University of Stavanger

Faculty of Science and Technology

BACHELOR'S THESIS

Study program/ Specialization: Control Engineering and Circuit Design - Bachelor's Degree Programme	Spring semester, 2022 Open
Writer: Mathias Dyvik	Mathias Dyvik (Writer's signature)
Faculty supervisor: Damiano Rotondo	
Thesis title: *Polytopic quasi-LPV modeling, identification, and LQR control of a Quanser Aero*	
Credits (ECTS): 20	
Key words: Control theory Modeling Simulation	Pages: 78 + enclosure: 29 Stavanger, 15/5 - 22 Date/year

Acknowledgements

I would like to thank my supervisor Damiano Rotondo for his patience and excellent guidance during this project, as well as for presenting an interesting thesis topic. Also, I wish to thank Didrik Efjestad Fjereide for his assistance in setting up the Aero laboratory system. Additionally, I am grateful for the support I received from my colleagues at Troll C during this project. Lastly, I would like to thank my family and close friends for their patience and support.

- Mathias Dyvik

Abstract

This thesis describes the quasi-LPV modeling of a 2DOF dual-rotor lab experiment known as the Quanser Aero. The quasi-LPV model is based on a nonlinear model derived from Newton's law and Euler's rotational dynamics. The unknown model parameters have been identified through an experimental approach, and the model has been validated through simulation and online testing. In addition, a robust LQR state feedback controller has been designed based on LMI theory. To make the LMIs computationally solvable, the quasi-LPV model has been approximated in a polytopic way. Two different bounding box approaches have been applied to approximate the quasi-LPV model in order to compare the effects of a simple polytopic representation and a more advanced representation. The more straightforward representation derived by a bounding box method and the more advanced approach derived by an SVD-based box method are described and compared in detail. No control experiments have been conducted due to infeasible LMI solutions due to too large variations within the parameters. However, suggestions have been made as to how a feasible solution may be found.

Sammendrag

Denne avhandlingen beskriver utviklingen av en quasi-LPV modell for et laboratorieoppsett med to frihetsgrader, kalt Quanser Aero. Quasi-LPV modellen er utviklet med bakgrunn i en ulineær dynamisk modell av Aero systemet. Denne ulineære modellen er utarbeidet ved bruk av Newtons lover og Eulers rotasjons lover. Ukjente parametere som inngår i modellen, er identifisert gjennom en eksperimentell tilnærming. Modellen er verifisert ved bruk av simulering og sammenligninger mot målinger fra det faktiske Aero systemet. I tillegg er det utviklet en LQR regulator basert på tilstandstilbakekobling (*State feedback*), ved bruk av lineære matrise ulikheter (LMI). For å matematisk kunne løse disse ulikhetene er quasi-LPV modellen approksimert gjennom konveks polytopisk tilnærming. Avhandlingen ser på to ulike typer polytopisk approksimasjoner med hensikt i å analysere effekten av å bruke en avanserte plytopisk metode kontra em enkel. De to anvendte metodene består av en enklere metode kjent som *bounding box* metoden og en mer avansert metode kalt *SVD-based box* metoden. Metodene er sammenlignet og beskrevet i detalj. Denne avhandlingen har ikke sett på en faktisk regulering av Aero systemet grunnet at de lineære matrise ulikhetene knyttet til regulator designet, ikke lot seg løse. Dette skyldes at de varierende LPV parameterne inneholder for mye variasjon. Det er allikevel presentert flere løsninger som kan lede til ulikheter som kan løses og dermed resultere i en brukbar regulator.

Contents

Nomenclature

Symbol	Description
$\mathbf{A}$	State matrix in a state-spacemodel
$\mathbf{B}$	Input matrix in a state-spacemodel
$\mathbf{C}$	Output matrix in a state-spacemodel
$\mathbf{D}$	Feedthrough matrix in a state-spacemodel
$\mathbf{K}$	Gain matrix in a state-feedback controller
N	Number of vertices in a polytopic representation
$\mathbf{u}$	Input vector in a state-spacemodel
$\mathbf{P}$	Lyapunov matrix
$\mathbf{x}$	State vector in a state-spacemodel
$\mathbf{y}$	Output vector in a state-spacemodel
$\mathbf{r}$	Control reference
μ	Coefficient of a polytopic presentation
$\boldsymbol{\Psi}$	Varying parameter vector
ψ	Varying parameter

List of Acronyms

Abbreviation	Description
DAQ	Data Acquisition Device
DC	Direct current
DOF	Degrees of freedom
Eq.	Equation
IMU	Inertial Measurement Unit
LMI	Linear Matrix Inequality
LPV	Linear Parameter Varying
LQR	Linear Quadratic Regulator
MIMO	Multiple-inputs-multiple-outputs
I/O	Input/Output
UAV	Unmanned aerial vehicle
SPI	Serial Peripheral Interface Bus
SVD	Singular Value Decomposition
PWM	Pulse Width Modulation

List of Symbols

Symbol	Description	Unit
θ_p	Pitch angle of the Aero	rad
$\dot{\theta}_p = \Omega_p$	Angular velocity of the Aero around the pitch axis	rad/s
θ_y	Yaw angle of the Aero	rad

