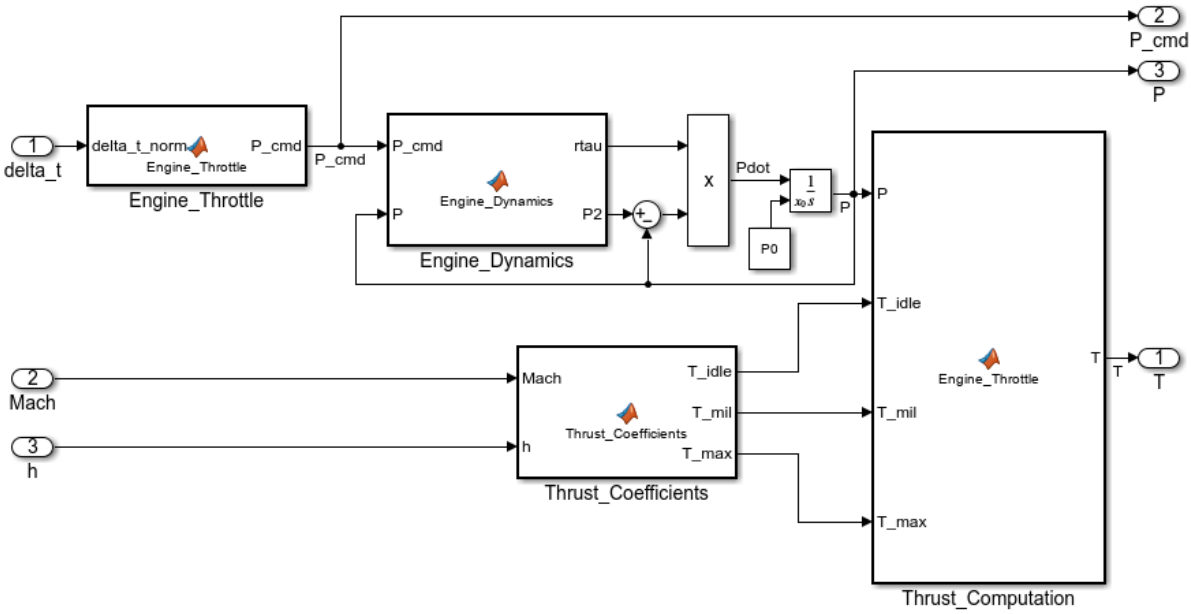



**Figure C.1:** Simulink implementation of engine model.

Starting with the first step, the conversion from normalized throttle command to power (in percentage) command. This is done using a piece wise linear equation, where the region of $\delta_t > 0.77$ activates the afterburner and the region $\delta_t \leq 0.77$ is referred to as the military mode:

$$\begin{cases} P_{cmd} = 64.94\delta_t; & \delta_t \leq 0.77 \text{ (military)} \\ P_{cmd} = 217.38\delta_t - 117.38; & \delta_t > 0.77 \text{ (afterburner)} \end{cases} \tag{C.1}$$

Subsequently, the commanded power is the input for the engine dynamics. The logic diagram to compute the differential equation is shown in Figure C.2. It splits the computation of $P_2$ and $\frac{1}{\tau}$ into four regions based on the value of the commanded engine power $P_{cmd}$ and the current engine power $P$ depending if these are bigger or smaller than $50$. The values of $P_2$ and $\frac{1}{\tau}$ are used for the differential

equation. If necessary the following formula is used to calculate the time constant:

$$f_\tau \begin{cases} \frac{1}{\tau} = 1; & (P2 - P_{cmd}) < 25 \\ \frac{1}{\tau} = 1.9 - 0.036(P2 - P_{cmd}) & 25 \leq (P2 - P_{cmd}) \leq 50 \\ \frac{1}{\tau} = 0.1; & (P2 - P_{cmd}) > 50 \end{cases} \tag{C.2}$$

