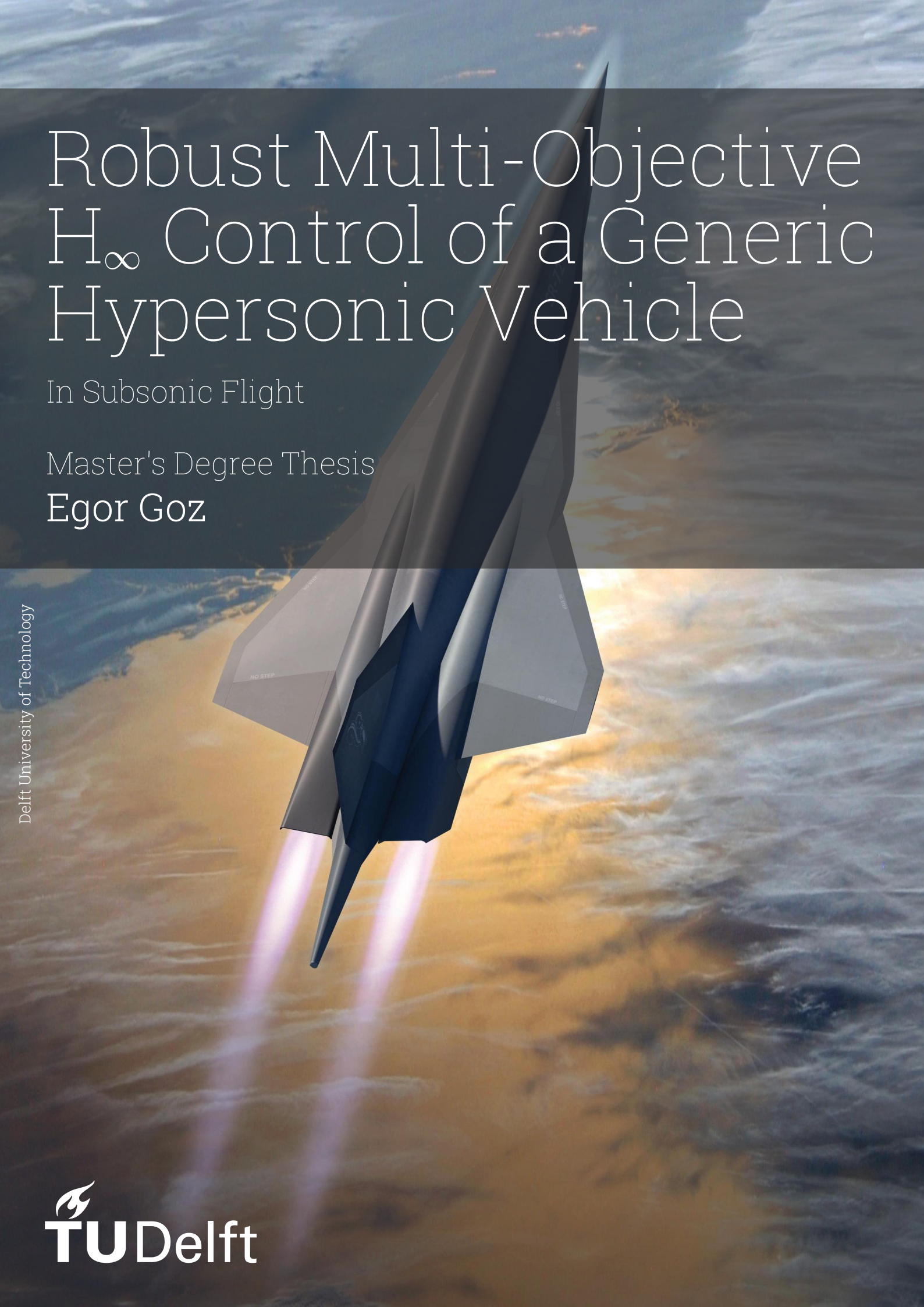
Robust Multi-Objective H_∞ Control of a Generic Hypersonic Vehicle

In Subsonic Flight

Master's Degree Thesis

Egor Goz



Delft University of Technology

TU Delft

[This page intentionally left blank]

Robust Multi-Objective H_∞ Control of a Generic Hypersonic Vehicle

In Subsonic Flight

by

Egor Goz

to obtain the Master of Science degree
in Aerospace Engineering
at Delft University of Technology

Student number:	4676459
Associate Professor:	S. Theodoulis
Project Duration:	June, 2023 - July, 2024
Faculty:	Faculty of Aerospace Engineering, Delft

Cover: An artist's concept of the SR-72, newatlas.com

TU Delft

Preface

”If I were asked back then where I would be in 10 years...” The completion of this thesis marks exactly 10 years since I left home in pursuit of education and opportunities. Back then, I had no clue where I would end up or what a long and eventful journey lay ahead. Now, I find myself graduating with a Master’s Degree in aerospace engineering, and a part of me cannot believe I am saying it. This thesis is a culmination of my academic endeavors, a finish line for my student life, and the grand start of a new journey as an aerospace engineer. Let’s see where the next 10 years will take me.

I would like to express my deepest gratitude to the people who supported me along the way. I am eternally grateful to my family, especially my father, who shared my dreams and without whom this achievement would not have been possible. I want to thank my friends and those I hold dearest, with whom I shared my brightest and hardest moments. And, of course, I want to thank my supervisor, Spilios, for mentoring me and teaching me to always reach for higher goals. Thank you all for believing in me.

Egor Goz
Delft, July 2024

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
CG	Center of Gravity
CL	Closed-Loop
DCM	Direction Cosine Matrix
DM	Disk Margin
DGM	Disk Gain Margin
DoF	Degrees-of-Freedom
DPM	Disk Phase Margin
FCS	Flight Control System
FP	Flight Point
GHAME	Generic Hypersonic Aerodynamic Model Example
GM	Gain Margin
HF	High-Frequency
HV	Hypersonic Vehicle
IMU	Inertial Measurement Unit
ISA	International Standard Atmosphere
LF	Low-Frequency
MIMO	Multi-Input-Multi-Output
MMIO	Multi-Loop Input-Output
MOI	Moment of Inertia
NMP	Non-Minimum-Phase
PM	Phase Margin
SIMO	Single-Input-Multi-Output
SISO	Single-Input-Single-Output
SSTO	Single-Stage-To-Orbit

Symbols

Symbol	Definition	Unit
α	Angle of attack	deg
β	Angle of sideslip	deg
δ_a	Aileron deflection	deg
δ_e	Elevator deflection	deg
δ_r	Rudder deflection	deg
δ_t	Throttle tab deflection	[0-2]
δ_{vl}	Left elevon deflection	deg
δ_{vr}	Right elevon deflection	deg
γ	Tuning performance metric	–
ω	Radial frequency	rad/s
ρ	Density of air	kg/m^3
ϕ	Roll angle	deg
θ	Pitch angle	deg
ψ	Yaw angle	deg
σ	Singular value	–

Symbol	Definition	Unit
ζ	Damping ratio	–
a	Speed of sound	m/s
A_c	Engine cowl area	m^2
b	Reference wing span	m
C_a	Capture area ratio	–
C_D	Drag coefficient	–
C_{D_0}	Zeroth drag coefficient	–
C_{D_α}	Drag due to angle of attack coefficient	1/deg
C_{D_1}	Modified partial drag coefficient	–
C_L	Lift coefficient	–
C_{L_0}	Zeroth lift coefficient	–
C_{L_α}	Lift due to angle of attack coefficient	1/deg
$C_{L_{\delta_e}}$	Lift due to elevon deflection coefficient	1/deg
C_{L_1}	Modified partial lift coefficient	–
C_{l_p}	Roll moment due to roll rate coefficient	1/rad
C_{l_r}	Roll moment due to yaw rate coefficient	1/rad
C_{l_β}	Roll moment due to sideslip angle coefficient	1/deg
$C_{l_{\delta_a}}$	Roll moment due to aileron deflection coefficient	1/deg
$C_{l_{\delta_r}}$	Roll moment due to rudder deflection coefficient	1/deg
C_m	Pitch moment coefficient	–
C_{m_0}	Zeroth pitch moment coefficient	–
C_{m_q}	Pitch moment due to pitch rate coefficient	1/rad
C_{m_α}	Pitch moment due to angle of attack coefficient	1/deg
$C_{m_{\delta_e}}$	Pitch moment due to elevon deflection coefficient	1/deg
C_{m_1}	Modified partial pitch moment coefficient	–
C_{n_p}	Yaw moment due to roll rate coefficient	1/rad
C_{n_r}	Yaw moment due to yaw rate coefficient	1/rad
C_{n_β}	Yaw moment due to sideslip angle coefficient	1/deg
$C_{n_{\delta_a}}$	Yaw moment due to aileron deflection coefficient	1/deg
$C_{n_{\delta_r}}$	Yaw moment due to rudder deflection coefficient	1/deg
C_X	Longitudinal force coefficient	–
C_{Y_β}	Side force due to sideslip angle coefficient	1/deg
$C_{Y_{\delta_a}}$	Side force due to aileron deflection coefficient	1/deg
$C_{Y_{\delta_r}}$	Side force due to rudder deflection coefficient	1/deg
C_Z	Normal force coefficient	–
c	Reference aerodynamic chord	m
D	Drag force	N
f_P	Thrust force	N
g_0	Gravitational constant	m/s^2
I_{11}	Roll mass moment of inertia	kg-m^2
I_{13}	Mass cross-product of inertia	kg-m^2
I_{22}	Pitch mass moment of inertia	kg-m^2
I_{33}	Yaw mass moment of inertia	kg-m^2
I_{sp}	Specific impulse	N-s
L	Lift force	N
M	Mach number	–
m	Mass	kg
m_f	Fuel mass	kg
m_{f_e}	Fuel mass expended	kg
n_z	Vertical load factor at IMU location	–
p	Roll rate	rad/s
q	Pitch rate	rad/s
$\bar{q}$	Dynamic pressure	N/m^2
r	Yaw rate	rad/s

Symbol	Definition	Unit
S	Reference surface area	m^2
u	Body longitudinal velocity	m/s
v	Body lateral velocity	m/s
w	Body vertical velocity	m/s
V	Airspeed	m/s
$\|\cdot\|_\infty$	H_∞-norm	–

1

Introduction

1.1. Context

Hypersonic vehicles (HV), capable of traveling at speeds and altitudes unreachable by conventional supersonic aircraft, represent a significant advancement in aerospace technology. Their exceptional speed enables drastic reductions in travel time, making them ideal for high-priority transport and emergency response scenarios. Additionally, the high altitudes achieved by HVs facilitate reconnaissance missions and extend beyond Earth's atmosphere, positioning them as promising candidates for reduced-cost space access as single-stage-to-orbit (SSTO) reusable vehicles.

Nevertheless, hypersonic flight is a relatively unexplored field. A lot of research is put into the development of this new technology by engineers all around the world. The HV modelling and control system design is a challenge of its own, as very limited open-source data is available. Extreme operating conditions, unique airframe configurations, and lack of high-fidelity aerodynamic models introduce significant uncertainties into the design, necessitating robust flight control systems (FCS) that can address them directly.

These conditions naturally fall under the H_∞ control framework. Since its development in the 1980's, it has become an industry standard because of its direct focus on stability and robust performance under system uncertainties and perturbations. In spite of that, classic H_∞ methods that rely on linear matrix inequalities or bi-linear matrix inequalities fail to address the modern needs of structured control systems with complex demands on performance objectives. However, recent advancements in optimization methods for H_∞ control synthesis approaches enable FCS design with pre-defined structures. Additionally, multi-objective and multi-model extensions with the new optimization methods open new possibilities for complex handling of performance and robustness requirements with reduced conservatism, thereby completely outclassing the original H_∞ control synthesis methods.

The new structured multi-objective H_∞ control was implemented into the toolboxes of MATLAB® and Simulink® just a few years ago. At this point, its implementation on hypersonic vehicles was not found in the open literature. This presents a major research gap, as the aforementioned HV field is precisely the one that needs advanced robust control systems of such kind. This paper aims to address this issue and apply the most modern H_∞ framework on a hypersonic vehicle.

1.2. Research goals

The AErospace dynamics and RObust CONtrol (AEROCON) research group aims to investigate robust hypersonic vehicle control using modern H_∞ design methods. This study explores the hypersonic vehicle design traits and operating conditions, as well as the previous FCS designs applied on them. A flexible framework would be established by developing a 6-degree-of-freedom (DoF) non-linear model of a hypersonic vehicle in MATLAB and Simulink, incorporating parametric uncertainty. The scope involves setting up programs for trimming, linearization, and synthesizing a structured robust flight control system using multi-objective H_∞ mixed sensitivity techniques. The FCS would then be implemented

and tested in a non-linear simulation to verify and showcase the structured controller performance.

Research Questions

Main research question:
How effectively can a structured flight control system designed with multi-objective H_∞ methods handle subsonic short-period dynamics of a hypersonic vehicle under multiple performance and robustness requirements?

Sub-questions:

1. **What makes the HV different from conventional aircraft?**
   (a) What are the HV mission profiles?
   (b) What are the HV unique design characteristics?
   (c) What are the challenges associated with HV stability and control?

2. **How to incorporate a non-linear HV model into Simulink?**
   (a) How to incorporate an aerodynamic model?
   (b) How to model the non-linear dynamics and kinematics?
   (c) How to model the HV actuators and sensors?

3. **How to develop a linear HV model with uncertainty?**
   (a) How is uncertainty modelled?
   (b) Which parameters are uncertain and in what range?
   (c) Which operating point to consider?
   (d) How is the non-linear model trimmed and linearized?
   (e) Which linear dynamics are to be controlled?

4. **How to design a robust FCS with multi-objective H_∞ methods?**
   (a) What are the working principles behind the new methods?
   (b) How is the multi-objective H_∞ control implemented practically?
   (c) What are the FCS robustness and performance design requirements?
   (d) How are the requirements transferred into practical constraints?
   (e) Which controller structure to choose?

5. **How to assess FCS performance and robustness?**
   (a) How is the FCS robust performance assessed in frequency domain?
   (b) How is the FCS robust performance assessed in time domain?
   (c) How to implement the FCS in the non-linear model?
   (d) How can the FCS robust performance be verified with disturbances and uncertainty in the non-linear simulation?

1.3. Research paper structure

This document consists of 5 chapters. Literature study is in chapter 2 and includes investigation on hypersonic vehicles in section 2.1, modelling of aerospace vehicles in section 2.2, robust control theory and FCS design basics in section 2.3, and the uncertainty modelling and robustness principles in section 2.4. Scientific article in chapter 3 forms the core part of this research. It is a standalone document with its own formatting, section and page numbering, introduction, conclusion, and references. The scientific article describes the entire procedure from the start to the end. Therefore, some of the information in chapter 3 may be repeating specific parts of chapter 2, but also present new information in-between. Chapter 4 then presents the details and nuances of GHAME modelling, linearization, and FCS design. The research then finishes with conclusion and recommendations in chapter 5.

2

Literature Research

2.1. Hypersonic vehicles

In general, the term ”hypersonic vehicle” refers to an aerospace vehicle that operates at speeds above Mach 5 within (but not limited to) the Earth’s atmosphere, normally above 25 km altitude. These vehicles have to deal with extreme conditions that arise at such velocities, which include aerodynamic heating from skin friction, vibrations and structural loads due to shock waves and boundary layer transitions, high dynamic pressure on the airframe, stability issues due to rapid changes in the aerodynamic environment, and more [1]. Such conditions pose significant engineering challenges to ensure the safety and success of hypersonic missions.

Many types of hypersonic vehicles exist, and each vehicle is built specifically to its mission profile. The differences can be related to horizontal or vertical take-off, flight altitude and space operation, payload type, etc. A ballistic missile that reaches hypersonic speed can be considered a hypersonic vehicle [2], as well as a boost-glide vehicle, which is carried to the top of the atmosphere by a rocket and then performs an unpowered glide at hypersonic speeds [3]. However, hypersonic vehicles considered within the scope of this research are those that perform a safe landing. Re-entry capsules with parachutes and expendable rockets will not be assessed.

2.1.1. Brief History

Despite the challenges mentioned earlier, the theory of hypersonic flight has actually been present since the 1930’s, when a boost-glide vehicle concept Silbervogel was proposed in Germany. The first operational vehicle that technically reached hypersonic speeds, although expendable, was the infamous Nazi Germany’s rocket V-2 developed in the 1940’s, which travelled at Mach 5 at its maximum. However, the actual pioneer among the hypersonic vehicles is considered to be the North American Aviation X-15, because not only did it reach 7273 km/h, but also because it was piloted and utilised a safe return. This rocket powered vehicle was developed in the 1950’s specifically as a hypersonic flight test vehicle, with 3 machines produced, which made a total of 199 flights [1].

As the Cold War went on, the two major superpowers kept financing bold and extremely expensive projects for reusable hypersonic vehicles, such as the Space Shuttle program in the USA, and the Buran in the USSR. Both were reusable and capable of orbital operations. The Space Shuttle came earlier, with its first flight in 1981. It was capable of delivering 25 tons of cargo to LEO and also return certain payloads from space back to the ground. The velocity at the moment of re-entry into the atmosphere was around Mach 25. However, its external fuel tank was expendable, the maintenance costs and turnaround times were enormous. Due to these reasons, multiple disasters, and other factors the program ended in 2011. The Buran was the Soviet competitor to the Space Shuttle. With the first and only flight in 1988, it had a similar appearance and technical characteristics, while the flight was actually autonomous. However, the main thrust was provided by the Energia rocket, so unlike the Space Shuttle, the Buran and its rocket were two separate vehicles. The program was quickly cancelled mainly because of lack of funding as the USSR came to its dusk. The USA then produced the Boeing X-37

Vehicle	Country of origin	First launch or flight (planned)	Maximum altitude	Maximum speed	Payload
Concept vehicles					
Hammer	Russia	No data	500 km	Orbital	800 kg
ZEHST	Europe	-2040	32 km	Mach 4	50–100 passengers
Lockheed SR-72	USA	-2030	No data	Mach 6	No data
PAK-DA	Russia	2020–2030	No data	Hypersonic	No data
SL-12 "Vimana"	UK	-2020	LEO	Orbital	6000 kg
Skylon	UK	-2019	200 km+	Orbital	15,000 kg
SOAR	Switzerland	2016 (test flight)	80 km	No Data	Up to 250 kg
Lynx	USA	-2015	100 km	Mach 2.7	1 passenger or 120 kg
LEA	France	-2015	No data	Mach 8	No data
Operational vehicles					
New Shepard	USA	29/04/2015	100 km	No data	11.3 kg
IXV	Europe	11/02/2015	Not applicable	7700 m/s	No data
Orion Multi Purpose Crew Vehicle	USA	05/12/2014	Beyond LEO	Not applicable	4 Passengers
EADS Spaceplane	Europe	05/06/2014 (drop test)	100 km	Over 3000 km/h	4 Passengers
Dream Chaser	USA	26/10/2013 (drop test)	Not applicable	Not applicable	7 Passengers
Falcon 9	USA	19/09/2013	GTO	Orbital	13,150 kg LEO, 4850 kg GTO
Space Ship Two	USA	29/04/2013	110 km	4200 km/h	6 Passengers
STIG B	USA	06/10/2012	100 km	No data	50 kg
Masten Xaero	USA	2011	30 km	No data	10 kg
Dragon	USA	08/12/2010	Not applicable	Not applicable	3310 kg
X-51	USA	26/05/2010	21 km	Mach 5.1	No data
X-37	USA	22/04/2010	LEO	Orbital	No data
HIFIRE	Australia/USA	07/05/2009	Various	Hypersonic	No data
Shenlong	China	11/12/2007 (drop test)	No data	No data	No data
X-43 Hyper-X	USA	02/06/2001	33.5 km	Mach 9.6	No data

Figure 2.1: Overview of modern hypersonic vehicles under development [1]

as an unmanned Orbital Test Vehicle, for which most of the information is unfortunately classified. Its development started in 1999, and multiple flights took place in 2010's [1].

The next big step in the evolution of reusable hypersonic vehicles came with commercialisation of space, as the Scaled Composites SpaceShipOne (although technically not a HV) opened a potential market for space tourism. Air-launched and climbing above 100 km altitude, it paved the way for SpaceShipTwo. The vehicles such as the Rocketplane, the XCOR Lynx, Airbus Spaceplane are among some of the ones that followed. Falcon 9 made by SpaceX is perhaps the biggest achievement in commercial space flights, as it is currently the only reusable launcher capable of orbital operations. Lately, a lot of work has been going on in further development of reusable hypersonic vehicles. So far, most of the practically working HVs are those primarily designed for sub-orbital flight trajectories for space tourism or for orbital operations as alternative means for satellite deployment. Figure 2.1 provides a comprehensive overview of current developments.

There are, however, ongoing conceptual designs for air-breathing hypersonic vehicles which remain within the atmosphere. Most of the work done is research-based, but there are also actual vehicles in development. The most notable would be the Lockheed SR-72, which is said to be the next generation of the legendary SR-71. The SR-72 is expected to be utilised for surveillance purposes, or as a high speed weapon carrier. Not surprisingly, there is not much open information on it, apart from featuring an air-breathing combined cycle engine and Mach 6 design speed. It is expected to make its first flight in 2025 [1].

2.1.2. Practical Overview

With the ongoing trends, it is possible to highlight two main types of hypersonic vehicles. First are the Space Launchers, as comes from the name, refer to vehicles that can "reach, operate in and return from orbit without expending the vehicle" [1]. With the primary goal being the orbit, they are usually designed to exit the earth's atmosphere in shortest possible time. Thus, they accelerate through hypersonic speeds during the climb to orbit, and they have to decelerate from hypersonic speeds after re-entry for landing. Therefore, their time spent in hypersonic flight regime within the atmosphere is limited, but the velocities encountered can be as high as Mach 25. An example of a space launcher would be the Boeing X-37B, shown in Figure 2.2.

The second type is the Hypersonic Transport. It refers to the vehicles which are designed to cruise at hypersonic speeds within the Earth's atmosphere, normally with an air-breathing engine, which means they have to deal with aerodynamic forces across their entire flight envelope. That poses a different kind of challenge, one where the velocities are smaller, but they have to be sustained throughout the flight.

Figure 2.2: Boeing X-37B as an example of Space Launcher class [1]

A perfect example of Hypersonic Transport would be Lockheed-Martin SR-72, shown in Figure 2.5.

Figure 2.3: Lockheed-Martin SR-72 as an example of Hypersonic Transport class [1]

Distinct from one another, the two ”classes” of hypersonic vehicles considered have a large potential for the aerospace industry. Apart from apparent military use as a fast weapon delivery system, hypersonic vehicles can be used in other applications, with the two types sharing some similarities. Space Launchers offer potential for commercial human space flights, alternative way of satellite deployment, research and technology testing. Hypersonic Transport, on the other hand, opens new possibilities for fast point-to-point transportation, upper atmosphere research, surveillance at unreachable speeds and altitudes without dependency on orbits, and essentially all the potential benefits of an aircraft with much greater flight velocity and service ceiling. Both the Space Launchers and Hypersonic Transports lean towards full reusability, with faster turnaround times and supposedly lower launch costs than expendable vehicles. However, the combination of the mission profiles and the stated goals makes it a great engineering challenge. At this moment of time, only SpaceX managed to produce reusable Space Launchers that are competitive on the market, while Hypersonic Transport has not yet been successfully operational in the industry at all [1].

This is because the design of hypersonic vehicles must encounter unique problems associated with their mission profile, for some of which the technology is simply not mature enough. Some of them are presented below:

- **Aerodynamic issues.** The ultimate goal is to reach operational speed and altitude (or orbit)

with the least amount of drag. Therefore, an ideal trajectory would be a near-vertical climb at the highest acceleration. Ballistic type vehicles with rocket propulsion can accomplish it, but at a cost of cross range and controllability, thus being more suitable for space missions. The winged or lifting-body vehicles, on the other hand, require smaller propulsion system, offer a more controllable flight, and can provide safe launch-abort capabilities. Furthermore, a horizontal runway is simpler than a vertical launch site. However, the climb angle is reduced compared to non-lifting vehicles, thus requiring greater time to reach operational conditions. Apart from that, there is also another trade-off between winged vehicles, which offer a better lift/drag (L/D) ratio at subsonic speeds compared to lifting-body, but become disadvantageous at hypersonic regimes as the wings produce larger drag and have to be reinforced to sustain structural loads. Regardless of the choice, the L/D ratio for hypersonic vehicles is generally much lower than for conventional aircraft [4]. Designing the aerodynamic configuration thus becomes a serious task. For Hypersonic Transport, the entire configuration strongly affects the propulsion system, too, as the engine inflow compresses along the body [5].

- **Propulsion issues.** This is one of the most problematic aspects for hypersonic vehicles. Not only that a large amount of thrust has to be provided, but for the Hypersonic Transport that thrust has to be maintained during cruise, and the propulsion system can have both upper and lower operational speed limits. Most common propulsion systems can be broken into three types [1]:
  - Air-breathing engines include turbojets, ramjets, and scramjets, with ascending operational velocities respectively. Basically, the previous engine upper speed limit is the next ones lower limit. This is precisely why an air-breathing Hypersonic Transport cannot use just one air-breathing engine, but needs a combination.

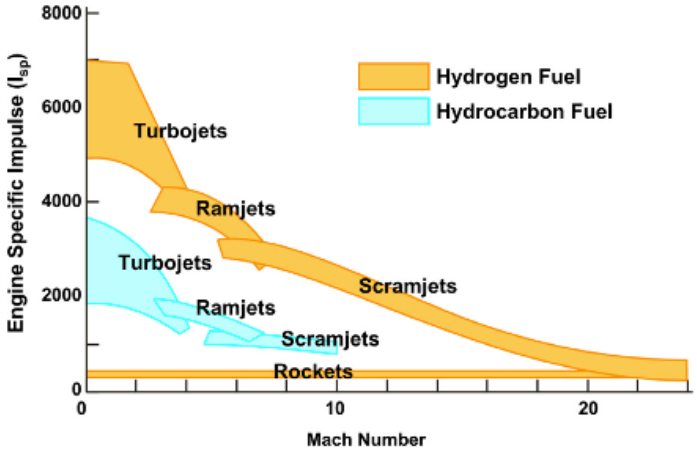


Figure 2.4: Properties and operational ranges of various propulsion system cycles [1]

  - Rocket propulsion, which does not rely on the atmospheric conditions and can operate across entire velocity range. Requires large amounts of fuel onboard, provides higher thrust but lower specific impulse compared to air-breathers.
  - Hybrid engines are essentially all kinds of possible combinations between the two, and are primarily needed for Hypersonic Transports as a method to operate at subsonic and hypersonic regimes. Of all hybrids, a turbine based combined cycle stands out, which uses turbojet or low bypass turbofan engine to speed up the vehicle from ground standstill up to Mach 3, and then switches to a ramjet. This is the system on the future SR-72, shown below.

- **Airframe issues.** The extensive vibrations and loads on the structure require a strong hull to deal with. However, the bounds on the airframe weight push for expensive materials to be used, as the conventional ones will not withstand the conditions. Another issue is the extensive heat loads that the airframe is subject to. Therefore, designing the airframe is a complex task, even though a hypersonic vehicle is less subject to extreme g-loads than a fighter jet. On a separate

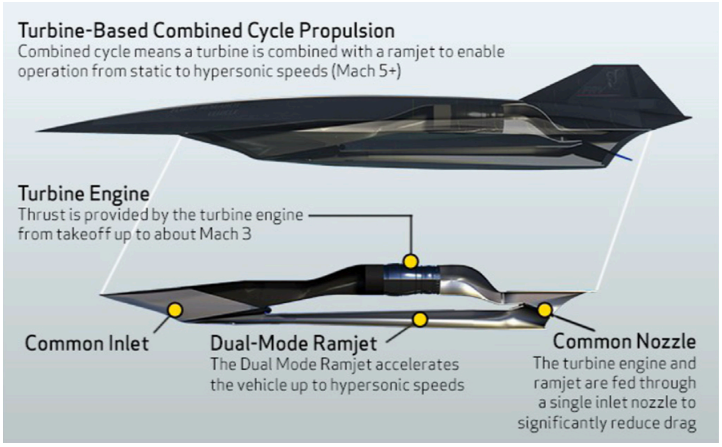


Figure 2.5: Combined cycle propulsion system of SR-72 [1]

note, it is of paramount importance to keep the airframe from longitudinal bending, as it affects the flight control for these vehicles.

Apart from technical troubles the hypersonic vehicles face operational ones, such as poor reliability (estimates), costly maintainability due to complex technology employed, and legislation issues [1].

2.1.3. Stability and control issues

The scope of the further research, however, is focused on the stability and control of the air-breathing hypersonic vehicles that belong to the Hypersonic Transport class. From this point, the term Hypersonic Vehicle (HV) refers to this class only.

Fidan [6] points out that the hypersonic research only started to focus on the flight control aspects since the 1990's. A wide range of operating conditions and mass distributions result in significant variations of dynamic characteristics of HVs over the flight envelope, more than any other aircraft. During hypersonic flight these vehicles have to deal with extreme conditions, which include aerodynamic heating from skin friction, vibrations and structural loads due to shock waves and boundary layer transitions, high dynamic pressure on the airframe, stability issues due to rapid changes in the aerodynamic environment, and more [1, 6]. All of them can affect the dynamic behavior of the system, so even when a high-fidelity model is available, some changes simply can not be predicted. When compared with conventional aircraft, the longitudinal models of hypersonic vehicles are usually unstable, non-minimum phase systems perturbed by considerable amount of model uncertainty [7]. Therefore, the control system design for hypersonic vehicles must address the model uncertainties and be robust to changes in the system parameters [4].

The center of gravity (CG) and center of pressure (CP) positions migrate significantly across the flight envelope. The engines are tightly integrated into the airframe and usually take up most of the aircraft volume, and are normally mounted in the back. Due to the mission characteristics typical for a hypersonic vehicle, there is a significant change in fuel quantity (which also takes up a significant portion of take-off weight). Thus, as the fuel is burned, the CG moves aft towards the engines. Meanwhile, the aerodynamic center also migrates aft when the vehicle velocity exceeds Mach 1, also affecting the stability characteristics of the vehicle. Logically, the CP moves back to its original position at the approach and landing phases (when the speeds are subsonic), while the CG remains aft at the end off mission due to aforementioned reasons. Those aft CG positions and hypersonic aerodynamics result in long, slender and sharp planforms for hypersonic vehicles that rely on aerodynamic lift [1].

Structural deformations can greatly affect control of the vehicle at hypersonic speeds. Excessive heat loads on the structure create vibrations in the airframe, and can also result in structural deformations, causing parameter variations [4]. Not only the airframe can deform from thermal loads, it can also bend due to aerodynamic loads. This is especially the case for Hypersonic Transport, with their typically long slender bodies and large engines tightly integrated into the airframe with light-weight structures. The elastic-body interactions represent a significant uncertainty source due to low-frequency vibrations and

non-uniform aerodynamic heating [6], whereas sudden changes in pitch rate can cause the frame to slightly bend at hypersonic speeds [1], intervening in flight control. The strong couplings between the aerodynamics, the airframe structure, and the propulsion system of HV are argued to be the most complex of any aircraft [8].

This strong longitudinal coupling between the aerodynamics, the airframe and the propulsion system is the most notable uncertainty source, and has been at the center of HV modelling and control research since the 1990's [6]. Various control design approaches have been implemented over this time, for which a detailed description can be found in the introduction of the scientific article in chapter 3.

In a nutshell, the aforementioned non-linearities and uncertainties naturally need robust control approaches, thus most of the initial designs were using the H_∞ and μ-synthesis methods first, such as in [5, 8, 9, 10, 11]. However, the tools were not developed enough at the time, and were generally underperforming. Specifically, HV controllers with classical H_∞-synthesis suffered performance degradation under simultaneous uncertainties [8], whereas μ-synthesis is conservative when there are real perturbations in the system [12, 13]. Furthermore, both methods lead to high-order controllers, while reducing the controller order showed loss of robustness [12]. The classical H_∞ and μ-synthesis were also not flexible enough to conveniently handle a large and diverse amount of performance specifications [12, 13]. Due to these drawbacks, the focus then mostly switched to non-linear and adaptive control methods for HV in 2000's [7, 14].

Meanwhile, the H_∞ theory had a major breakthrough in the applied algorithms that solved the aforementioned problems [15]. These new non-smooth optimization methods were first introduced in 2006 and then implemented into MATLAB Robust Control Toolbox [16] in 2015. However, no application of these modern H_∞ techniques on HV were found in the literature.

2.2. Modelling

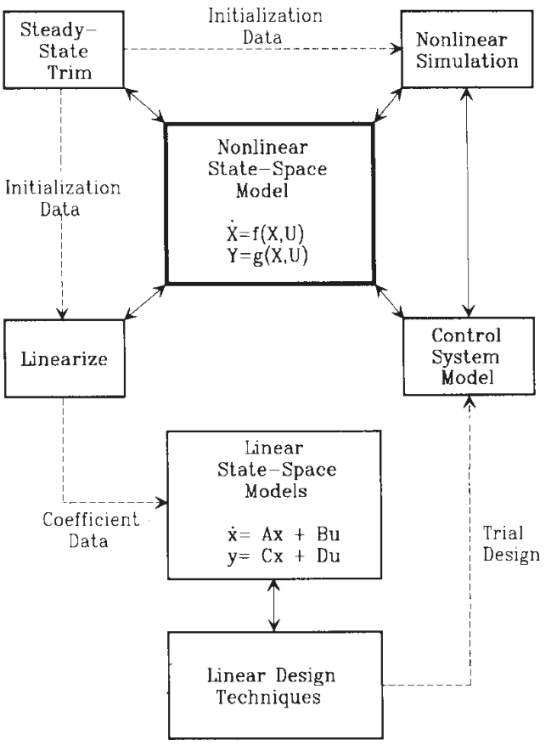


Figure 2.6: State-space modelling overview [17]

The general flight control system development process can be outlined as follows. The modelling starts with deriving the non-linear equations of motion. The system is then trimmed and linearized to arrive at the desired state-space representation. The linear robust control techniques are applied to it to

synthesize the controllers. The design is then evaluated with various robustness measures and tested in a non-linear simulation. The basic outline of the entire process is shown in Figure 2.6.

2.2.1. Non-linear modelling of aerospace vehicles

Assembling a non-linear vehicle model serves as a ground base for all the follow-up work. The motion of a point mass through the air is governed by physics laws, and is thus described by universal equations of dynamic and kinematic relations, which have all been derived and studied extensively. P. Zipfel [18] provides a comprehensive overview of those equations. The model choice falls on 6 degrees-of-freedom (DoF) aircraft equations, which assume flat, non-rotating Earth. This is considered an acceptable simplification from the elliptical Earth equations, since the model will be used for control law development at selected operating points rather than for full-flight simulation. Generally, non-linear modelling of an aerospace vehicle can be broken down into equations of motion, kinematic equations, aerodynamics, and forces and moments relations.

Equations of motion

The equations of motion originate from the first principles of Newton's and Euler's laws. The translational equations are based on the Newton's second law in the inertial frame of reference I, but the chosen subsonic flight conditions allow flat-Earth assumption, and thus the Earth frame of reference E becomes identical to I. The Newtons second law can thus be expressed in one formula:

$$mD^E v_B^E = f_{a,p} + mg \tag{2.1}$$

where m is mass of the vehicle (assumed constant), v_B^E is the velocity vector of the vehicle's center of mass wrt the inertial frame E, D^E is the rotational time derivative wrt to frame E, $f_{a,p}$ is a vector of aerodynamic and propulsion forces, and g is the gravity force acting on the vehicle. The equations of motion are not yet complete, however, because a term relative to tangential acceleration is missing, and the coordinate system needs to be identified. Choosing the body frame of reference B as the most convenient to work with, the rotational time derivative can also be replaced with ordinary time derivative when expressed wrt to frame B instead of E. The equations of motion in their general form then become:

$$m\left[\frac{dv_B^E}{dt}\right] + m\left[\Omega^{BE}\right]^B\left[v_B^E\right]^B = \left[f_{a,p}\right]^B + m\left[g\right]^B \tag{2.2}$$

Where the second term on the left-hand side corresponds to the aforementioned tangential acceleration, with Ω^{BE} the angular velocity vector of the vehicle wrt frame E. Lastly, the gravitational acceleration is the only term that is better analysed in the local-level coordinate system, because in that case it is conveniently comprised only of $[\bar{g}]^L = [0 \quad 0 \quad g]$. To include this term into the body reference frame, a direction cosine matrix (DCM) $[T]^{BL}$ is needed to provide that rotation from frame B to frame L. The DCM will be elaborated on in next section. The updated and final general equation of motion is then:

$$m\left[\frac{dv_B^E}{dt}\right] + m\left[\Omega^{BE}\right]^B\left[v_B^E\right]^B = \left[f_{a,p}\right]^B + m\left[T\right]^{BL}\left[g\right]^L \tag{2.3}$$

and with expanded matrices:

$$m\left\{\begin{bmatrix}du/dt\\dv/dt\\dw/dt\end{bmatrix}^B + \begin{bmatrix}0 & -r & q\\r & 0 & -p\\-q & p & 0\end{bmatrix}^B\begin{bmatrix}u\\v\\w\end{bmatrix}^B\right\} = \begin{bmatrix}f_{a,p_1}\\f_{a,p_2}\\f_{a,p_3}\end{bmatrix}^B + \begin{bmatrix}t_{11} & t_{12} & t_{13}\\t_{21} & t_{22} & t_{23}\\t_{31} & t_{32} & t_{33}\end{bmatrix}^{BL}\begin{bmatrix}0\\0\\mg\end{bmatrix}^L \tag{2.4}$$

These can then be re-arranged into non-linear state-space representation, and also in scalar form by working out the matrices, if necessary.

Cancelling out mass and integrating the equations twice leads to the distance travelled s_{BE} of body mass relative to frame E as $\left[D^E s_{BE}\right] = \left[v_B^E\right]$. The integration is best done in local reference frame, meaning the DCM is again present:

$$\begin{bmatrix}(ds_{BE}/dt)_1\\(ds_{BE}/dt)_2\\(ds_{BE}/dt)_3\end{bmatrix} = \overline{\begin{bmatrix}t_{11} & t_{12} & t_{13}\\t_{21} & t_{22} & t_{23}\\t_{31} & t_{32} & t_{33}\end{bmatrix}}^{BL}\begin{bmatrix}u\\v\\w\end{bmatrix}^B \tag{2.5}$$

The rotational EoM are governed by Euler's law, and are worked out in a similar fashion as the translational ones. Taking the E frame as the reference, the time rate of change of angular momentum l_B^{BE} equals the applied moment m_B:

$$D^E l_B^{BE} = m_B \tag{2.6}$$

in which $l_B^{BE} = I_B^B \omega^{BE}$, the body moment of inertia wrt CG multiplied by the angular velocity. Once again choosing the body frame B as the reference, the Euler's transformation yields:

$$D^B l_B^{BE} + \Omega^{BE} l_B^{BE} = m_B \tag{2.7}$$

Expanding the angular momentum vector in the rotational derivative term by using the chain rule leads to:

$$D^B l_B^{BE} = D^B I_B^B \omega^{BE} + I_B^B D^B \omega^{BE} = I_B^B D^B \omega^{BE} \tag{2.8}$$

which is where one of the key assumptions takes place, as the hypersonic vehicle under subsonic flight regime is assumed to have rigid body, resulting in the time rate of change of MOI to be zero $D^B I_B^B = 0$. Lastly, choosing once again the body coordinate system for simple time derivative and constant tensor MOI, the rotational EoM become

$$\left[\frac{d\omega^{BE}}{dt}\right]^B = \left(\left[I_B^B\right]^B\right)^{-1} \left(-\left[\Omega^{BE}\right]^B \left[I_B^B\right]^B \left[\omega^{BE}\right]^B + \left[m_B\right]^B\right) \tag{2.9}$$

For additional fidelity for an aircraft model, the expression can be upgraded to include angular momentum from propulsion devices, such as turbojets. In that case, with an assumption of constant rotary speed, the equation now includes the term $l_{B_R}^{B_R E}$ corresponding to that momentum, and becomes:

$$\left[\frac{d\omega^{BE}}{dt}\right]^B = \left(\left[I_B^B\right]^B\right)^{-1} \left[-\left[\Omega^{BE}\right]^B \left(\left[I_B^B\right]^B \left[\omega^{BE}\right]^B + l_{B_R}^{B_R E}\right) + \left[m_B\right]^B\right] \tag{2.10}$$

For an aircraft-like vehicle, the moment of inertia has a trait that the 2nd axis is a principle MOI axis due to symmetrical shape (hypersonic vehicles are expected to have a symmetry plane), so the MOI matrix is:

$$\left[I_B^B\right]^B = \begin{bmatrix} I_{11} & 0 & I_{13} \\ 0 & I_{22} & 0 \\ I_{31} & 0 & I_{33} \end{bmatrix} \tag{2.11}$$

Finally, the expanded Equation 2.10 in the matrix form is shown below:

$$\begin{aligned} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix}^B = & \left(\begin{bmatrix} I_{11} & 0 & I_{13} \\ 0 & I_{22} & 0 \\ I_{31} & 0 & I_{33} \end{bmatrix}^B \right)^{-1} \left(- \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}^B \times \right. \\ & \left. \left(\begin{bmatrix} I_{11} & 0 & I_{13} \\ 0 & I_{22} & 0 \\ I_{31} & 0 & I_{33} \end{bmatrix}^B \begin{bmatrix} p \\ q \\ r \end{bmatrix}^B + \begin{bmatrix} l_R \\ 0 \\ 0 \end{bmatrix}^B \right) + \begin{bmatrix} m_{B_1} \\ m_{B_2} \\ m_{B_3} \end{bmatrix}^B \right) \end{aligned} \tag{2.12}$$

in which the l_R term can be left out if the angular momentum from the rotary propulsion system is not accounted for. This concludes the main equations of motion for a rigid-body aircraft-like aerospace vehicle under flat-Earth assumption, with Equation 2.4 describing translational motion, and Equation 2.10 the rotational motion.

Kinematics and environment

The kinematic relations include the computation of the angle of attack α, the angle of sideslip β, and the DCM matrix from the body frame to the local frame T^{BL}. The kinematics involving L-frame velocity and distance were covered in Equation 2.5, as they conveniently followed the flow of EoM derivation.

The angles α and β are easily computed with the access to the velocity components $[u \quad v \quad w]$, as follows:

$$\begin{aligned} \alpha &= \arctan\left(\frac{w}{u}\right) \\ \beta &= \arcsin\left(\frac{v}{\sqrt{u^2+v^2+w^2}}\right) \end{aligned} \tag{2.13}$$

The DCM is needed to compute the orientation of the vehicle wrt the inertial I (in this case also E) reference frame with Euler angles $[\phi \quad \theta \quad \psi]$ using the angular velocities in the body reference frame $[p \quad q \quad r]$. It can be computed by 3 different methods. The first is using the rotation tensor differential equations, in which the transformation matrix T^{BL} from body frame B to local frame L. The differential equations are expressed in the following form:

$$\left[\frac{dT}{dt}\right]^{BL} = \overline{[\Omega^{BE}]}^{B} [T]^{BL} \tag{2.14}$$

where

$$[T]^{BL} = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\theta\sin\phi \\ \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{2.15}$$

and $\overline{[\Omega^{BE}]}^{B}$ is the angular velocity tensor. This method leads to 6 linear differential equations to be solved, making it the least computationally effective of the tree. The Euler angles are not directly available, and it requires initial computations. However, the transformation matrix $[T]^{BL}$ is available straight away.

The second method is using the Euler angle differential equations, where the it converges to the following expression:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi / \cos\theta & \cos\phi / \cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.16}$$

This approach naturally allows for Euler angles to be directly calculated, and has half as many differential equations to be solved as the rotation tensor method. However, the differential equations are non-linear, and the transformation matrix is not directly available. Another important trait is that there is a singularity occurring at $\theta = \pm\pi/2$, where there is no distinction between roll and yaw motion anymore.

Last but not least, there is a quaternion approach. An entire quaternion algebra exists, but in a nutshell the quaternions are a way to describe 3D rotations using 4-dimensional vectors. Not diving into derivation theory, the differential equations for this case take the following form:

$$\begin{Bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} = \frac{1}{2} \begin{Bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{Bmatrix} \begin{Bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{Bmatrix} \tag{2.17}$$

where the q_i are the four quaternions with their linear relationship to the respective rates of change $\dot{q}_i$. They have to be initialized, which can be achieved with Euler angles:

$$\begin{aligned} q_0 &= \cos\left(\frac{\psi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) + \sin\left(\frac{\psi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi}{2}\right) \\ q_1 &= \cos\left(\frac{\psi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi}{2}\right) - \sin\left(\frac{\psi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) \\ q_2 &= \cos\left(\frac{\psi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) + \sin\left(\frac{\psi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi}{2}\right) \\ q_3 &= \sin\left(\frac{\psi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi}{2}\right) - \cos\left(\frac{\psi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi}{2}\right) \end{aligned} \tag{2.18}$$

The Euler angles and the DCM $[T]^{BL}$ can then be computed from the quaternions directly:

$$\begin{aligned} tan\,\phi &= \frac{2(q_2q_3 + q_0q_1)}{q_0^2 - q_1^2 - q_2^2 - q_3^2} \\ tan\,\theta &= -2(q_1q_3 + q_0q_2) \\ tan\,\psi &= \frac{2(q_1q_2 + q_0q_3)}{q_0^2 + q_1^2 - q_2^2 - q_3^2} \end{aligned} \tag{2.19}$$

The first of the Equations 2.19 has singularities at $\phi = \pm 90°$ and the third at $\psi = \pm 90°$. Fortunately, they are not inside the differential equations, and can thus be programmed around to avoid them.

$$[T]^{BL} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{2.20}$$

The quaternion method uses only four differential equations, all of which are linear, and does not have any singularities within the differential equations. On the downside, the transformation matrix and Euler angles are not available directly, but can be straightforwardly calculated. Zipfel [18] argues for this method to be the most reliable and effective of the three. However, using the quaternions is not as convenient for trimming and linearization purposes. In that case the quaternions are passed as system states, which is not as intuitive as using the Euler angles. Therefore, the Euler angle differential equations method was used for linearization, whereas the quaternion method was applied for the non-linear simulation.

Aerodynamic model, forces and moments

The forces $f_{a,p}$ from Equation 2.4 and moments m_B from Equation 2.10 are necessary for the equations of motion and can be computed using familiar straightforward relationships, presented below:

$$[f_{a,p}]^B = \begin{bmatrix} f_{a,p_1} \\ f_{a,p_2} \\ f_{a,p_3} \end{bmatrix} = \begin{bmatrix} \overline{q}SC_X + f_P \\ \overline{q}SC_Y \\ \overline{q}SC_Z \end{bmatrix} \tag{2.21}$$

$$[m_B]^B = \begin{bmatrix} m_{B_1} \\ m_{B_2} \\ m_{B_3} \end{bmatrix} = \begin{bmatrix} \overline{q}SbC_l \\ \overline{q}ScC_m \\ \overline{q}SbC_n \end{bmatrix} \tag{2.22}$$

where $\overline{q}$ is the dynamic pressure and f_P is the thrust force acting through the longitudinal axis of the body. However, acquiring the other components in the above equations is a challenge of its own.

Unlike all the equations above, which focused on an arbitrary object flying through a 3D space, this stage of the non-linear modelling process is directly dependent on the vehicle type and its characteristics. The parameters such as the reference area S, the reference span b, and the chord length c are geometrical and come directly from the configuration. The 6 aerodynamic force and moment coefficients C_i depend on a number of parameters, such as Mach number M, aerodynamic angles α, β, altitude h, control surface deflections δ_s, thrust f_P, and vehicle configuration [17]. Not only the configuration includes the basic geometry, but also its variations with landing gear position, flaps, etc.

Moreover, the aerodynamic coefficients are composed of their static and dynamic components. The former ones cover the stationary flight and can simply be measured by fixing a physical vehicle model in a wind tunnel, or using theoretical methods such as computational fluid dynamics (CFD) or a combination of analytical and empirical data (e.g., DATCOM [19]). The latter ones capture the changes due to angular velocities and translational accelerations, and thus can only be measured with an oscillating model in the wind tunnel, or during an actual flight test. Basically, for a coefficient i the general equation is:

$$C_i = C_i(\alpha, \beta, M, h, \delta_s, T_c) + \sum \Delta C_{i_j}(\alpha, \beta, M, h, \delta_s, T_c) \tag{2.23}$$

where the first term on the right-hand-side is the static coefficient, and the second term is a general combination of the derivative terms related to accelerations and angular velocities. Notice that the parameters in the brackets are a generalisation of what the coefficients are usually a function of. The examples of the derivative terms C_{i_j} include

$$C_{l_p}, \quad C_{m_q}, \quad C_{n_r}, \quad C_{l_r}, \quad C_{n_p}, \quad C_{L_q}, \quad C_{Y_p}, \quad C_{Y_r}, \quad C_{L_{\dot{\alpha}}}, \quad C_{m_{\dot{\alpha}}}.$$

The last two terms are called acceleration derivatives [17]. When accelerating, the aerodynamic angles derivatives become non-zero, altering flow-field and creating an airspeed-dependent delay before downwash and sidewash changes take effect at the tail. The acceleration derivatives capture those conditions. The terms related to the angular rates $[p \quad q \quad r]$ are called the damping derivatives, and originate from the fact that a naturally stable vehicle would resist angular velocities and damp them using a moment in the opposite direction. Generally, for a dimensionless damping force or moment from Equation 2.23, the equation is:

$$\Delta C_{i_j} = C_{i_j} \times \frac{k}{2V} \times rate \tag{2.24}$$

where the C_{i_j} can be one of the $[p \quad q \quad r]$ derivatives coefficients listed above, k is either the reference span for roll and yaw rates, or mean aerodynamic chord for pitch rate, and V is the airspeed.

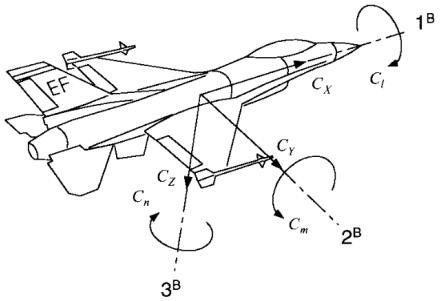


Figure 2.7: Force and moment coefficients in the body reference frame [18]

When the aerodynamic model is derived, the coefficients can be expressed in various ways, ranging from Taylor-series expansion to multi-dimensional look-up tables. According to P. Zipfel [18], the coefficients are normally defined in the body coordinates, with alignment as shown in Figure 2.7 [18], while the use of stability axis is more old-fashioned. Nevertheless, the aerodynamic model data is not an easy source to come by for some vehicle configurations, including hypersonic vehicles. Therefore, it is still common to see the use of C_L and C_D in the stability frame instead of C_X and C_Z in the body frame. The stability frame has its longitudinal axis vertically aligned with with velocity vector rather than the body axis. The picture shows an overview of the two reference frames Figure 2.8, and also the wind reference frame, which is angled with the velocity vector completely.

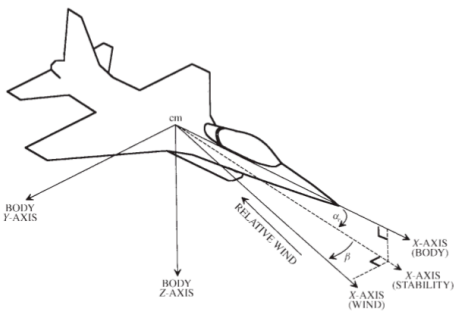


Figure 2.8: Axes definitions and aerodynamic angles [17]

The translation from the body axis to the stability axis can be achieved by a simple rotation of the frame, using the following transfer matrix (and its inverse for reverse direction) [13]:

$$[T]^{BS} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{2.25}$$

As mentioned earlier, acquiring the aerodynamic coefficients for a certain configuration is a challenge of its own. While many aerodynamic models exist for conventional aircraft, this is not a similar case for hypersonic vehicles, mainly due to relatively limited research and complexity of hypersonic flight tests. Therefore, the field is not saturated with publicly available models for hypersonic vehicle configurations. Due to the ongoing research of the strongly coupled structure and propulsion with aerodynamics, as mentioned in subsection 2.1.3, majority of the Hypersonic Transport models are only longitudinal [20]. Nevertheless, there are three models that are most widely used for control design of HV.

- **NASA Winged-cone HV.** The most frequently appearing 6-DoF model in HV research field is the NASA Winged-cone configuration hypersonic vehicle [21], with full technical memorandum and aerodynamic data open for public use. It is a manned, horizontal take-off and landing, single-stage-to-orbit (SSTO), air-breathing Hypersonic Transport. The model, however, is built solely with analytical programs in the 1990's using Aerodynamic Preliminary Analysis System (APAS) [22], and does not include any actual flight test data.

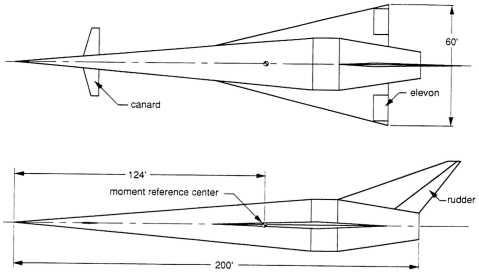


Figure 2.9: NASA Winged-cone HV configuration [21]

- **Dryden GHAME.** Another publicly available 6-DoF model is the Generic Hypersonic Aerodynamic Model Example (GHAME) developed at Dryden Flight Research Facility in the 1990's [23]. The main purpose was to allow possibility of control and guidance public research of hypersonic vehicles for commercial and military purposes. The GHAME was also developed with SSTO concept in mind, with a generalised winged configuration for an air-breathing hypersonic vehicle [24]. Zipfel [18] uses GHAME data for hypersonic vehicle simulations. The data comes in the form of aerodynamic tables, and is composed of both theoretical methods, such as Newtonian impact flow, and empirical data of similar aircraft and the Space Shuttle [24]. There is also a second set of data computed with APAS.

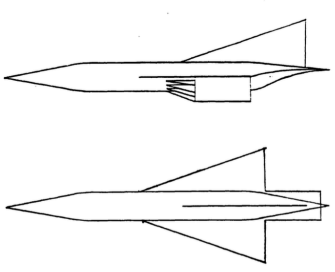


Figure 2.10: Dryden GHAME original configuration [24]

- **Bolender and Doman HSV.** The latest model was developed by Bolender and Doman in 2005 at the Air Force Research Laboratory [25]. It was derived from first principles with basic geometry and includes structural bending modes to explicitly account for the aeroelastic effects in hypersonic flight. The model is thus flexible-body, but lacks lateral degrees of freedom and empirical data. Although the model was found to be implemented in Simulink with further trimming, linearization, and control law development in [26, 27], the aerodynamic data were not found readily available to the public.

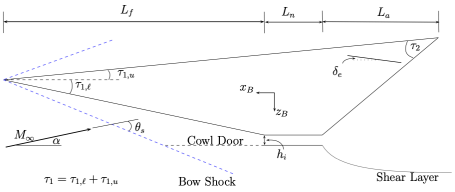


Figure 2.11: Bolender and Doman flexible HSV model [25]

For this research, GHAME model was selected as the most appropriate choice for multiple reasons. First, the aerodynamic tables of this model are ready to use from Zipfel's FORTRAN code [18]. Secondly, GHAME is the only one of the three models that has empirical data incorporated into the aerodynamics. It has 6 degrees of freedom, making future research possible in the lateral motion as well. Additionally, it rarely appears in literature compared to the other two.

2.2.2. Trimming & Linearization

After establishing a non-linear model, linearization is the next step in order to arrive at the state space representation of the system and be able to use the vast variety of linear control design techniques. The non-linear equations of motion derived earlier can be represented with [17]:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= g(x(t), u(t)) \end{aligned} \tag{2.26}$$

Where f and g are arrays of continuous single-valued functions, x is a vector of state variables, u is the vector of input variables and y is the vector of output variables.

Trimming at operating points

The operating points are normally established as a grid of operating conditions within a flight envelope. The organization of the grid is a separate process out of the scope of this research, but the potential points themselves can be taken on an individual basis for model analysis. The system is then trimmed to these operating points.

A trimmed condition is reached when the aircraft is in steady-state, meaning that the state derivatives $\dot{u}$, $\dot{v}$, $\dot{w}$, and $\dot{p}$, $\dot{q}$, $\dot{r}$ are essentially zero [17]. As a result, the velocity, altitude, and attitude remain constant without the need to continuously adjust the control inputs. When the equilibrium is reached, the model is linearized by calculating the partial derivatives of the non-linear equations of motion with respect to the constant state variables and inputs, specific to that operating point.

Using the earlier notation, an equilibrium point can be represented by [28]:

$$\begin{aligned} 0 &= f(\bar{x}, \bar{u}) \\ \bar{y} &= g(\bar{x}, \bar{u}) \end{aligned} \tag{2.27}$$

for some state values $\bar{x}$ at equilibrium and the necessary (trimmed) control inputs $\bar{u}$ to hold the system at that equilibrium. The trimmed output is then $\bar{y}$.

Some additional constraints are applied based on the desired trim condition. The common operating

points are outlined below [17]:

$$\begin{array}{ll} \text{steady wings-level flight:} & \phi, \dot{\phi}, \dot{\theta}, \dot{\psi} \equiv 0 \quad (\therefore p, q, r \equiv 0) \\ \text{steady turning flight:} & \dot{\phi}, \dot{\theta} \equiv 0, \quad \dot{\psi} \equiv \text{turn rate} \\ \text{steady pull-up:} & \phi, \dot{\phi}, \dot{\psi} \equiv 0, \quad \dot{\theta} \equiv \text{pull-up rate} \\ \text{steady roll:} & \dot{\theta}, \dot{\psi} \equiv 0, \quad \dot{\phi} \equiv \text{roll rate} \end{array} \tag{2.28}$$

Analytically, the process of trimming involves setting the aforementioned state derivatives to zero and the operating parameters to the desired values, and then backward calculating the necessary control inputs. Practically, the trimming becomes a cost function for which numerical optimisation algorithms are readily available [17].

One of such programs is the $findop$ function within the MATLAB environment. To use it properly, some additional parameter constraints must be considered for trimming. Each state, input, and output has an initial assigned (guessed) value, a free/fixed trigger to allow the initial value to vary, steady-state trigger, a range of allowed values, and a range of allowed rates in case a parameter is not in steady-state. These need to be carefully considered when using the trimming program, since it is very easy to overconstrain the algorithm with unnecessary requirements. For instance, it makes no sense to constrain the x^E or y^E coordinates of the vehicle, they do not impact the linear system, yet may disrupt the trimming process. The actuators must be constrained with their respective deflection limits. Unless specific requirements are defined for outputs that are part of the state vector, it is best to leave them free and constrain their respective states instead.

Jacobian linearization

The standard linearization method for non-linear equations of motion is the Jacobian linearization. It consists of calculating the Jacobian matrices with the small perturbation method.

By definition, the Jacobian matrix for a system of n differential equations f is given as [28]:

$$\nabla f(x_1, ..., x_n) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \cdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{2.29}$$

After the vehicle is trimmed to an operating point $(\bar{x}, \bar{u})$ as described in the previous subsection, its motion variables have their steady-state (equilibrium) values. A perturbed flight is defined relative to that steady state, where each variable is now re-defined as a sum of its steady state and perturbed values [29].

For a state x, input u and output y, the perturbed variables relative to the equilibrium are defined as:

$$\begin{aligned} \delta x(t) &= x(t) - \bar{x} \\ \delta u(t) &= u(t) - \bar{u} \\ \delta y(t) &= y(t) - \bar{y} \end{aligned} \tag{2.30}$$

The non-linear system from Equation 2.26 can now be rewritten.

$$\begin{aligned} \dot{\delta x}(t) + \dot{\bar{x}} &= f(\bar{x} + \delta x(t), \bar{u} + \delta u(t)) \\ \delta y(t) + \bar{y} &= g(\bar{x} + \delta x(t), \bar{u} + \delta u(t)) \end{aligned} \tag{2.31}$$

Using the Taylor series expansion on the right-hand side and neglecting all terms of order higher than 1, the expression becomes:

$$\begin{aligned} \dot{\delta x}(t) = \dot{x}(t) - \dot{\bar{x}} &\approx f(\bar{x}, \bar{u}) + \left.\frac{\partial f}{\partial x}\right|_{\bar{x},\bar{u}} \delta x(t) + \left.\frac{\partial f}{\partial u}\right|_{\bar{x},\bar{u}} \delta u(t) \\ \delta y(t) = y(t) - \bar{y} &\approx g(\bar{x}, \bar{u}) + \left.\frac{\partial g}{\partial x}\right|_{\bar{x},\bar{u}} \delta x(t) + \left.\frac{\partial g}{\partial u}\right|_{\bar{x},\bar{u}} \delta u(t) - g(\bar{x}, \bar{u}) \end{aligned} \tag{2.32}$$

however, $f(\bar{x}, \bar{u}) = 0$, as was stated in Equation 2.27. Thus, the expression is further reduced to just:

$$\begin{aligned} \dot{\delta x}(t) &\approx \left.\frac{\partial f}{\partial x}\right|_{\bar{x},\bar{u}} \delta x(t) + \left.\frac{\partial f}{\partial u}\right|_{\bar{x},\bar{u}} \delta u(t) \\ \delta y(t) &\approx \left.\frac{\partial g}{\partial x}\right|_{\bar{x},\bar{u}} \delta x(t) + \left.\frac{\partial g}{\partial u}\right|_{\bar{x},\bar{u}} \delta u(t) \end{aligned} \tag{2.33}$$

As long as the perturbation variables δ are small, the linear approximation holds. For a system with n states, m inputs, and l outputs the partial derivatives of f and g become matrices with the following definitions:

$$\begin{aligned} A &:= \left.\frac{\partial f}{\partial x}\right|_{\bar{x},\bar{u}} \in \mathbf{R}^{n\times n}, \quad B := \left.\frac{\partial f}{\partial u}\right|_{\bar{x},\bar{u}} \in \mathbf{R}^{n\times m} \\ C &:= \left.\frac{\partial g}{\partial x}\right|_{\bar{x},\bar{u}} \in \mathbf{R}^{l\times n}, \quad D := \left.\frac{\partial g}{\partial u}\right|_{\bar{x},\bar{u}} \in \mathbf{R}^{l\times m} \end{aligned} \tag{2.34}$$

where each matrix is expanded in a similar fashion to Equation 2.29. The linear system becomes the Jacobian Linearization of the original non-linear system.

$$\begin{aligned} \dot{\delta x}(t) &\approx A\delta x(t) + B\delta u(t) \\ \delta y(t) &\approx C\delta x(t) + D\delta u(t) \end{aligned} \tag{2.35}$$

Linearization in Simulink

Since Simulink is the main tool to be utilised, it is important to understand how it linearizes the systems, as there are some features unique to it [30].

- **Trimming.** Simulink allows for direct control over variables that have to be fixed to a certain value, or that are allowed to vary, and then calculates the necessary inputs to trim the system.
- **Separate linearization.** Instead of linearizing the entire system, Simulink rather linearizes each block separately, and then combines them together into a linear system
- **Numerical approach to custom blocks.** When a standard pre-defined block is used, there is also a built-in Jacobian for that block. However, for a custom block with hand-written code, Simulink slightly perturbs the input and measures the output to determine the linear slope, essentially using the central-difference numerical approximation of the Jacobian. That perturbation step needs to be noted, as too large deviation can result in averaging of the function, while too small deviation can be problematic for high-frequency functions.

2.3. Robust control

2.3.1. Robust control history and purpose

To understand the meaning of robust control, it is best to look into the historical events of control theory. Up until 1950's, the control theory was more analytical, with methods like Bode-Nyquist plots, root-locus and frequency response analysis. The paradigm shifted in the 1960's and early 1970's with the introduction of state-space models. The linear models opened up new potential for design and analysis methods, as well as utilisation of the emerging computers. Control design became more and more mathematical, with optimal control methods dominating in the field [31].

The so called "modern" methods of Linear Quadratic Regulator (LQR) and Linear Quadratic Gaussian (LQG) showed huge potential as they were essentially mathematical optimisation problems which converged to the best solution for required performance. However, a gap started to emerge between theoretical and practical work. The classical control methods were seen as outdated, and the models were assumed to be sufficiently accurate to achieve desired system performance and stability, without any explicit robustness concerns. Even the books on control theory of that time barely touched on the phase margin topics, stability margin topic was basically abandoned [31].

The engineers managed to achieve promising results with optimal control methods, until problems started to emerge. The attempts to apply theoretical optimization methods to multivariable systems

resulted in multiple failures. The LQG controller, which incorporates a Kalman filter for state estimation, turned out to be extremely sensitive to model perturbations, meaning uncertainties and inaccuracies can easily drive a system to instability. Some notable examples include simulations of the Trident submarine which suddenly surfaced under moderate sea conditions, and the F-8C aircraft in 1977 for which it was concluded that more pragmatic techniques should be implemented [31].

Meanwhile, Doyle's paper of 1978 [32] showed that LQG regulators, in fact, do not guarantee any robustness margins. Moreover, he stated that "like their more classical colleagues, modern LQG designers are obliged to test their margins for each specific design" [32]. The focus started to shift back to the classical methods to be integrated with the new ones, so that the robustness is addressed explicitly, especially for multivariable systems.

Following the mentioned events, the H_∞ control was born in the 1980's, which explicitly aimed at robust performance and stability under uncertainties and disturbances in the system. The control problem itself was introduced in 1981, and was solved for a simple single-input-single-output (SISO) system in the same year [31]. Since then, the H_∞ robust control framework was undergoing huge developments, including μ-synthesis and loop-shaping techniques.

It is of paramount importance to address uncertainties in the system, not only due to modelling inaccuracies, but also because the system parameters can change with time and exploitation. When treated improperly or neglected, the system can become unstable, and unstable systems can result in serious consequences. A light example is that of SAAB JAS 39 Gripen fighter jet prototype crash due to the control system not managing the pilot-induced oscillations [33]. A more tragic example of unstable system consequences was the infamous Chernobyl nuclear power plant catastrophe in 1986. It is therefore necessary to address robustness of the system explicitly to keep it stable and avoid disasters. The H_∞ control is precisely about that goal, and finds itself inseparable from the control industry today.

2.3.2. Robust control theory

Robust control theory was developed specifically to target uncertainties in the system model. Now that the system model is established, the next step is the robust controller synthesis.

Singular values and the H_∞ norm

Singular values form the backbone for robustness analysis and the H_∞ framework. The problem starts with defining some methods of how to measure a size of a system. For SISO systems, the Bode diagrams provide a good estimate: The output is amplified for frequencies where Bode is larger than 1, and suppressed where it is less than 1. However, for MIMO systems the size depends also on the input and output direction relative to the system, as well as on frequency. The singular values are the perfect tool in this case. For any $l \times m$ matrix, call it G, the individual singular values σ_i can be calculated from the eigenvalues λ_i, according to [34]:

$$\sigma_i(G) = \sqrt{\lambda(G^H G)} \tag{2.36}$$

where G^H is the complex conjugate transpose of G. Furthermore, the matrix G can be conveniently written in the singular value decomposition form, given by:

$$G = U\Sigma V^H \tag{2.37}$$

where Σ is an $l \times m$ diagonal matrix with singular values σ_i of G placed in descending order, U is $l \times l$ unitary matrix with its columns as output singular vectors, and V is $m \times m$ unitary matrix whose columns are input singular vectors. The output singular vectors provide the directional axes for the system, while input vectors are mapped onto them with gains as respective singular values. As a result, the minimum and maximum singular values ($\underline{\sigma}$ and $\bar{\sigma}$) provide lower and upper bounds on the system gain, and can be plotted separately on the Bode diagram.

The H_∞ norm of a stable multivariable LTI system is defined as the peak value over frequency of the largest singular value of the frequency response [34]:

$$\|G(s)\|_\infty = \max_\omega \bar{\sigma}(G(j\omega)) \tag{2.38}$$

This is an induced norm, meaning that it provides a measure of the largest possible steady-state gain between input and output [34].

The H_∞ control problem

The general H_∞ control problem can be formulated as follows. Given are the rational transfer matrix $P(s)$, and the controller space $\mathcal{K}$ composed of rational transfer matrices $K(s)$ in the feedback loop, as illustrated in Figure 2.12, with a general state-space representation:

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix} \tag{2.39}$$

The objective is to compute the optimal solution $K*$ in the following optimization algorithm [15]:

$$\begin{aligned} &\text{minimize} && \|T_{w\to z}(P,K)\|_\infty \\ &\text{subject to} && K \text{ stabilizes } P \text{ internally} \\ & && K \in \mathcal{K} \end{aligned} \tag{2.40}$$

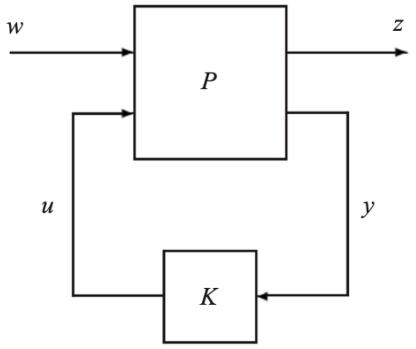


Figure 2.12: General structure for the H_∞ control problem [15]

In other words, the optimal control problem is minimizing the H_∞ norm of the closed-loop performance channel $T_{w\to z}(P,K)$ over all stabilising controllers [15]. In practice, it is computationally simpler to design a sub-optimal controller which approaches the optimal solution [20]. Let γ_{min} be the (optimal) minimum value of $\|T_{w\to z}(P,K)\|_\infty$, and the sub-optimal solution be $\gamma > \gamma_{min}$. The sub-optimal H_∞ control problem can then be stated as:

$$\|T_{w\to z}(P,K)\|_\infty < \gamma \tag{2.41}$$

where the sub-optimal solution is approached via iterative algorithms.

2.3.3. Robust control system design

Design specifications

Before some H_∞ controller synthesis methods are presented, it is paramount to discuss the design specifications which they target. The general requirements for a linear control system are be formulated below [13, 34]:

1. Nominal stability - the controller must provide stability for the nominal linearized model.
2. Nominal performance - the controller must provide adequate attenuation of disturbance and noise signals, as well as adequate tracking and decoupling of reference commands, for the nominal linearized model.
3. Robust stability - the controller must provide stability to unstructured uncertainty for all linear models within some structured uncertainty set.

4. Robust performance - the controller must provide adequate attenuation of disturbance and noise signals, as well as adequate tracking and decoupling of reference commands, for all linear models within some structured uncertainty set.

These requirements can be further defined in terms of singular values of transfer functions within a system. For that purpose, consider a typical closed-loop system with plant G, controller K, reference signal r, controller input signal e, controller output signal u, plant input signal u_p, plant output y_p, system output y, input and output disturbances d_I and d_O, and measurement noise n, shown in Figure 2.13.

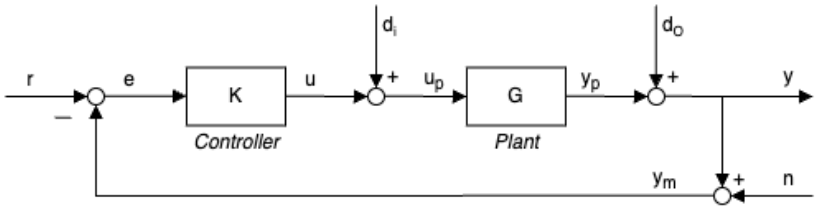


Figure 2.13: Typical closed-loop system with disturbances and noise

When relations between the elements are worked out, the system is represented by the transfer functions [34]:

$$\begin{aligned} e &= S_O r - S_O G d_I - S_O d_O + T_O n \\ y &= T_O r + S_O G d_I + S_O d_O - T_O n \\ u &= K S_O r - T_I d_I - K S_O d_O - K S_O n \\ u_p &= K S_O r - K S_O d_O + S_I d_O - K S_O n \end{aligned} \tag{2.42}$$

With some key definitions [13]:

$$\text{Input and output loop transfer matrices} \quad L_I = KG, \quad L_O = GK \tag{2.43}$$

$$\text{Input sensitivity function} \quad S_I = (I + KG)^{-1} \tag{2.44}$$

$$\text{Output sensitivity function} \quad S_O = (I + GK)^{-1} \tag{2.45}$$

$$\text{Input complementary sensitivity function} \quad T_I = KG(I + KG)^{-1} = L_I(I + L_I)^{-1} \tag{2.46}$$

$$\text{Output complementary sensitivity function} \quad T_O = GK(I + GK)^{-1} = L_O(I + L_O)^{-1} \tag{2.47}$$

With that in mind, some design objectives using the singular values can be formulated for the provided system [13, 34]:

1. For output disturbance rejection at the plant output, make $\bar{\sigma}(S_O)$ small
2. For input disturbance rejection at the plant input, make $\bar{\sigma}(S_I)$ small
3. For input disturbance rejection at plant output, make $\bar{\sigma}(S_O G)$ small
4. For noise attenuation at the plant output, make $\bar{\sigma}(T_O)$ small
5. For reference tracking, make $\bar{\sigma}(T_O)$ and $\underline{\sigma}(T_O) \approx 1$. This condition is equivalent to specification 1, because when $S_O \to 0$ then $T_O \to I$
6. For control energy reduction / output disturbance rejection / measurement noise attenuation at the plant *input*, make $\bar{\sigma}(K S_O)$ small
7. For robust stability to input multiplicative uncertainty, make $\bar{\sigma}(T_I)$ small
8. For robust stability to output multiplicative uncertainty, make $\bar{\sigma}(T_O)$ small
9. For robust stability to input additive uncertainty, make $\bar{\sigma}(K S_O)$ small

However, $S_I + T_I = I$ and $S_O + T_O = I$, so the objective specifications above can not be met simultaneously, presenting fundamental trade-offs in the system design. Fortunately, they have different significance over the frequency range. For example, disturbance rejection is favoured at low frequencies, while noise attenuation - at high frequencies. These trade-offs are directly tackled by the mixed sensitivity H_∞ method.

Classical mixed-Sensitivity approach

This method focuses on shaping the sensitivity function S_O together with one ore more closed-loop transfer functions, usually KS_O or complementary sensitivity T_O. The goal is to establish a cost function with weighting filters, which becomes an H_∞-norm minimization problem.

As an example, consider a system with disturbance at the plant output and insignificant measurement noise. The objective is to provide disturbance rejection, thus minimizing S_O. Additionally, KS_O needs to be limited in size and bandwidth in order to avoid actuator saturation and energy use. With the weights assigned to them, the cost function for the minimization problem is formulated [13]:

$$\left\| \begin{matrix} W_1 S_O \\ W_2 K S_O \end{matrix} \right\|_\infty \tag{2.48}$$

The choice of the weighting filters depends on the specific conditions. In this case, disturbance is normally present at low frequencies, so W_1 is a low-pass filter with bandwidth similar to the disturbance. Actuators should be less responsive to high frequencies, so a natural choice for W_2 would be a high-pass filter. Therefore, the minimization of the infinity norm of S_O is prioritised at low frequencies (disturbance rejection), while minimization of the infinity norm of KS_O is the priority at high frequencies (control energy reduction) [13].

This problem can be fitted to the general H_∞ problem structure from Figure 2.12. The schematic of the system is shown in Figure 2.14.

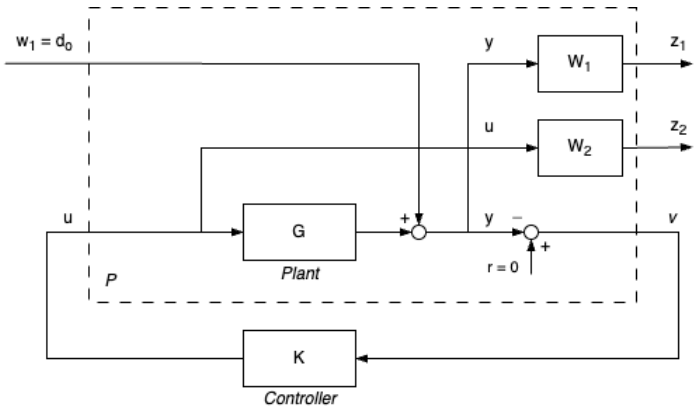


Figure 2.14: Standard form of the S/KS optimization

Here, the disturbance $d_0 = w_1$ and the error signal $z = [z_1^T \; z_2^T]^T$ where $z_1 = W_1 y$ and $z_2 = W_2 u$, so that the generalised plant from Equation 2.39 is [13]:

$$\left[\begin{array}{c} z_1 \\ z_2 \\ \hline v \end{array}\right] = \left[\begin{array}{cc} W_1 & W_1 G \\ 0 & W_2 \\ \hline -I & -G \end{array}\right] \begin{bmatrix} w \\ u \end{bmatrix} \tag{2.49}$$

and the closed-loop channel from Equation 2.40:

$$T(P,K) = \begin{bmatrix} W_1 S_O \\ -W_2 K S_O \end{bmatrix} \tag{2.50}$$

which is then put into the minimization algorithm. In a similar manner, the system can be optimized for reference tracking and noise attenuation.

Mixed-sensitivity H_∞ method is a powerful tool when there is a simple trade-off between two or three ($S/KS/T$) complementary functions. However, this classical type of mixed-sensitivity H_∞-synthesis has a tendency to result in controllers with zeros that cancel out the stable poles of the plant. The problem lies in the fact that a two-block or a three-block problem can only consider a single disturbance either at the plant input or output. The optimization then disregards potential disturbances at the other

points in the loop which can excite the plant poles, and therefore attempts to cancel the feedback path from the considered disturbance to the plant input with controller zeros. However, in the presence of system uncertainty the poles may not be canceled exactly, resulting in poor robust performance [34]. This problem can be surpassed by explicit consideration of disturbances at the plant input and output. Considering again Figure 2.14 with additional disturbance at the plant input $d_i = W_3 w_2$, as shown in Figure 2.15, it becomes a four-block problem of the form below:

$$T(P,K) = \begin{bmatrix} W_1 S_O & W_1 S_O G \\ -W_2 K S_O & -W_2 T_I \end{bmatrix} \tag{2.51}$$

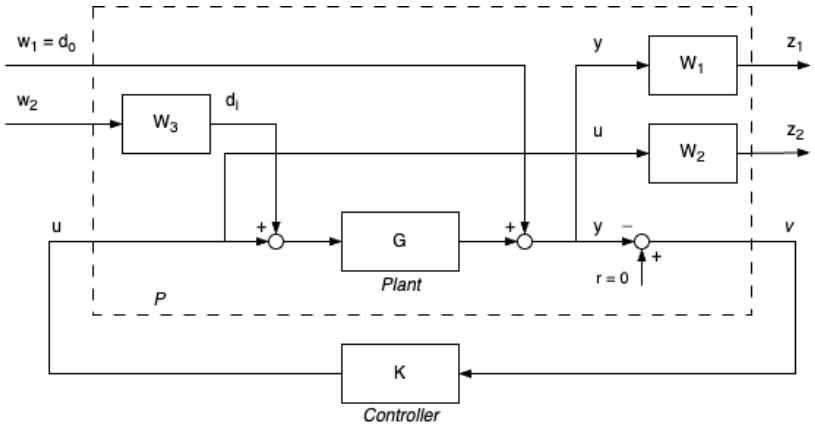


Figure 2.15: Four-block mixed-sensitivity problem with input and output disturbances

This can be further enhanced by applying additional weights on the disturbances for more precise control of the constraints. However, the process of choosing the weights becomes more complicated when more than two closed-loop functions are considered under a single performance metric γ, and therefore might lack flexibility [13, 34].

The original H_∞ computation methods of the 1980's and 1990's produced unstructured controllers - a major drawback for practical applications. That was the case until 2006 when P. Apkarian and D. Noll solved the H_∞ problem again with non-smooth optimization algorithms [35]. The new approach completely redefined the H_∞ framework, allowing controller synthesis with predefined fixed structure and multiple performance objectives. This methodology, called multidisk, now permits flight control system design for multiple plants and performance channels $T_{w \to z}$ simultaneously, thus significantly extending the classical mixed-sensitivity approach capabilities, along with explicit parametric uncertainty incorporation and mixed H_2/H_∞ constraints [15]. A more detailed overview of the multidisk approach and its practical difference from the older methods is discussed in section IV of chapter 3.

2.4. Uncertainty and robustness

Now that the system model is available, it is time to consider its deviation from the actual system, because it is impossible simulate the real dynamics perfectly. Uncertainty is referred to this divergence from the physical system, and modelling the uncertainties is the key concept of robustness [13]. The ultimate goal of designing a robust control system is to make sure that the plant successfully performs under worst-case deviations from the model the controller were designed for. Therefore, modelling uncertainty serves as a basis for the H_∞ control framework and robustness analysis. Additionally, with the complex mission profile of hypersonic vehicles one can expect that the system may be altered after the cruise phase, and thus should anticipate the uncertainties in the plant.

The main sources of uncertainty include the following [13]:

- Some parameters of the system can not be known in their exact values, and those that can may be identified inaccurately.
- Parameters can vary from non-linearities in the modelled linear system or due to changes in operating conditions.

- Errors in the measurements from imperfect sensors introduce uncertainty to control inputs.
- A model loses its fidelity at high frequencies. At some point with increasing frequency, the model structure and order will be unknown.
- Sometimes it is preferred to work with a reduced order model and intentionally treat left-out system dynamics as uncertainty, even when the full model is available.
- The implemented controller can deviate from the originally designed one, so it may be beneficial to include uncertainty due to implementation inaccuracies.

2.4.1. Disk margins

Essentially, the uncertainties transform into changes in gain and phase of the system. Classic gain and phase margins (GM and PM) are simple, but cannot provide a comprehensive indication of the system robustness if the gain and phase change simultaneously. Disk margins (DM), on the other hand, are defined using complex perturbations of the plant that account for simultaneous gain and phase variations across all frequencies [36]. The disk $D(\alpha, \sigma)$ contains a set of perturbations, where the maximum size α is the maximum allowable perturbation for which the closed-loop remains stable, given a disk skew σ. Note that α and σ are not related to the angle of attack and singular value here, and will always be explicitly defined in these cases. The disk margin is then computed with [36]:

$$\alpha_{max} = \frac{1}{\|S + \frac{\sigma-1}{2}\|_\infty} \tag{2.52}$$

where S is the familiar sensitivity function. When $\sigma = 1$, it becomes the S-based margin, effectively meaning that the DM is the equal to the inverse of H_∞ norm of sensitivity function. The disk is then located exactly at the critical point -1 in the Nyquist plane, and the DM equals the distance from the open loop to the critical point [36]. This further signifies that bounding the H_∞ norm of S has a directly positive effect on the size of DM. A case where $\sigma = -1$ is referred to the T-based disk margin. However, the most useful case for analysis is the $S-T$ symmetrical margin of $\sigma = 0$, where the disk is not skewed, and the gain and phase variations are considered equal in both directions on a log scale [34]. That way, a typical requirement of $\pm$6 dB GM for aerospace applications can be verified.

The disk margins can be computed for both SISO and MIMO cases with appropriate (complementary) sensitivity functions incorporation. The case of plotting the $S - T$ disk margin for MIMO case is a little special, though. The expression itself yields:

$$S - T = (I - L)(I + L)^{-1} \tag{2.53}$$

where I is the identity matrix and L is the open loop at the plant output. When the uncertainty varies inside a complex disk $re^{-j\theta}$, the system is stable. θ in this case is the angular space between 0 and 2π, and r is the radius at each angle. The uncertainty is complex, and thus has the form of $(1 - re^{-j\theta})/(1 + re^{-j\theta})$, meaning the extreme gain variations occur at $\theta = [0,\ \pi]$ and phase variations at $\theta = \pm\frac{\pi}{2}$. The guaranteed MIMO gain margin is then [34]:

$$GM_{ST} = \left[\frac{1 - r_{\text{min}}}{1 + r_{\text{min}}},\ \frac{1 + r_{\text{min}}}{1 - r_{\text{min}}}\right] \tag{2.54}$$

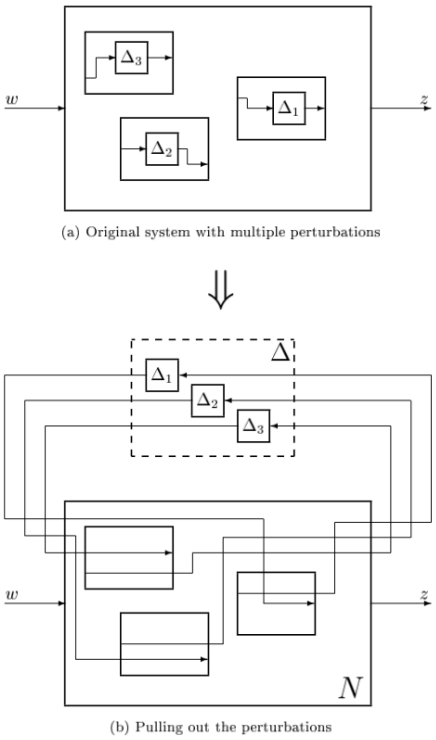
where r_{min} is equal to half of the disk margin α value. The guaranteed MIMO phase margin is:

$$PM_{ST} = \left[-2\tan^{-1}(r_{\text{min}}),\ 2\tan^{-1}(r_{\text{min}})\right] \tag{2.55}$$

These margins only include simultaneous gain or phase changes across loops, but to assess simultaneous gain and phase variations across all loops, the $(1 - r_{\text{min}}e^{-j\theta})/(1 + r_{\text{min}}e^{-j\theta})$ can be plotted in the Nichols plane [34]. Nichols charts are very informative, as the open loops are plotted on a gain vs phase axes for all frequencies.

2.4.2. General plant model with uncertainty

When uncertainty is explicitly modelled, the entire system can be rearranged into the $N - \Delta$ structure, as depicted in Figure 2.16. The system has inputs w and outputs z, while all perturbations are collected in the Δ block.