Lastly, the conversion from percentage power to thrust has to be performed. Here, the values of $T_{idle}$, $T_{mil}$ and $T_{max}$ are retrieved from lookup tables, depending on the Mach number and altitude. The tables can be found in Nguyen [33].

$$\begin{cases} T = T_{idle} + (T_{mil} - T_{idle})\frac{P}{50}; & P < 50 \\ T = T_{mil} + (T_{max} - T_{mil})\frac{P-50}{50}; & P \geq 50 \end{cases} \tag{C.3}$$

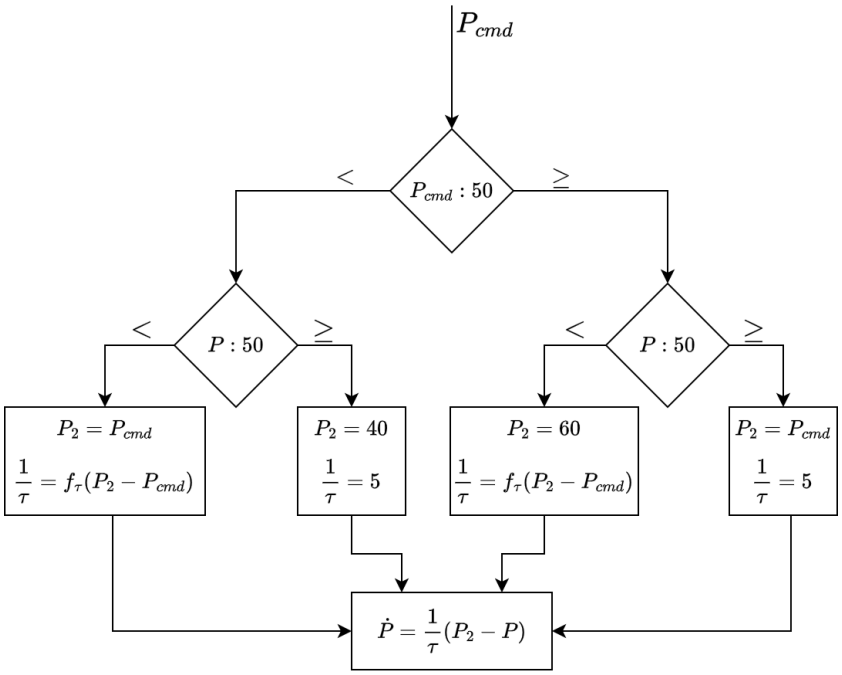
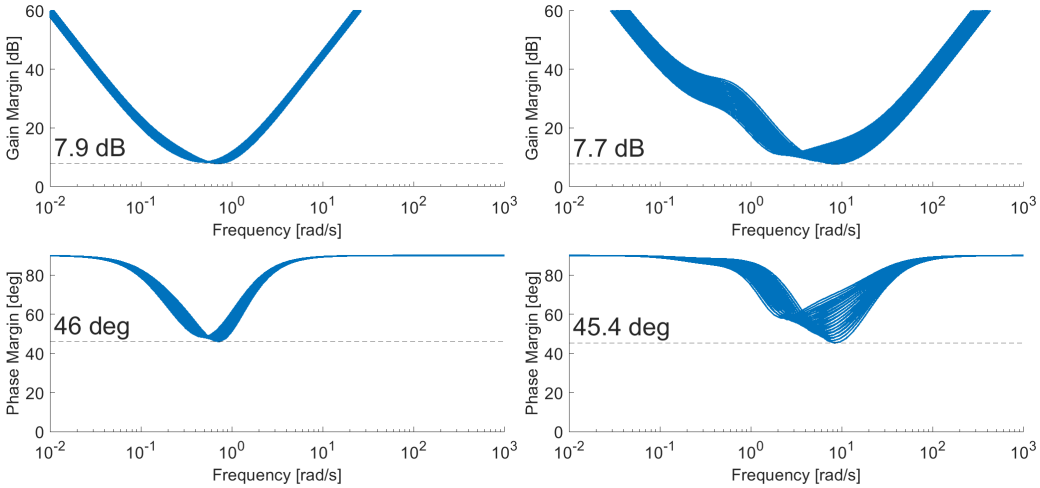