(a) Original system with multiple perturbations

(b) Pulling out the perturbations

Figure 2.16: $N - \Delta$ representation of an uncertain plant [13]

With the use of upper linear fractional transformation, the uncertain closed-loop transfer function F from w to z is given by [13]:

$$F = F_u(N, \Delta) = N_{22} + N_{21}\Delta(I - N_{11}\Delta)^{-1}N_{12} \tag{2.56}$$

Basically, the system can now be seen as a collection uncertain plants. The aim of properly modelling uncertainty is to obtain the least conservative global uncertain plant, for which a robust controller can be later synthesized.

2.4.3. Unstructured uncertainty

In this case the origins of uncertainty can not be traced to specific points in the system [34]. Basically, unstructured uncertainty consists of all cases of neglected and unmodelled dynamics, either due to deliberate negligence or lack of knowledge [13].

The uncertainty is now represented by a complex perturbation of the plant in the complex plane. Let G_0 be the nominal plant model without uncertainty, and the uncertain plant to be G_P. Then the three forms of unstructured uncertainty and their inverses are given below, and their corresponding schematic outlines shown in Figure 2.17.

- (a) Additive uncertainty: $G_P = G_0 + w_A\Delta_A$
- (b) Multiplicative input uncertainty: $G_P = G_0(I + w_I\Delta_I)$
- (c) Multiplicative output uncertainty: $G_P = (I + w_O\Delta_O)G_0$
- (d) Inverse additive uncertainty: $G_P = G_0(I - w_{iA}\Delta_{iA}G_0)^{-1}$
- (e) Inverse multiplicative input uncertainty: $G_P = G_0(I - w_{iI}\Delta_{iI})^{-1}$
- (f) Inverse multiplicative output uncertainty: $G_P = (I - w_{iO}\Delta_{iO})^{-1}G_0$

The w represent the weights and Δ is the normalized perturbation matrix. The uncertainties are assumed to be unknown, but stable and bounded, so that the H_∞ norm is bounded by some constant. When normalized, the infinity norm of Δ is bounded by 1.

For SISO, the input and output uncertainties are identical, but for MIMO they should be considered separately. The output uncertainty is more preferable to be considered first, as it is normally less restrictive on control performance than input uncertainty [37].

The nominal model can be chosen out of three options: use a simplified model (delay-free or low order), use the mean parameter values $\bar{G}$, or use the central plant in the Nyquist plot.

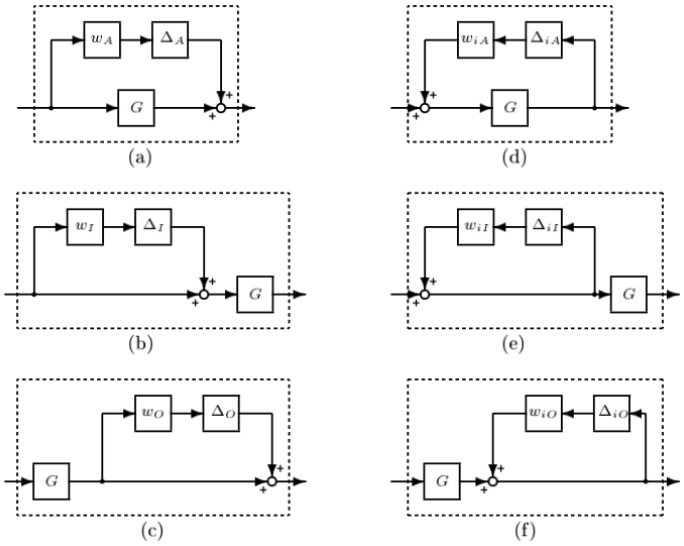


Figure 2.17: Unstructured uncertainty forms [13]

It is most efficient to model unstructured uncertainty in the frequency domain, where it is easier to find the global bounded perturbed plant. Bounding uncertainty means obtaining the weights for whichever form of uncertainty is considered. The process involves calculation of the smallest radius $l_M(\omega)$ of a disk in the complex plane for all possible plants within finite number of $\omega \leq \omega_{max}$; G_P can take any complex value inside the disk. The points $l_M(\omega)$ are then approximated with a filter $w'(j\omega) \geq l_M(\omega)$ for all $\omega \leq \omega_{max}$. Applying model order reduction results in the weighted transfer function, which can then be seen as the global uncertain plant. This process of bounding unstructured uncertainty is shown in Figure 2.18.

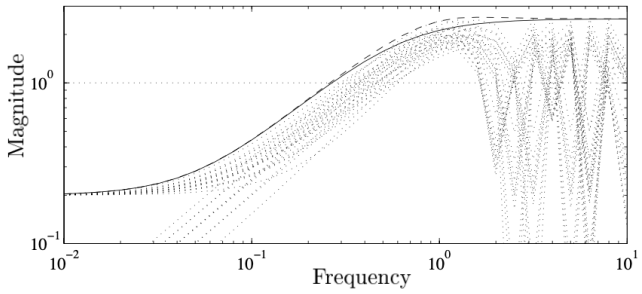


Figure 2.18: Example of a collection of perturbed plant variations (dotted) and the upper bound on uncertainty as first order (solid) and third order (dashed) weight w_I [13]

From this point, the Small Gain Theorem can be defined. For robust stability analysis, consider the system arranged into the $M - \Delta$ structure presented in Figure 2.19.

The theorem states that for a stable *open-loop* transfer function matrix $L(s) = M\Delta(s)$ with no hidden unstable modes, the *closed-loop* system is stable if [34]:

$$\|L(j\omega)\| < 1 \quad \forall\omega \tag{2.57}$$

In other words, if the system gain is less than 1, then it is stable. The theorem in this form is conservative, as it provides sufficient condition for stability, but not a necessary one, because it is defined with any

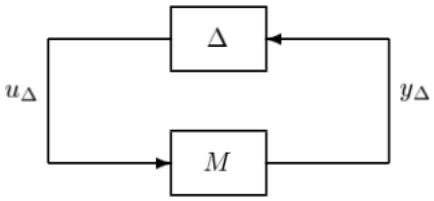


Figure 2.19: M-Δ structure for robustness analysis [13]

induced norm. However, using the H_∞ norm with the Small Gain Theorem is non-conservative, as it then becomes both *sufficient and necessary condition* [34].

In the case of defined $\|\Delta\|_\infty \leq 1$, the Small Gain Theorem for the $M - \Delta$ system can be written in a simpler format: the $M - \Delta$ system is stable if

$$\|M\|_\infty < 1 \tag{2.58}$$

The matrix M can be found by isolating the perturbations and deriving the transfer matrix from perturbation output to perturbation input. The matrix M then can be written as

$$M = W_1 M_0 W_2 \tag{2.59}$$

where, neglecting negative signs as they do not affect stability conditions, M_0 for each of the 6 uncertainty models becomes [13]:

- (a) Additive uncertainty: $M_0 = K(I + GK)^{-1} = KS_O$
- (b) Multiplicative input uncertainty: $M_0 = K(I + GK)^{-1}G = T_I$
- (c) Multiplicative output uncertainty: $M_0 = GK(I + GK)^{-1} = T_O$
- (d) Inverse additive uncertainty: $M_0 = (I + GK)^{-1}G = S_O G$
- (e) Inverse multiplicative input uncertainty: $M_0 = (I + KG)^{-1} = S_I$
- (f) Inverse multiplicative output uncertainty: $M_0 = (I + GK)^{-1} = S_O$

Then, substituting into Equation 2.58, the system is stable if [13]:

$$\|W_1 M_0 W_2\|_\infty < 1 \quad \forall\omega \tag{2.60}$$

2.4.4. Structured uncertainty

The parametric uncertainty assumes that each disputed parameter, call it α_P, is bounded within some region $[\alpha_{min}, \alpha_{max}]$. The uncertainty Δ is thus assumed to be real. Each disputed parameter can then be represented by

$$\alpha_P = \bar{\alpha}(1 + r_\alpha \Delta) \tag{2.61}$$

Where $\bar{\alpha}$ is the mean parameter value, $r_\alpha = (\alpha_{max} - \alpha_{min})/(\alpha_{max} + \alpha_{min})$, and Δ is a real scalar such that $|\Delta| \leq 1$.

The parametric uncertainty is more effectively used for MIMO systems, as it can represent the coupling between uncertain transfer functions elements. The parametric uncertainty is structured, so all of the individual uncertainties are grouped into a diagonal or block diagonal matrix structure $\Delta = diag(\delta_1, \delta_2, \dots)$.

3

Scientific Article

Robust Multi-Objective H_∞ Control of GHAME Hypersonic Vehicle in Subsonic Flight

E. Goz* and S. Theodoulis†
Delft University of Technology, Delft, 2628CD, Netherlands

This research is aimed at developing a comprehensive approach for robust hypersonic vehicle (HV) control utilizing modern H_∞ techniques. Initial focus is placed on subsonic flight condition to validate the framework and controller design in a relatively untreated field, for which the HV are not primarily optimized. A 6-degree-of-freedom non-linear model of the GHAME hypersonic vehicle was constructed in MATLAB and Simulink, incorporating tensor-based equations of motion and embedded parametric uncertainty in the aerodynamic coefficients. The linear short-period longitudinal dynamics were then extracted at multiple operating points. A controller of fixed structure was synthesized using multi-objective (multidisk) H_∞ mixed-sensitivity techniques with various performance and robustness requirements covering the pitch moment coefficient parametric uncertainty domain. Additionally, the design is extended to handle variations in Mach number, altitude, and fuel mass around the trim point using a multi-model approach. A single structured control system successfully stabilized, rejected input and output disturbances and provided reference tracking for the uncertain short-period models and met the minimum robustness margin requirements for the entire grid. It was then tested on the non-linear model and successfully performed the same tasks under parameter variations across the flight point grid.

I. Nomenclature

α	=	Angle of attack
β	=	Angle of sideslip
δ_a	=	Aileron deflection
δ_e	=	Elevator deflection
δ_r	=	Rudder deflection
δ_t	=	Throttle tab deflection
δ_{vl}	=	Left elevon deflection
δ_{vr}	=	Right elevon deflection
γ	=	Tuning performance metric
ρ	=	Density of air
σ	=	Singular value
a	=	Speed of sound
A_c	=	Engine cowl area
C_a	=	Capture area ratio
C_D	=	Coefficient of drag force
C_{D_1}	=	Modified partial drag coefficient
C_L	=	Lift coefficient
C_{L_1}	=	Modified partial lift coefficient
C_l	=	Roll moment coefficient
C_m	=	Pitch moment coefficient
C_{m_1}	=	Modified partial pitch moment coefficient
C_n	=	Yaw moment coefficient
C_X	=	Longitudinal force coefficient
C_Y	=	Side force coefficient
C_Z	=	Normal force coefficient
f_P	=	Thrust force
g_0	=	Gravitational constant
M	=	Mach number
n_z	=	Vertical load factor at IMU location
$\bar{q}$	=	Dynamic pressure

II. Introduction

Hypersonic vehicles (HV), capable of traveling at speeds and altitudes unreachable by conventional supersonic aircraft, represent a significant advancement in aerospace technology, and require a careful consideration of challenges associated with market, operational infrastructure, and, of course, engineering [1]. Their extraordinary speed

*MSc, student, Control & Simulation division, Faculty of Aerospace Engineering, P.O. Box 5058, 2600GB Delft, Netherlands; eggoz55@gmail.com.

†Associate Professor, Control & Simulation division, Faculty of Aerospace Engineering, P.O. Box 5058, 2600GB Delft, Netherlands; S.Theodoulis@tudelft.nl. Associate Fellow AIAA.

capabilities enable drastic reductions in travel time, presenting potential applications for high-priority point-to-point transport and emergency response scenarios where time is a critical factor. Furthermore, the high altitudes achieved by these vehicles not only facilitate reconnaissance missions above conventional flight levels, but also extend beyond the Earth's atmosphere. This capability positions hypersonic vehicles as promising candidates for reduced-cost space access, functioning as single-stage-to-orbit (SSTO) reusable vehicles [2, 3]. Such a development could revolutionize space utilization for the Low Earth Orbit operations. Although HV designs vary depending on mission profiles, such as rocket-propelled space launchers [1], in the context of this paper the term HV specifically refers to the horizontal take-off air-breathing hypersonic transport vehicle, which is able to sustain cruise speeds above Mach 5 within the Earth's atmosphere.

In order to reach these operating conditions, some unique design characteristics must be employed. In general, the HV tend to have long, slender bodies, where the propulsion system is tightly integrated into the airframe, with lightweight yet strong structure able to withstand aero-thermodynamic loads [1, 2]. The forward-extended fuselage is needed to create a series of shock-waves that compress the airflow fed into the engine and to create lift force from underneath, whereas the rear fuselage is shaped for expanding the exhaust gases externally, hence also contributing to the lift force. The propulsion system is typically mounted underneath, below the center of gravity (CG), producing a pith-up moment [2].

Among many engineering challenges, the HV operating conditions and design traits introduce significant problems related to their stability and control. The vibrations of the fuselage change the pressure distributions across the airframe, resulting in lift, drag, pitching moment, and intake airflow perturbations [2, 4]. Due to the airframe structure, the vibrations tend to be of relatively low frequencies posing elastic-body mode interactions, whereas the changes in the angle of attack during pitch-up maneuvers further alter the flow field at the inlet and can result in variations in thrust vector magnitude and direction [5], or even in engine flameout altogether [6]. In fact, the airframe-propulsion interactions are argued in [7] to possibly be the most complex of any vehicle. All of the aforementioned problems introduce uncertainties into the system, along with a lack of high-fidelity models of accurate aerodynamic data (at least among those available to the public). Therefore, the flight control system (FCS) designs for HV must account for the unmodelled nonlinearities, as well as consider specific operational requirements such as atmospheric turbulence rejection [7], angle of attack variation limitation [6, 8], etc.

The discussed aeroelastic effects have been the primary concern in almost all HV-related works referenced in this paper, so the modelling and control design approaches must be carefully considered. There are only a handful of HV aerodynamic models available to the general public for research. Among the widely spread ones are the Winged-Cone Configuration Hypersonic Vehicle [9] and the Generic Hypersonic Vehicle Model Example (GHAME) [10]. Both are rigid-body, 6 degrees-of-freedom (DoF), and developed in the 1990's. However, the former is developed completely via analytical computer modelling programs, whereas GHAME is a combination of analytical and real empirical data. There is another model developed later in 2005 [11, 12], which aims to capture the interactions between the airframe, propulsion system, and aerodynamics, although in longitudinal plane only. The model was derived from first principles, and incorporates structural bending into the equations of motion. It was then incorporated into Simulink® framework, for which the initial flight control system was designed with Linear Quadratic Regulator (LQR) [13, 14]. However, the uncertainties in the original model were not quantified [2], and the LQR does not guarantee any robustness margins at the plant output. The focus switched later to make the developed LQR controller robust to uncertainties using the servomechanism theory in [15], but the unmodelled dynamics were considered implicitly as parameter variations with changes of fuel mass. On the other hand, the H_∞ robust control framework allows to address the uncertainties directly, and guarantees robustness margins.

One of the early attempts of implementing H_∞ mixed-sensitivity method using basic S/T structure, described in [16], concluded that the real parameter uncertainties were too large to handle by this method and switched to H_∞ μ-synthesis method instead, which treats the worst-case uncertainty scenario. In fact, μ-synthesis was mostly used in the earlier designs of robust H_∞ controllers for HV. In [7] it was also concluded that H_∞ controllers suffer performance degradation with introduction of simultaneous uncertainties into the system, and called for μ-synthesis as a better performing control design methodology for this application. A major drawback of μ-synthesis is that it leads to controllers of very high order, which is computationally inefficient and complicates follow-up gain-scheduled designs. Additionally, it cannot address real parametric uncertainty directly and rather treats it as complex, presenting conservatism in the solution. Further work on HV control in [17–19], where the aeroelastic effects are primarily modelled as parametric uncertainty in C_m partial coefficients, concluded that the controller order reduction does not provide any robustness guarantees, whereas synthesizing a fixed-order controller still results in the orders of 5 to 9. Ref. [5] further outlines the aforementioned problems, stating the need for fixed-structure robust controllers of reduced conservatism, whereas

μ-synthesis treats real parameter uncertainty as complex. It also stated that it is crucial to attenuate the effects of atmospheric turbulence in hypersonic flight, for which an additional H_2 constraint is required. This results in a problem of mixed H_2/H_∞, which had no solution in the 1990's even in a full-order variant [5]. The public focus then shifted more towards non-linear and adaptive control methods for HV [20, 21].

Fortunately, all of the aforementioned problems associated with H_∞ controllers have been solved later by Apkarian in [22, 23] using non-smooth optimization algorithms to synthesize controllers of predefined fixed structure. This led to multi-objective and multi-model approaches (known as multidisk problem) [24], making it possible to synthesize a single controller for multiple performance specifications for both power and energy signals simultaneously, as well as for multiple plant models and directly specified parametric uncertainty. The latter is of special interest, as it presents a direct competitor to μ-synthesis. The multi-disk solution not only outperforms μ-synthesis because of controller structure specification ability, but also because it addresses real parametric uncertainty directly, reducing conservatism [25]. The new methods were incorporated into functions *hinfstruct* and *systune* added to the MATLAB® Robust Control Toolbox [26] in 2015. The main difference between the two is that the former uses a joint performance metric for all constraints like the classical mixed-sensitivity, whereas the latter fully utilizes multidisk capabilities and puts a performance measurement on each constraint individually, thereby making the controller less conservative and the constraining filter selection more convenient. Surprisingly, there is an evident lack of application of either of these methods to HV control. The few recent (after 2015) works on H_∞ control of HV still use the basic techniques under a single performance metric that lead to controllers of full order, such as in [27–30].

The field of hypersonic vehicle control is relatively new and under-researched, partly due to the scarcity of experimental data and the complexity of the operating conditions. At this point, most focus is aimed at modelling or accounting for the highly non-linear aeroelastic effects at hypersonic speeds. The models available to the public are of low fidelity for high Mach numbers, and are either completely mathematical, or approximations from methods like DATCOM [31] combined with flight data of other aircraft. Most of the implemented H_∞ designs are simply not in line with the modern tools, which are able to provide flexible frameworks and satisfy various control design requirements in time and frequency domains. There is still much to be done for the HV flight control system development. One example of a relatively untreated area is the hypersonic airframe behavior in subsonic flight. The HV are not primarily designed for it, with long slender airframes and typically limited control surfaces, making them potentially challenging to optimize for robust performance within that flight regime. An adaptive non-linear control method has been used for that purpose in [32], but the implementation of any other methods was not found during the literature research.

The AErospace dynamics and RObust CONtrol (AEROCON) research group, based at the Faculty of Aerospace Engineering in TU Delft, aims to start investigating robust hypersonic vehicle control using modern H_∞ robust control design methods and tools, for which this research establishes a foundation. The scope consists of constructing a 6-degree-of-freedom (DoF) non-linear model of the GHAME vehicle in MATLAB [33] and Simulink [26, 34] using tensor-based equations of motion with a flexible subsystem-separated block structure and embedded parametric uncertainty, setting up trimming and linearization programs that produce separated longitudinal and lateral linear models in the form of uncertain state-space systems at any given flight point and condition, synthesizing a flight control system of fixed structure at one of the flight points using multi-objective (multidisk) H_∞ mixed sensitivity techniques to explicitly design for robustness against structured parametric uncertainty. The design is further extended to be robust against trim condition variations in airspeed, altitude, and fuel mass using a multi-modelling approach. A model grid is formed around the nominal flight point (FP), and a single controller is synthesized for all parametrically uncertain models in the grid simultaneously, and then implemented and tested on the non-linear model. As a first initial step, the focus would be put on the short-period longitudinal dynamics at a subsonic flight condition. It seems as an appropriate starting point to test the framework and the controller design on an untreated field.

This paper is structured in three consecutive phases: modelling in section III, FCS design in section IV, and non-linear implementation and simulation in section V. The modelling phase consists of GHAME HV model description and analysis in subsection III.A, its non-linear tensor-based model description in subsection III.B, and trimming and linearization process outlined in subsection III.C. The FCS design includes a brief theoretical background in subsection IV.A, the FCS structure and synthesis description in subsection IV.B, and the results and analysis discussion in subsection IV.C. Finally, conclusion and recommendations for future work are presented in section VI.

III. Modelling

A. GHAME model

1. General description

The hypersonic vehicle model selected is the NASA's Generic Hypersonic Aerodynamic Model Example developed at the Dryden Flight Research Facility. The original publication of the model with full description can be found in [10]. GHAME was specifically developed to provide realistic data of hypersonic flight that is unclassified and available to the general public for performance calculations, trajectory optimization, simulation, and control design purposes. The main two arguments for selecting this vehicle model are the fact that its aerodynamic data are based on a combination of realistic data from existing aircraft and theoretical data of a double-delta wing configuration, and the fact that it was directly available in FORTRAN code written by P. Zipfel and extensively discussed in his book [35]. Additionally, it is 6-DoF and therefore allows future work with lateral motion.

GHAME is a horizontal take-off SSTO vehicle with a gross take-off weight of 300,000 pounds ($\approx$ 136,080 kg) and a dry weight of 120,000 pounds ($\approx$ 54,432 kg). Its mass and size properties were based on the XB-70, and the moments of inertia were estimated from simplified geometry. The airframe features a 70° delta-wing configuration with a single vertical tail rudder and two elevons. The vehicle is 243 feet ($\approx$ 74.07 m) long with a span of 80 feet ($\approx$ 24.38 m). Its configuration is built from simple geometrical shapes and excludes landing gear and any variable geometry apart from the aforementioned control surfaces. The wing area is 6,000 square feet ($\approx$ 557.42 m^2). The fuselage consists of a 10° half-angle cone ending in a cylinder with a 20-foot ($\approx$ 6.1 m) diameter, terminating in an integrated boattail/nozzle. The schematics are shown in Fig.1.

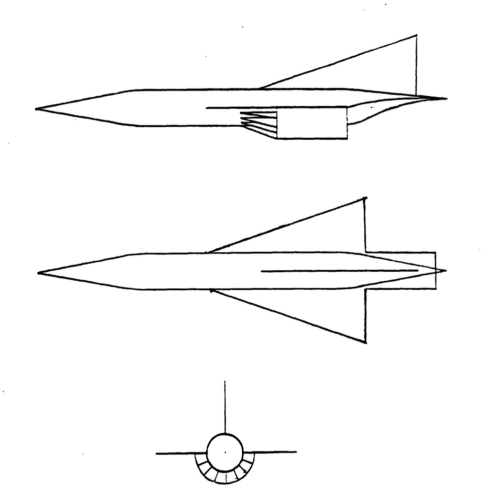


Fig. 1 Schematic of GHAME configuration [36]

2. Aerodynamic model

As has been mentioned, the aerodynamic model was comprised of analytical and empirical data. For lower Mach numbers, it is a combination of a swept double-delta wing using modified Newtonian Impact Flow method and an actual flight test data from the Space Shuttle. Above Mach 8, the data is exclusively based on the latter, but properly scaled. The force and moment coefficient equations have been linearized around a range of angle of attack α numbers for a range of Mach numbers at zero sideslip angle β. The force coefficient equations are presented in Eq. 1, and the moment coefficient equations in Eq. 2.

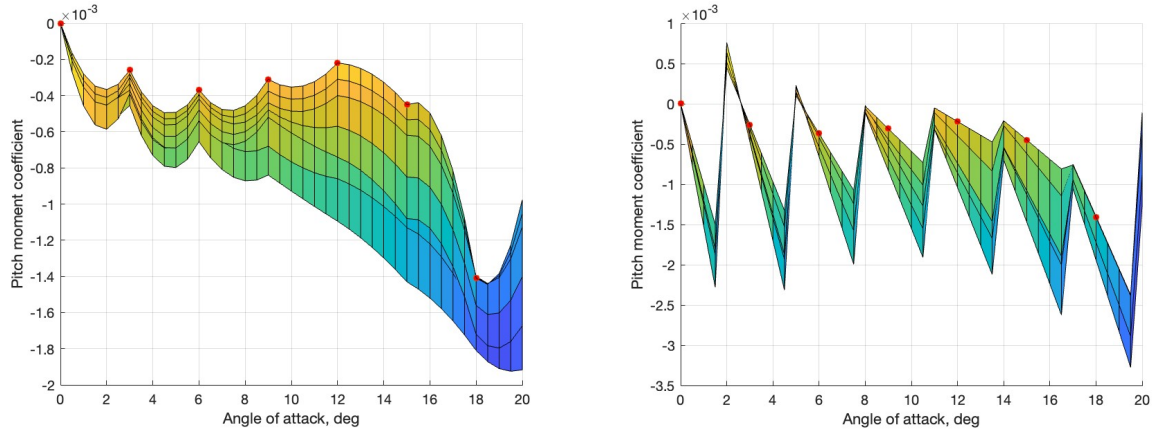
$$
\begin{aligned}
C_L &= C_{L_0}(M,\alpha) + C_{L_\alpha}(M,\alpha)\alpha + C_{L_{\delta_e}}(M,\alpha)\delta_e \\
C_D &= C_{D_0}(M,\alpha) + C_{D_\alpha}(M,\alpha)\alpha \\
C_Y &= C_{Y_\beta}(M,\alpha)\beta + C_{Y_{\delta_a}}(M,\alpha)\delta_a + C_{Y_{\delta_r}}(M,\alpha)\delta_r
\end{aligned} \tag{1}
$$

$$
\begin{aligned}
C_m &= C_{m_0}(M,\alpha) + C_{m_\alpha}(M,\alpha)\alpha + C_{m_{\delta_e}}(M,\alpha)\delta_e + C_{m_q}(M,\alpha)\frac{qc}{2V} \\
C_l &= C_{l_\beta}(M,\alpha)\beta + C_{l_{\delta_a}}(M,\alpha)\delta_a + C_{l_{\delta_r}}(M,\alpha)\delta_r + C_{l_p}(M,\alpha)\frac{pb}{2V} + C_{l_r}(M,\alpha)\frac{rb}{2V} \\
C_n &= C_{n_\beta}(M,\alpha)\beta + C_{n_{\delta_a}}(M,\alpha)\delta_a + C_{n_{\delta_r}}(M,\alpha)\delta_r + C_{n_p}(M,\alpha)\frac{pb}{2V} + C_{n_r}(M,\alpha)\frac{rb}{2V}
\end{aligned} \tag{2}
$$

All partial coefficients w.r.t. angles have units of 1/°, and all partial coefficients w.r.t. rotation rates have units of 1/rad. Lift coefficient C_L is defined positive upwards and perpendicular to velocity vector, drag coefficient C_D is positive aft and parallel to velocity vector. The rest of the coefficients have conventional positive directions. Each of the partial coefficients is given in the form of a look-up table of 9 α rows and 13 Mach number columns. Their indices, respectively, are shown in Eq. 3.

$$
\begin{aligned}
\alpha &= [-3 \quad 0 \quad 3 \quad 6 \quad 9 \quad 12 \quad 15 \quad 18 \quad 21] \\
M &= [0.4 \quad 0.6 \quad 0.8 \quad 0.9 \quad 0.95 \quad 1.05 \quad 1.2 \quad 1.5 \quad 2.0 \quad 3.0 \quad 6.0 \quad 12.0 \quad 24.0]
\end{aligned} \tag{3}
$$

Jumping a little ahead, a fundamental discrepancy in the aerodynamic data was discovered during the implementation of this aerodynamic model in the simulation. The solution process led to a certain alternative adaptation of the partial coefficients, so it makes more sense do discuss it in this subsection. The problem only concerns the first two terms in the computations of C_L, C_D, and especially C_m, essentially at subsonic Mach numbers. Since its impact on C_m is more severe than on the first two, the following discussion is explained on C_m.



(a) Pitch moment coefficient computed with interpolation for M = 0.4-1.0

(b) Pitch moment coefficient as a set of linear functions for M = 0.4-1.0

Fig. 2 Comparison of pitch moment coefficients using different methods

Naturally, the aerodynamic model with look-up tables means interpolating each of the partial coefficients for active (i.e., currently set) α and M values in the simulation. However, it was directly stated by the authors in [10] that the equations have been linearized around α values. When linearizing the entire coefficient equations w.r.t. α at equilibrium at some Mach number, it is the first two terms that contribute to it. Plotting $C_m = C_{m_0}(\alpha) + C_{m_\alpha}(\alpha)\alpha$, when the terms are interpolated at (and multiplied with) respective α first and then summed, for a set of Mach numbers between 0.4 and

1.0, leads to the graph shown in Fig. 2a. The red dots are placed at intersections with α indices along $M = 0.4$ for better comprehension.

This is clearly an unnatural pattern, and it contradicts the values of C_{m_α} look-up table, which has all values as negative, whereas here the slope becomes positive just before the next α index. As an even better indication of data self-contradiction, note that the red dots at the indices show a positive slope w.r.t. α, whereas the C_{m_α} table states that the slope is always negative. The phenomena of clearly positive slope between the indices only happens at Mach < 1. There is a possible explanation to this discrepancy. The coefficient equations have already been linearized in the aerodynamic tables, so $C_m = C_{m_0} + C_{m_\alpha}\alpha$ can be perceived as a basic linear function $y = ax + b$. However, the coefficients at some fixed Mach number $C_{m_0}(\alpha)$ and $C_{m_\alpha}(\alpha)$ should not be treated as continuous functions of alpha, but rather as a "scheduled" set of linear functions $y = ax + b$ at discreet α values. This may be confusing, so it is best illustrated in Fig. 2b for the same set of Mach numbers.

In this interpretation, the closest linear function at some active α is selected and then its deviation from the index is multiplied with C_{m_α}. Although a linear pitching moment coefficient slope may be a useful simplification around the indices, the "saw" shape is not suitable for full scale simulation, either. As a result, it was decided to compute the $C_m = C_{m_0} + C_{m_\alpha}\alpha$ at all α indices first, and then interpolate with straight lines between the indices for the entire Mach range, producing a new aerodynamic table. The same procedure was applied to C_L and C_D, since they have the same structure. The new partial coefficients are called C_{m_1}, C_{L_1}, C_{D_1}, and are defined as a summation of the first two corresponding aerodynamic tables at the indices (i, j), as shown in Eq. 4 and plotted in Fig. 3 (red dots again at M = 0.4 for comprehension). The new partial coefficients are then substituted into their respective formulas previously shown in Eq. 1 and Eq. 2, and they are the ones to be interpolated at the active Mach and α.

$$\begin{aligned} C_{m_1}(M,\alpha) &= C_{m_0}(M_i,\alpha_j) + C_{m_\alpha}(M_i,\alpha_j)\alpha_j \\ C_{L_1}(M,\alpha) &= C_{L_0}(M_i,\alpha_j) + C_{L_\alpha}(M_i,\alpha_j)\alpha_j \\ C_{D_1}(M,\alpha) &= C_{D_0}(M_i,\alpha_j) + C_{D_\alpha}(M_i,\alpha_j)\alpha_j \end{aligned} \tag{4}$$

3. Actuator model

The model features two elevons that function both as elevators and ailerons, located at the trailing edge of the wing. The conversions between them are shown in Eq. 5:

$$\delta_e = \frac{\delta_{vl} + \delta_{vr}}{2}, \qquad \delta_a = \frac{\delta_{vl} - \delta_{vr}}{2} \tag{5}$$

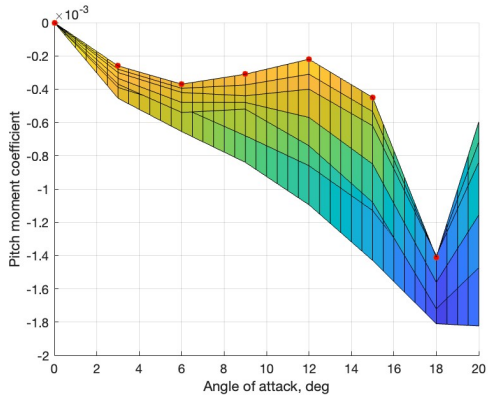
where δ_{vl} and δ_{vr} are, respectively, left and right elevon deflections, both defined positive downwards. There is also a single rudder, which is defined positive trailing edge left. There is no further information on the actuators in the original model, so it was assumed that limits on deflection angles and rates are intentionally left free to choose. Actuator modelling will be further discussed in subsection III.B.

4. Propulsion system model

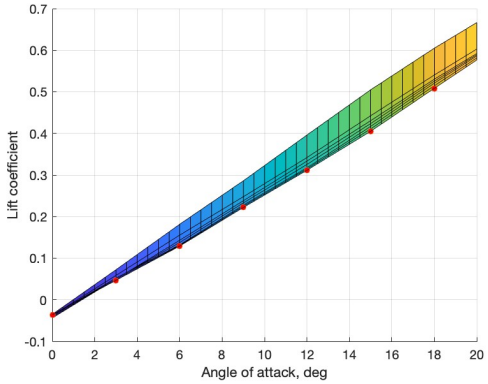
Being not the primary focus of research, the propulsion system was intentionally simplified by the authors to just fly the aerodynamic model [10]. The generic engine model approximates a combined-cycle propulsion system, where a turbojet operates between Mach 0-2, ramjet operates between Mach 2-6, and supersonic combustion ramjet operates above Mach 6. The engine cycles are assumed to change automatically. The inlet has variable geometry and its size is scheduled w.r.t. α and Mach. The thrust f_P is then computed with Eq. 6.

$$f_P = 0.029\delta_t I_{sp}(M,\delta_t) g_0 \rho M a C_a(M,\alpha) A_c, \qquad \delta_t = [0-2] \tag{6}$$

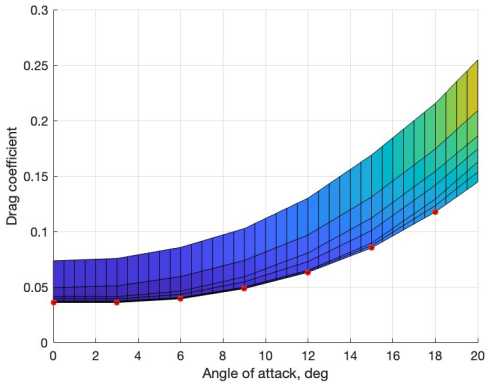
Where δ_t is throttle deflection tab, 0.029 comes from the fact that the pilot indirectly regulates the stoichiometric ratio with throttle tab, a is the speed of sound, and A_c is engine cowl area factor constant. Specific impulse I_{sp} is dependent on the mass flow, so it is provided in the form of a look-up table for indices of Mach (same as in Eq. 3) and δ_t. The effective capture area C_a of the engine inlet is determined by the properties of bow shock wave under the vehicle fore-body, which is dependent on the angle of attack and dynamic pressure [4], and is therefore given as a look-up table for indices as from Eq. 3. Evidently, the thrust force is computed instantaneously, i.e., the propulsion system model has no dynamics. The thrust force is assumed to be aligned with the center of gravity, and parallel to the airframe x-axis.



(a) Pitch moment coefficient C_{m_1}



(b) Lift coefficient C_{L_1}



(c) Drag coefficient C_{D_1}

Fig. 3 Modified lift C_{L_1}, drag C_{D_1}, and pitch moment C_{m_1} coefficients against angle of attack for Mach = 0.4-1.0

B. Non-linear model

The modelling of GHAME was largely based on techniques discussed by Zipfel in [35] and his model of GHAME written in FORTRAN code, which was built for purposes of full simulation with elliptical Earth equations of motion. The model of GHAME presented in this paper, however, has its own unique traits and assumptions, which are generally outlined below:

1) The model is 6-DoF, built in MATLAB and Simulink environment, with subsystem-separated blocks mindset.
2) The EoM assume flat, non-rotating Earth, which is an understandable simplification since the model is built for control law development purposes.
3) The mass of the vehicle m and the moment of inertia (MOI) are assumed constant in the EoM, for the same reason as the previous assumption.
4) The model is tensor-based, using a mix of relations for an aircraft and a hypersonic vehicle, both described in [35].
5) The center of gravity (CG) position is fixed, since its variation is not included explicitly in the original aerodynamic model.
6) The inertial measurement unit (IMU) is loosely approximated to be located 80 ft (24.384 m) ahead of CG along the center line, which is geometrically just before the conical nose in Fig. 1. It is merely an initial coarse reference value for load factor computation near pilot location, and can easily be changed later.
7) The measurement noise is not explicitly included in the initial model and will be added in the future.

The forces and moments computations directly utilize the aerodynamic coefficients derived in subsection III.A, and have the following expressions in body coordinates, respectively:

$$[f_{a,p}]^B = \begin{bmatrix} f_{a,p_1} \\ f_{a,p_2} \\ f_{a,p_3} \end{bmatrix} = \begin{bmatrix} \overline{q}SC_X + f_P \\ \overline{q}SC_Y \\ \overline{q}SC_Z \end{bmatrix} \tag{7}$$

$$[m_B]^B = \begin{bmatrix} m_{B_1} \\ m_{B_2} \\ m_{B_3} \end{bmatrix} = \begin{bmatrix} \overline{q}SbC_l \\ \overline{q}ScC_m \\ \overline{q}SbC_n \end{bmatrix} \tag{8}$$

Where f_P is propulsion force (thrust), and the body-frame coefficients C_X and C_Z are computed directly from C_L, C_D, and α using Eq. 9.

$$C_X = -C_D\cos(\alpha) + C_L\sin(\alpha), \qquad C_Z = -C_D\sin(\alpha) - C_L\cos(\alpha); \tag{9}$$

The equations of motion originate from the first principles of Newton's and Euler's laws. The translational equations are based on the Newton's second law in the inertial frame of reference I, but with the flat-Earth assumption the Earth frame of reference E becomes identical to I. The general equation of motion, expressed in body coordinate system B, is given in Eq. 10 in matrix form:

$$m\left[\frac{dv_B^E}{dt}\right]^B + m\left[\Omega^{BE}\right]^B\left[v_B^E\right]^B = \left[f_{a,p}\right]^B + m\,[T]^{BL}\,[g]^L \tag{10}$$

where v_B^E is the velocity vector of the vehicle's CG w.r.t inertial frame E, Ω^{BE} is the angular velocity vector of the vehicle wrt frame E, $[T]^{BL}$ is the direction cosine matrix (DCM) which is used to transform the gravity vector $[\bar{g}]^L = [0 \quad 0 \quad g]$ from local frame L to the frame B. In coordinate form the translational EoM become:

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

The rotational EoM are derived from Euler's law, and are worked out in a similar fashion as the translational ones, with the final general expression shown in Eq. 12:

$$\left[\frac{d\omega^{BE}}{dt}\right]^B = \left(\left[I_B^B\right]^B\right)^{-1}\left[-\left[\Omega^{BE}\right]^B\left(\left[I_B^B\right]^B\left[\omega^{BE}\right]^B\right) + [m_B]^B\right] \tag{12}$$

where ω^{BE} is the vehicle's angular velocity vector, $\left[I_B^B\right]^B$ is the vehicle's moment of inertia, and $[m_B]^B$ is from Eq. 8. Assuming symmetry in longitudinal plane, the MOI has $I_{13} = I_{31}$. Expanded matrices of Eq. 12 are shown in Eq. 13.

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix}^B = \left(\begin{bmatrix} I_{11} & 0 & I_{13} \\ 0 & I_{22} & 0 \\ I_{31} & 0 & I_{33} \end{bmatrix}^B\right)^{-1}\left(-\begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}^B \times \left(\begin{bmatrix} I_{11} & 0 & I_{13} \\ 0 & I_{22} & 0 \\ I_{31} & 0 & I_{33} \end{bmatrix}^B \begin{bmatrix} p \\ q \\ r \end{bmatrix}^B\right) + \begin{bmatrix} m_{B_1} \\ m_{B_2} \\ m_{B_3} \end{bmatrix}^B\right) \tag{13}$$

Next, the kinematic relations include familiar expressions for the angle of attack and the angle of sideslip:

$$\alpha = \arctan\left(\frac{w}{u}\right), \qquad \beta = \arcsin\left(\frac{v}{\sqrt{u^2+v^2+w^2}}\right) \tag{14}$$

The DCM introduced earlier in Eq. 10 is needed to compute the orientation of the vehicle w.r.t. the E reference frame. It is derived with Euler angles $[\phi \quad \theta \quad \psi]$ using the angular velocities in the body reference frame $[p \quad q \quad r]$. The Euler angles method was chosen to compute the DCM because of simplicity of implementation, as it directly uses Euler angles as states in the program calculated from rotational rates, and then the DCM is easily calculated afterwards. The

Euler angles are updated with with Eq. 15, and the DCM is then calculated with Eq. 16. There is a singularity at $\theta = \pm 90°$, which can be avoided with simple program operations. Additionally, the HV are not high maneuverability vehicles, so the singularity region is not their usual operating condition.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi \tan\theta & \cos\phi \tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi / \cos\theta & \cos\phi / \cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{15}$$

$$T^{BL} = \begin{bmatrix} \cos\psi \cos\theta & \sin\psi \cos\theta & -\sin\theta \\ \cos\psi \sin\theta \sin\phi - \sin\psi \cos\phi & \sin\psi \sin\theta \sin\phi + \cos\psi \cos\phi & \cos\theta \sin\phi \\ \cos\psi \sin\theta \cos\phi + \sin\psi \sin\phi & \sin\psi \sin\theta \cos\phi - \cos\psi \sin\phi & \cos\theta \cos\phi \end{bmatrix} \tag{16}$$

The speed of sound a and air density ρ are computed using International Standard Atmosphere. The load factor at the CG is calculated as $n = [a_B^E]^B / g_0$ (acceleration/gravity), in body reference frame. However, it is desirable to measure the load factor at the pilot location, where additional forces due to centrifugal and angular acceleration take place. Therefore, the acceleration at the IMU location is calculated using the Grubin's form of Newton's second law, where all the terms are divided by common mass [35]:

$$[a_S^E]^B = [a_B^E]^B + \left[\Omega^{BE}\right]^B \left[\Omega^{BE}\right]^B [s_{SB}]^B + \left[\dot{\Omega}^{BE}\right]^B [s_{SB}]^B \tag{17}$$

where $[a_S^E]^B$ is the acceleration vector at IMU location, $[a_B^E]^B = [f_{a,p}]^B / m$ is the acceleration vector at CG, and $[s_{SB}]^B$ is IMU location vector relative to CG. The load factor at the pilot's location is then $[n_S^E]^B = [a_S^E]^B / g_0$, and the vertical load factor is $n_z = -[n_S^E]^B(3)$ to make it positive upwards.

The non-linear model is initialized at a specific flight point using a separate file containing the initial conditions and airframe constants. The vehicle mass and MOI are specified for gross take-off weight and at burn-out (dry mass). The active mass and MOI at initial condition are regulated with fuel mass fraction, e.g., 0.5 of fuel tank corresponds to the median values of vehicle mass and MOI. The actuator model is taken directly from Zipfel's GHAME model in FORTRAN [35], and put into Simulink. Both elevons and the rudder have exactly the same characteristics, and are modelled as second order systems. The initialization parameters and their values are outlined in Table 7.

C. Uncertainty, Trimming and Linearization

The objective is to retrieve a linear state-space model at some operating point with uncertainty in aerodynamic parameters. For that purpose, 2 additional Simulink models were created, one for the airframe excluding the actuators, and one for the propulsion system. The process can be outlined as follows: first, the uncertainty is incorporated into both models. Then, the airframe is trimmed for equilibrium, so that the actuator deflections and thrust that keep the airframe in desired state are computed. The throttle tab deflection is then trimmed in the propulsion model to produce the required thrust. Both models are then linearized at their operating points, producing a full linear uncertain state-space (uss) system with all states together. The system is then decoupled into lateral, longitudinal, and short-period dynamics.

1. Uncertainty implementation

The created framework allows the parametric uncertainty to be explicitly defined for all partial aerodynamic coefficients independently, and can be easily extended to include additional parameters. The real parametric uncertainty can be implemented in MATLAB using $ureal$ function, which takes the mean value and the uncertainty range as inputs. The latter can be specified using ± absolute value range or percentage variation, while the former is trickier. Unlike some constant parameter such as actuator natural frequency, the partial aerodynamic coefficients vary along the flight path, which means their values are computed in the simulation directly, and calculating their value at trim condition by hand is cumbersome. It is significant for the framework to be flexible, such that parameter variation would be specified at the start, and then it would be carried over to the uncertain state-space system directly and automatically. Therefore, the nominal values of the partial aerodynamic coefficients computations are left unchanged, and there are generally two ways to incorporate their variations into Simulink, depending on whether uncertainty is specified as percentage or absolute value range.

When specifying it as percentage, a separate set of $ureal$ parameters is created during initialization for all partial aerodynamic coefficients, with a mean value of 1 and the desired percentage, as in the example of uncertainty variable

for C_{m_q} at the top of Eq. 18. They are then added as uncertain state-space uss blocks in Simulink along the path of nominal parameter values, so that they are multiplied with a mean of 1 and vary by the percentage.

$$\begin{aligned} uCm\_q\_percent &= \text{ureal}('\text{uCm\_q}', 1, '\text{ Percentage}', 20) \\ uCm\_q\_range &= \text{ureal}('\text{uCm\_q}', 0, '\text{ PlusMinus}', 0.0001) \end{aligned} \tag{18}$$

When the uncertainty is desired to be in absolute value range format, the initialization is done in a similar way, but with a different specification as at the bottom of Eq. 18, for example. However, in this case, the pitch rate signal q is split in two in the Simulink model. The first branch is multiplied with the nominal computed value of C_{m_q}, while the other branch is multiplied with its twin uncertainty variable uCm_q_range with a mean of 0 and absolute value ± range. They are then added together to produce an uncertain partial aerodynamic coefficient with a range specified for uCm_q, while the 0-mean does not intervene into calculations.

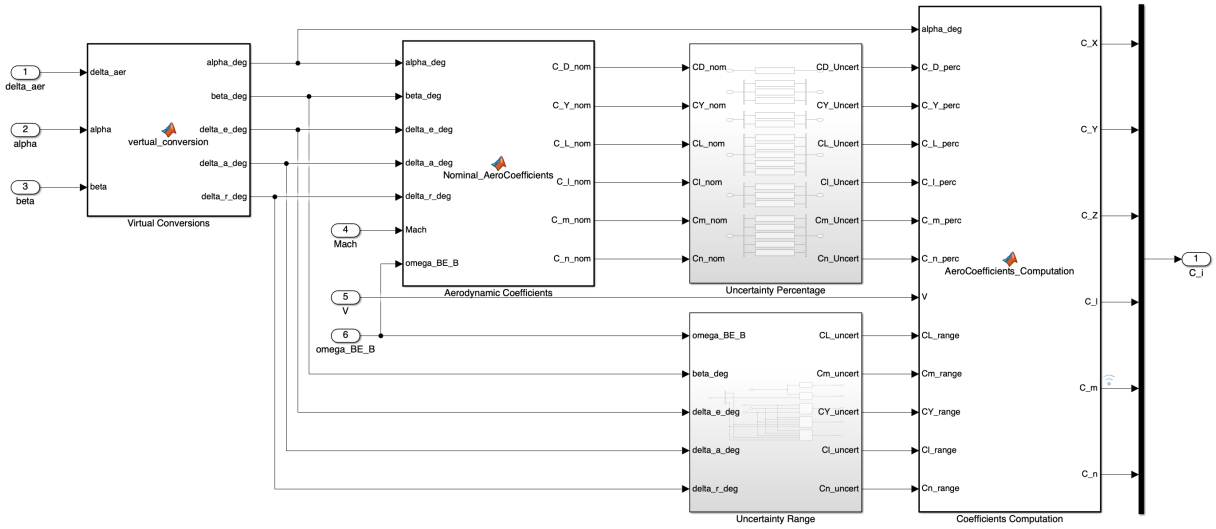


Fig. 4 Simulink model snippet of uncertainty implementation into aerodynamic coefficients

In this model, both variants of uncertainty were implemented into the framework. The schematic is shown in Fig. 4, where the *Uncertainty Percentage* block contains the percentage $ureal$ variables that are multiplied with their respective coefficients, whereas the *Uncertainty Range* block contains the absolute value range $ureal$ variables that are multiplied with their respective control deflections and rotational rates. The *Coefficients Computation* block is where all of them are added together. When one uncertainty variant is chosen, the other must be replaced by a real value of its mean, i.e., if percentage variation is chosen, then its twin variable uCm_q_range must be set to 0. Likewise, the uncertainty for some specific parameter can be "turned off" altogether by setting both of its respective uncertainty variables to their mean values. The uncertainty variant can be selected independently between parameters - one can be specified in percentage, and the other in range. However, the absolute value variation can not be implemented on parameters that are both continuously computed and not multiplied with some continuous signal. This is the case for C_{m_1}, C_{L_1}, and C_{D_1}, as well as engine parameters. These can only be specified in percentage.

For this initial design, the uncertainty was specified for C_{m_1} (which includes C_{m_0} and C_{m_α}), $C_{m_{\delta e}}$, and C_{m_q}. All three partial aerodynamic coefficients were arbitrarily chosen to vary by ±20%, while the uncertainty variables of the other coefficients were turned off.

2. Trimming

Before the model is linearized, it must be trimmed to some operating point. At that point, the control inputs are computed that would keep the system at steady-state (forces and moments are either 0 or constant). The appropriate selection of which states to keep constant, zero, or free depending on the equilibrium flight condition are outlined in [37].

The nominal operating point was selected at 0.75 Mach, 5000 m altitude, 0.5 fuel fraction of tank, in a steady, straight, wings-level flight. The trimming process was done via $findop$ function in MATLAB, which allows for code trimming specifications for the Simulink model. At this point, the inputs and outputs that would be used in the next steps must be selected. The uncertain airframe model has 4 inputs: thrust f_P, left and right elevon deflections δ_{vl} and δ_{vr}, and rudder deflection δ_r. Note that these are actuator outputs, not the commanded inputs. The actuator models will be added separately in the next step. The outputs of the uncertain airframe are the IMU-measured load factor and the rotational rates.

The states of the model and their trimming specifications are outlined for the nominal operating point in Table 1. The model is then trimmed automatically to find the necessary control inputs. The required thrust is then used to trim the propulsion system model in a similar fashion.

Table 1 Trimming Specifications for Model States and Inputs

Description	Known	Initial Value	Steady State	Min	Max
Roll rate p	Yes	0	Yes	$-\infty$	∞
Pitch rate q	Yes	0	Yes	$-\infty$	∞
Yaw rate r	Yes	0	Yes	$-\infty$	∞
Roll angle ϕ	Yes	0	Yes	$-90°$	$90°$
Pitch angle θ	No	$3°$	Yes	$-90°$	$90°$
Yaw angle ψ	No	0	Yes	$-360°$	$360°$
Airspeed V	Yes	240.4 m/s	Yes	0	∞
Angle of attack α	No	$3°$	Yes	0	$21°$
Sideslip angle β	No	0	Yes	$-\infty$	∞
Position x	No	0	No	$-\infty$	∞
Position y	No	0	Yes	$-\infty$	∞
Position z	Yes	-5000 m	Yes	$-\infty$	0

3. Linearization

The uncertain airframe model in Simulink can then be conveniently linearized around the selected operating point using the function $ulinearize$. The advantage of this method is that it recognizes the uss blocks with specified uncertainty in the model, and directly produces an uncertain state space model with embedded uncertain parameters specified earlier.

The computed full linear state-space model of the airframe has the following properties:

$$\begin{aligned} \text{Inputs} &= [f_P \quad \delta_{vl} \quad \delta_{vr} \quad \delta_r] \\ \text{States} &= [p \quad q \quad r \quad \phi \quad \theta \quad \psi \quad V \quad \alpha \quad \beta \quad z^E] \\ \text{Outputs} &= [p \quad q \quad r \quad n_x \quad n_y \quad n_z] \end{aligned}$$

The objective is to retrieve the longitudinal short-period model with virtual elevator input δ_e from the full model. The selected states are angle of attack α and pitch rate q, and the regulated outputs for the control system are the vertical load factor n_z measured by IMU and the pitch rate q. The short-period state-space model is retrieved by selecting appropriate rows and columns from the full state-space model matrices corresponding to the desired states and outputs. The two elevon inputs are converted to a single elevator input δ_e by simply adding the two elevons together in the rows corresponding to the longitudinal motion. The reason why the Eq. 5 is not used here is because the two elevons are represented separately with identical values for longitudinal states and outputs in the full linear model, so in this case δ_e action is divided equally between them.

The resulting short-period state-space model is represented by Eq. 19 with respective stability derivatives, according to [37]. The lateral and full longitudinal models were obtained in a similar way, but any further discussion on them is outside the scope of this research.

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -Z_\alpha/V & 1 \\ M_\alpha & M_q \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} -Z_{\delta_e}/V \\ M_s \end{bmatrix} \delta_e = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}$$
$$\begin{bmatrix} n_z \\ \alpha \end{bmatrix} = \begin{bmatrix} -A_\alpha/g & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} -A_{\delta_e}/g \\ 0 \end{bmatrix} \delta_e = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{19}$$

The stability derivatives were verified by using Jacobian linearization on the EoM analytically. The computed nominal short-period system matrices are presented in Eq. 32 in the Appendix.

The nominal short-period model input-output pole-zero map is shown in Fig. 5. It has a non-minimum-phase (NMP) zero in the right-half-plane for vertical load factor, which is consistent with this vehicle type since the elevons are located aft the vehicle's CG. Small frequencies indicate the system is relatively slow, and real negative pole locations indicate that it is open-loop stable and critically damped. The short-period model is in the form of uncertain state-space system, its properties and operating point conditions are summarized in Table 2.

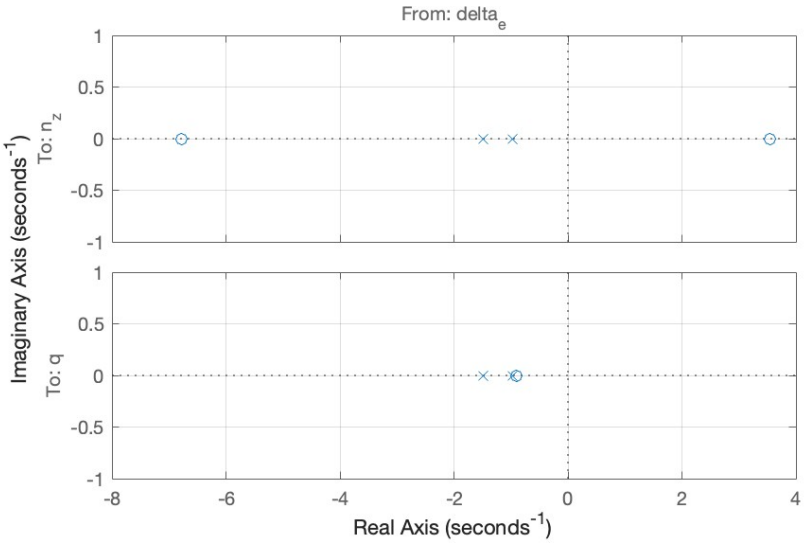


Fig. 5 Short-period input-output pole-zero map

Table 2 Short-period nominal linear system properties

Trim condition property	Value	Linear system property	Value
Mach number	0.75	Pole 1 λ_1	-0.965
Altitude	5000 m	Pole 2 λ_2	-1.48
Fuel fraction	0.5	NMP zero z_{nmp}	3.55
Angle of attack α	3.89°	Damping ratio ζ	1.0
Pitch angle θ	3.88°	Uncertainty C_{m_1}	±20%
Elevator deflection δ_e	-1.925°	Uncertainty $C_{m_{\delta_e}}$	±20%
		Uncertainty C_{m_q}	±20%

The virtual elevator δ_e is modelled as a 2-nd order linear system with the same natural frequency and damping ratio as for the elevons, presented in Table 7.

4. Model grid

To account for variations around the nominal flight condition, additional models were trimmed and linearized at specified deviations from the trim point. As mentioned earlier, the variations include fuel mass, Mach number, and altitude. The fuel mass is measured as a fuel fraction that is present in the tank, effectively varying vehicle mass and MOI. The variations around the nominal flight point are summarized in Table 3.

Table 3 Nominal flight point and variations around it

Property	Nominal FP value	Variation
Fuel fraction	0.5	0.25-0.75
Mach number	0.75	0.70-0.80
Altitude	5000 m	4500-5500 m

Taking into account the fuel fraction variation at the nominal trim point, a total grid of 11 models was assembled. All of them are short-period uss models and include parametric uncertainty in C_m specified earlier. The produced models and their properties are outlined in Table 4, where the first model corresponds to the nominal flight point

Table 4 Full short-period uncertain linear model grid

Model number	Model code name	Fuel fraction	Mach number	Altitude
1	G_sp_05_75_50	0.5	0.75	5000 m
2	G_sp_025_75_50	0.25	0.75	5000 m
3	G_sp_075_75_50	0.75	0.7	5000 m
4	G_sp_025_70_45	0.25	0.7	4500 m
5	G_sp_025_70_55	0.25	0.7	5500 m
6	G_sp_025_80_45	0.25	0.8	4500 m
7	G_sp_025_80_55	0.25	0.8	5500 m
8	G_sp_075_70_45	0.75	0.7	4500 m
9	G_sp_075_70_55	0.75	0.7	5500 m
10	G_sp_075_80_45	0.75	0.8	4500 m
11	G_sp_075_80_55	0.75	0.8	5500 m

IV. Robust flight control system design

A. H_∞ mixed-sensitivity theory

Control system performance and robustness to uncertainty specifications can be addressed with closed-loop (CL) transfer functions within the system. Consider a typical multiple-input-multiple-output (MIMO) control system shown in Fig. 6 with plant G, controller K, reference signal r, controller input signal e, controller output signal u, plant input signal u_p, plant output y_p, system output y, input and output disturbances d_I and d_O, and measurement noise n. If the system is closed-loop stable, then the fundamental relations between the signals can be derived as in Eq. 20.

$$\begin{aligned} e &= S_O r - S_O G d_I - S_O d_O + T_O n \\ y &= T_O r + S_O G d_I + S_O d_O - T_O n \\ u &= K S_O r - T_I d_I - K S_O d_O - K S_O n \\ u_p &= K S_O r - K S_O d_O + S_I d_O - K S_O n \end{aligned} \tag{20}$$

where S_I and S_O are input and output sensitivity transfer functions, and T_I and T_O are input and output complementary sensitivity transfer functions, respectively, with following relations: $S_I + T_I = I$, $S_O + T_O = I$. Basic performance

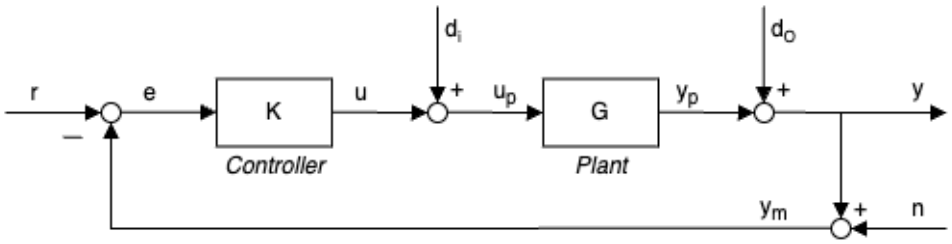


Fig. 6 General closed loop control system

objectives include disturbance rejection, noise attenuation, control effort minimization, reference tracking. Attenuating some signal's effects on the system involves minimizing the H_∞ norm of the transfer function between that signal and the analysis point that it affects. On top of that, enforcing robust stability to uncertainty involves minimizing the H_∞ norms of KS_O and both complementary sensitivity functions. A more detailed description of the relations between design specifications and the CL transfer functions can be found in [38]. However, fulfilling all of the requirements simultaneously at all frequencies is impossible, and thus trade-off must be made between robustness and performance.

The general H_∞ control problem can be formulated as [22]:

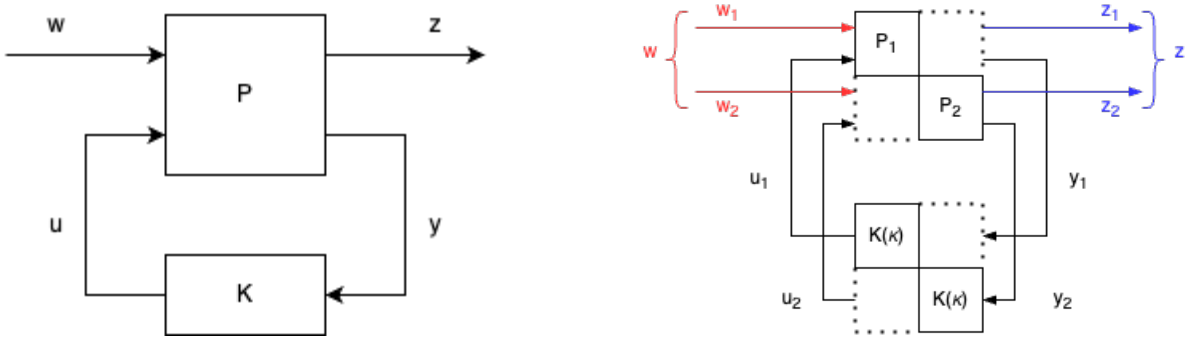
$$\begin{aligned} \text{minimize} \quad & ||T_{w\to z}(P,K)||_\infty \\ \text{subject to} \quad & K \text{ stabilizes } P \end{aligned}$$

which is schematically represented in Fig. 7a. In its core, the classical mixed-sensitivity H_∞ method is a way to synthesize a controller by putting appropriate weights on the selected closed loop (CL) transfer functions that attenuate the signals at certain frequencies. Notation is important here: a high frequency (HF) gain of inverted filter corresponds to HF maximum gain constraint for the transfer function (TF), and low frequency (LF) gain of inverted filter thus constraints the LF gain of the TF. Essentially, the filters are inverted transfer function gain constraints. For example, for output disturbance rejection at y, one can specify a performance channel $T_{w\to z}$ from d_O to y with a low-pass filter W_{S_O}, so that the disturbances are attenuated at low frequencies, where they normally happen. Likewise, for minimization of control effort at high frequencies and robustness to additive uncertainty, a high-pass filter W_{KS_O} can be put on on the channel from d_O to u. The classic mixed-sensitivity S/KS bundle is then put in a single minimization cost function under a single performance metric γ [38]:

$$\left\| \begin{bmatrix} W_{S_O} S_O \\ W_{KS_O} KS_O \end{bmatrix} \right\|_\infty \leq \gamma$$

The original H_∞-synthesis algorithms developed in 80-90's that use linear matrix inequalities (LMI) to solve this problem produce full-order controllers (order is the same as number of states in P + order of all filters), which is in many cases undesirable. Furthermore, the optimization of the S/KS two-block problem can result in cancellation of the stable poles in the plant by the controller zeros because the disturbance is considered only at the plant input or output [38]. On the other hand, a four-block problem with both input and output disturbances complicates filter selection [39]. The use of LMI-based optimization corresponds to $hinfsyn$ command in MATLAB. Relatively recently, in 2006, P. Apkarian and D. Noll developed non-smooth optimization algorithms that allow synthesis of controllers with predefined structure [22, 23]. The cost of this method is that it converges to local optima instead of the global one, like the former method does. However, it was shown that in practice the convergence to local optima performs even better, as it can handle problems of large scale, unlike the classical method which experiences numerical problems [23]. The non-smooth optimization then led to new solutions of mixed H_2/H_∞ and multidisk problems. The latter is of special interest in this case, as it allows controller synthesis for multi-objective and multi-modelling problems.

The multidisk problem formulation in a nutshell means having multiple separate performance channels $T_{w^i\to z^i}$, for which multiple plants P_i are formulated and the controller $K(\kappa)$ of fixed structure is connected to all of them simultaneously. The multidisk problem for two performance channels is illustrated in Fig. 7b and can be formulated as



(a) General H_∞ control problem illustration **(b) H_∞ multidisk control problem illustration**

Fig. 7 Standard and multidisk H_∞ control problems [22]

[22]:

$$\begin{aligned} \text{minimize} \quad & ||T_{w_1 \to z_1}(P_1, K)||_\infty \\ \text{subject to} \quad & ||T_{w_2 \to z_2}(P_2, K)||_\infty \leq \gamma_2 \\ & K \text{ stabilizes } P_1 \text{ and } P_2 \\ & K = K(\kappa) \text{ is structured} \end{aligned}$$

Naturally, the multidisk approach can be extended to include more than two performace channels, so that it can be used to design for multiple models, multiple performance objectives, and parametric uncertainty. Its broader formulation is then [22]:

$$\min_{\kappa \in \mathbb{R}^n} \max_{i=1,\ldots,N} \alpha_i \, \|F_\ell(P_i, K(\kappa))\|_\infty \tag{21}$$

Where α_i is cost weight per channel, and N can also be an uncertainty set Δ. The trick lies in the fact that the maximum H_∞ norm among multiple separate norms is equal to the joint H_∞ norm. Thus, minimizing the maximum H_∞ among the channels also minimizes the total H_∞ norm of the system. This process iterates on a worst-case basis, where at each iteration a new maximum H_∞ norm is found and the controller is updated according to it, until a satisfactory solution is reached. In the case of real parametric uncertainty, it uses an inner relaxation method with generally same logic as above. μ-analysis is used to certify the solution with a conservative metric in the end rather than during synthesis, which outperforms outer relaxation methods such as for μ-synthesis. A more detailed description of the multidisk problem solution and application can be found in the original publication paper [24], and its application for parametric uncertainty with inner relaxation can be found in [25]. The non-smooth optimization methods have been put into the *hinfstruct* and *systune* MATLAB functions, and therefore both offer structured H_∞ controllers synthesis with multi-modelling capabilities. The focus in this paper is put on *systune*, because it optimizes each constraint individually in a separate performance channel with its own γ metric for analysis, which allows a more flexible oversight over the performance objectives.

B. FCS synthesis

The controlled variables for the flight control system are the vertical load factor n_z and the pitch rate q, and the control input is the commanded elevator deflection angle $\delta_{e,cmd}$. Due to the exploratory nature of this research it is unknown how much robust performance can the GHAME provide, so there are no initial concrete numbers on achievable robustness margins or on how fast the disturbances can be rejected. The general margin requirements for flight control systems include a minimum of 6 db gain margin (GM) and 35°phase margin (PM) evaluated on a Nichols exclusion region [38]. These requirements are a good first indication of available robustness margins and are thus included in the design specifications for the FCS. Therefore, the design requirements are of general nature at this point, and are outlined below:

1) The FCS must robustly stabilize the vehicle for the nominal flight point under parametric uncertainties Δ in C_m partial coefficients of ±20%.

2) The FCS must provide a minimum disk GM of 6 dB and a minimum disk PM of 35°at the plant input and outputs evaluated on a Nichols exclusion region for all Δ.
3) The FCS must provide adequate n_z disturbance rejection at the plant output for all Δ.
4) The FCS must provide adequate δ_e disturbance rejection at the plant input for all Δ.
5) The FCS must provide adequate tracking of n_z reference commands for all Δ.
6) The closed loop short-period natural frequency and damping ratio must abide the flying qualities criteria outlined in [40].
7) The FCS must attenuate high-frequency control gains near actuator bandwidth.
8) The commanded actuator response overshoot must be within adequate limits to avoid actuator saturation.
9) The FCS must meet the design requirements using fixed structure controllers of minimum possible order.

1. Set-up

The Simulink set-up is shown in Fig. 8. The gray G_a block represents the actuator model with parameters presented in Table 7, and the gray G_{sp} block represents the uncertain short-period state-space model from Eq. 19. The blue blocks are the tunable controllers, and the green blocks are the inputs into the system, namely: reference n_z value represented by signal r, input disturbance d_I, and output disturbance d_O.

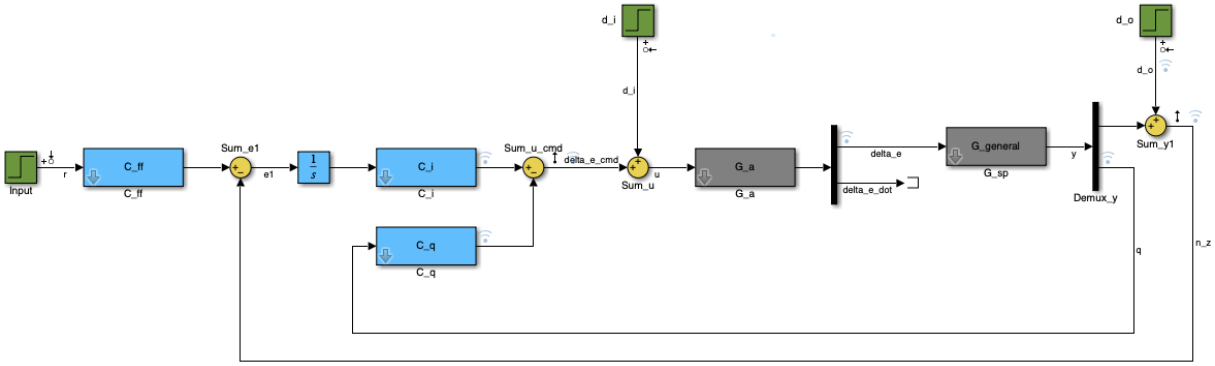


Fig. 8 Short-period FCS layout

The actuator model G_a has 2nd-order dynamics, takes commanded elevator deflection $\delta_{e,cmd}$ as input, and produces the elevator deflection δ_e and elevator deflection rate $\dot{\delta}_e$ as output. The first output δ_e is then fed as an input into the short-period airframe model G_{sp}. The joint actuator-airframe plant G_{asp} can be represented in zero-pole-gain format:

$$G_{asp}(\delta_{e,cmd} \rightarrow n_z): \quad \frac{291.44(s+6.785)(s-3.547)}{(s+0.9654)(s+1.479)(s^2+70s+2500)} \tag{22}$$

$$G_{asp}(\delta_{e,cmd} \rightarrow q): \quad \frac{-316.85(s+0.9046)}{(s+0.9654)(s+1.479)(s^2+70s+2500)} \tag{23}$$

Here, the left half plane (LHP) complex poles correspond to the actuator, and the real negative poles correspond to the airframe, they were already identified back in Table 2. The NMP zero in the n_z channel is of particular interest here, as it is located at a relatively low frequency of 3.547 rad/s. This zero imposes a limitation on achievable bandwidth of the system. The maximum possible crossover frequency ω_c (and thus the bandwidth ω_B) for a real RHP zero is [39]:

$$\omega_B \approx \omega_c < \frac{z_{NMP}}{2} = 1.7735 \quad \text{rad/s} \tag{24}$$

This is the case for an "ideal" controller without any penalty on the input u, thus making it the theoretical ceiling. In practice, however, the input would be weighted in the cost function, and the controller would be structured, so the achievable bandwidth is expected to be considerably lower. Additionally, the bandwidth ω_B is defined here as a frequency where the sensitivity S crosses the -3.01 dB line, so $\omega_B < \omega_c$, thereby decreasing it further. In a nutshell, this makes tight control (reference command following and disturbance rejection) possible only in the lower frequency range.

It is also possible to flip it around and have tight control in the frequencies above the NMP zero, but makes no practical sense in this case.

A way to bypass the bandwidth limitation is to use non-casual controllers where the output depends on future inputs, which is unfortunately unrealizable in practice [39]. Only casual controllers can be implemented in real world systems, and with them the bandwidth limitation from NMP zero simply has to be tolerated. To ensure that the controllers are casual and thus realizable, the controller transfer function must be proper, i.e., numerator cannot be of higher order than the denominator.

2. Controller structure

There are a total of 3 controllers to be tuned. Integral controller C_i is put on the error between reference signal and the output n_z. An integral itself is placed on the line with C_i to ensure zero steady-state error. Proportional controller C_q utilizes the pitch rate q to stabilize rotation. A feed-forward controller C_{ff} is placed outside of the loop to tune transient response to reference commands, for which a convenient choice is a lead-lag compensator [39]. A proportional controller on n_z was found to be redundant. The C_i and C_q controller blocks were initially set as tunable gains, which led to good stability margins yet poor performance in time domain, so the order had to be increased. A final well-balanced structure was found to be the following:

$$C_i = \frac{K_i(s+z_i)}{(s+p_i)}, \quad C_q = K_q, \quad C_{ff} = \frac{K_{ff}(s+z_{ff})}{(s+p_{ff})} \tag{25}$$

Which corresponds to setting C_i and C_{ff} blocks to tunable 1-st order transfer functions with one zero and one pole, and the C_q to tunable gain.

3. Feedback tuning goals

Unlike the conventional mixed-sensitivity approach, there is no need to restructure the Simulink model from Fig. 8 into a $T_{w\to z}$ performance channel configuration with $systune$. Here, the analysis points are used to specify the desired inputs and outputs of the performance channels for constraints using the related signal names. The program then computes the corresponding transfer functions and applies the constraints on them. The selected analysis points are the reference signal r, the disturbance inputs d_i and d_i, the commanded actuator deflection $\delta_{e,cmd}$, and the output n_z. They can be seen as input perturbation and output measurement arrows on the corresponding signals in Fig. 8.

The Simulink model, analysis points, tuned blocks, and model substitutions are input into the $slTuner$ object within $systune$, which is used to create tuning goals. The tuning goals are basically the performance channels, and can handle time-domain, frequency domain, pole placement, minimum margins, and other constraints. The process is now similar to setting the conventional weighting filters for classical mixed-sensitivity. The procedure below is described for the nominal flight point.

First, the output sensitivity function S_O is constrained for output disturbance rejections at the controlled output. The $Gain$ tuning goal is created with input d_o and output n_z. The disturbances are expected to be in LF, so the inverse weighting function must be a high-pass filer. That way, output sensitivity is attenuated at low frequencies, and allowed to pass at HF, where the disturbances are not present. LF gain is thus set to a near-zero value of -60 dB. The HF gain constraint corresponds to the maximum peak value of S_O, which is directly related to the modulus disk margin. Generally, the disk margin α_{max} is computed as in Eq. 26 [41]:

$$\alpha_{max} = \frac{1}{\|S + \frac{\sigma-1}{2}\|_\infty} \tag{26}$$

where σ is the disk skew. When $\sigma = 1$, it becomes the S-based (modulus) margin with the disk centered exactly at the critical point in the Nyquist plot, and thus corresponds to the distance from the n_z open-loop curve to the critical point. Constraining $\|S\|_\infty$ puts a bound on the minimum distance to the critical point in the Nyquist plot. A common maximum peak criteria is to constrain the peak gain of S to be ≤ 2, which guarantees a GM ≥ 6 dB and PM $\geq 30°$[39] within the n_z loop. To slightly increase the margins, the HF gain of S_O was set to 1.9. The achievable bandwidth of S_O is part of the trade-off between performance and robustness, and is specified with a gain of -3.01 dB at the selected frequency. The achieved bandwidth ω_B of S_O is 0.4 rad/s. The low bandwidth is consistent with the expected range from Eq. 24, and increasing it any further leads to actuator saturation. The inverted filter is set as a proper 1st order transfer function using a $makeweight$ MATLAB command. It is important to note that the inverted weights are passed directly into $systune$, and are inverted back automatically within the cost function.

The second tuning goal is to constrain the gain of $-KS_O$ (which is actually not a negative equivalent of KS_O since it is a multiple output system, but the name is kept for convention). The tuning goal analysis points are d_o as input and $\delta_{e,cmd}$ as output. To attenuate HF control effort and provide robustness to additive uncertainty, the gain of $-KS_O$ must roll-off sufficiently at the actuator bandwidth. The LF gain is effectively unconstrained and set to 60 dB, since the actual gain will be the DC-gain of actuator. The HF-gain must approach zero, so it is set to -60 dB. The bandwidth frequency is equal to actuator bandwidth of 50 rad/s, and the gain at that point is the performance-robustness trade-off parameter set to -16 dB for sufficient attenuation. The function is scaled with inverse of a singular value of the G_{asp} model. The weighting filter is 1st order proper TF.

Two more constraints follow a similar pattern for input disturbance. The tuning goal on S_OG constraints the sensitivity of output n_z to input disturbance d_i, so the analysis points are set accordingly. The DC gain limit is set to -60 dB for disturbance rejection at low frequency range. The HF-gain is set to 2, although the function rolls off at HF due to system gain G. The bandwidth at -3.01 dB is set to 0.8 rad/s, which is again a trade-off parameter. The filter is again a 1st order TF.

Finally, constraining the input complementary sensitivity T_i bounds the control effort wrt input disturbance. The transfer function T_i is set from input d_i to output $\delta_{e,cmd}$. Its peak value is located in the LF range and is constrained to magnitude of 2. The attenuation at the actuator bandwidth of 50 rad/s is set to -23 dB. The T_i has a steeper slope than the transfer functions above, a 3rd order filter is necessary to properly constrain the roll-off, and the high-frequency gain is set to -80 dB. The singular values of all the respective filter inverses are shown in Fig. 9.

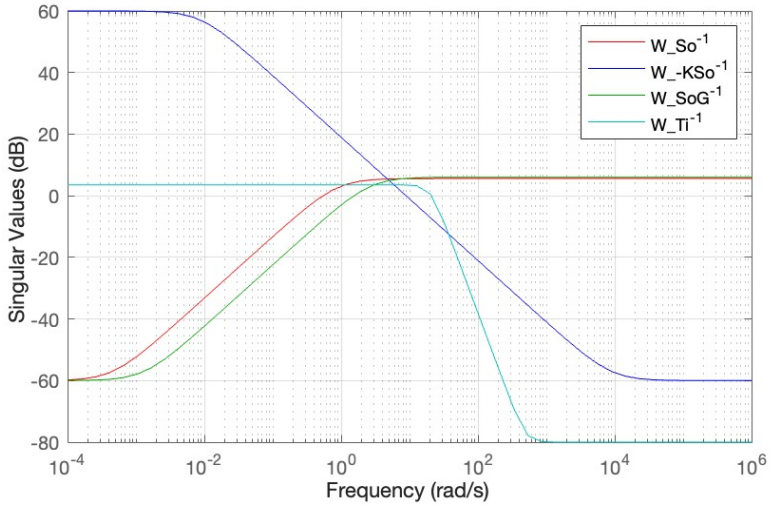


Fig. 9 Inverse weighting filters in frequency domain, nominal flight point

4. Reference tracking

The reference tracking behavior can be specified with a reference model. The H_∞ norm is then placed on the error between the actual model and the reference model responses. The reference model is a 2nd order transfer function with identified NMP zero, and has a form:

$$T_r = \frac{-\frac{\omega_{\text{ref}}^2}{z_{\text{nmp}}}s + \omega_{\text{ref}}^2}{s^2 + 2\omega_{\text{ref}}\zeta_{\text{ref}}s + \omega_{\text{ref}}^2} \tag{27}$$

The natural frequency and damping ratio are selected according to the short-period handling qualities criteria. A handling qualities analysis was performed on GHAME in [40] for all modes of motion, where GHAME was identified as a Class III aircraft - large, heavy, low-to-meduim maneuverability. For a non-terminal flight condition of Category B

(climb, cruise, descent) corresponding to the selected trim point, the short-period requirements are:

$$\begin{array}{lll} \text{Level 1:} & 0.46 \leq \omega_{\text{ref}} \leq 3.50 & 0.30 \leq \zeta_{\text{ref}} \leq 2.00 \\ \text{Level 2:} & 0.36 \leq \omega_{\text{ref}} \leq 6.00 & 0.20 \leq \zeta_{\text{ref}} \leq 2.00 \\ \text{Level 3:} & 0.36 \leq \omega_{\text{ref}} & 0.05 \leq \zeta_{\text{ref}} \end{array}$$

Instead of guessing the optimal ω and ζ values, an optimization program $fmincon$ is used to compute them based on desired settling time and overshoot values. It is set to minimize the weighted sum of the squared errors in settling time and overshoot between the set values and the achieved ones. The settling time is set to 4 seconds, overshoot to 1%. To meet Level 1 requirements, the bounds on optimal ω_{ref} are set between 0.5 and 3.5, and bounds on optimal ζ_{ref} are between 0.7 and 1.

The optimal values computed to be $\omega_{\text{ref}} = 1.0597$, and $\zeta_{\text{ref}} = 0.8279$, they are substituted into Eq. 27 along with the NMP zero in the n_z channel. The constraint is then set directly as $StepTracking$ tuning goal within $systune$ environment, which takes the reference model T_{ref} to optimize the transient response, and a relationship gap that is set to 15%. The input is r and the output is n_z, forming the final fifth performance channel. This concludes the process general constraint set-up for the nominal flight point model. All 5 performance specifications are passed as constraints of equal weight, and are summarized in Table 5. Finally, $systune$ function is called to compute the controllers.

5. Multi-modelling

In the multi-modelling extension, the design requirements identified earlier in this section now must be met for the entire model grid using a single controller. Practically, the multi-modelling procedure is very similar to the one for the nominal flight point. Now, however, a model array is substituted into the G_{sp} block in Fig 8. The array consists of the uncertain state space models from the grid in Table 4. Given a broad variety of conditions that the controller must satisfy, there are now two sets of tuning goals. One set contains the hard constraints that are applied on the uncertain model at the nominal flight point (model 1 from Table 4), whereas the second set contains soft constraints, which are applied on all the other uncertain models including those with fuel variation at the nominal trim point (models 2-11). Evidently from the name, the hard constraints are prioritized over the soft ones in the algorithm. The new hard and soft constraints cover the same performance channels as before, carry similar names, and are specified with the same logic. They are, however, generally more relaxed compared to designing for nominal flight point only, which is understandable since a single controller now has to satisfy the full grid of models. The final values for hard and soft constraints for multi-modelling are outlined in Table 5.

C. Results and analysis

The results are analysed in consecutive steps. First, the γ performance metric is given, one per tuning goal. This is a first indication if the constraints have been met successfully. The transfer functions are then plotted in frequency domain with their respective inverse filters to check how well they comply. The robustness is assessed with disk margins at the plant input and outputs. The open loops are then plotted on the Nichols chart. Each transfer function is then plotted in the time domain to assess performance w.r.t. disturbance rejection and control effort.

The tuned controllers for the nominal FP design and for the full grid design are presented in Eq. 28.

$$\begin{array}{rlrl} & \text{Nominal FP:} & & \text{Full Grid:} \\ C_i = & \dfrac{-3.7014(s+0.9311)}{(s+3.25)} & C_i = & \dfrac{-4.0888(s+0.8258)}{(s+2.434)} \\ C_q = & -24.587 & C_q = & -34.412 \\ C_{ff} = & \dfrac{0.80293(s+1.182)}{(s+0.9454)} & C_{ff} = & \dfrac{0.62306(s+1.466)}{(s+0.9135)} \end{array} \tag{28}$$

The resulting γ values for the nominal flight point are:

$$\text{Nominal: } \gamma(S_O) = 0.9446, \quad \gamma(-KS_O) = 0.9964, \quad \gamma(S_OG) = 0.9968, \quad \gamma(T_i) = 0.9964, \quad \gamma(T_{\text{ref}}) = 0.9935 \tag{29}$$

The closer γ is to 1 from the lower side, the better performance is; a value larger than 1 indicates a constraint violation. The values from Eq. 29 can be related to the graphs in Fig. 10, where the respective transfer functions' singular values

Table 5 Weighting filters for the nominal flight point (left) and for multi-modelling (right)

Constraint specification	Nominal flight point	Full grid, hard	Full grid, soft
$d_o \rightarrow n_z$: S_O tuning goal, 1st order TF			
LF-gain	-60 dB	-60 dB	-60 dB
Bandwidth ω_B	0.4 rad/s	0.35 rad/s	0.3 rad/s
Gain at ω_B	-3.01 dB	-3.01 dB	-3.01 dB
HF-gain	1.9	1.9	1.9
$d_o \rightarrow \delta_{e,cmd}$: $-KS_O$ tuning goal, 1st order TF			
LF-gain	60 dB	60 dB	60 dB
Bandwidth ω_B	50 rad/s	50 rad/s	50 rad/s
Gain at ω_B	-16 dB	-16 dB	-16 dB
HF-gain	-60 dB	-60 dB	-60 dB
$d_i \rightarrow n_z$: S_OG tuning goal, 1st order TF			
LF-gain	-60 dB	-60 dB	-60 dB
Bandwidth ω_B	0.8 rad/s	1.4 rad/s	1.3 rad/s
Gain at ω_B	-3.01 dB	-3.01 dB	-3.01 dB
HF-gain	2	2	2
$d_i \rightarrow \delta_{e,cmd}$: T_i tuning goal, 3rd order TF			
LF-gain	2	2	2
Bandwidth ω_B	50 rad/s	50 rad/s	50 rad/s
Gain at ω_B	-23 dB	-20 dB	-15 dB
HF-gain	-80 dB	-80 dB	-80 dB
$r \rightarrow n_z$: T_{ref} tuning goal, using relationship gap			
ω_{ref}	1.0597	1.0597	1.0597
ζ_{ref}	0.8279	0.8279	0.8279
Relationship gap	15 %	15 %	20%

are plotted with their filter inverses. The γ values close to 1 from the lower side correspond to singular values from Fig. 10 (blue) being located close to the borders of the weighting filters (yellow), but not violating them. This indicates a successful constraint selection which uses full potential of the system. It is worth noting that γ for the reference tracking constraint is measured in the time domain according to how much the relationship gap is violated. It is therefore possible that the fifth γ is violated by one of the sampled models, but the average step tracking for all of them is generally in line with requirements, so it is better to assess the fifth tuning goal compliance visually. Notice also how a 3rd order weighting filter on T_i smoothly constraints its entire roll-off slope, whereas lower order filter was found not steep enough, basically affecting T_i only at break-out frequency.