**Figure C.2:** Logic diagram for engine dynamics [33].

# D

# Appendix: Stability margins of the direct trajectory controller

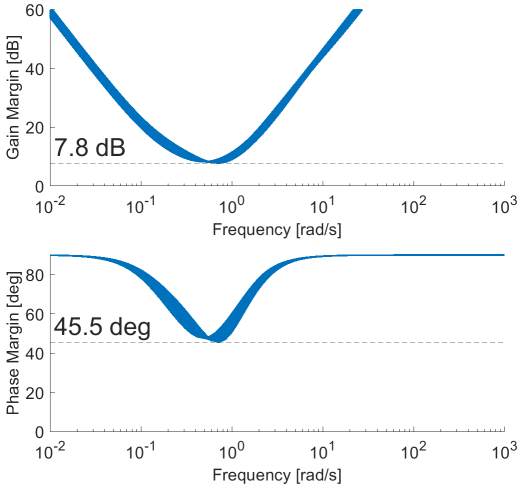
In Chapter 5, only the worst-case stability margins were shown. For completeness, the margins along the most important frequency range are added here.



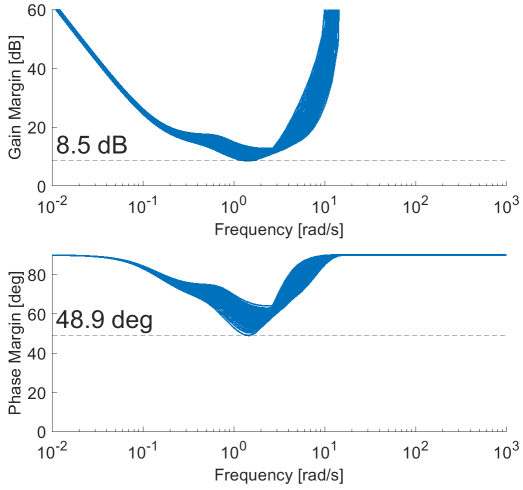

**(a)** Loop at a time input ($\delta_{t,cmd}$) stability margin.




**(b)** Loop at a time input ($\delta_{e,cmd}$) stability margin.

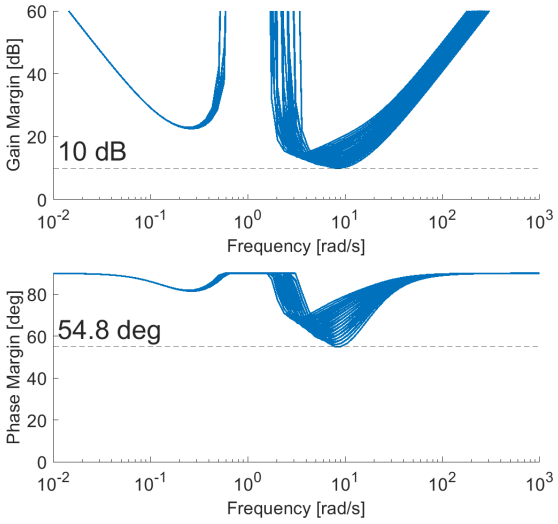



**(c)** Loop at a time output ($V$) stability margin.

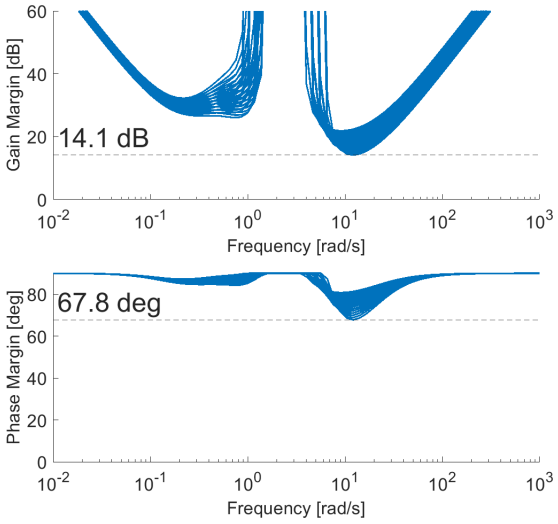