For the multi-modelling tuning with hard and soft constraints, two γ values are now given per performance channel: one for soft constraint and one for hard constraint, giving a total of 10 performance metrics. The respective γ values for the multi-modelling case are presented below:

$$\text{Hard: } \gamma(S_O) = 1.0000, \quad \gamma(-KS_O) = 1.0000, \quad \gamma(S_OG) = 1.0000, \quad \gamma(T_i) = 1.0000, \quad \gamma(T_{\text{ref}}) = 0.7156 \tag{30}$$

$$\text{Soft: } \gamma(S_O) = 1.0317, \quad \gamma(-KS_O) = 1.0505, \quad \gamma(S_OG) = 1.0009, \quad \gamma(T_i) = 1.0815, \quad \gamma(T_{\text{ref}}) = 1.2557 \tag{31}$$

Naturally, there are also 10 tuning goal compliance graphs, one per hard and one per soft constraint. They are shown Fig. 11, and follow similar relationship as in the nominal FP case. As can be seen from the γ values and the charts, the

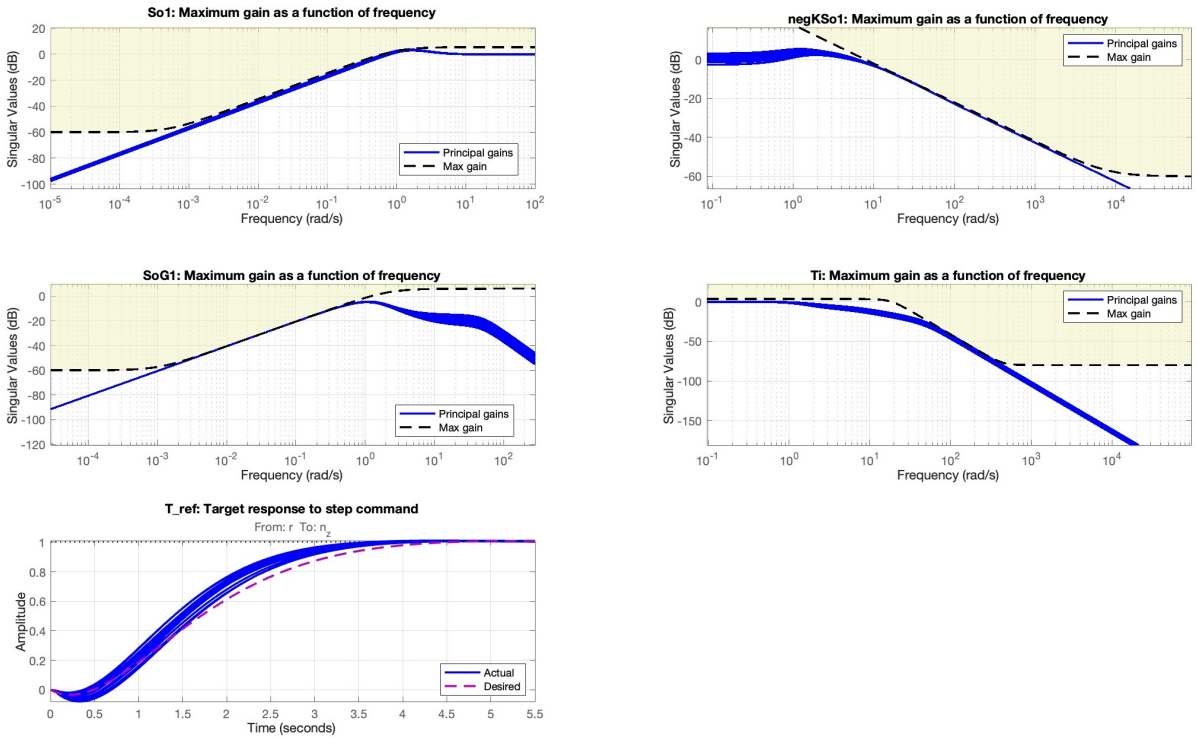


Fig. 10 Tuning goals results for nominal flight point model

hard constraints are perfectly met, whereas some slight violation is present in the soft requirements. This is considered an acceptable violation, given the variety of trim conditions and the fact that less weight is put on the soft constraints than on the hard ones. The γ of reference tracking is understandably large for the soft case, but the average step pattern follows the desired one closely, and the models have approximately similar transient time, as can be seen in the same Fig. 11. Overall, the controller satisfies the tuning goals quite well for all 11 uss models. Additionally, the observed successful roll-off of T_i and $-KS_O$ at the actuator bandwidth frequency of 50 rad/s satisfies FCS requirement 7, whereas the successful fulfillment of step tracking goal fulfills FCS requirement 6.

For the SISO open-loop analysis, the uncertain state-space models are sampled at random values of uncertain parameters in C_{m_α}, $C_{m_{\delta_e}}$, C_{m_q} using $usample$ function in MATLAB. For illustration purposes, 20 state-space systems were randomly sampled per model in the grid. This amounts to 20 models for the nominal flight point design, and $11 \times 20 = 220$ models for the full grid design. The SISO open-loops were then computed with $loopsens$ MATLAB function which connects the model array G_{asp} to controller matrix K and automatically opens the loops at inputs and outputs. It also indicates if the closed-loops are stable and, to no surprise, confirmed the stability of all closed loops in the model arrays, thus FCS design requirement 1 is satisfied. The SISO open loops were then extracted, and the $diskmargin$ MATLAB function was called on all of the samples to compute the $S-T$ disk margin (DM) corresponding to Eq. 26 with $\sigma = 0$. This way the DM is symmetrical, with equal probability that gain and phase can vary in either direction. The smallest DM, disk gain margin (DGM), disk phase margin (DPM), and multi-loop input-output margin (MMIO) among all samples were extracted. It is worth noting that the multi-loop disk margin incorporates the structured singular value μ in the computation process [41].

The DM's were then used to plot the disks of the worst-case margins on the Nichols charts (orange), and the 6 dB DGM requirement was used to plot the exclusion regions (red). The opened SISO loops were then plotted on their respective Nichols charts, together with DGM (green) and DPM (purple) lines for better comprehension.

The opened loops at the plant input are presented in Fig. 12. Naturally, the margins for the worst case loop among those sampled at the nominal flight condition in Fig. 12a are significantly larger than the worst-case margins for the full grid samples in Fig. 12b. Nevertheless, both worst-case sampled scenarios pass the FCS design requirement 2 with flying colors. In contrast, the open loops at the plant output n_z, shown in Fig. 13, are significantly closer to the

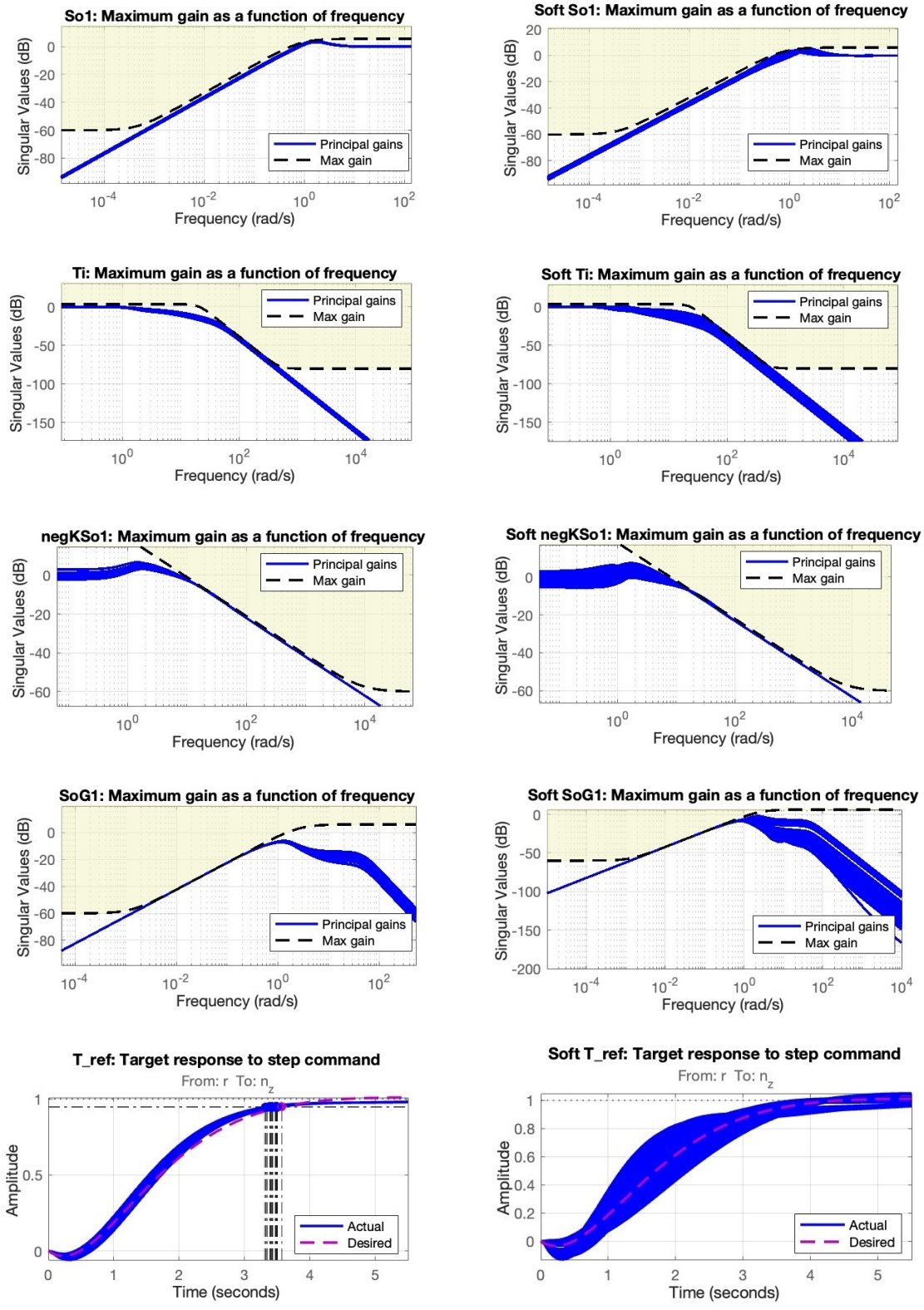
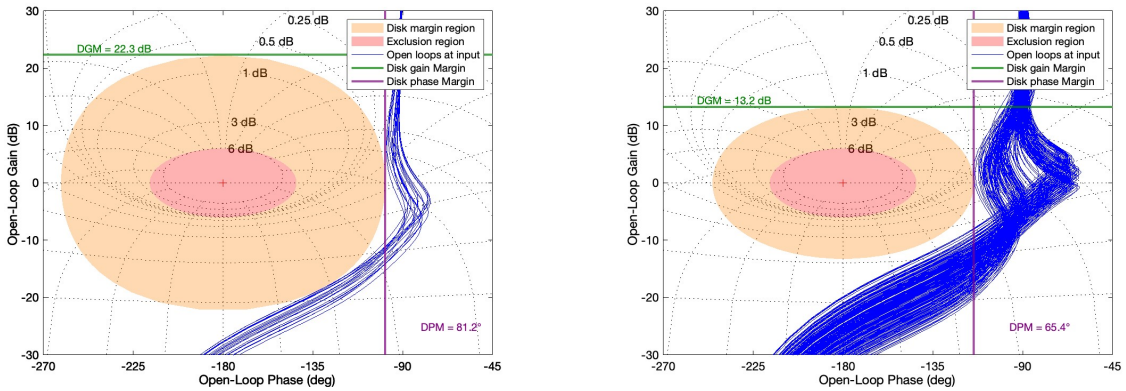


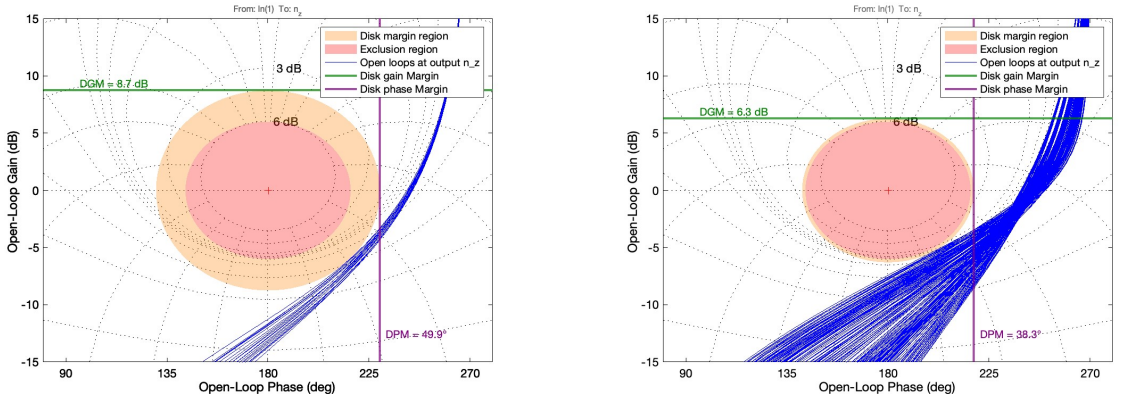
Fig. 11 Tuning goals results for full grid



(a) Nominal flight point **(b) Full grid**

Fig. 12 Nichols chart of sampled SISO loops opened at plant input

exclusions region. In a similar fashion, the margins for the full grid case in Fig. 13b are smaller than for the nominal point in Fig. 13a, but still pass the design requirement 2.

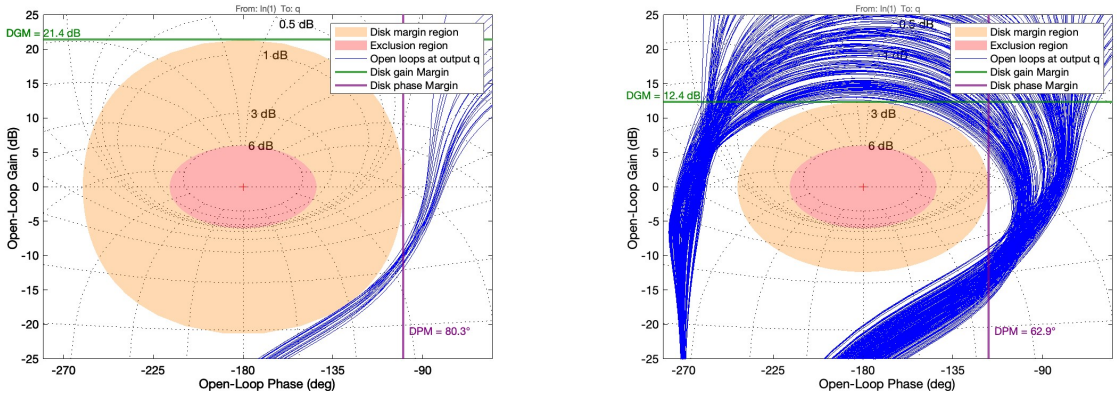


(a) Nominal flight point **(b) Full grid**

Fig. 13 Nichols chart of sampled SISO loops opened at plant output n_z

Although the n_z channel is the most important to analyze since that is the output to be controlled by the reference signal and that is the loop on which the output disturbance is acting, it is still necessary to consider all inputs and outputs in the system. The Nichols charts for the sampled SISO loops opened at plant output q are shown in Fig. 14, and follow the same trend as in the previous two charts with good DGM and DPM.

Therefore, a single set of controllers successfully handles the short-period dynamics under the specified parametric uncertainty of 20% in all C_m partial coefficients, and still meets the requirement of 6 dB and 35°at the inputs and outputs of the plant (FCS requirement 2 satisfied). This is especially a considerable achievement in the multi-modelling case, where 11 uncertain models from the grid are sampled into 20 each. The controller still guarantees robustness to unstructured uncertainty with larger margins than required, and that is the case for the worst of the 220 samples. To summarize the disk margin analysis, the DGM and DPM values from the Nichols charts are presented all together in Table 6, along with simultaneous MMIO gain and phase margins. The MMIO margins are indeed lower, to no surprise. However, those are, again, the worst case parametric uncertainty margins, and there are still robustness guarantees against simultaneous multi-loop changes at inputs and outputs.



(a) Nominal flight point **(b) Full grid**

Fig. 14 Nichols chart of sampled SISO loops opened at plant output q

Table 6 Minimum achieved symmetrical disk margins under sampled parametric uncertainty

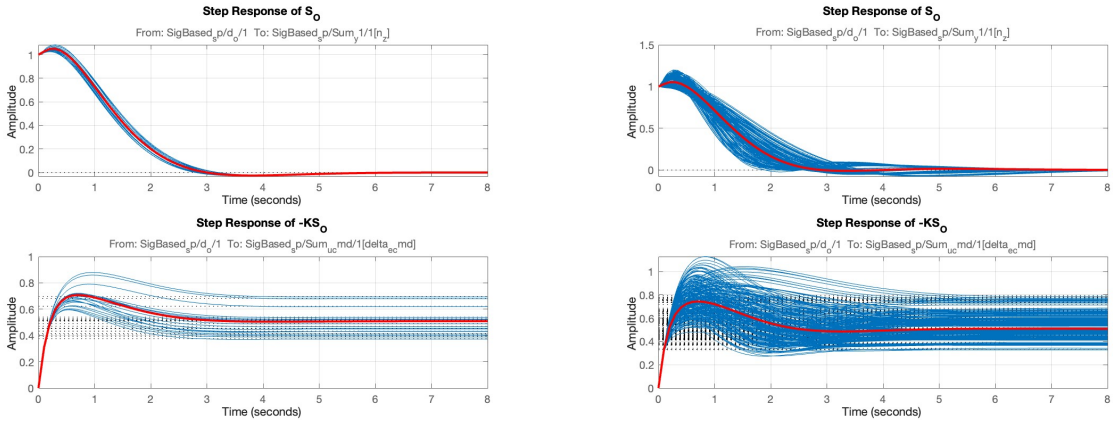
Margin type	Nominal flight point	Full grid	Units
Disk gain margin at input $\delta_{e,cmd}$	22.28	13.93	dB
Disk gain margin at output n_z	8.75	6.47	dB
Disk gain margin at output q	21.40	13.10	dB
Multi-loop input-output gain margin	3.90	2.88	dB
Disk phase margin at input $\delta_{e,cmd}$	81.20	67.25	deg
Disk phase margin at output n_z	49.87	39.22	deg
Disk phase margin at output q	80.28	65.04	deg
Multi-loop input-output gain margin	27.92	25.41	deg

Lastly, the CL performance of disturbance rejection and reference tracking were analysed in the time domain. In a similar fashion as for the SISO loops, the CL transfer functions were sampled in the parametric uncertainty domain Δ, and the nominal flight point design is presented next to the full grid multi-modelling design. The step responses of the transfer functions are grouped together according to output disturbance rejection case, input disturbance rejection case, and the reference tracking case. In each of the plots, the thick red line indicates the nominal system response (no uncertainty) at the nominal operating point.

For output disturbance rejection, the step response of S_O indicating n_z response to d_o is plotted together with step response of $-KS_O$, which represents the commanded control response to reject d_o. The step responses are plotted in Fig. 15. It is evident that the disturbance is rejected efficiently in under about 3 seconds, thus fulfilling FCS design requirement 3. Both the overshoot and steady-state amplitudes of the $-KS_O$ plots exceed the actuator deflection limit of $20° = 0.3491$ rad. This indicates that the maximum achievable n_z value is significantly less than 1, which can not be accounted for in linear simulations directly. The overshoot, however, is contained to not exceed the steady-state value too far, implicitly avoiding potential actuator saturation and satisfying FCS requirement 8. The control step responses ($-KS_O$) for the full grid are especially vastly spread, which is expected. For example, look at the plots exactly at 2 seconds into the step in Fig. 15b. The slowly descending response with the highest amplitude at 2 seconds for both S_O and $-KS_O$ corresponds to the 9th model in the array (refer to grid in Table 4). G_sp_075_70_55 flies at the top altitude in the grid (smallest speed of sound), and at the smallest Mach in the grid, which together mean it has the smallest airspeed V among the entire model grid. Additionally, it has largest fuel fraction in the tank, thus having larger mass and MOI, so G_sp_075_70_55 has the least control authority in the grid, predictably making its response slow and required actuator deflection large.

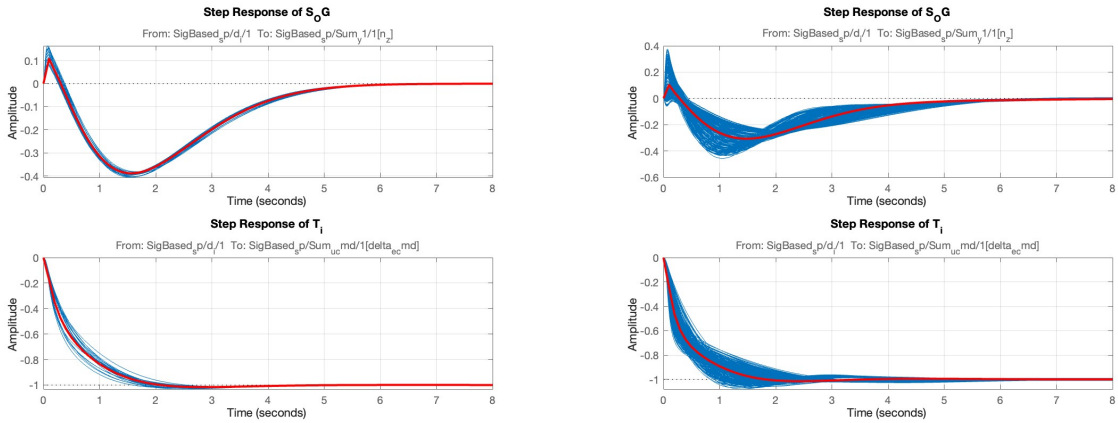
In the same manner, the input disturbance rejection is assessed with $S_O G$ for output and T_i for control response in Fig. 16. Since the disturbance is acting on the commanded input, it is the actuator deflection that disturbs the output. Therefore, the n_z value oscillates around 0 until the input disturbance is completely rejected. As can be seen in the figures, the input disturbance is rejected smoothly without overshoot in the commanded deflection path, and n_z is brought back to zero for all models. The FCS requirement 4 is thus fulfilled.

Lastly, the reference n_z step tracking is presented in Fig. 17 with the transfer functions $T_{r\rightarrow n_z}$ and $[K_{ff}KS_O]_{r\rightarrow\delta_{e,cmd}}$. In the nominal flight point design in Fig. 17a, where the samples correspond only to previously defined parametric uncertainty, it can be observed for $K_{ff}KS_O$ how different the steady-state actuator deflections are. Needless to say that adding another 10 models to the nominal FP in Fig. 17b fills the chart completely. Nevertheless, that is the reason why the n_z tracking in the T charts is smoothly performed. Although it was possible to set a desired step tracking transient time 1 second faster than it is here, it resulted in severe actuator response overshoot (about double the steady state value), so a 1 second slower reference tracking response time was set to avoid actuator overload. Thus, the FCS requirements 5 and 8 are satisfied. As a result, the control system successfully meets all the design requirements specified back in subsection IV.B, both for the nominal flight point design and for the multi-modelling design. The next step is to test it in the non-linear simulation.



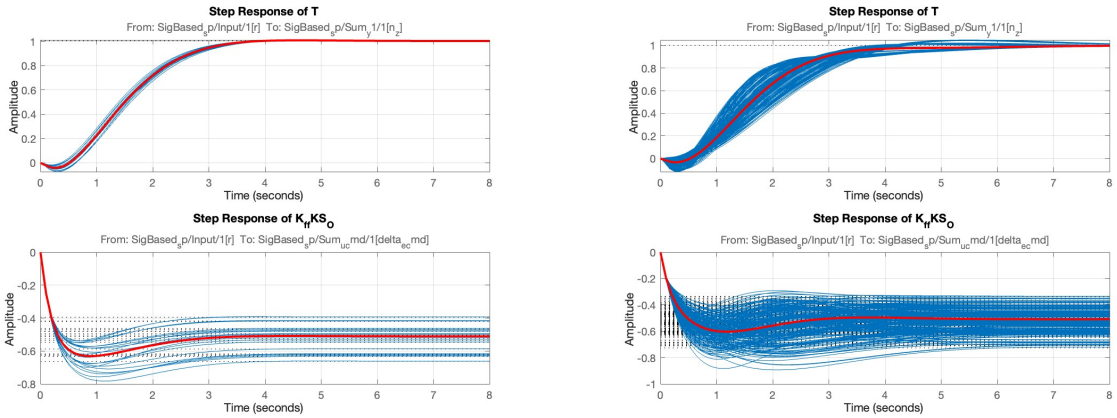
(a) Nominal flight point model **(b) All models in the grid**

Fig. 15 Output disturbance rejection on linear system



(a) Nominal flight point model **(b) All models in the grid**

Fig. 16 Input disturbance rejection on linear system



(a) Nominal flight point model **(b) All models in the grid**

Fig. 17 Reference step tracking on linear system

V. Non-linear implementation and simulation

A. Implementation

To test the synthesized controllers, a flight control system block is added to the Simulink model, as well as input and output disturbances. The complete non-linear Simulink model layout is shown in Fig. 18. The *FCS* block is placed before the actuators, with output measurements fed back into it. The input disturbance is positioned to act simultaneously on both elevons in the *Actuators* block, and the output disturbance acts directly on the n_z component of the load factor in the *Sensors* block.

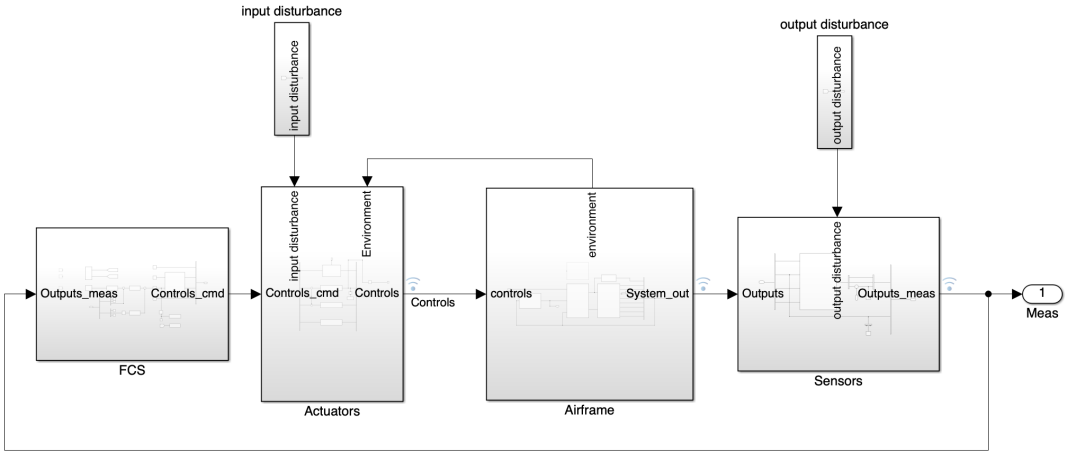


Fig. 18 Flight control system implementation into non-linear model

The expanded *FCS* block is shown in Fig. 19. The n_z and q are extracted from the output measurements and fed into the familiar controller block structure. This time, however, $\delta_{e,cmd}$ must be initialized with the trim value, for which the integrator was moved past the summation point with C_q. It also contains saturation limits to avoid out-of-bounds commands. Since the q channel is now in the integrator path, it must be differentiated beforehand [42]. Therefore, a derivative is placed before the C_q block. The time step of the simulation is set to 0.001 s, so the derivative block sampling is set to 0.01 s to have a smaller discrete time step relative to the simulation.

Commanded deflection $\delta_{e,cmd}$ is then fed into conversion block with $\delta_{a,cmd}$ to transform the commanded virtual elevator and aileron deflections into real elevon deflections. They are then added to the bus with trimmed throttle and trimmed rudder values and sent into the *Actuator* block. Lastly, in the top left corner of Fig. 19 there are reference and disturbance step inputs, each next to its destination: output disturbance and input disturbance into the $Goto$ blocks, and the reference signal fed directly to feed-forward controller.

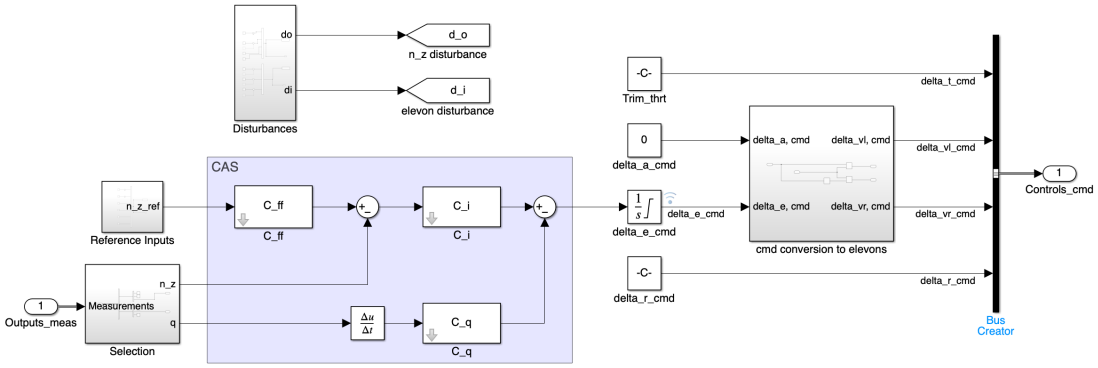


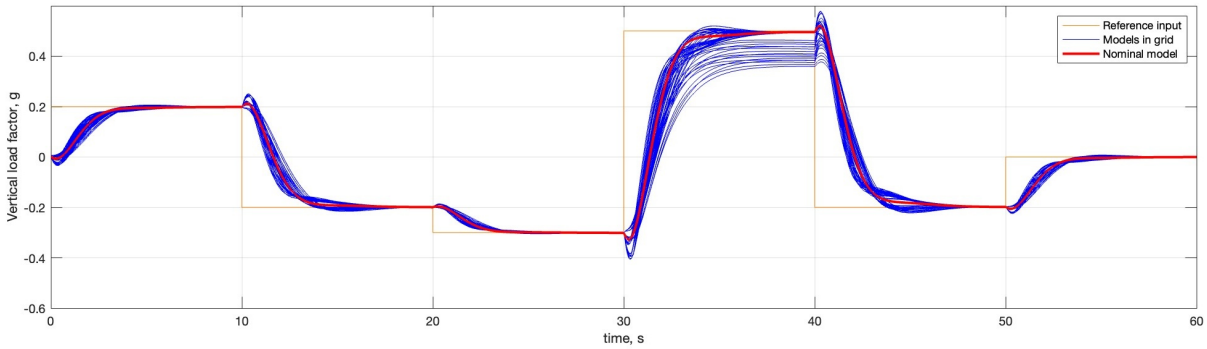
Fig. 19 Expanded flight control system block of the non-linear model

B. Simulation

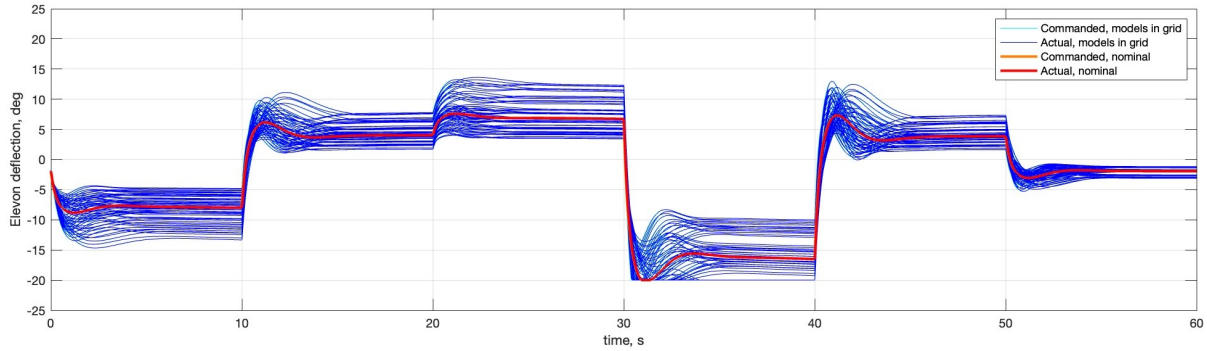
The controller blocks were set to the full-grid variant from Eq. 28. The model was initialized, trimmed, and simulated a total of 65 times to cover the operational window of 4500-5500 m, 0.7-0.8 Mach, 0.25-0.75 fuel fraction, and the 20% variations in the pitch moment partial coefficients. The latter were perturbed by directly multiplying the computed coefficient values with 0.8 or 1.2, depending on the iteration. The simulations were carried out separately for the reference tracking and the disturbances. As before, the nominal model response is highlighted with a thick red line.

The results of the step commands tracking are displayed in Fig 20. The vertical load factor in Fig 20a smoothly tracks the reference value for all models. The exception is for the command at 30 seconds into simulation, where the command was purposely set to a value which is unreachable for some of the models, in order to display the limitations of achievable load factor commands for GHAME vehicle at this flight point. Both commanded and actual elevon deflections are plotted in Fig 20b, where the reached deflection limit is seen for the same reference step at 30 seconds. The respective actuator deflection rates are shown in Fig 20c, they are far from saturating the deflection rate limits of 400 °/s. Lastly, the pitch rate is displayed in Fig 20d. Note that the maximum achievable pitch rate with saturated elevons is only around 1.5 °/s at these conditions.

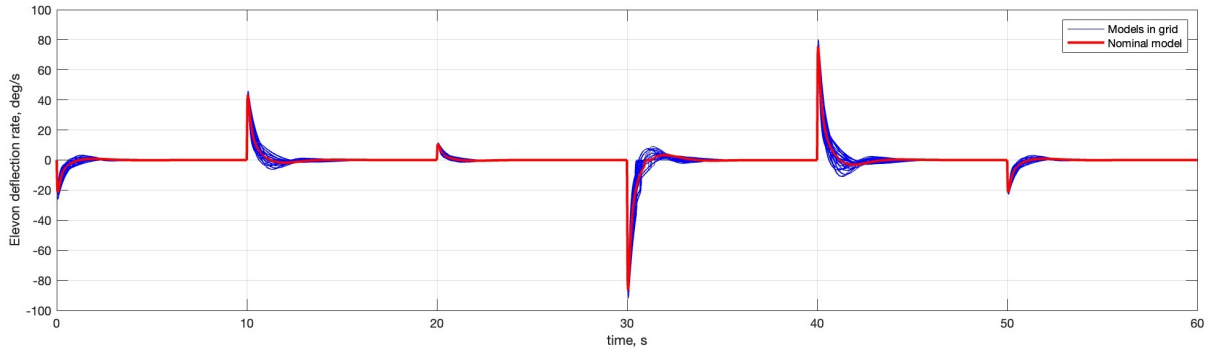
The disturbance rejection responses are shown in Fig 21. The simulation starts with two consecutive input disturbances, then 3 consecutive output disturbances, and then a final joint input-output disturbance which superimposes for n_z. The input disturbance magnitudes can be read from the elevon deflection graph in Fig 21b, where the deflection angle is instantaneously changed to disturbance value. Note how the commanded deflection separates from the actual one due to input disturbances, yet controls the elevons well. The input disturbances are also easily spotted in the deflection rate graph in Fig 21c - they appear as vertical lines that move past the boundaries of deflection rates. That is because disturbance step inputs act instantaneously on the deflection angle. Nevertheless, all disturbances are successfully rejected, which is clearly shown in Fig 21a where the vertical load factor is brought to zero.



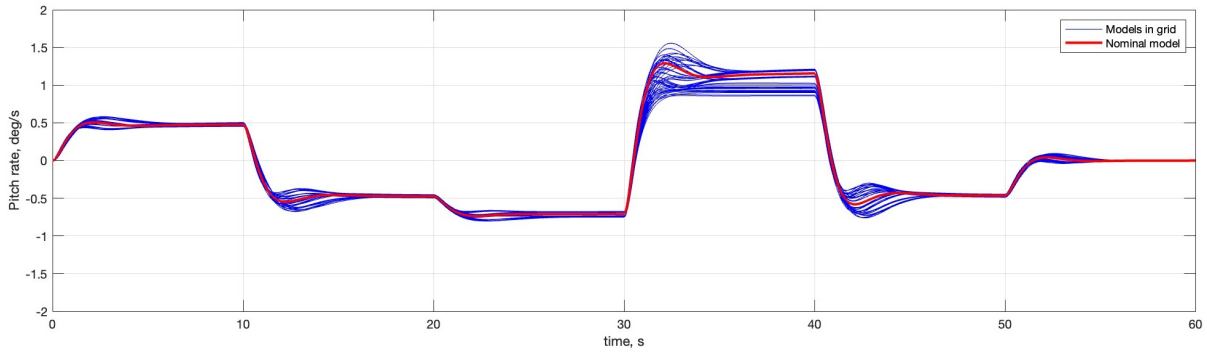
(a) Vertical load factor response



(b) Elevon deflection response

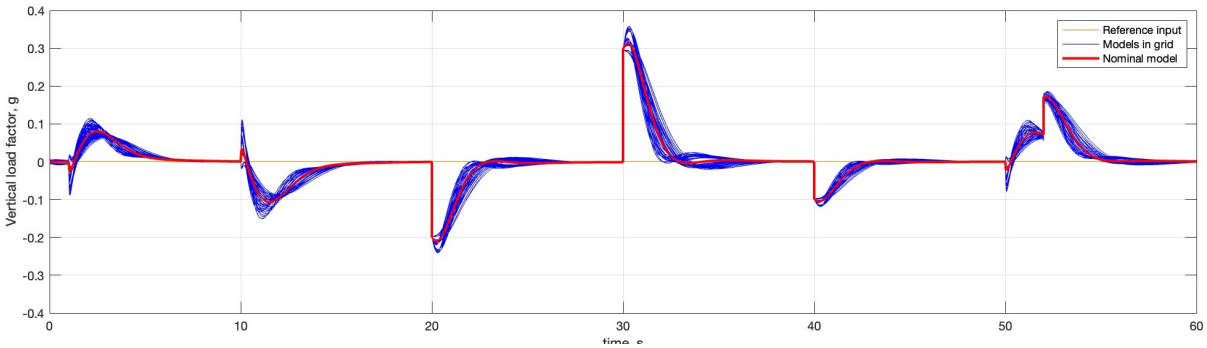


(c) Elevon deflection rate response

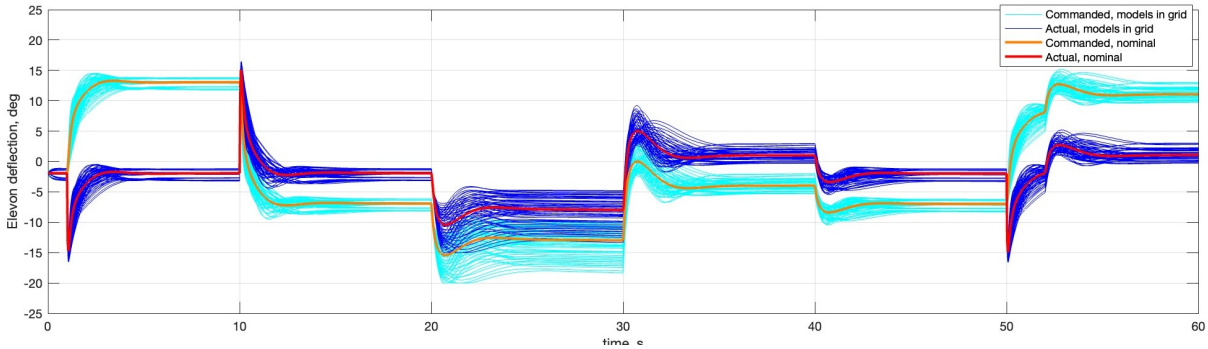


(d) Pitch rate response

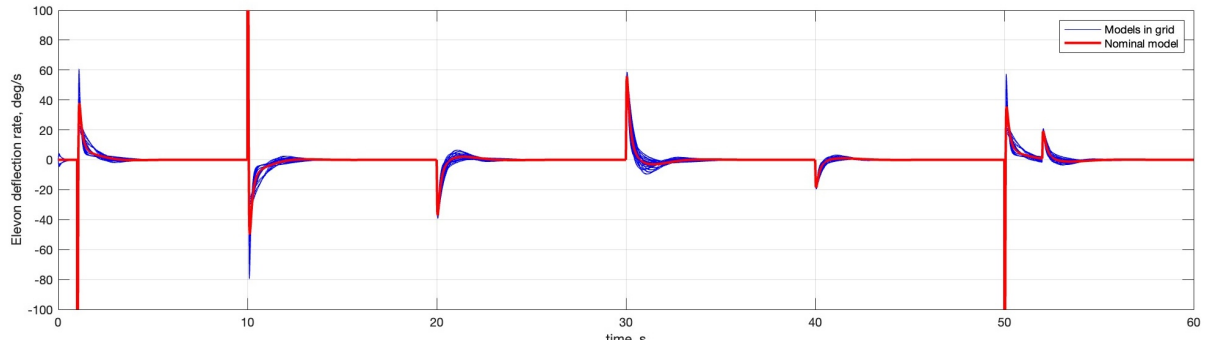
Fig. 20 Non-linear simulation responses to reference load factor commands



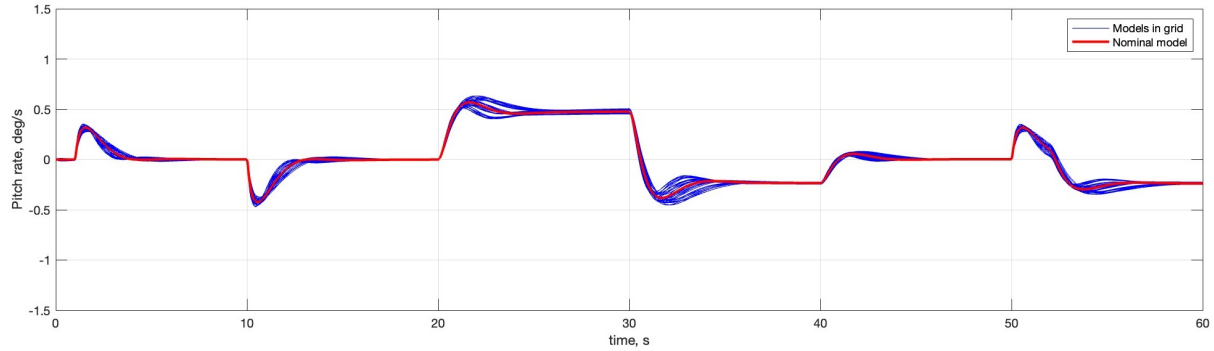
(a) Vertical load factor response



(b) Elevon deflection response



(c) Elevon deflection rate response



(d) Pitch rate response

Fig. 21 Non-linear simulation responses to input and output disturbances

VI. Conclusion and Recommendations

This research focused on developing a robust flight control system for the GHAME hypersonic vehicle's subsonic short-period model under parametric uncertainties. A non-linear GHAME model was integrated into Simulink, with a flexible subsystem-separated structure. During linearization, discrepancies in aerodynamic data were resolved by adjusting interpolation methods for aerodynamic coefficients. The model was trimmed and linearized at 11 operating points, incorporating 20% uncertainty in pitch moment coefficients, and the short-period dynamics were characterized.

A fixed-structure flight control system was synthesized using multi-objective H_∞ mixed-sensitivity methods for disturbance rejection, high-frequency control attenuation, and reference tracking at the nominal flight point. This design was extended to a full grid of uncertain models, and the controller successfully stabilized all sampled models, meeting robustness margin requirements. The results showed effective disturbance rejection at the plant input and output at low frequencies, and smooth control signal roll-off at actuator bandwidth. The FCS was incorporated into the non-linear model and successfully tested for disturbance rejection and reference tracking under parameter variations, demonstrating that a single fixed-structure controller can manage the GHAME model's subsonic dynamics under uncertainty.

Future work should address model limitations and control system enhancements. Incorporating aeroelastic effects, updating engine dynamics, and adding CG and in-flight mass variations are important for hypersonic speeds. It is significant to consider HV-specific scenarios like angle of attack hold during maneuvers for future FCS designs. Additionally, extending the framework to include phugoid and lateral dynamics, adding output disturbances for pitch rate, and incorporating gain and phase margin constraints can improve robustness. Finally, developing gain-scheduled controllers to cover a larger part of flight envelope would be a natural extension of this framework.

Appendix

Table 7 Mechanical and Control Constants

Name	Symbol	Value	Unit
Take-off gross vehicle mass	m_0	136077	kg
Total fuel mass	$m_{f_{\text{tot}}}$	81646	kg
Vehicle mass at burn-out	m_1	54431	kg
Moment of Inertia take-off (I11)	I_{11_0}	1.573×10^6	kg.m^2
Moment of Inertia take-off (I22)	I_{22_0}	31.6×10^6	kg.m^2
Moment of Inertia take-off (I33)	I_{33_0}	32.54×10^6	kg.m^2
Moment of Inertia take-off (I13)	I_{13_0}	0.38×10^6	kg.m^2
Moment of Inertia burn-out (I11)	I_{11_1}	1.18×10^6	kg.m^2
Moment of Inertia burn-out (I22)	I_{22_1}	19.25×10^6	kg.m^2
Moment of Inertia burn-out (I33)	I_{33_1}	20.2×10^6	kg.m^2
Moment of Inertia burn-out (I13)	I_{13_1}	0.38×10^6	kg.m^2
Wing area	S	557.42	m^2
Wing span	b	24.38	m
Wing chord	c	22.86	m
Engine cowl area factor	A_c	27.87	-
IMU position	s_{SB}	24.4	m
Standard gravity	g_0	9.80665	m/s^2
Maximum actuator deflection	$\delta_{v_{\max}}$	±20	°
Maximum actuator deflection rate	$\dot{\delta}_{v_{\max}}$	±400	°/s
Natural frequency of actuator	ω_v	50	rad/s
Damping ratio of actuator	ζ_v	0.7	-

The state-space matrices for the nominal short-period model at nominal trim condition are shown in Eq. 32.

$$A = \begin{bmatrix} -0.9091 & 1 \\ -0.03207 & -1.535 \end{bmatrix},\ B = \begin{bmatrix} -0.01765 \\ -0.1267 \end{bmatrix},\ C = \begin{bmatrix} 22.16 & -3.817 \\ 0 & 1 \end{bmatrix},\ D = \begin{bmatrix} 0.1166 \\ 0 \end{bmatrix} \tag{32}$$

Acknowledgments

The authors would like to thank Prof. Peter H. Zipfel for his contribution to this research and support with the GHAME aerodynamic model implementation.

References

[1] Sziroczak, D., and Smith, H., "A review of design issues specific to hypersonic flight vehicles," *Progress in Aerospace Sciences*, Vol. 84, 2016, pp. 1–28. https://doi.org/10.1016/j.paerosci.2016.04.001.

[2] Bolender, M. A., "An Overview on Dynamics and Controls Modelling of Hypersonic Vehicles," , 2009.

[3] Erbland, P. J., "Current and Near-Term RLV/Hypersonic Vehicle Programs," , 2004.

[4] Fidan, B., Mirmirani, M., and Ioannou, P., "Flight dynamics and control of air-breathing hypersonic vehicles: review and new directions," *12th AIAA international space planes and hypersonic systems and technologies*, 2003, p. 7081.

[5] Calise, A. J., "Research In Robust Control For Hypersonic Aircraft," Report, 1993.

[6] Vu, P., and Biezad, D. J., "Longitudinal Control of Hypersonic Aircraft: an Alpha Follow-Up Scheme," , 1993. https://doi.org/10.1109/AEROCS.1993.720973.

[7] Gregory, I., McMinn, J., Shaughnessy, J., and Chowdhry, R., "Hypersonic vehicle control law development using H infinity and mu-synthesis," *AIAA 4th International Aerospace Planes Conference*, 1992. https://doi.org/https://doi.org/10.2514/6.1992-5010.

[8] Vu, P., and Biezad, D. J., "A pseudo-loop design strategy for the longitudinal control of hypersonic aircraft," , 1993. https://doi.org/10.2514/6.1993-3814.

[9] Shaughnessy, J. D., Pinckney, S. Z., McMinn, J. D., Cruz, C. I., and Kelley, M.-L., "Hypersonic Vehicle Simulation Model: Winged-Cone Configuration," Report, 1990.

[10] White, D., and Sofge, D., *Handbook of Intelligent Control*, Van Nostrand Reinhold, 1992. Ch. 11.

[11] Bolender, M., and Doman, D., "A Non-Linear Model for the Longitudinal Dynamics of a Hypersonic Air-breathing Vehicle," , 2005. https://doi.org/10.2514/6.2005-6255.

[12] Parker, J. T., Serrani, A., Yurkovich, S., Bolender, M. A., and Doman, D. B., "Control-Oriented Modeling of an Air-Breathing Hypersonic Vehicle," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 3, 2007, pp. 856–869. https://doi.org/10.2514/1.27830.

[13] Groves, K., "Modelling, simulation, and control design of an air-breathing hypersonic vehicle," Thesis, 2005. URL http://rave.ohiolink.edu/etdc/view?acc_num=osu1302726196.

[14] Groves, K., Sigthorsson, D., Serrani, A., Yurkovich, S., Bolender, M., and Doman, D., "Reference Command Tracking for a Linearized Model of an Air-Breathing Hypersonic Vehicle," , 2005. https://doi.org/10.2514/6.2005-6144.

[15] Sigthorsson, D. O., Jankovsky, P., Serrani, A., Yurkovich, S., Bolender, M. A., and Doman, D. B., "Robust Linear Output Feedback Control of an Airbreathing Hypersonic Vehicle," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 4, 2008, pp. 1052–1066. https://doi.org/10.2514/1.32300.

[16] Anderson, M., Emami-Naeini, A., and Vincent, J., "Robust Cotrol Law Development for a Hypersonic Cruise Aircraft," *1991 American Control Conference*, IEEE, 1991, pp. 839–845. https://doi.org/10.23919/ACC.1991.4791490.

[17] Buschek, H., and Calise, A., "Fixed order robust control design for hypersonic vehicles," *Guidance, Navigation, and Control Conference*, AIAA, 1994, p. 3662. https://doi.org/10.2514/6.1994-3662.

[18] Buschek, H., and Calise, A. J., "Robust Control of Hypersonic Vehicles Considering Propulsive and Aeroelastic Effects," , 1993.

[19] Buschek, H., and Calise, A., "Uncertainty Modeling and Fixed-Order Controller Design for a Hypersonic Vehicle Model," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, 1997, pp. 42–48. https://doi.org/10.2514/2.4031.

[20] Duan, H., and Li, P., "Progress in control approaches for hypersonic vehicle," *Science China Technological Sciences*, Vol. 55, 2012, p. 2965–2970. https://doi.org/https://doi.org/10.1007/s11431-012-5036-x.

[21] Mirmirani, M., Wu, C., Clark, A., Choi, S. B., and Colgren, R., "Modeling for control of a generic airbreathing hypersonic vehicle," *AIAA guidance, navigation, and control conference and exhibit*, 2005, p. 6256. https://doi.org/10.2514/6.2005-6256.

[22] Apkarian, P., and Noll, D., "The H Control Problem is Solved," *Aerospace Lab*, 2017. https://doi.org/10.12762/2017.AL13-01.

[23] Apkarian, P., and Noll, D., "Nonsmooth H-infinity synthesis," *IEEE Transactions on Automatic Control*, Vol. 51, No. 2, 2006, pp. 382–382.

[24] Apkarian, P., and Noll, D., "Nonsmooth optimization for multidisk H synthesis," *European Journal of Control*, Vol. 12, No. 3, 2006, pp. 229–244.

[25] De Aguiar, R. S. D. S., Apkarian, P., and Noll, D., "Structured Robust Control Against Mixed Uncertainty," *IEEE Transactions on Control Systems Technology*, Vol. 26, No. 5, 2018, pp. 1771–1781. https://doi.org/10.1109/TCST.2017.2723864.

[26] MathWorks, "Robust Control Toolbox, version: 23.2 (R2023b)," , 2022. URL https://www.mathworks.com.

[27] Echols, J. A., Puttannaiah, K., Mondal, K., and Rodriguez, A., "Fundamental Control System Design Issues for Scramjet-Powered Hypersonic Vehicles," , 2015. https://doi.org/10.2514/6.2015-1760.

[28] Wu, N., and Yu, J., "Robust Controller Design of Hypersonic Vehicle in Uncertainty Models," Conference paper, 2018.

[29] Zhu, Y., Shen, H., Liu, Y., Zhang, G., and Lu, Y., "Optimal control and analysis for aero-elastic model of hypersonic vehicle," *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, 2016, pp. 1911–1915. https://doi.org/10.1109/CGNCC.2016.7829081.

[30] Wenbiao, Z., Dong, L., Yong, M., and Yicheng, C., "Method of velocity controller design for an airbreathing hypersonic cruise vehicle," *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*, 2014, pp. 1817–1821. https://doi.org/10.1109/CGNCC.2014.7007458.

[31] Williams, J., and Vukelich, S., "The USAF stability and control digital DATCOM. Volume I. Users manual," , 1979.

[32] Alsuwian, T. M., Ordonez, R., and Jacobsen, L., "Adaptive Control for Longitudinal Dynamics of Hypersonic Vehicle at Subsonic Speeds," , 2017. https://doi.org/10.2514/6.2017-4009.

[33] MathWorks, "MATLAB, version: R2023b," , 2023. URL https://www.mathworks.com.

[34] MathWorks, "Simulink® Control Design, version: 23.2 (R2023b)," , 2022. URL https://www.mathworks.com.

[35] Zipfel, P. H., *Modeling and Simulation of Aerospace Vehicle Dynamics*, 2nd ed., AIAA, 2000.

[36] Araki, J. J., "Reentry Dynamics and Handling Qualities of a Generic Hypersonic Vehicle," Thesis, 1992.

[37] Stevens, B. L., Lewis, F. L., and Johnson, E. N., *Aircraft control and simulation - dynamics, controls design, and autonomous systems*, John Wiley & Sons, 2015.

[38] Bates, D., and Postlethwaite, I., *Robust Multivariable Control of Aerospace Systems*, Vol. 8, IOS Press, 2002.

[39] Skogestad, S., and Postlethwaite, I., *Multivariable Feedback Control: Analysis and Design*, John Wiley & Sons, 2005.

[40] Choi, J., "Application of Hypersonic Vehicle Flying Qualities Criteria and Computational Considerations," Thesis, 1994.

[41] Seiler, P., Packard, A., and Gahinet, P., "An introduction to disk margins," *IEEE Control Systems Magazine*, Vol. 40, No. 5, 2020, pp. 78–95.

[42] Kaminer, I., Pascoal, A. M., Khargonekar, P. P., and Coleman, E. E., "A velocity algorithm for the implementation of gain-scheduled controllers," *Automatica*, Vol. 31, No. 8, 1995, pp. 1185–1191.

4

Framework details

4.1. Modelling

The GHAME model implementation into MATLAB and Simulink needs additional clarifications in terms of its working principles and nuances. The directory folder consists of a total of 10 main files and one more folder, with the following general overview:

1. **Init_Airframe.m**: model initialization function which takes airspeed, altitude and fuel fraction in the tank as inputs and initializes the non-linear model at that point. It contains the aerodynamic tables, airframe constants such as take-off and dry MOI, computations such as mass based on fuel fraction in the tank, and the uncertainty values of the parameters. It outputs two structure variables - $Data$ with initialized and computed constants, and $Unertain$ with the uncertainty variables.
2. **Non_Linear.slx**: the full non-linear model of GHAME with autopilot and disturbances. Note that in its latest version it requires not only file 1 at some flight point, but also the controller gains loaded into the workspace.
3. **Airframe_uncertain.slx**: used to trim and linearize the airframe. Generally speaking, a copy of the non-linear model without the actuators and autopilot. Some of the subsystems were restructured completely to include the parametric uncertainty blocks.
4. **Engine_uncertain.slx**: the propulsion system block which is used to trim and linearize the engine. Slightly redesigned to include uncertainty and to use parameters from the workspace, since there is no airframe connected to it to compute them.
5. **Trim_Linearize.m**: a function which trims the airframe and engine from files 3 and 4, linearizes them, and outputs the uncertain state-space short-period, full-longitudinal, and lateral models of the airframe, and the uss model of the engine, as well as the trimmed parameters.
6. **Main_Modelling.m**: the master script that sets an initial flight point, calls the *Init_Airframe.m* function, and then function 5 to compute the linear state-space models. It then resets the flight point initial conditions from function 1 to the trimmed ones. Basically, all that is needed to trim the non-linear model and obtain the linear systems at some flight point is to run this file. Note that without opening the files 3 and 4 it can throw an error.
7. **Cm_CL_CD.m**: a standalone file which calculates and plots various interpretations of the named coefficients. It was used to analyze the discrepancies in the aerodynamic model.
8. **coef_check.m**: a script which analytically linearizes the equations of motion at the same flight point as the trimmed one to check that short-period model is linearized correctly. Requires a run of file 5, but the flight point has to be manually set again.
9. **Simulate.m**: sets custom steps for reference tracking or disturbances to simulate in the *Non_Linear.slx* model. Beware that the script is written specifically to simulate over a range of operating conditions, including explicit aerodynamic coefficients variation. For that, it trims the

models 3 and 4 and runs simulation of file 2 with the specified parameters. The process repeats for parameter values in the domain, so it takes about 15 minutes to run.

10. **Trim.m**: used by *Simulate.m* to trim the models, since no linearization is needed here. It is a direct snippet from *Trim_Linearize.m* function.
11. **Controller**: a folder containing the controller synthesis files and their *.mat* outputs.
    - **SigBased_sp_.m**: the main script for controller synthesis.
    - **SigBased_sp.slx**: the Simulink file used by the script above containing the FCS layout and the linear models.
    - **Compute_ReferenceModelParameters.m**: a partial function of the optimization process to compute optimal ω and ζ for reference tracking. It calculates the weighted sum of the squared errors.

To summarize, only the files 1-6 are needed to fully use the GHAME model. Files 7-8 are supplementary and used for analysis, whereas files 9-10 are needed for FCS test in non-linear simulation, but are specifically written to address certain conditions. The controller synthesis files have a flexible and clear framework that can be utilized for future designs with $systune$. The files outlined above will be elaborated on further in this chapter when necessary.

4.1.1. Further on GHAME aerodynamic data

Although the encountered problem with GHAME aerodynamic model has been discussed in Section III of chapter 3, there is some additional information which was not included there due to page limit. Some of it may clarify the problem further.

As has been mentioned, the slopes in the C_{m_α} aerodynamic table contradict the locations of index points when combined with respective C_{m_0} values. Both tables can be found in Appendix A in Table A.1, A.2, A.3, and A.4. The problem is best illustrated by a simple schematic shown in Figure 4.1. The dark-red dots represent the C_m values calculated at the α indices from the aforementioned tables, and the red lines represent the slopes at those indices as indicated by the C_{m_α} table. However, the slopes clearly contradict the locations of C_m at the indices: there are instances in the data where the slope is stated as negative, but the next data point is located higher. This leads to the ”wavy” shape of pitch moment coefficient slope when interpolating each table directly, which was shown before in chapter 3.

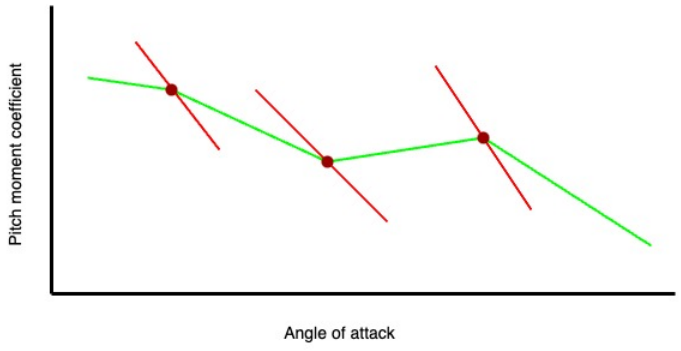


Figure 4.1: Pitch moment coefficient slope discrepancy schematic

There were 3 initial solutions proposed. The first was to treat the given data as a set of linear functions and interpret the slope to start at the α index, while the second was to put the slope line in the middle of each index. The former is named ”forward difference” and the latter is ”central difference”, due to their resemblance of differentiation methods. The third option was to compute the C_m values at the respective indices, connect them with straight lines and then interpolate along those lines. All 3 options are shown together in Figure 4.2. Note that the straight lines in both ”difference” interpretations directly correspond to the C_{m_α} values from the aerodynamic table. Similar figures for lift and drag coefficients can be found in Figure A.1 and Figure A.2, respectively.

The problem with the forward difference are its apparently long slope lines, and problem with the central

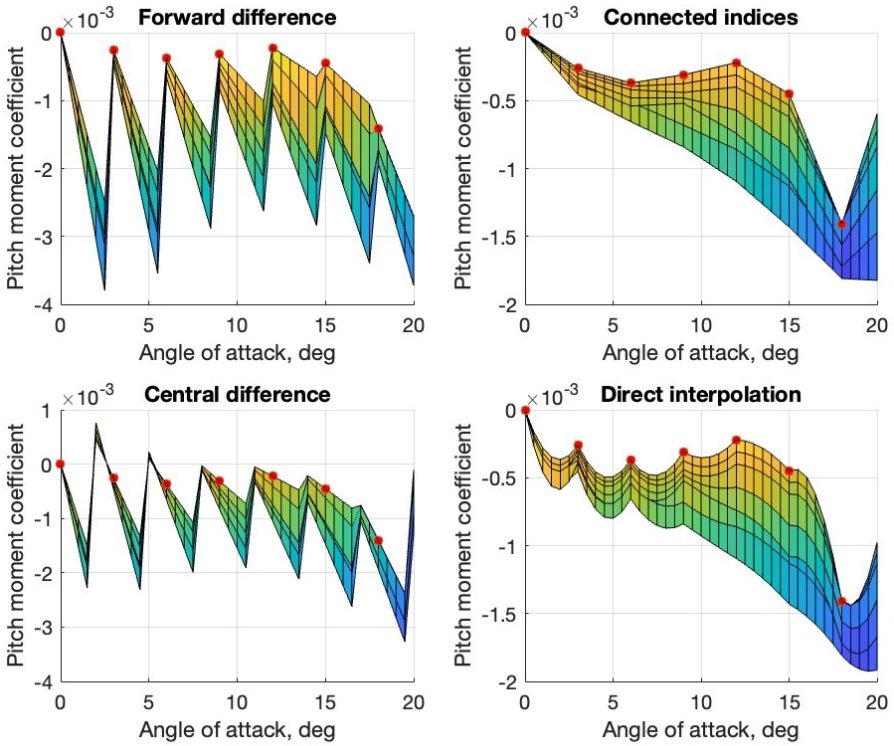
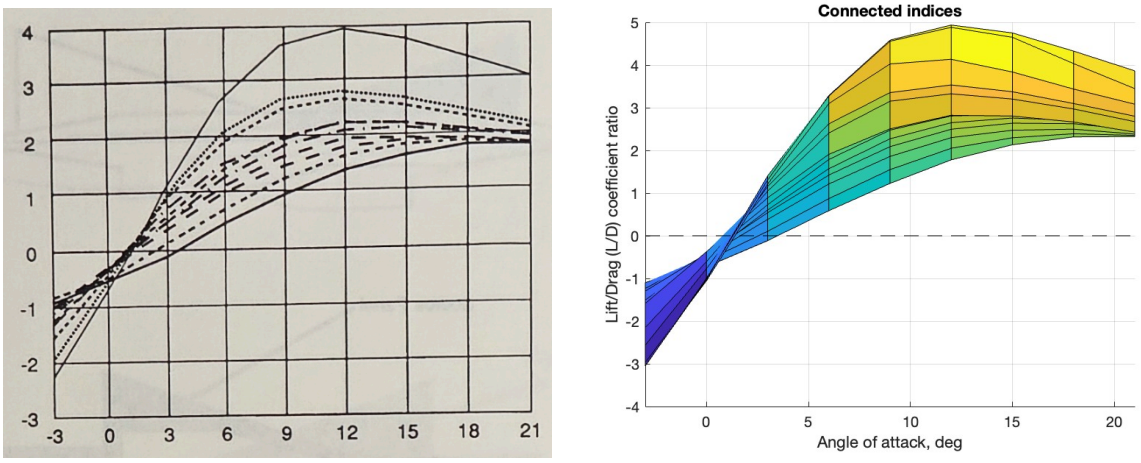


Figure 4.2: Various interpretations of pitch moment coefficient data

difference are its positive moment coefficient values, which are absent in the original data. Furthermore, the ”saw” shapes of both of them are not realistic, and can thus only work near the angle of attack indices. The problem with the connected indices is that it still has a positive slope, which directly contradicts the aerodynamic table data of C_{m_α}. Nevertheless, the connected indices method was selected, as it resembles the realistic aerodynamic data most. Additional confirmation was found using the original publication document of GHAME [23].



(a) Lift to drag ratio in original document of GHAME [23]

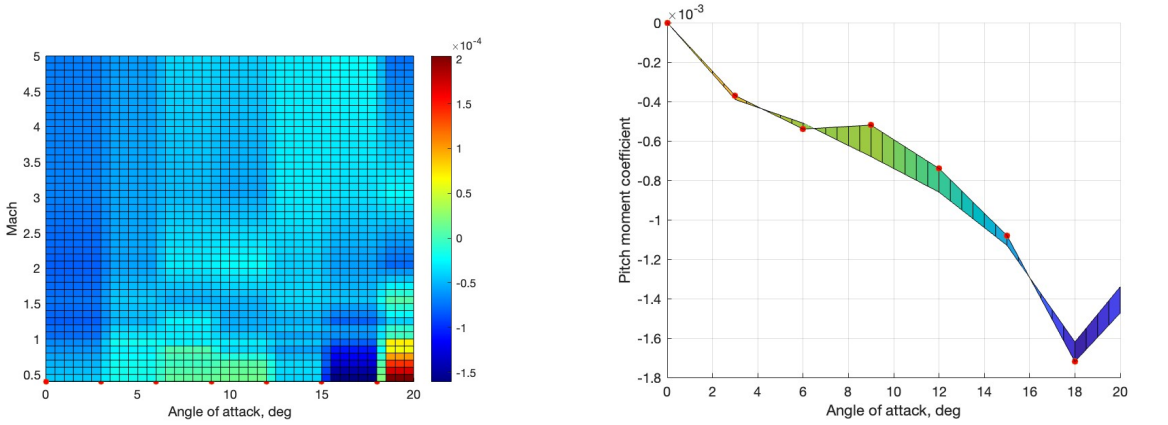
(b) Lift to drag ratio computed with connected indices method

Figure 4.3: Comparison of original and computed lift-to-drag ratio graphs against angle of attack for all Mach

In the document, there is a graph of a lift-to-drag (L/D) ratio versus the angle of attack, where the indices of α are connected with straight lines, as shown in Figure 4.3a. Apparently, when computing the L/D

ratio using the connected indices method, it produces a similar result, shown in Figure 4.3b. Therefore, the connected indices method was confirmed, and the previously computed C_{m_1}, C_{L_1} and C_{D_1} were used, while the C_{m_0}, C_{m_α}, C_{L_0}, C_{L_α}, C_{D_0}, and C_{D_α} were neglected onwards.

The new pitch moment coefficient positive slope only happens in the middle α, low Mach domain. Specifically, the C_{m_1} slope becomes negative across the entire domain after Mach 0.8. This is illustrated in Figure 4.4a. Note the warmer colors only at the bottom - corresponding to positive slope at low mach numbers. The side view of the transition is shown in Figure 4.4b. Thus, the new C_{m_α} is always negative for Mach > 0.8. The table for the updated pitch moment coefficient C_{m_1} can be found in Table A.5 and A.6, whereas the complete graphs for C_{L_1}, C_{D_1}, and C_{m_1} are in Figure A.3, all in Appendix A.



(a) Slope of pitch moment coefficient wrt angle of attack, connected indices

(b) Pitch moment coefficient transition from positive slope to always negative slope at Mach 0.8-0.9

Figure 4.4: Pitch moment coefficient slope overview across data domain

All of the charts and methods presented above can be plotted with file 7 of the directory. In the script, there is an initialization at a dummy flight point to run function 1 in order to retrieve the tables. The desired Mach and α ranges for the plots can be set at the very top, along with desired sample steps.

4.1.2. Simulink non-linear model

The overall layout of the GHAME non-linear Simulink model was presented earlier in the final section of the scientific article, but is displayed again in Figure 4.5.

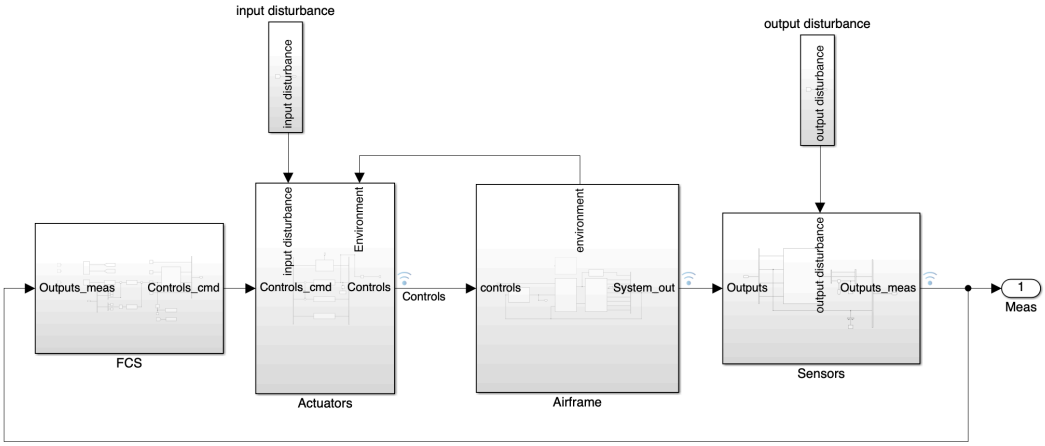


Figure 4.5: General layout of the full non-linear GHAME model in Simulink

Diving deeper into each subsystem from left to right, the *FCS* block, expanded in Figure 4.6, has the same structure as in the scientific article.The steps are set by the *Simulate.m* file from the directory. It

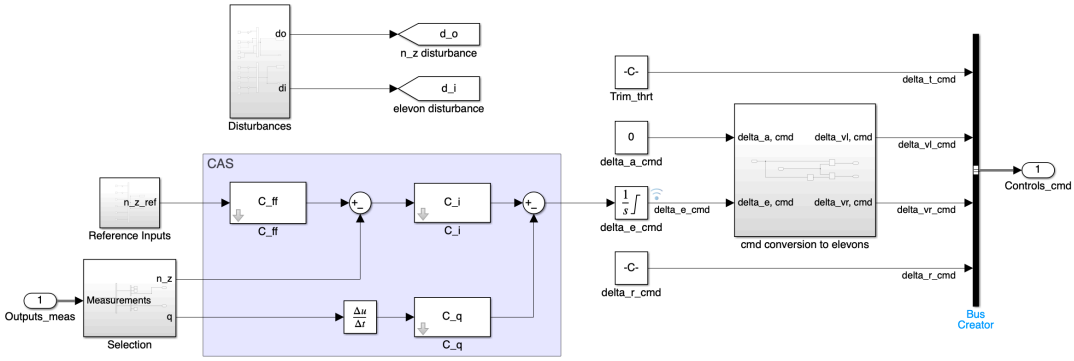


Figure 4.6: *FCS* block of the non-linear model

works the following way: the nominal flight point is simulated first, and then all the combinations of the parameter variations are simulated in a for loop. Beware that simulation time variable $sim\_seconds$ must be set to the same value as the simulation duration of the Non_Linear.slx file.

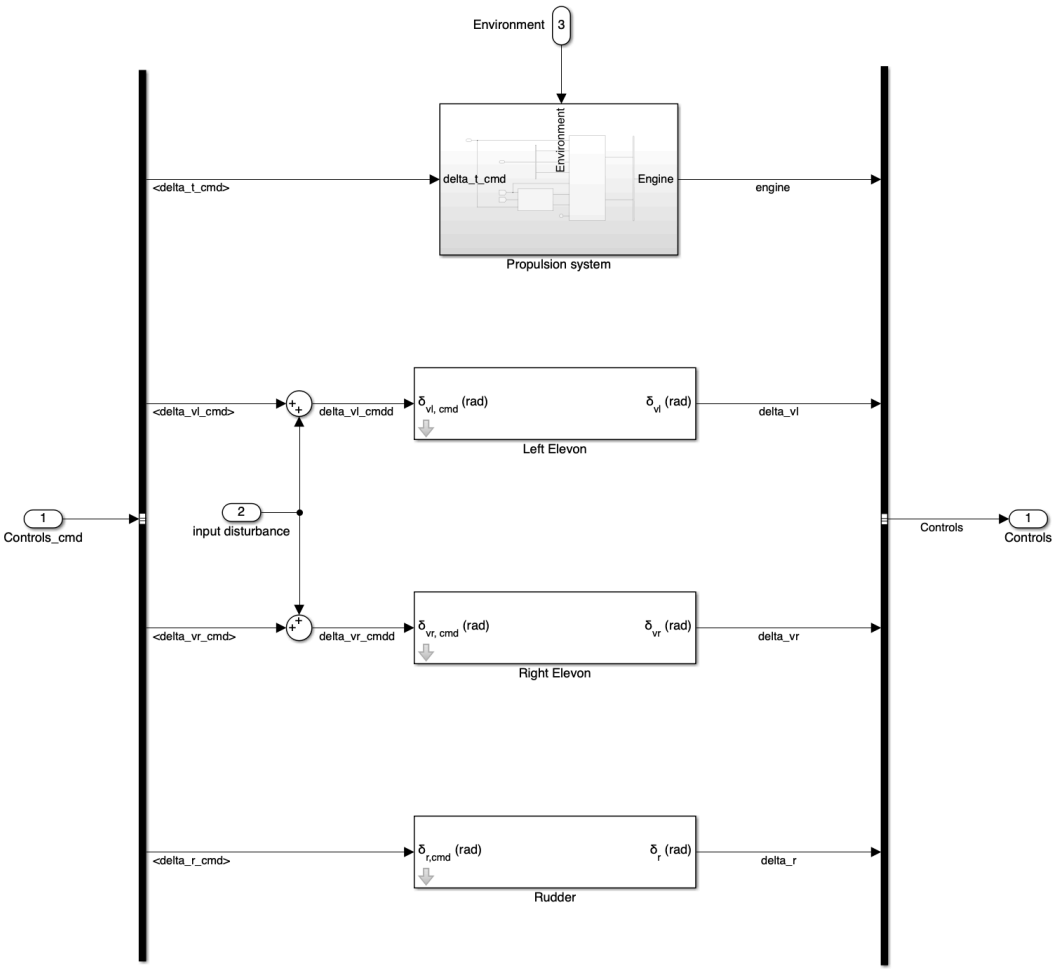


Figure 4.7: *Actuators* block of the non-linear model

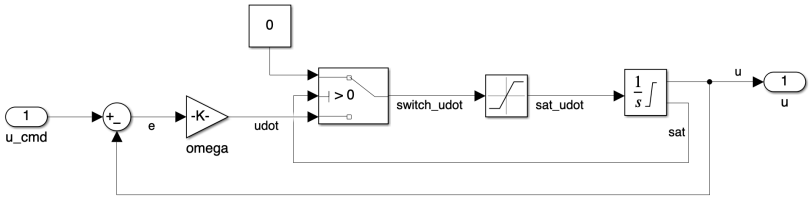


Figure 4.8: Elevon model from the *Actuators* block

The commanded actuator deflections are fed into the *Actuators* block, the contents of which are shown in Figure 4.7. The actuators consist of the propulsion system and 3 tailfins: rudder and two elevons. The tailfins have exactly the same parameters and are all modelled just as regular ailerons or elevons. The expanded elevon block is shown in Figure 4.8. The error between the commanded and actual elevon deflection is multiplied with *omega* gain, which corresponds to the natural frequency of the actuator ω_a. It is then sent through a switch that stops deflection rate when the angle is saturated. The deflection rate is checked for saturation limits $\dot{\delta}_{v_{max}}$, and then integrated with angle saturation limits $\delta_{v_{max}}$. All three parameters can be found in the appendix of the scientific article chapter 3.

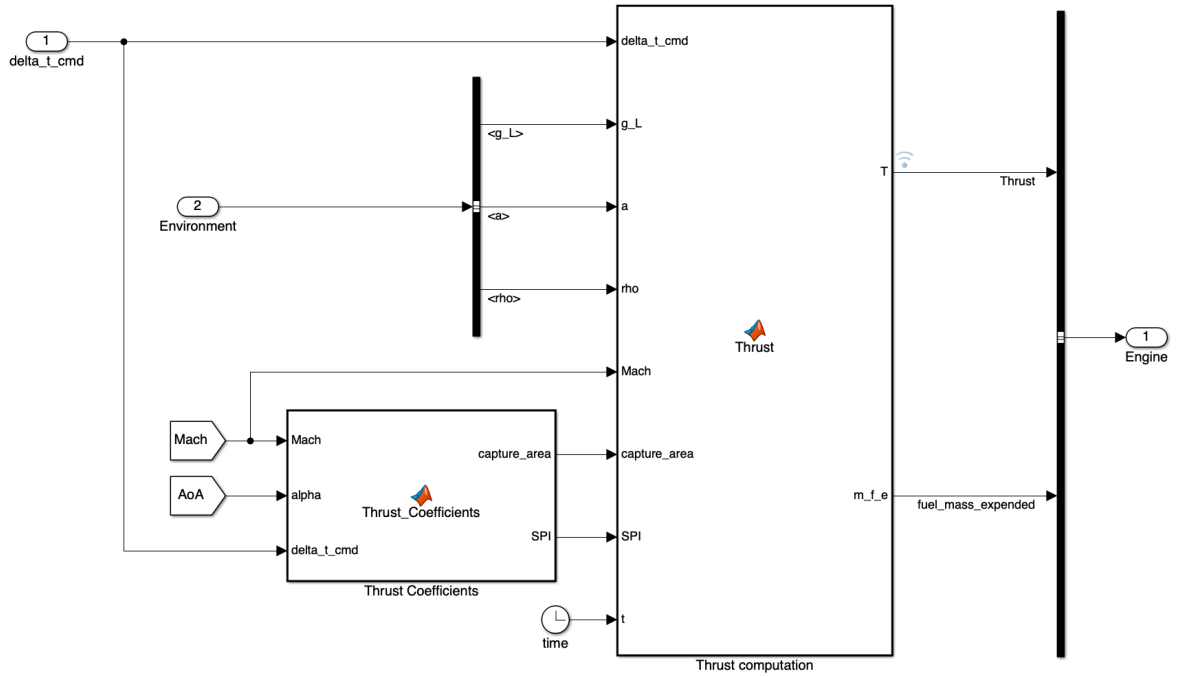


Figure 4.9: Propulsion system model from the $Actuators$ block

The propulsion system is modelled exactly as described earlier, with a layout shown in Figure 4.9. The aerodynamic tables of capture area and specific impulse are interpolated and then directly multiplied with the other parameters in the *Thrust computation* block. There is, however, a detail which was left out in the previous description. The GHAME model includes mass and MOI variation calculated from the expended fuel mass. For that purpose a clock is also fed into the block, and a second computation takes place. The fuel mass flow is calculated with [23]:

$$\dot{m_f} = \frac{f_P}{I_{sp} g_0} \tag{4.1}$$

Where f_P is thrust, I_{sp} is specific impulse, and g_0 is gravity. The expended fuel mass is then simply $m_{f_e} = \dot{m_f} t$. Both thrust and expended fuel mass are combined into an $Engine$ bus, which is then added to the main actuator bus in Figure 4.7.

The control signals from the *Actuators* block are fed directly into the *Airframe* from Figure 4.5. The *Airframe* block is expanded in Figure 4.10. In the airframe, the control signals are currently fed only into the *Aerodynamics* subsystem, but can also be fed into the *MOI computation* block. That block retrieves the previously computed m_{f_e} and recalculates the MOI of the vehicle. The mass and MOI variations during simulation were assumed to be zero for this research (they are preset with fuel fraction during initialization), but can be used for the future FCS designs or other purposes.

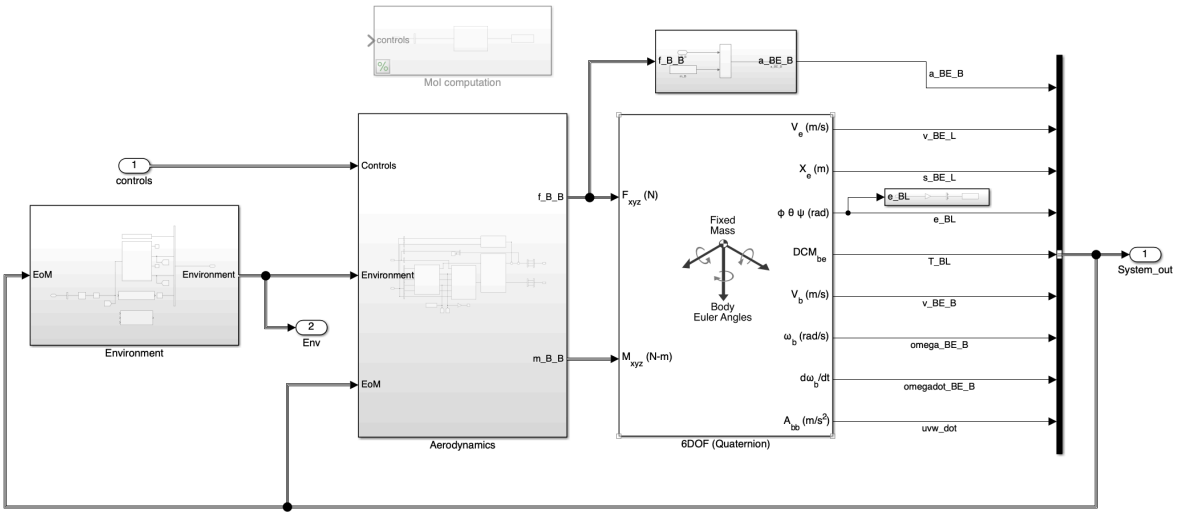


Figure 4.10: *Airframe* subsystem of the non-linear model

The *Environment* block takes the active z coordinate (altitude) from the EoM to compute the speed of sound and air density based on International Standard Atmosphere. It also provides a wind simulation capability by setting a wind vector $[v_A^E]^L$ in local coordinate system (flat Earth-fixed), which is now set to zero (no wind). It also retrieves the standard gravity g_0 from the workspace and puts it on a bus with other signals. The bus is then fed into the *Aerodynamics* block, as well as to output 2 that feeds it back into the propulsion system.

The *Aerodynamics* block is another major subsystem, its contents are shown Figure 4.11. Here, the actuator deflections, thrust, and the environment variables are all used in tandem with some outputs of the EoM to calculate the forces and moments acting on the vehicle. The *Air Data Computer* (ADC) block has a function of performing some basic kinematic calculations. The wind velocity vector $[v_A^E]^L$ is subtracted from vehicle velocity $[v_B^E]^L$ resulting in the airspeed vector in L frame. It is then matrix-multiplied with the DCM $[T]^{BL}$ to produce the airspeed vector in body coordinates $[v_B^A]^B$. Some basic relations are then used to compute the total airspeed and Mach number, whereas α and β are calculated with Equation 2.13 from subsection 2.2.1.

The *Aerodynamic coefficients* block contains a small function which converts the elevon deflections to virtual aileron and elevator deflections, and a large function which interpolates the partial aerodynamic coefficients and computes the respective total aerodynamic coefficients in body coordinates. The conversion from C_L, C_D to C_X, C_Z also happens there. Additionally, there is a small change to the calculation of C_m to include the parametric uncertainty, which follows the formula:

$$C_m = C^u_{m_1} C_{m_1} + C^u_{m_{\delta_e}} C_{m_{\delta_e}} \delta_e + C^u_{m_q} C_{m_q} \frac{qc}{2V} \tag{4.2}$$

The difference between the nominal case and the alternative are the three uncertain variables $C^u_{m_1}$, $C^u_{m_{\delta_e}}$, $C^u_{m_q}$ which are used to perturb their respective partial coefficients for the non-linear simulation. They are part of the $Uncertain$ structure variable initialized in *Init_Airframe.m*, and are all set to 1 by default (no perturbation). That way, the model functions nominally until explicit intervention into the workspace. This structure is used by *Simulate.m* file which alternates them between 0.8 and 1.2 to deliberately perturb the partial C_m coefficients by 20% for the FCS non-linear test.

Finally, the last two blocks of the *Aerodynamics* subsystem are used to calculate the forces and moments according to Equation 2.21 and Equation 2.22. Note that gravity vector is multiplied with DCM

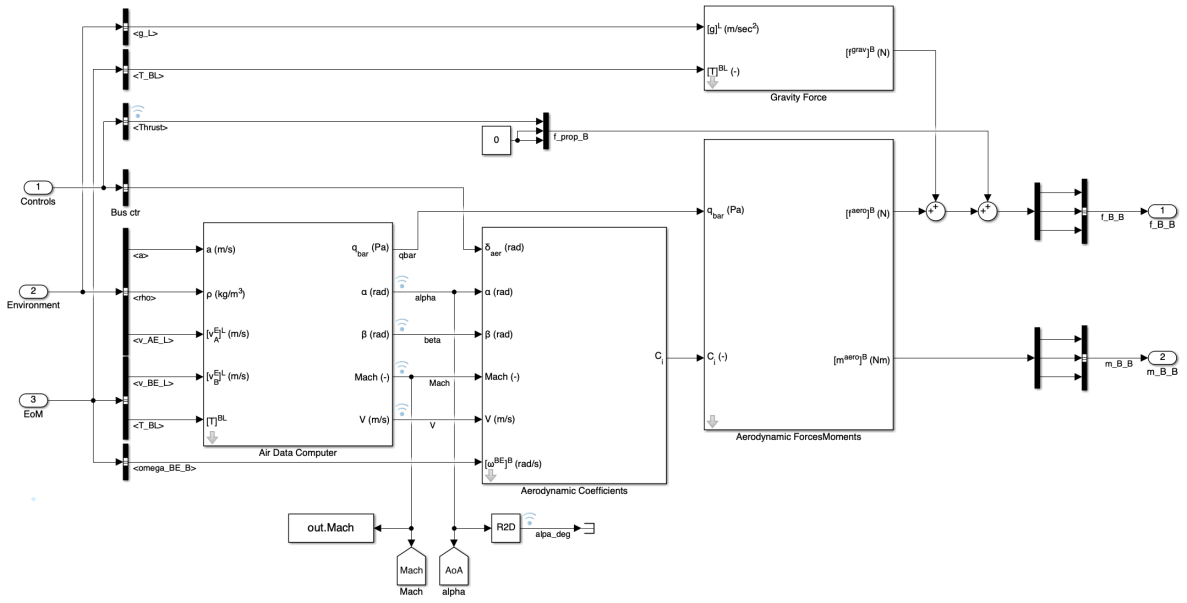


Figure 4.11: *Aerodynamics* subsystem of the *Airframe* block

to transform it to the body reference frame.