**(d)** Loop at a time output ($V_z$) stability margin.

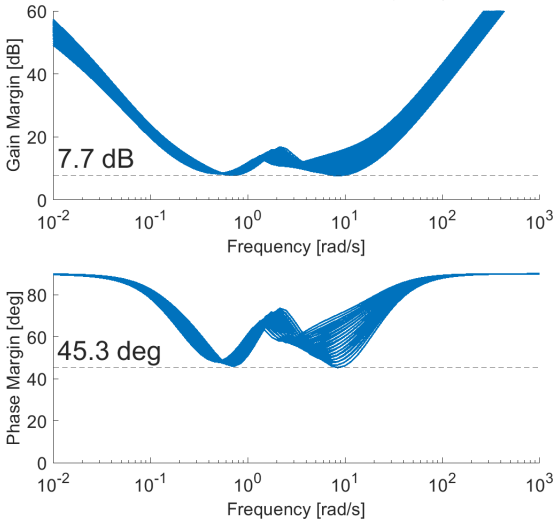
**Figure D.1:** Direct trajectory controller disk margins.



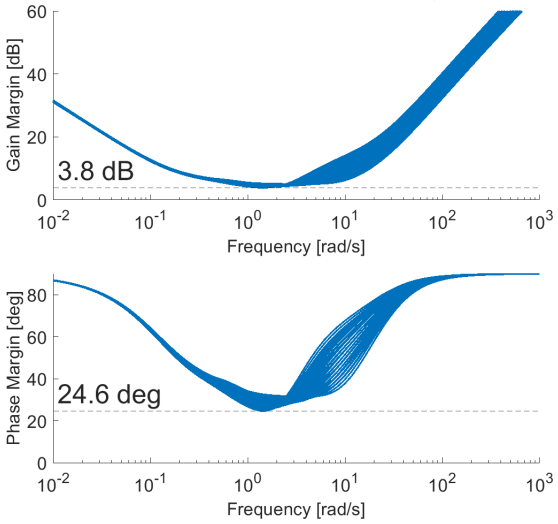

**(e)** Loop at a time output ($n_z$) stability margin.



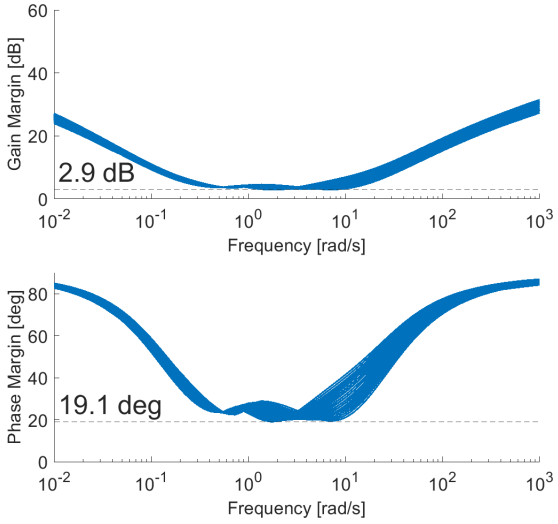

**(f)** Loop at a time output ($q$) stability margin.




**(g)** Simultaneous input stability margin.




**(h)** Simultaneous output stability margin.




**(i)** Simultaneous input/output stability margin.

**Figure D.1:** Direct trajectory controller disk margins (continued).