Going back to Figure 4.10, the forces and moments are fed into the *EoM* block from the Aerospace Blockset of Simulink. It is a predefined block which uses standard flat-Earth non-linear EoM presented earlier in subsection 2.2.1. Its particular advantage apart from its certified verification is the ability to quickly switch between quaternion and Euler angle DCM computations, which were discussed in the same subsection 2.2.1. A small subsystem at the top, however, is needed to calculate the accelerations in body frame without effects of rotational rates. These accelerations are used for the IMU load factor computations later, for which the Grubin transformation is applied separately accounting for the IMU location. The *EoM* block computes body accelerations only together with the rotational rates.

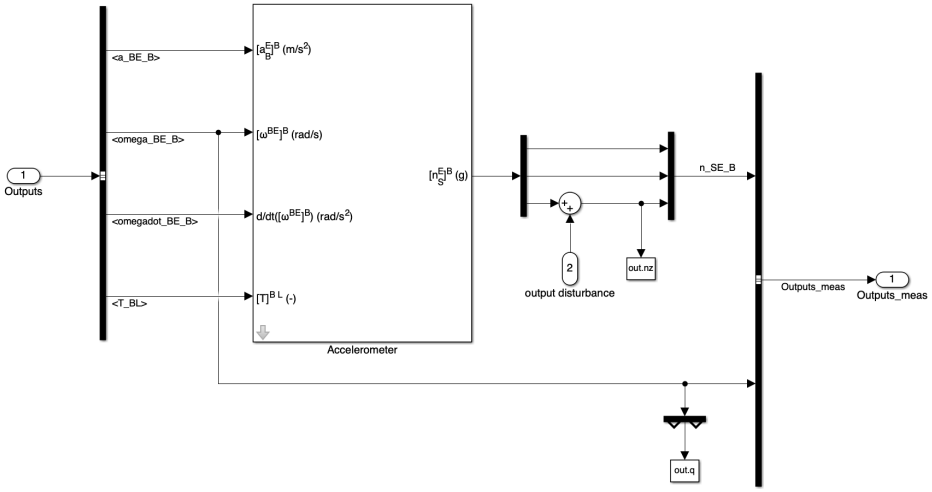


Figure 4.12: *Sensors* block of the non-linear model

At last, the *Sensors* block from Figure 4.5 is expanded in Figure 4.12. The sensor noise modelling is not included in the scope of this research, so this subsystem essentially contains only load factor computation at the IMU location. These calculations were already discussed in detail in Section III of

chapter 3. However, there is a small detail. The load factor full equation in that block actually contains a gravity term at the end:

$$[a_S^E]^B = [a_B^E]^B + \left[\Omega^{BE}\right]^B \left[\Omega^{BE}\right]^B [s_{SB}]^B + \left[\dot{\Omega}^{BE}\right]^B [s_{SB}]^B - [T]^{BL} g^L \tag{4.3}$$

That way the IMU can calculate the normal acceleration including the gravity. However, for the FCS design it has to be force-specific, so the last term was commented out.

4.1.3. Uncertainty, Trimming, and Linearization

As has been mentioned multiple times, to obtain the uncertain state-space systems at some operating points, two more Simulink files were created: *Airframe_uncertain.slx* and *Engine_uncertain.slx* (refer to the file directory in the beginning of this chapter). *Main_Modelling.m* calls the initialization function *Init_Airframe.m* and then launches *Trim_Linearize.m*, which uses these two Simulink models. The full non-linear model is not a part of this process.

Airframe

The *Airframe_uncertain.slx* model is structurally the same as the non-linear model, but without the actuators and FCS. There are major differences in three of its subsystem blocks, however. First of all, the aerodynamic coefficients are now computed with uncertainty, the way it was stated in chapter 3. A general schematic of uncertainty implementation on C_m is presented in Figure 4.13, where $C_{m_{\delta_e}}$ is not shown for simplicity. Displaying the *Aerodynamic coefficients* subsystem again in Figure 4.14 for comprehension, the two uncertainty blocks can now be assessed.

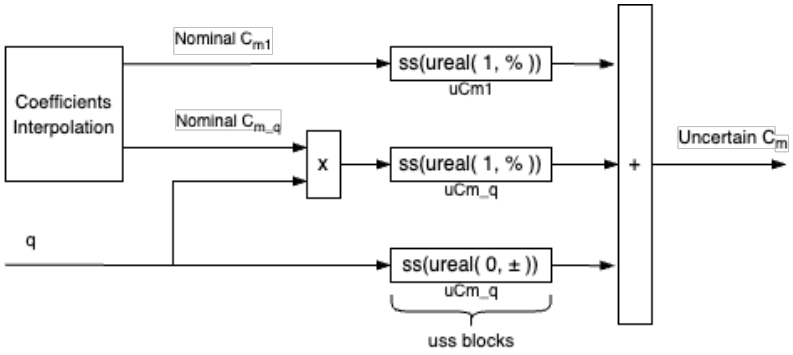


Figure 4.13: A simplified schematic of uncertainty implementation into pitch moment coefficient

Important note concerning limitations of uncertainty embedment. Back in chapter 3 it was discussed that the real parametric uncertainty was implemented into the model in two variants - in terms of percentage and absolute value. It was stated that the latter can not be applied on parameters which are both continuously computed and not multiplied with some continuous signal (e.g, pitch rate). However, there is also a limitation specific to the percentage case. If a percentage uncertainty variable, say uCm_q (in MATLAB notation), is multiplied with the nominal computed coefficient Cm_q first, and the result is then later multiplied with the pitch rate q (as during the total C_m computation), the uncertain parameter will not be detected during linearization. The reason is that during trimming some signals are set to be steady-state zero, such as rotational rates. Because of that, the uncertain variables are disregarded by the program. The work-around this problem is to pre-multiply the nominal coefficients with their respective signals first, e.g., $Cm\_q * q$, and only then multiply the result by the uncertainty variables (uCm_q in this case). Mathematically it leads to the same result, but conceptually it can be perceived as % variation of the product, thereby including the signal into it. Despite this limitation, the result is nevertheless the same mathematically, and the percentage uncertainty system is more comprehensible than the absolute value range.

In Figure 4.14 the *Aerodynamic coefficients* block actually pre-multiplies all the nominal partial coefficients with their respective signals first, apart from C_{m_1}, C_{L_1}, and C_{D_1}, which have α incorporated in them and thus do not suffer from the detection problem. The results are then fed into the

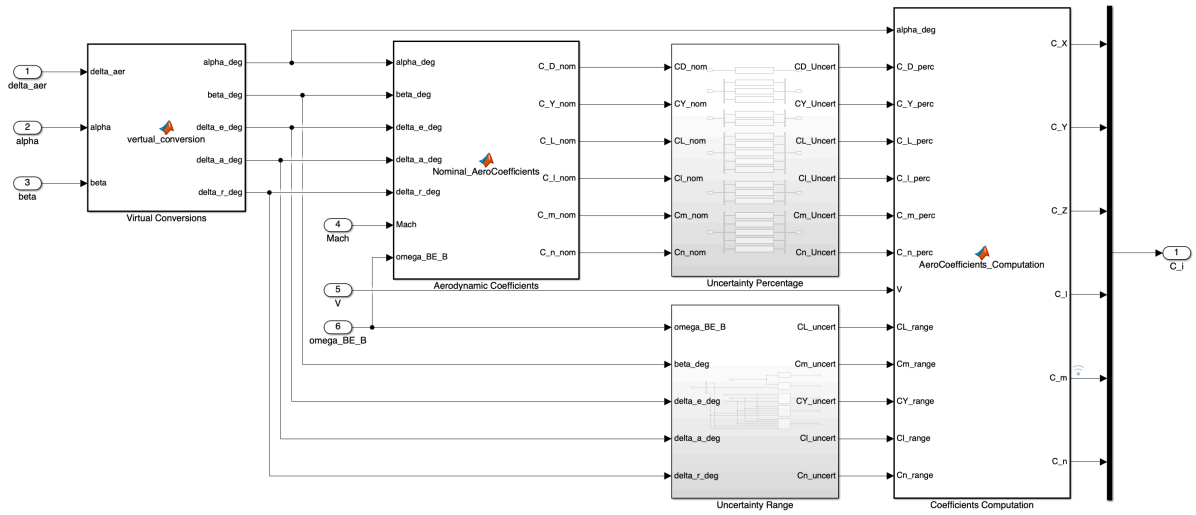


Figure 4.14: *Aerodynamic coefficients* block of the uncertain airframe model

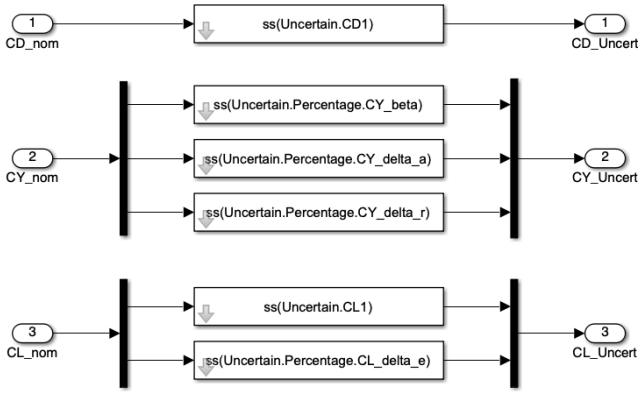


Figure 4.15: Part of the *Uncertainty Percentage* block of the *Aerodynamic coefficients* subsystem

Uncertainty Percentage block, the top half of which is shown in Figure 4.15 for illustration. The white blocks are the uncertain state-space blocks which are filled with their respective uncertainty $ureal$ variables, initialized in file 1 with specified percentage. These uss blocks are what allows the uncertainty variables to be multiplied with other signals in the model and be detected by the $ulinearize$ function. The other three coefficients follow a similar pattern.

The case of absolute value range uncertainty is more direct, although has a more complicated structure. It is implemented in the *Uncertainty Range* block shown in Figure 4.16. Sideslip angle, rotational rates and virtual actuator deflections are multiplied with respective uncertainty variables of the partial aerodynamic coefficients. As before, the uncertainty variables are put into uss blocks as $ureal$ variables, initialized in *Init_Airframe.m*. Mathematically, the absolute value range uncertainty is implemented similar to additive uncertainty. The process can be described by an example for lift coefficient:

$$C_L = C_{L_1} + C_{L_{\delta_e}}\delta_e + C^u_{L_{\delta_e}}\delta_e \tag{4.4}$$

where $C^u_{L_{\delta_e}}$ is a $ureal$ variable with specified value range. This final summation is what happens in the *Coefficients Computation* block from Figure 4.14. Once again, uncertainty for the same partial coefficient can not be turned on simultaneously as percentage and range. Looking back to Equation 4.4, either $C_{L_{\delta_e}}$ is perturbed with percentage variation in *Uncertainty Percentage* block while $C^u_{L_{\delta_e}}$ is set to zero, or $C_{L_{\delta_e}}$ is multiplied with 1 in that block to produce nominal value while $C^u_{L_{\delta_e}}$ is set to vary in range. As has been stated, the uncertainty for C_{m_1}, C_{L_1}, and C_{D_1} can only be specified in percentage or turned off completely by setting their uncertain variables to 1 in the *Init_Airframe.m* file, because they

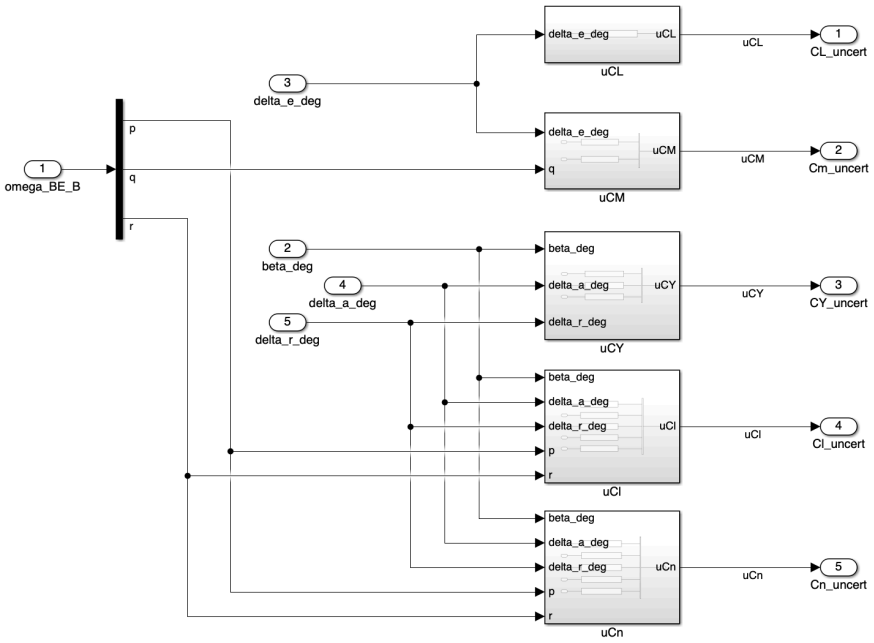


Figure 4.16: *Uncertainty Range* block of the *Aerodynamic coefficients* subsystem

are not multiplied with any signal. Again, the entire process described above can be referred back to Figure 4.13. There, only one of the uCm_q blocks can be specified as a $ureal$ variable, while the other must be set to its mean value. Naturally, it is also possible to remove the percentage version of the uCm_q uss block, or in contrast remove the entire bottom line to be left with percentage uncertainty variant.

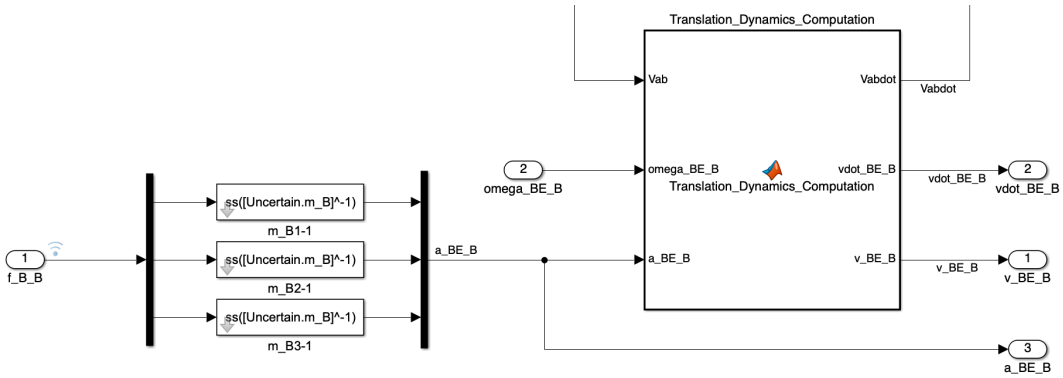
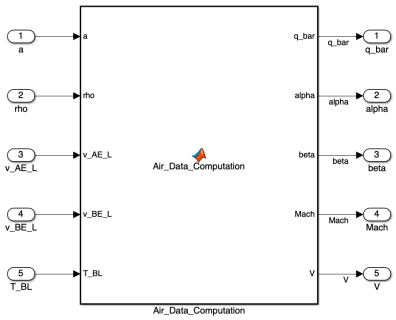


Figure 4.17: Part of *Translation Dynamics* block of the *EoM* with uncertain mass

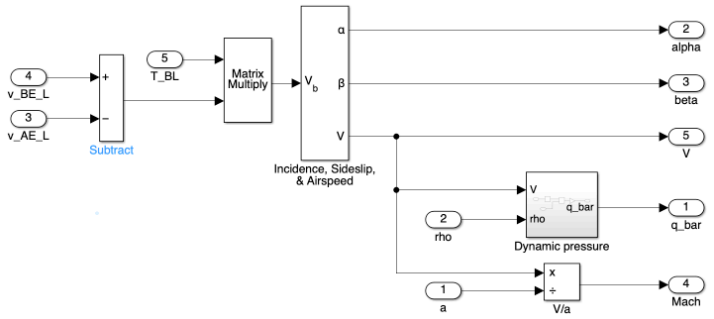
The *EoM* block of the *Airframe_uncertain.slx* model changed from standard to custom-build, but retaining the same dynamic and kinematic equations and following a similar structure. The reasons are the ability use $[V\ \alpha\ \beta]$ as states for linearization directly and the ability to add parametric uncertainty in the form of uss blocks to any desired variables. In this case, only the vehicle mass was added to the framework. The implementation is shown in Figure 4.17, where the forces are directly multiplied by inverse of uncertain mass variable. Although in this research the mass was varied explicitly between within the flight point grid, it is also possible to add mass to the uncertain variables by changing its uncertain value in *Init_Airframe.m* from m_B^B to any desired form of $ureal$ variable. Unlike the coefficients, however, it is not a continuously computed parameter here, so the mean value of the mass $ureal$ variable must be set to m_B^B. **Take caution** that the mass m_B^B was not changed to its uncertain variant of $Uncertain.m\_BB$ in of all its appearances, since the mass uncertainty was included in multi-modelling separately. Thus, the framework of mass uncertainty with $ureal$ function is not verified. Changing all m_B^B instances in

Init_Airframe.m was not attempted.

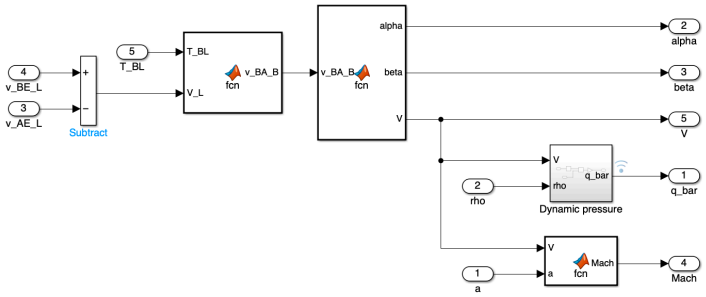
Important note on problems with the *Air Data Computer* block during linearization. There were a total of 3 variants of that block, two of which were implemented. The variants of ADC are shown in Figure 4.18, and they have all been checked and verified to produce exactly the same outputs, even superimposed. The first variant in Figure 4.18a is written as a single code function. This block initially worked fine with $linearize$ function. However, it was discovered to be disrupting the linearization process with $ulinearize$ for an unknown reason. The code was transformed into a form shown in Figure 4.18b with standard Simulink blocks. This variant, however, disrupted the trimming process, as the standard blocks always started off by sending out-of-bounds outputs, which in turn disrupted interpolation of aerodynamic coefficients. The solution was found by combining the two into a ”hybrid” ADC shown in Figure 4.18c. Since Simulink linearizes each block separately and then combines them into a single linear system, it seems that $ulinearize$ could not handle the original version of a large single block of code, so the code was broken down into smaller blocks. Note that combining even any two of the smaller code blocks can result in incorrect linearization.



(a) Original coded *ADC* block (unused)



(b) Redesigned *ADC* with standard blocks



(c) Hybrid *ADC* block

Figure 4.18: *Air Data Computer* block implementations

Propulsion system

The propulsion system model for trimming and linearization, displayed in Figure 4.19, is almost identical to the one shown in Figure 4.9 from non-linear model. The two major differences are the inclusion of uncertain variables for the aerodynamic parameters, and the replacement of previously fed-back parameters with their trimmed values.

Linearization

Trim_Linearize.m function is called to perform the trimming and linearization of the above Simulink models. Its general outline is as follows:

1. Set up the operating point specification with $operspec$ function for the *Airframe_uncertain.slx* model. The settings for the states were already discussed in Section IV of chapter 3, whereas the inputs and outputs are kept free within their limits. For every state, the initial guess is set to initial

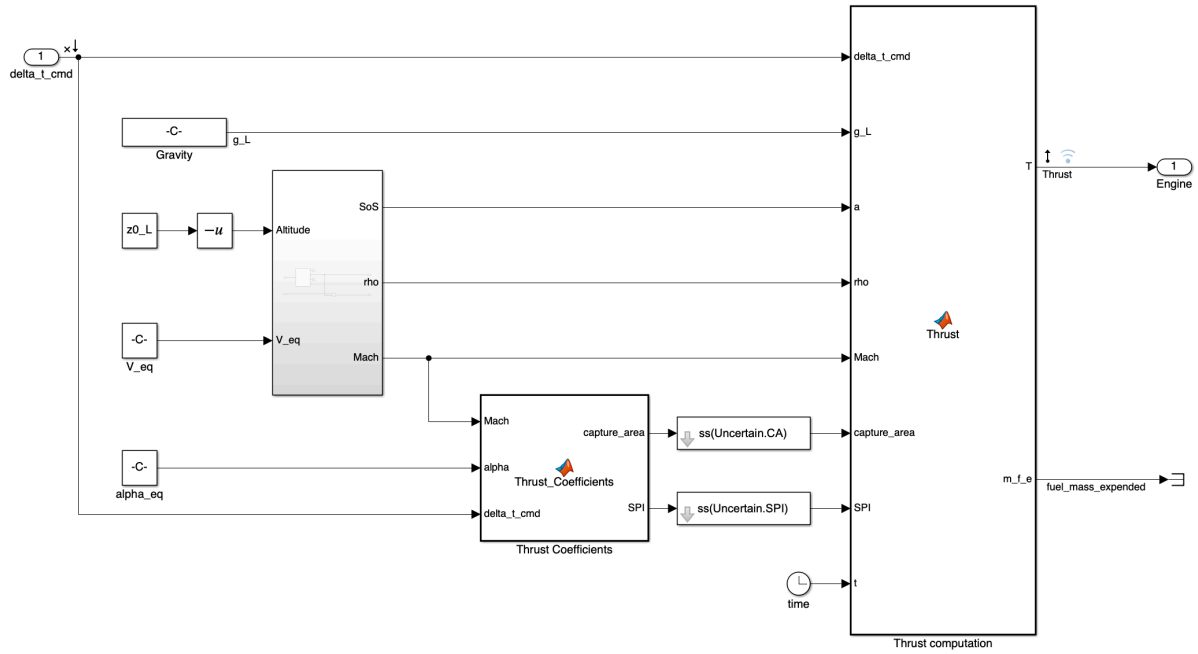


Figure 4.19: *Engine_uncertain.slx* file layout

flight point specification. The code follows the structure similar to the automatically generated version of the *Linearization app* in Simulink.

2. The airframe is trimmed with $findop$ function. Gradient-descent elimination optimizer type with sequential quadratic programming algorithm were found to be the best performing setting.
3. The trimmed parameter values are retrieved, the necessary ones for the propulsion system are assigned to workspace. The model is now switched to *Engine_uncertain.slx*.
4. The propulsion system is trimmed in the same manner as the airframe.
5. The inputs and outputs for linearization are retrieved directly from specified points in the respective Simulink models. $ulinearize$ is called on the airframe and the engine to linearize the models.
6. There are two linear models now, one is the full uncertain state-space model of the airframe that was introduced earlier in Section III of chapter 3, and one is uncertain state-space model of the engine. The airframe model is decoupled and post-processed. The outputs of the *Trim_Linearize.m* function are the short-period model, full-longitudinal model, lateral model, engine model, and the trimmed parameters.

The short-period, longitudinal and lateral models are retrieved by selecting and reorganizing specific rows and columns of the full linear model of the airframe. This is also the stage where the two separate elevon deflection inputs are converted to virtual aileron and elevator inputs for the linear systems. This conversion happens after the longitudinal and lateral models have been decoupled. The conversion to elevator inputs is done by simply adding the elevon columns together. The conversion to ailerons is trickier, as the column values have opposite signs: right elevon column must be subtracted from the left elevon column, and then divided by two. However, the elevon matrix entries have the same magnitude for lateral model, so the aforementioned operation corresponds to just multiplying the left elevon column by 2, and removing the right elevon column completely. Additionally, the longitudinal and lateral models with real actuators (elevons) are also retrieved for any potential future use. The aforementioned procedures are shown in the code snippet below.

```
%% Model postprocessing (airframe, virtual inputs)

% Rename states
sys_airframe.InputName = {'T'; 'delta_r'; 'delta_vl'; 'delta_vr'};
sys_airframe.StateName = {'p'; 'q'; 'r'; 'phi'; 'theta'; 'psi'; 'V'; 'alpha'; 'beta'; 'z_e'};
sys_airframe.OutputName = {'p'; 'q'; 'r'; 'n_x'; 'n_y'; 'n_z'};
```

```
7
8  % Longitudinal model
9  A_long      = sys_airframe.A([8 2 7 5],[8 2 7 5]);
10 B_long      = sys_airframe.B([8 2 7 5], [1 3]);
11 %B_l = B_long.NominalValue
12 B_long(:,2) = B_long(:,2)*2; % convert to de
13 %B_long.NominalValue
14 C_long      = sys_airframe.C([6 2], [8 2 7 5]);
15 D_long      = sys_airframe.D([6 2], [1 3]);
16 D_long(:,2) = D_long(:,2)*2; % convert to de
17 sys_long    = ss(A_long, B_long, C_long, D_long);
18
19 sys_long.InputName  = {'T'; 'delta_e'};
20 sys_long.StateName  = {'alpha'; 'q'; 'V'; 'theta'};
21 sys_long.OutputName = {'n_z'; 'q'};
22
23 sys_long.InputUnit  = {'N'; 'rad'};
24 sys_long.StateUnit  = {'rad'; 'rad/s'; 'm/s'; 'rad';};
25 sys_long.OutputUnit = {'g'; 'rad/s'};
26
27
28 % Longitudinal short-period
29 A_long_sp      = sys_airframe.A([8 2],[8 2]);
30 B_long_sp      = sys_airframe.B([8 2], 3);
31 B_long_sp(:,1) = B_long_sp(:,1)*2; % convert to de
32 C_long_sp      = sys_airframe.C([6 2], [8 2]);
33 D_long_sp      = sys_airframe.D([6 2], 3);
34 D_long_sp(:,1) = D_long_sp(:,1)*2; % convert to de
35 sys_long_sp    = ss(A_long_sp, B_long_sp, C_long_sp, D_long_sp);
36
37 sys_long_sp.InputName  = {'delta_e'};
38 sys_long_sp.StateName  = {'alpha'; 'q'};
39 sys_long_sp.OutputName = {'n_z'; 'q'};
40
41 sys_long_sp.InputUnit  = {'rad'};
42 sys_long_sp.StateUnit  = {'rad'; 'rad/s'};
43 sys_long_sp.OutputUnit = {'g'; 'rad/s'};
44
45 % Lateral/directional model
46 A_lat = sys_airframe.A([4 1 9 3 6],[4 1 9 3 6]);
47 B_lat = sys_airframe.B([4 1 9 3 6], [3 2]);
48 B_lat(:,1) = B_lat(:,1)*2; % convert to da
49 C_lat = sys_airframe.C([5 1 3], [4 1 9 3 6]);
50 D_lat = sys_airframe.D([5 1 3], [3 2]);
51 D_lat(:,1) = D_lat(:,1)*2; % convert to da
52 sys_lat = ss(A_lat, B_lat, C_lat, D_lat);
53
54 sys_lat.InputName  = {'delta_a'; 'delta_r'};
55 sys_lat.StateName  = {'phi'; 'p'; 'beta'; 'r'; 'psi'};
56 sys_lat.OutputName = {'n_y'; 'p'; 'r'};
57
58 sys_lat.InputUnit  = {'rad'; 'rad'};
59 sys_lat.StateUnit  = {'rad'; 'rad/s'; 'rad'; 'rad/s'; 'rad'};
60 sys_lat.OutputUnit = {'g'; 'rad/s'; 'rad/s'};
```

Important note on uncertainty distribution among linear models. The main feature of $ulinearize$ is that it detects the previously defined uncertainty blocks and embeds them into the linear systems automatically. If some uncertainty block is set to a real value rather than a $ureal$ variable, it will not be included as one of the uncertain parameters in the obtained uss system. Furthermore, if uncertainty is turned off completely (no $ureal$ variables detected), then $ulinearize$ outputs a standard ss system. Note that the related uncertainty variables are carried over and distributed among the decoupled models automatically. In other words, if there is uncertainty in C_{n_r} coefficient, it will not be present in the extracted short-period model. However, beware that if the program finds some lateral coefficient's uncertainty affecting the longitudinal motion, then it will be carried over to the extracted longitudinal model, and vice-versa. Unfortunately, no way around it was found, apart from careful uncertainty variables selection. I.e., if longitudinal model is of interest, then turn off uncertainty for lateral coefficients, and vice-versa.

The extracted models' states, inputs, and outputs are presented in Equation 4.5. The current propulsion system model is essentially just a D matrix with throttle δ_t as input and thrust T as output.

$$\begin{array}{lll} \text{Longitudinal model} & \text{Short-period model} & \text{Lateral model} \\ \text{States} = [\alpha \quad q \quad V \quad \theta] & \text{States} = [\alpha \quad q] & \text{States} = [\phi \quad p \quad \beta \quad r \quad \psi] \\ \text{Inputs} = [T \quad \delta_e] & \text{Inputs} = [\delta_e] & \text{Inputs} = [\delta_a \quad \delta_r] \\ \text{Outputs} = [n_z \quad q] & \text{Outputs} = [n_z \quad q] & \text{Outputs} = [n_y \quad p \quad r] \end{array} \tag{4.5}$$

Important note: of the three linear models only the short-period model was verified analytically. The other two were outside the scope of this research.

An example of analytical verification is the A_{21} entry of the short period matrix, corresponding to $d\dot{q}/d\alpha$. Taking the EoM corresponding to $\dot{q}$ from Equation 2.12, the expression is:

$$\dot{q} = \frac{1}{I_{22}}((I_{33} - I_{11})pr + I_{13}(p^2 - r^2) + m_{B_2}) \tag{4.6}$$

where I_{ii} are MOI entries and m_{B_2} is the pitch moment. Since the trimmed condition sets rotational rates p and r to zero, the expression simplifies to just:

$$\dot{q} = \frac{m_{B_2}}{I_{22}} \tag{4.7}$$

where $m_{B_2} = \bar{q}ScC_m$ from Equation 2.22. The C_m is calculated as specified in the aerodynamic model of GHAME. Therefore, the complete expression at trim is:

$$\dot{q} = \frac{\bar{q}Sc}{I_{22}}(C_{m_1}(M,\alpha) + C_{m_{\delta_e}}(M,\alpha)\delta_e + C_{m_q}(M,\alpha)\frac{qc}{2V}) \tag{4.8}$$

However, q is also zero at trim, and looking at the table of $C_{m_{\delta_e}}(M,\alpha)$ it is apparent that the coefficient changes only with Mach, unless the angle of attack is almost at the upper limit. Therefore, the differentiation expression to be used for calculation is:

$$\frac{d\dot{q}}{d\alpha} = \frac{\bar{q}Sc}{I_{22}}\left(\frac{dC_{m_1}(M,\alpha)}{d\alpha}\right) \tag{4.9}$$

For analytical verification, the *coef_check.m* file is used. For any given flight point that was specified in the *Main_Modelling.m* file, it retrieves the calculated trimmed values and performs separate calculations for each entry in the matrices. The equations for each entry were derived by hand, the code only plugs in the constant values and differentiates the aerodynamic tables either wrt Mach or α via custom functions.

For almost all entries the match between the analytical and actual values is exact. The only problem is the conversion from C_L and C_D to C_Z and C_X. Naturally, if differentiation of C_Z or C_X is needed, then it makes sense to differentiate the conversion equations:

$$C_X = -C_D\cos(\alpha) + C_L\sin(\alpha), \qquad C_Z = -C_D\sin(\alpha) - C_L\cos(\alpha); \tag{4.10}$$

and substitute the respective expressions for C_L and C_D from the aerodynamic model. However, that did not lead to any meaningful results - perhaps an oversight that was not identified. A quick workaround was needed to approximate the differentiation of C_Z and C_X to maintain the project time limits. The solution is the following, with example on C_Z: for some trimmed condition (M_t, α_t), compute a column of virtual C_Z table for trimmed Mach number and compute a row of virtual C_Z table for trimmed α with:

$$\begin{aligned} C_Z(M_t,\alpha) &= -C_D(M_t,\alpha)\cos(\alpha) + C_L(M_t,\alpha)\sin(\alpha) \\ C_Z(M,\alpha_t) &= -C_D(M,\alpha_t)\cos(\alpha_t) + C_L(M,\alpha_t)\sin(\alpha_t) \end{aligned} \tag{4.11}$$

for a range of M and α values. Depending on what the C_Z needs to be differentiated with, the virtual row or column is differentiated with a custom function. This does not lead to exact match between analytical computation and the corresponding actual linear model matrix entry, but the matching error is about 1%, so it was considered acceptable within the scope of this research. This method was applied to all instances of C_Z and C_X differentiation.

The *coef_check.m* file already contains some of the full-longitudinal matrix entries computations (some of which were not finalized), and can be extended further if needed. Everything but the short-period matrices was beyond the project scope.

4.2. Controller synthesis

The theory behind multi-objective H_∞ control has already been extensively discussed in this paper. This section aims to provide details on practical matters with regards to its implementation. The whole process is self-contained in the *Controller* folder of the project directory. The *SigBased_sp_.m* file approximately follows the same structure as the automatically generated code from *Control System Tuner* Simulink app.

Some notes regarding the file set-up and controls:

- The model grid is loaded from *model_grid.mat* contained in the folder. A model array $G\_sp(:,:,i)$ is assembled from the models. If a single model is needed, the array can contain only a single model. The models must be in uss format for this file.
- Actuator is modelled by assembling a state-space for it manually using its ω_a and ζ_a. The input is $\delta_{e,cmd}$, states are $[\dot{\delta}_e,\ \ddot{\delta}_e]$, and outputs are $[\delta_e,\ \dot{\delta}_e]$.
- The $samples$ variable sets the amount of samples of the models to be used in the analysis stage for open loops assembly. This does not set the sampling for multi-modelling controller synthesis.
- The $models\_hard$ and $models\_soft$ variables set the model indices for hard and soft constraints. By default, model 1 in the array is the nominal flight point model, whereas the rest of the 10 models are its trim variations. Therefore, model index 1 is passed to $models\_hard$, and model indices 2:11 are passed to soft constraints with $models\_soft$.
- The $models\_switch$ is used to control the switch structure within the file. Setting it to 1 corresponds to using a single uss model instead of an array, and applying only hard constraints on it. The loops are sampled for uncertainty within that model only. Setting the switch to 2 corresponds to using a model array, for which the soft constraints are turned on. The loops are then sampled for all uncertain models in the array.

The outline of the controller synthesis procedure in *SigBased_sp_.m* file is the following:

1. Set-up the file using the controls above. The model array is set manually, so to use a single model the other instances must be commented out.

```
load model_grid

omega_a = 50; %rad/s
zeta_a  = 0.7;
% u_max = 20 deg, u_dot_max = 400 deg/s

A_a = [0, 1; -omega_a^2, -2*zeta_a*omega_a];
B_a = [0; omega_a^2];
C_a = [1, 0; 0, 1];
D_a = [0; 0];
G_a = ss(A_a, B_a, C_a, D_a);

samples = 20;

models_hard = 1;
models_soft = 2:11;

models_switch = 2;% 1 for one uss, 2 if multi-model

G_sp(:,:,1)  =  G_sp_05_75_50;
G_sp(:,:,2)  = G_sp_025_75_50;
G_sp(:,:,3)  = G_sp_075_75_50;
G_sp(:,:,4)  = G_sp_025_70_45;
G_sp(:,:,5)  = G_sp_025_70_55;
G_sp(:,:,6)  = G_sp_025_80_45;
G_sp(:,:,7)  = G_sp_025_80_55;
G_sp(:,:,8)  = G_sp_075_70_45;
G_sp(:,:,9)  = G_sp_075_70_55;
G_sp(:,:,10) = G_sp_075_80_45;
G_sp(:,:,11) = G_sp_075_80_55;
```

2. The models are connected to the actuator in series and sampled for post-analysis. In the multi-modelling case it becomes a 2-dimensional model array, where the rows correspond to model numbers and columns correspond to samples of each model. The nominal value of each model is put in the first column. Function $genss$ is used on one of the models to define the general state-space block of the same structure in Simulink into which the model array is substituted later.

```
G_general = genss(G_sp_05_75_50);
temp_G_asp      = series(G_a(1), G_sp);
G_asp           = usample(temp_G_asp, samples);

switch models_switch
    case 1
        G_asp(:,:,1)  = temp_G_asp.NominalValue;
    case 2
        G_asp(:,:,:,1)  = temp_G_asp(:,:,:).NominalValue;
end
```

3. Set the filters for soft and hard constrains using the theory discussed in chapter 3. The weights are created using $makeweight$ function. Here is a sample for S_O constraint:

```
% So
dcgain_So   = db2mag(-60);
freq_So     = 0.35;
mag_So      = db2mag(-3.01);
hfgain_So   = 1.9;
%.....
W_So1_inv       = makeweight(dcgain_So,[freq_So,mag_So],hfgain_So,0,1);
```

4. Define tunable controller structures with $tunableGain$ or $tunableTF$. Several are included, and can be selected using $controller\_type$ switch. Below is the structure used in this research:

```
% C_i TF
SigBased_C_i            = tunableTF('SigBased_C_i',1,1);
SigBased_C_i.Numerator.Value   = [1 1];
SigBased_C_i.Denominator.Value = [1 1];
SigBased_C_i.Numerator.Free    = [1 1];
SigBased_C_i.Denominator.Free  = [0 1];
C_i = ss(SigBased_C_i);

% C_q Gain
SigBased_C_q            = tunableGain('SigBased_C_q',1,1);
SigBased_C_q.Gain.Value = 0;
SigBased_C_q.Gain.Free  = 1;
C_q = ss(SigBased_C_q);

% C_ff TF
SigBased_C_ff            = tunableTF('SigBased_C_ff',1,1);
SigBased_C_ff.Numerator.Value   = [1 1];
SigBased_C_ff.Denominator.Value = [1 1];
SigBased_C_ff.Numerator.Free    = [1 1];
SigBased_C_ff.Denominator.Free  = [0 1];
C_ff = ss(SigBased_C_ff);
```

5. Create system data with slTuner interface. This step defines the model substitution using $struct$ function. That is how the model array is passed into the Simulink block. This step also defines the tuned blocks and analysis points in the selected Simulink model. Note that there is no need to specify the input perturbations or output measurements in the Simulink model itself - it is done in code by using appropriate signal names.

```
TunedBlocks = {'SigBased_sp/C_q'; ...
               'SigBased_sp/C_ff';
               'SigBased_sp/C_i'};
AnalysisPoints = {'SigBased_sp/Input/1'; ...
                  'SigBased_sp/d_o/1'; ...
                  'SigBased_sp/d_i/1'; ...
                  'SigBased_sp/Sum_u_cmd/1'; ...
                  'SigBased_sp/Sum_y1/1'};
```

```
9
10 modelSub = struct('Name','SigBased_sp/G_sp','Value', G_sp);
11
12
13 % Specify the custom options
14 Options = slTunerOptions('AreParamsTunable',false);
15 % Create the slTuner object
16 CL0 = slTuner('SigBased_sp',TunedBlocks,AnalysisPoints,modelSub, Options);
```

6. Soft and hard constraints are defined for the signals specified between the analysis points in the model. The selected signals were outlined in chapter 3. This step includes reference model computation for the reference tracking, which was also discussed in detail already. Below is a code snippet of the reference model computation and reference-tracking tuning goal:

```
1  %% ____________________________REF MODEL ___________________________________
2  % Parameters
3  ts_desired = 4;
4  Mp_desired = 0.01;
5  t_step     = linspace(0, 7, 2e4+1);
6
7  % Optimizer options
8  options                          = optimoptions('fmincon');
9  %options.Display                  = 'off';
10 options.OptimalityTolerance     = 1e-12;
11 options.StepTolerance           = 1e-12;
12 options.MaxFunctionEvaluations = 1e12;
13 options.MaxIterations           = 1e6;
14
15 % Find optimal values
16 zeros_G  = zero(G_asp(1,1));
17 zero_nmp = zeros_G(2,1);
18 lb = [0.7, 0.5]; % Lower bounds
19 ub = [1, 3.5];    % Upper bounds
20 x0             = [0.8 1]; % [zeta omega]
21 x_optimal      = fmincon(@(x)Compute_ReferenceModelParameters(x, ts_desired, Mp_desired,
       zero_nmp, t_step), x0, [], [], [], [], lb, ub, [],  options);
22 zeta_optimal  = x_optimal(1,1)
23 omega_optimal = x_optimal(1,2)
24 % Form reference model
25 T_t = ss(tf([-omega_optimal^2/zero_nmp omega_optimal^2],[1 2*omega_optimal*zeta_optimal
       omega_optimal^2]));
26
27
28
29 % Create tuning goal to shape how the closed-loop system responds to a specific input
       signal
30 % Inputs and outputs
31 Inputs = {'SigBased_sp/Input/1[r]'};
32 Outputs = {'SigBased_sp/Sum_y1/1[n_z]'};
33 % Create tuning goal for step tracking
34 T_ref = TuningGoal.StepTracking(Inputs,Outputs,T_t);
35 T_ref.RelGap = RelGap_hard;
36 T_ref.Name = 'T_ref'; % Tuning goal name
37 T_ref.Models = models_hard;
38
39 % Create tuning goal for step tracking
40 soft_T_ref = TuningGoal.StepTracking(Inputs,Outputs,T_t);
41 soft_T_ref.RelGap = RelGap_soft;
42 soft_T_ref.Name = 'Soft T_ref'; % Tuning goal name
43 soft_T_ref.Models = models_soft;
```

7. Function $systune$ is called to synthesize the controllers.

```
1 %% ______________________________ Tune ___________________________________
2 % Create option set for systune command
3 Options = systuneOptions();
4 Options.RandomStart = 20; % Number of randomized starts
5 Options.UseParallel = true; % Parallel processing flag
6
```

```
% Set soft and hard goals
HardGoals = [ So1 ; ...
              negKSo1; ...
              SoG1; ...
              Ti;   ...
              T_ref];

switch models_switch
    case 1
        SoftGoals = [];
    case 2
        SoftGoals = [soft_So1 ; ...
                     soft_negKSo1; ...
                     soft_SoG1; ...
                     soft_Ti;   ...
                     soft_T_ref];
end
% Tune the parameters with soft and hard goals
[CL1,gamma_Soft,gamma_Hard,Info] = systune(CL0,SoftGoals,HardGoals,Options);
```

8. The controllers are put into a single matrix K and the sampled loops are closed with K in the feedback. Function $loopsens$ is called for opening the loops at the plant input and the outputs, forming a total of 220 sampled open loops each.

```
C_i = getTunedValue(CL1,'SigBased_sp_C_i');
C_q = getTunedValue(CL1,'SigBased_sp_C_q');
C_ff = getTunedValue(CL1,'SigBased_sp_C_ff');
K_gains = [zpk(C_i),...
           zpk(C_q),...
           zpk(C_ff)]
K = [1 1]*[C_i*tf(1,[1 0]) 0; 0 C_q];
loops = loopsens(G_asp, K);

% SISO output loops
Lo1_temp = feedback(loops.Lo,1,2,2,-1); % feedback(G,K,feedin,feedout,-1);
Lo2_temp = feedback(loops.Lo,1,1,1,-1);
Li1      = loops.Li;
Lo1      = Lo1_temp(1,1);
Lo2      = Lo2_temp(2,2);
```

9. Disk margins are computed for every single open loop, the smallest among them are retrieved for analysis and plotting disk margin regions on Nichols charts. The disk margin regions are plotted using the principles from subsection 2.4.1. Below is a snippet example of Nichols chart plot with exclusion regions for open loops at input:

```
% Disk margin region
rmin = DMi / 2;
theta_vec = linspace(-pi, pi, 1e2);
locus_Li1_2DoF = (1 - rmin * exp(-1i * theta_vec)) ./ (1 + rmin * exp(-1i * theta_vec));
patch(ax, rad2deg(angle(locus_Li1_2DoF) - pi), mag2db(abs(locus_Li1_2DoF)), disk_color,
    'FaceAlpha', 0.3, 'LineStyle', 'none', 'DisplayName', 'Disk margin region');

% Exclusion region
rmin_required = 0.33228; % (for 6 dB)
locus_new = (1 - rmin_required * exp(-1i * theta_vec)) ./ (1 + rmin_required * exp(-1i *
     theta_vec));
patch(ax, rad2deg(angle(locus_new) - pi), mag2db(abs(locus_new)), 'r', 'FaceAlpha', 0.3,
     'LineStyle', 'none', 'DisplayName', 'Exclusion region');

% Loops on Nichols chart
nicholsplot(ax, Li1, plotoptions, 'b');
```

Back in subsection 2.3.3 it was identified that for robustness against multiplicative and additive uncertainty, the maximum singular values of $K_{ff}KS_O$ (equivalent to KS_O without feed-forward controller), T_I and T_O should be minimized. These transfer functions, however, were not shown in chapter 3 due to the article size limitations. Therefore, these three transfer functions for the sampled models in the grid are displayed in Figure A.4. Additionally, the open loops from the Nichols charts are also shown in

the Bode plot format in Figure A.5. It should be noted that the largest σ of $K_{ff}KS_O$ corresponds to a sample of model 7 ($G\_sp(:,:,7)$), which is the lightest and fastest flying of the whole grid, whereas the largest σ of T_I corresponds to a sample of model 9 ($G\_sp(:,:,9)$), which is the heaviest and slowest flying.

Important note on using $systune$ with uncertain model array. There are basically two ways to synthesize controllers for multiple uncertain plants: one is to pass a model array filled with uss models, and one is to sample the model array manually and pass all samples as nominal ss systems to use multi-modelling approach. Conceptually, they follow similar principles: sampling an infinite set of scenarios in the parametric uncertainty region into a finite set of samples. For that finite set, the multidisk H_∞ approach is applied, where the controller is connected to the sampled plants simultaneously and tuned. Although both methods lead to approximately same results, it was observed that pre-sampling the model array manually and passing 220 ss systems takes an enormous amount of time to tune the controllers when compared to tuning for 11 uss models. The most probable reason is that $systune$ does not choose the samples in the uncertainty set randomly. According to the inner-approximation technique from [38], there is a specific algorithm which selects the worst-case samples for tuning and thus stabilizing and controlling the plant for the entire uncertainty domain. Furthermore, it was explicitly stated in [38] that the tuning process succumbs for a large set of samples. As a result, $systune$ algorithm is well-optimized to handle multiple uss models with uncertainty. Pre-sampling, on the other hand, appears to overload the multi-disk H_∞ method with nominal models due to their large number. It is therefore strongly recommended to use the uss models with $systune$ as they are, instead of sampling them beforehand.

5

Conclusion & Recommendations

5.1. Conclusion

The objective of this research was to develop a robust flight control system for a generic hypersonic vehicle. It involved construction of a 6-degree-of-freedom non-linear model in MATLAB and Simulink, incorporation of parametric uncertainties, trimming and linearization, controller synthesis using multi-objective H_∞ mixed sensitivity techniques for the linear model, and implementation and testing of the developed controllers on the non-linear model. To achieve this goal, multiple research questions were formulated for this study and are outlined once again on the next page for reference.

The air-breathing hypersonic vehicles are among the toughest challenges in the modern aerospace industry. They fly at unprecedented speeds and altitudes that are unreachable by conventional supersonic aircraft and offer a potential low-cost space access as single-stage-to-orbit vehicles. Although no known operational air-breathing hypersonic vehicles exist yet, their conceptual development has been underway for a few decades. It was therefore possible to identify some general design traits unique to HV, extensively discussed in section 2.1. The common designs featured long, slender, lightweight airframe with tightly integrated propulsion system. Due to extreme flight conditions and the aforementioned characteristics, the HV dynamics exhibit strong non-linearities at hypersonic speeds from the aeroelastic effects, introducing significant challenges to their FCS design. The details of the intervention of the aeroelastic effects into HV control were also identified. It was found that high-fidelity aerodynamic models are not publicly available, which introduces significant uncertainties in the modelling process together with the aeroelastic effects. The literature study on the developed HV control system designs concluded that the implemented linear robust control methods relied on relatively old optimization algorithms that could neither provide flexible handling of performance specifications under system uncertainty, nor synthesize controllers of fixed-structure. The current focus was found to be switched to the non-linear and adaptive control methods, whereas the new multi-objective H_∞ methods were not found to be implemented yet. Therefore, research **questions 1(a-c)** were answered.

The GHAME 6-DoF non-linear model was successfully constructed in MATLAB and Simulink environment using the tensor-based equations of motion. For control system development the flat-Earth assumption was assumed to be sufficiently accurate. The aerodynamic model was incorporated into initialization code, whereas the aerodynamic coefficients were computed using 2-dimensional interpolation. A fundamental discrepancy in the aerodynamic data was discovered and fixed with the best fitted solution available. Each subsystem of the non-linear model was incorporated in a flexible block-separated structure. The propulsion system was modelled exactly as the one from the original GHAME model, whereas the actuator parameters were based on P. Zipfel's FORTRAN model of GHAME. The sensors included only the selected outputs and an IMU, which had an estimated location in the nose section of the airframe. IMU only calculated the load factor using Grubin's transformation. Sensor noise was not included in the model. As a result, the **questions 2(a-c)** were fully addressed.

Research Questions

Main research question:
How effectively can a structured flight control system designed with multi-objective H_∞ methods handle subsonic short-period dynamics of a hypersonic vehicle under multiple performance and robustness requirements?

Sub-questions:

1. **What makes the HV different from conventional aircraft?**
   (a) What are the HV mission profiles?
   (b) What are the HV unique design characteristics?
   (c) What are the challenges associated with HV stability and control?

2. **How to incorporate a non-linear HV model into Simulink?**
   (a) How to incorporate an aerodynamic model?
   (b) How to model the non-linear dynamics and kinematics?
   (c) How to model the HV actuators and sensors?

3. **How to develop a linear HV model with uncertainty?**
   (a) How is uncertainty modelled?
   (b) Which parameters are uncertain and in what range?
   (c) Which operating point to consider?
   (d) How is the non-linear model trimmed and linearized?
   (e) Which linear dynamics are to be controlled?

4. **How to design a robust FCS with multi-objective H_∞ methods?**
   (a) What are the working principles behind the new methods?
   (b) How is the multi-objective H_∞ control implemented practically?
   (c) What are the FCS robustness and performance design requirements?
   (d) How are the requirements transferred into practical constraints?
   (e) Which controller structure to choose?

5. **How to assess FCS performance and robustness?**
   (a) How is the FCS robust performance assessed in frequency domain?
   (b) How is the FCS robust performance assessed in time domain?
   (c) How to implement the FCS in the non-linear model?
   (d) How can the FCS robust performance be verified with disturbances and uncertainty in the non-linear simulation?

The uncertainty origins and types were identified, as well as their modelling methods. A flexible framework was established for parametric uncertainty implementation, where any of the partial aerodynamic coefficients could be varied independently and automatically carried over to the uncertain state-space systems during linearization process. Since the aeroelastic effects primarily influence the pitch motion of HV, all of the partial pitch moment coefficients were arbitrarily chosen to vary by 20%. A steady, straight, wings-level flight point was selected at 0.75 Mach number and 5000 m altitude to investigate HV control at subsonic speed. The model was trimmed and linearized with MATLAB software at 11 operating points, corresponding to the nominal flight point and the variations of fuel mass, Mach number, and altitude around the nominal point. As a result, 11 longitudinal short-period models were extracted with 20% uncertainty in all partial pitch moment coefficients. Therefore, research **questions 3(a-e)** were answered.

The classical H_∞-synthesis methods were compared to the new non-smooth optimization approach, for which the general working principles and practical differences were outlined. A fixed-structure flight control system was synthesized using multi-objective (multidisk) H_∞ mixed-sensitivity methods provided by MATLAB software. The robustness and performance requirements for the nominal flight point were specified as constraints on the closed-loop transfer functions according to input and output disturbance rejection, high-frequency actuator signal attenuation, and reference tracking criteria. The multi-modelling approach then extended the original design onto the full grid of uncertain models managed by a single controller. Each constraint applied on the controller was handled under a separate

performance metric. Therefore, **questions 4(a-e)** were addressed.

Finally, the analysis of the disk margins at the system input and outputs showed that a single controller can successfully stabilize all sampled models in the grid and guarantee minimum required disk robustness margins against uncertainty. The models were sampled and plotted on the Nichols charts for further analysis. The linear time-domain analysis was performed by plotting the step responses of the respective closed-loop transfer functions, confirming the correct functioning of the control system. The synthesized controller set was implemented into the non-linear model and tested in multiple simulations. It was shown that a single controller set of fixed structure can successfully handle the short-period subsonic dynamics of GHAME model under parametric uncertainty and considerable variations around the trim condition in the non-linear simulation. Conclusively, the **main research question** was thereby answered, along with **questions 5(a-d)**.

5.2. Limitations and Recommendations

This research established a framework for GHAME hypersonic vehicle control in MATLAB and Simulink, but only considered a specific scenario. Some of the recommendations for future work and development include updates on the model limitations and control system development directions.

Modelling

First of all, the non-linear model is rigid-body, and thus more careful consideration of the aeroelastic effects should be done in case of control at hypersonic speeds. These may include a more thorough research into parameter uncertainty ranges or explicit addition of structural modes into the EoM. Elliptical-Earth *EoM* block can be added to allow full flight envelope simulations.

CG variations were not included in the model, neither was the in-flight mass variation, both of which are significant at hypersonic speeds. The MOI variation block based on expended fuel mass was implemented into Simulink, but was disconnected from the non-linear model because this framework was out of the scope of this research, and therefore not finished.

The propulsion system model should be updated to include engine dynamics and perhaps even a switch between the engine cycles. The currently implemented model is simplified and is probably unsuitable for velocity control law development. Additionally, the actuator properties may need re-evaluation based on further literature research. Furthermore, the sensor modelling should be more carefully considered with sensor noise incorporation.

Finally, the wind and turbulence were also not included in the model, but can be added later to the *Environment* block.

Uncertainty, trimming and linearization

The parameter variation in the full non-linear model requires an update to a more flexible one - as of now, only variations in pitch moment partial coefficients are included. The airframe model for linearization needs an update if more parameters are to be included as uncertainty, such as MOI or the previously discussed vehicle mass uncertainty. Further investigation is required into the limitations of parametric percentage uncertainty implementation in the uncertain airframe model. Although both methods work as intended, it seems that the percentage uncertainty block cannot function properly without the additional framework of ranged uncertainty, even when all of the ranged uncertain values are set to zero. I.e., it is not recommended to remove the ranged uncertainty block from the model as that may disturb the detection of uncertainty defined in percentage during linearization.

The trimming function, although working properly during this research, needs additional testing across the flight envelope considering supersonic and hypersonic speeds. It was found to be glitchy during set-up because of its sensitiveness to trim condition constraints. For example, it once refused trimming until the x^E coordinate was allowed to be set in negative direction (which does not influence the linear model). For the same reasons, it was impossible to set a desired angle of attack during trimming, so it was kept free within bounded range. It would be necessary to investigate this further if linearization at specific angle of attack is desired. Further attention is needed to the coefficients interpolation, as

the active α and M signals can move out of the aerodynamic data bounds during trimming if the model undergoes changes in the future, as was the case with block version of the *ADC*. Additionally, batch-trimming and batch-linearization were not implemented in this research due to time constraints - a significant tool for further FCS designs with multi-modelling.

The uncertain linearization with $ulinearize$ also experienced a series of glitches, primarily due to the already described problem with the *ADC* block. If the model has been changed and is not linearizing properly, it could be because one of the coded blocks is not linearized properly on its own. This was the case with the coded version of the *ADC* block. The problem can be overcome by splitting large coded blocks into smaller ones. Furthermore, the full-longitudinal and the lateral linear models need analytical verification. Only the short-period model was verified sufficiently, but it would have been beneficial to verify the matrix entries corresponding to the conversion from C_L and C_D to C_Z and C_X without any error at all.

Control system design

Although the HV should be able to operate in the entire flight envelope (including straight subsonic flight), it is important to also consider more HV-related scenarios for control system development, such as angle of attack hold during maneuvers. The established framework allows easy extraction of phugoid and lateral dynamics, which were untreated here but can be considered for the future research.

In the case of a direct follow-up work on the short-period model with multi-objective H_∞ control, additional disturbances in the pitch rate channel can be added to the design, and explicit constraints on the gain and phase margins can be incorporated together with all the aforementioned performance specifications. The transfer functions related to sensor measurement noise were not included in the design process, but should definitely be in the next stages.

The next natural step would be to use the constructed framework for multi-objective H_∞ short-period control augmentation system at supersonic and hypersonic flight regimes. This can be further extended for gain-scheduled controller synthesis to cover the whole flight envelope.

As has been mentioned earlier, the mass and MOI variations could be added to the uncertain parameters in the uss models rather than varied between separate models, should the corresponding framework be implemented. Actuator parameter uncertainties were not included into the framework. Further rearrangement of uncertain parameter variations can be considered between those that are specified as uncertainty and those that vary with the linear models in the grid. This rearrangement also influences the selection of models for the tuning goals and their weight as soft or hard constraints when using $systune$.

Analysis

The robustness analysis was mostly focused on the symmetrical disk margins is separate SISO loops within the system. Although the multi-loop input-output disk margins were also analysed and found satisfactory, they were not explicitly designed for. Furthermore, the μ-analysis to assess robustness of the FCS was not performed due to available time limits, but it was noted that it is an integral part of multidisk optimization for systems with parametric uncertainty.

To further assess the robustness margins under parametric uncertainty, the uss models in the grid can be sampled into a larger number of random ss models. That, however, increases the computation time considerably, and does not influence the controller synthesis with $systune$ (unless all samples are passed into the multi-modelling approach, but that increases the computation time far more).

References

[1] Sziroczak, D. and Smith, H. “A review of design issues specific to hypersonic flight vehicles”. In: *Progress in Aerospace Sciences* 84 (2016), pp. 1–28. ISSN: 03760421. DOI: 10.1016/j.paerosci.2016.04.001.

[2] Van Wie, D. M. “Hypersonics: Past, present, and potential future”. In: *Johns Hopkins APL Technical Digest* 35.4 (2021), pp. 335–341.

[3] Besser, H. L. et al. “Hypersonic vehicles: State-of-the-art and potential game changers for future warfare”. In: *2016 NATO Science & Technology Symposium Brochure*. Science & Technology Organization. 2016, pp. 31–34.

[4] Coleman, C. C. and Farhan, A. F. *On Stability and Control of Hypersonic Vehicles*. Government Document. 2009.

[5] Buschek, H. and Calise, A. J. *Robust Control of Hypersonic Vehicles Considering Propulsive and Aeroelastic Effects*. Generic. 1993.

[6] Fidan, B., Mirmirani, M., and Ioannou, P. “Flight dynamics and control of air-breathing hypersonic vehicles: review and new directions”. In: *12th AIAA international space planes and hypersonic systems and technologies* (2003), p. 7081.

[7] Duan, H. and Li, P. “Progress in control approaches for hypersonic vehicle”. In: *Science China Technological Sciences* 55 (2012), pp. 2965–2970. DOI: https://doi.org/10.1007/s11431-012-5036-x.

[8] Gregory, I. et al. “Hypersonic vehicle control law development using H infinity and mu-synthesis”. In: *AIAA 4th International Aerospace Planes Conference*. 1992. DOI: https://doi.org/10.2514/6.1992-5010.

[9] Anderson, M.R., Emami-Naeini, A., and Vincent, J.H. “Robust Cotrol Law Development for a Hypersonic Cruise Aircraft”. In: *1991 American Control Conference*. IEEE, 1991, pp. 839–845. DOI: 10.23919/ACC.1991.4791490.

[10] Buschek, H. and Calise, A. “Fixed order robust control design for hypersonic vehicles”. In: *Guidance, Navigation, and Control Conference*. AIAA, 1994, p. 3662. DOI: 10.2514/6.1994-3662.

[11] Buschek, H. and Calise, A. “Uncertainty Modeling and Fixed-Order Controller Design for a Hypersonic Vehicle Model”. In: *Journal of Guidance, Control, and Dynamics* 20.1 (1997), pp. 42–48. ISSN: 0731-5090 1533-3884. DOI: 10.2514/2.4031.

[12] Calise, A. J. *Research In Robust Control For Hypersonic Aircraft*. Report. 1993.

[13] Skogestad, S. and Postlethwaite, I. *Multivariable Feedback Control: Analysis and Design*. John Wiley & Sons, 2005.

[14] Mirmirani, M. et al. “Modeling for control of a generic airbreathing hypersonic vehicle”. In: *AIAA guidance, navigation, and control conference and exhibit*. 2005, p. 6256. DOI: 10.2514/6.2005-6256.

[15] Apkarian, P. and Noll, D. “The H∞ Control Problem is Solved”. In: *Aerospace Lab* (2017). DOI: 10.12762/2017.AL13-01.

[16] MathWorks, Inc. *Robust Control Toolbox, version: 23.2 (R2023b)*. Natick, Massachusetts, United States, 2022. URL: https://www.mathworks.com.

[17] Stevens, B. L., Lewis, F. L., and Johnson, E. N. *Aircraft control and simulation - dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015.

[18] Zipfel, P. H. *Modeling and Simulation of Aerospace Vehicle Dynamics*. 2nd ed. AIAA, 2000.

[19] Williams, J.E. and Vukelich, S.R. *The USAF stability and control digital DATCOM. Volume I. Users manual*. 1979.

[20] Fida, B., Mirmirani, M., and Ioannou, P. A. *Air-Breathing Hypersonic Flight Control*. Conference Paper. 2004. DOI: 10.1016/s1474-6670(17)32234-6.

[21] Shaughnessy, J. D. et al. *Hypersonic Vehicle Simulation Model: Winged-Cone Configuration*. Report. 1990.

[22] Bonner, E., Clever, W., and Dunn, K. *Aerodynamic preliminary analysis system 2. Part 1: Theory*. Tech. rep. 1991.

[23] White, D.A. and Sofge, D.A. *Handbook of Intelligent Control*. Ch. 11. Van Nostrand Reinhold, 1992.

[24] Araki, J. J. "Reentry Dynamics and Handling Qualities of a Generic Hypersonic Vehicle". Thesis. 1992.

[25] Bolender, M. and Doman, D. *A Non-Linear Model for the Longitudinal Dynamics of a Hypersonic Air-breathing Vehicle*. Conference Paper. 2005. DOI: 10.2514/6.2005-6255.

[26] Groves, K. "Modelling, simulation, and control design of an air-breathing hypersonic vehicle". Thesis. 2005. URL: http://rave.ohiolink.edu/etdc/view?acc_num=osu1302726196.

[27] Groves, K. et al. *Reference Command Tracking for a Linearized Model of an Air-Breathing Hypersonic Vehicle*. Conference Paper. 2005. DOI: 10.2514/6.2005-6144.

[28] Packard, A. et al. *Dynamic Systems and Feedback*. University of California, 2018.

[29] Yechout, T. R. *Introduction to Aircraft Flight Mechanics*. American Institute of Aeronautics and Astronautics, 2000. ISBN: 9781563475771.

[30] MathWorks. *Linearization*. URL: https://nl.mathworks.com/help/slcontrol/linearization.html?s_tid=CRUX_lftnav.

[31] Safonov, M. G. *Origins of Robust Control: Early History and Future Speculations*. Conference Paper. 2012. DOI: https://doi.org/10.3182/20120620-3-DK-2025.00179.

[32] Doyle, J. "Guaranteed margins for LQG regulators". In: *IEEE Transactions on Automatic Control* 23.4 (1978), pp. 756–757. DOI: 10.1109/TAC.1978.1101812.

[33] Stein, G. "The practical, physical (and sometimes dangerous) consequences of control must be respected, and the underlying principles must be clearly and well taught." In: *IEEE Control Systems Magazine* (2003).

[34] Bates, D. and Postlethwaite, I. *Robust Multivariable Control of Aerospace Systems*. 2002.

[35] Apkarian, P. and Noll, D. "Nonsmooth optimization for multidisk H∞ synthesis". In: *European Journal of Control* 12.3 (2006), pp. 229–244.

[36] Seiler, P., Packard, A., and Gahinet, P. "An introduction to disk margins". In: *IEEE Control Systems Magazine* 40.5 (2020), pp. 78–95.

[37] Balas, Gary J. "Flight Control Law Design: An Industry Perspective". In: *European Journal of Control* 9.2-3 (2003), pp. 207–226. ISSN: 09473580. DOI: 10.3166/ejc.9.207-226.

[38] De Aguiar, R. S. D. S., Apkarian, P., and Noll, D. "Structured Robust Control Against Mixed Uncertainty". In: *IEEE Transactions on Control Systems Technology* 26.5 (2018), pp. 1771–1781. DOI: 10.1109/TCST.2017.2723864.

A

Appendix

Table A.1: C_{m_0} aerodynamic table (Part 1)

α \ M	0.4	0.6	0.8	0.9	0.95	1.05	1.2
-3	-0.00332	-0.00379	-0.00383	-0.00449	-0.00411	-0.00457	-0.00403
0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
3	0.00187	0.00216	0.00263	0.00270	0.00322	0.00304	0.00347
6	0.00245	0.00288	0.00360	0.00375	0.00468	0.00457	0.00545
9	0.00221	0.00298	0.00398	0.00454	0.00565	0.00599	0.00670
12	0.00182	0.00308	0.00502	0.00562	0.00760	0.00785	0.00857
15	0.00315	0.00463	0.00702	0.00757	0.01008	0.01001	0.01087
18	0.01029	0.01030	0.01232	0.01080	0.01408	0.01182	0.01396
21	0.00149	0.00406	0.00873	0.00846	0.01373	0.01117	0.01555

Table A.2: C_{m_0} aerodynamic table (Part 2)

α \ M	1.5	2.0	3.0	6.0	12.0	24.0
-3	-0.00403	-0.00364	-0.00331	-0.00312	-0.00275	-0.00254
0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
3	0.00310	0.00335	0.00292	0.00300	0.00267	0.00254
6	0.00512	0.00571	0.00526	0.00546	0.00507	0.00493
9	0.00699	0.00691	0.00734	0.00686	0.00721	0.00706
12	0.00912	0.00809	0.00932	0.00783	0.00895	0.00869
15	0.01121	0.00970	0.01121	0.00915	0.01075	0.01051
18	0.01241	0.01193	0.01229	0.01078	0.01203	0.01195
21	0.01229	0.01501	0.01280	0.01409	0.01349	0.01395

Table A.3: C_{m_α} aerodynamic table (Part 1)

α \ M	0.4	0.6	0.8	0.9	0.95	1.05	1.2
-3	-0.00127	-0.00145	-0.00146	-0.00171	-0.00157	-0.00174	-0.00154
0	-0.00100	-0.00120	-0.00125	-0.00152	-0.00142	-0.00162	-0.00145
3	-0.00071	-0.00082	-0.00100	-0.00103	-0.00123	-0.00116	-0.00132
6	-0.00047	-0.00055	-0.00069	-0.00071	-0.00089	-0.00087	-0.00104
9	-0.00028	-0.00038	-0.00050	-0.00058	-0.00072	-0.00076	-0.00085
12	-0.00017	-0.00029	-0.00048	-0.00054	-0.00072	-0.00075	-0.00082
15	-0.00024	-0.00035	-0.00054	-0.00058	-0.00077	-0.00076	-0.00083
18	-0.00065	-0.00065	-0.00078	-0.00069	-0.00089	-0.00075	-0.00089
21	-0.00008	-0.00022	-0.00048	-0.00046	-0.00075	-0.00061	-0.00085

Table A.4: Cm_α aerodynamic table (Part 2)

α \ M	1.5	2.0	3.0	6.0	12.0	24.0
-3	-0.00154	-0.00139	-0.00126	-0.00119	-0.00105	-0.00097
0	-0.00148	-0.00135	-0.00124	-0.00117	-0.00105	-0.00097
3	-0.00118	-0.00128	-0.00111	-0.00114	-0.00102	-0.00097
6	-0.00097	-0.00109	-0.00100	-0.00104	-0.00097	-0.00094
9	-0.00089	-0.00088	-0.00093	-0.00087	-0.00092	-0.00090
12	-0.00087	-0.00077	-0.00089	-0.00075	-0.00085	-0.00083
15	-0.00085	-0.00074	-0.00085	-0.00070	-0.00082	-0.00080
18	-0.00079	-0.00076	-0.00078	-0.00068	-0.00076	-0.00076
21	-0.00067	-0.00082	-0.00070	-0.00077	-0.00073	-0.00076

Table A.5: C_{m_1} aerodynamic table (Part 1)

α \ M	0.4	0.6	0.8	0.9	0.95	1.05	1.2
-3	0.0005	0.0006	0.0006	0.0006	0.0006	0.0006	0.0005
0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
3	-0.0003	-0.0003	-0.0004	-0.0005	-0.0005	-0.0004	-0.0004
6	-0.0004	-0.0005	-0.0007	-0.0008	-0.0007	-0.0007	-0.0007
9	-0.0003	-0.0004	-0.0005	-0.0007	-0.0008	-0.0008	-0.0007
12	-0.0002	-0.0006	-0.0011	-0.0014	-0.0010	-0.0014	-0.0014
15	-0.0014	-0.0014	-0.0017	-0.0019	-0.0017	-0.0017	-0.0015
18	-0.0002	-0.0006	-0.0014	-0.0012	-0.0020	-0.0016	-0.0012
21	0.0005	0.0004	0.0006	0.0006	0.0006	0.0005	0.0005

Table A.6: C_{m_1} aerodynamic table (Part 2)

α \ M	1.5	2.0	3.0	6.0	12.0	24.0
-3	0.0005	0.0005	0.0004	0.0004	0.0004	0.0004
0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
3	-0.0004	-0.0004	-0.0004	-0.0004	-0.0004	-0.0004
6	-0.0007	-0.0008	-0.0010	-0.0010	-0.0011	-0.0010
9	-0.0009	-0.0010	-0.0012	-0.0012	-0.0010	-0.0010
12	-0.0011	-0.0011	-0.0013	-0.0013	-0.0013	-0.0012
15	-0.0016	-0.0017	-0.0019	-0.0018	-0.0022	-0.0019
18	-0.0018	-0.0018	-0.0020	-0.0016	-0.0018	-0.0019
21	0.0005	0.0004	0.0006	0.0006	0.0006	0.0005

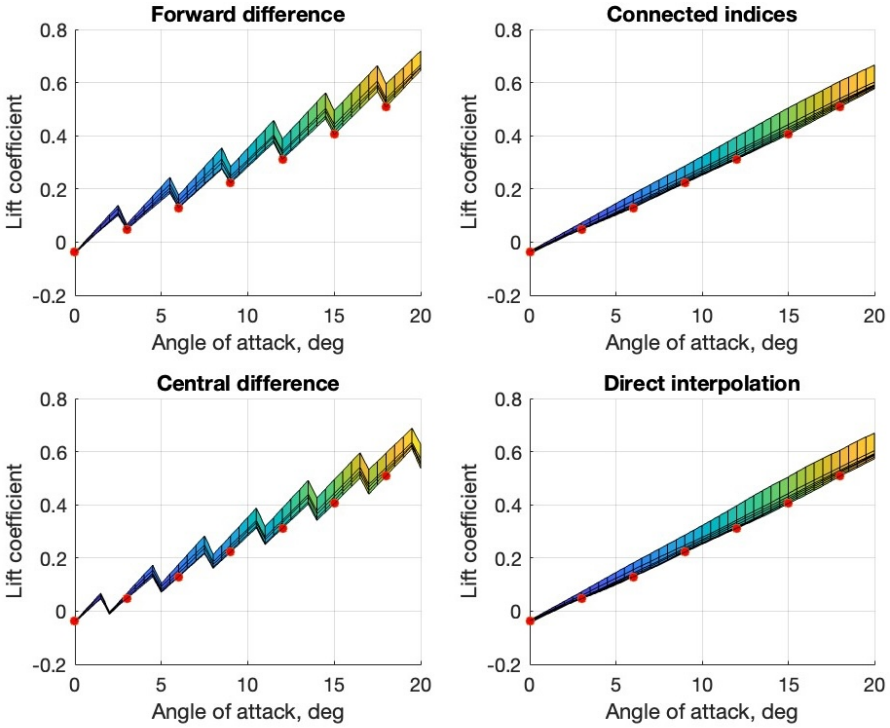


Figure A.1: Various interpretations of lift coefficient data, M = 0.4-1.0

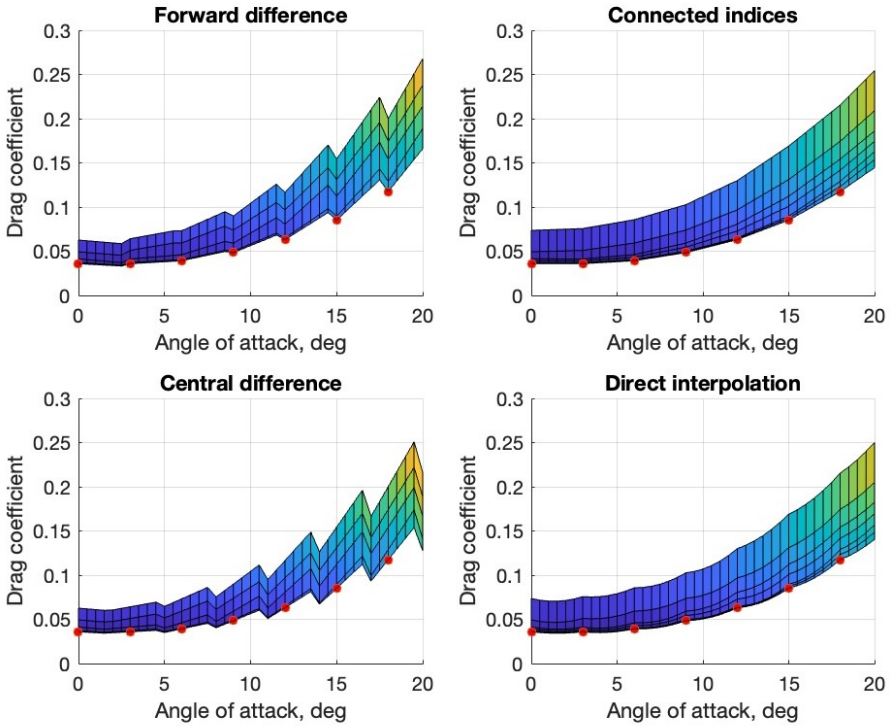
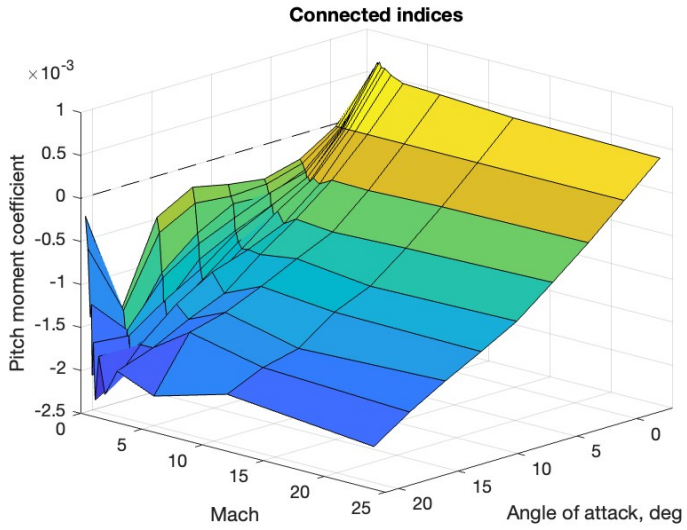
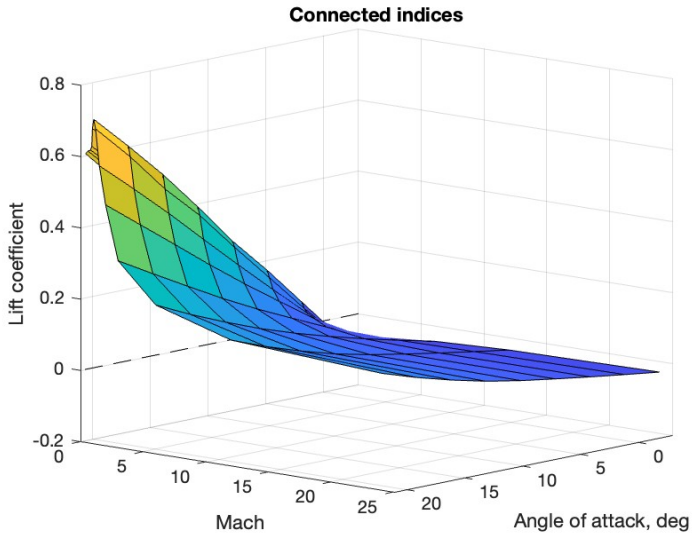


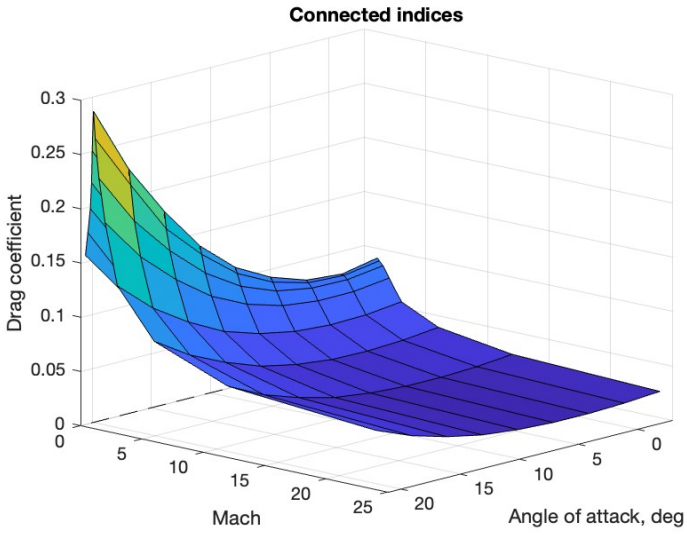
Figure A.2: Various interpretations of drag coefficient data, M = 0.4-1.0



(a) Pitch moment coefficient C_{m_1}

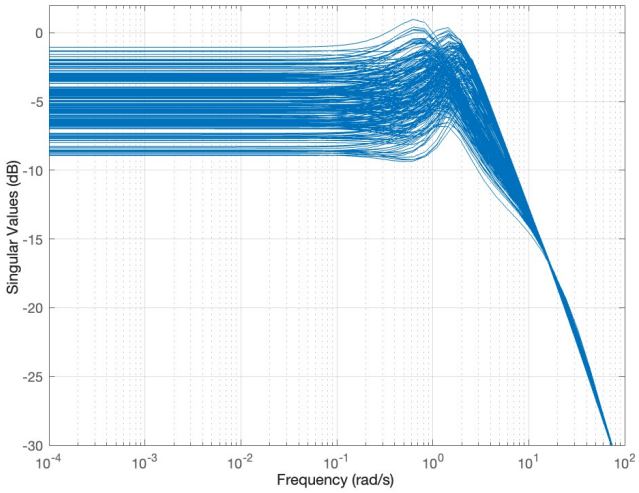


(b) Lift coefficient C_{L_1}

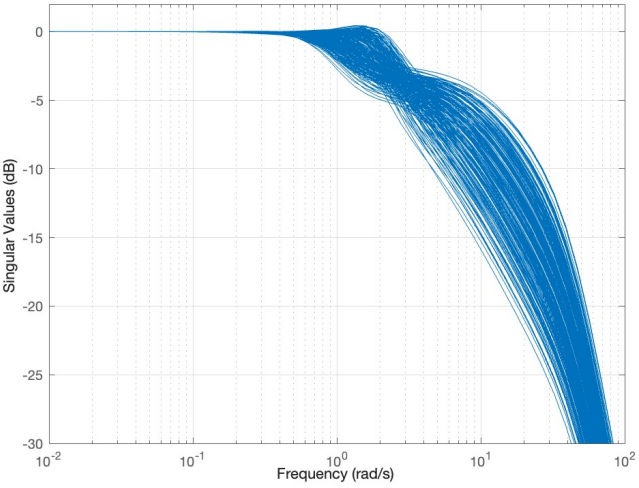


(c) Drag coefficient C_{D_1}

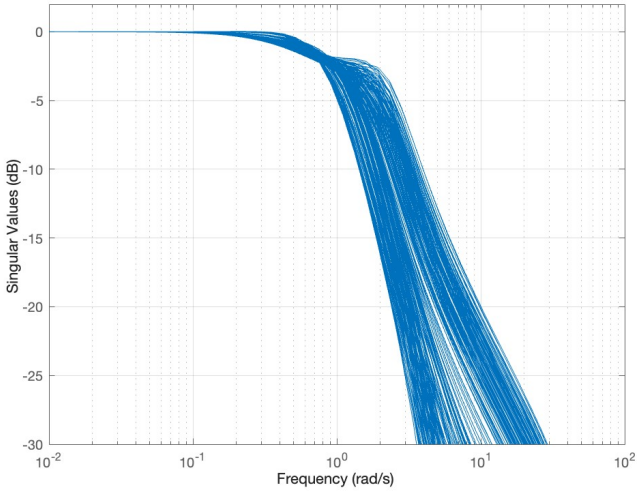
Figure A.3: Lift C_{L_1}, drag C_{D_1}, and pitch moment C_{m_1} coefficients for all angle of attack and Mach number indices



(a) Singular values of $K_{ff}KS_O$ $(r \rightarrow \delta_{e,cmd})$

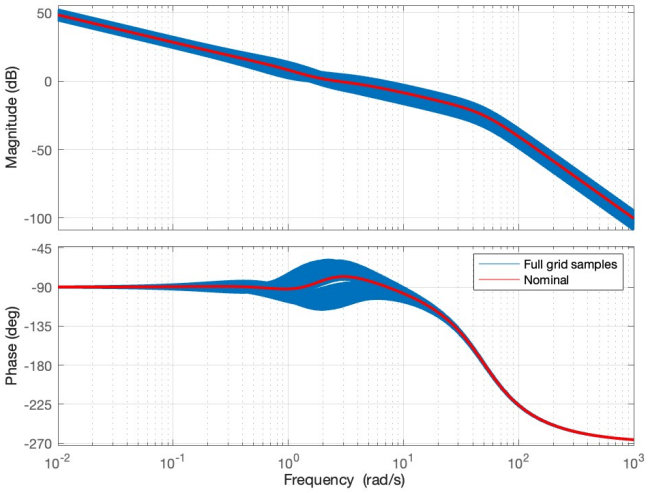


(b) Singular values of T_I $(d_i \rightarrow \delta_{e,cmd})$

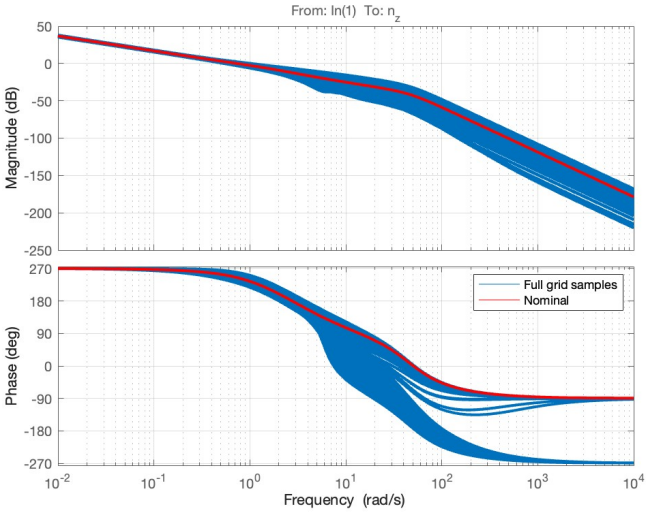


(c) Singular values of T_O $(r \rightarrow n_z)$

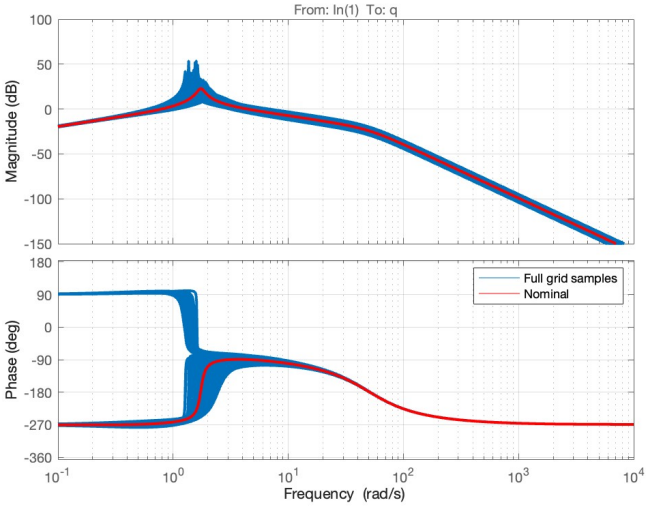
Figure A.4: Singular values of transfer functions related to robust stability against unstructured uncertainty



(a) Open loops at plant input



(b) Open loops at plant output n_z



(c) Open loops at plant output q

Figure A.5: Bode plots of sampled open loops across the model grid