$\dot{\theta}_y = \Omega_y$	Angular velocity of the Aero around the yaw axis	rad/s
ω_p	Angular velocity of the main rotor	rad/s
ω_y	Angular velocity of the tail rotor	rad/s
$\mathcal{T}_g$	Gravitational torque on pitch body	Nm
$\mathcal{T}_{Mp}$	Thrust torque from main rotor	Nm
$\mathcal{T}_{Ty}$	Thrust torque from tail rotor	Nm
$\mathcal{T}_{My}$	Cross-thrust torque from main rotor	Nm
$\mathcal{T}_{Tp}$	Cross-thrust torque from tail rotor	Nm
$\mathcal{T}_{Dp}$	Torque from damping of the pitch body	Nm
$\mathcal{T}_{Dy}$	Torque from damping of the yaw body	Nm
$\mathcal{T}_{Rp}$	Torque from Coulomb friction in the pitch joint	Nm
$\mathcal{T}_{Ry}$	Torque from Coulomb friction in the yaw joint	Nm
$\mathcal{T}_{Cm,p}$	Centripetal torque from the main rotor	Nm
$\mathcal{T}_{Ct,p}$	Centripetal torque from the tail rotor	Nm
$\mathcal{T}_{Cp}$	Total Centripetal torque	Nm
$\mathcal{T}_{Fy}$	Friction torque torque for yaw motion	Nm
F_{Fy}	Coulomb friction in the yaw joint	Nm
d_m	Center of mass displacement	m
d_t	Thrust displacement	m
F_{Mp}	Thrust force from main rotor	N
F_{Ty}	Thrust force from tail rotor	N
F_{My}	Cross-thrust force from main rotor	N
F_{Tp}	Cross-thrust force from tail rotor	N
F_g	Gravitational force	N
F_{cm}	Centripetal force on the main rotor	N
$F_{cm,p}$	Vertical Centripetal force on the main rotor	N
F_{ct}	Centripetal force on the tail rotor	N
$F_{ct,p}$	Vertical Centripetal force on the tail rotor	N
$f_a(\omega_p)$	Drag and air resistance torque main and tail rotor, defined in (2.8)	Nm
$f_{Mp}(\omega_p)$	Main motor thrust force, defined in (2.16)	N
$f_{Tp}(\omega_p)$	Tail motor cross-thrust force, defined in (2.16)	N
$f_{Dp}(\Omega_p)$	Viscous damping of the pitch body, defined in (2.18)	Nm
$f_{Dy}(\Omega_y)$	Viscous damping of the yaw body, defined in (2.34)	Nm
g	Gravitational acceleration at sea level	m/s^2
J_{beam}	Moment of inertia of the beam	kgm^2
J_{yoke}	Moment of inertia of the forked yoke	kgm^2
J_{prop}	Moment of inertia of the main and tail propeller	kgm^2
J_{motor}	Moment of inertia of the main and tail motor	kgm^2
J_{hub}	Moment of inertia collect clamp between the propellers and motors	kgm^2
J_r	Total moment of inertia of the rotors	kgm^2
J_p	Total moment of inertia of the pitch body	kgm^2
$J_y(\theta_p)$	Total moment of inertia of the yaw body, defined in (2.49)	kgm^2
L_a	Armature inductance for the DC motors	H
m_b	Mass of the Aero pitch body	kg
m_{rm}	Mass of main rotor assembly	kg

m_{tm}	Mass of tube connecting the main rotor	kg
m_{em}	Mass of the main DC motor	kg
m_{rt}	Mass of tail rotor assembly	kg
m_{tt}	Mass of tube connecting the tail rotor	kg
m_{et}	Mass of the tail DC motor	kg
m_{tc}	Mass of the tube clamp connecting the main and tail tubes	kg
K_{τ}	Torque constant	Nm/a
K_E	Back emf constant	$V/rad \cdot s$
k_{Mpp_1}	Thrust force coefficient of the main rotor for positive ω_p	$\frac{N}{(rad/s)^2}$
k_{Mpp_2}	Thrust force coefficient of the main rotor for positive ω_p	$\frac{N}{rad/s}$
k_{Mpn_1}	Thrust force coefficient of the main rotor for negative ω_p	$\frac{N}{(rad/s)^2}$
k_{Mpn_2}	Thrust force coefficient of the main rotor for negative ω_p	$\frac{N}{rad/s}$
k_{Myp_1}	Cross-thrust force coefficient of the main rotor for positive ω_p	$\frac{N}{(rad/s)^2}$
k_{Myp_2}	Cross-thrust force coefficient of the main rotor for positive ω_p	$\frac{N}{rad/s}$
k_{Myn_1}	Cross-thrust force coefficient of the main rotor for negative ω_p	$\frac{N}{(rad/s)^2}$
k_{Myn_2}	Cross-thrust force coefficient of the main rotor for negative ω_p	$\frac{N}{rad/s}$
k_{Tyn_1}	Thrust force coefficients of the tail rotor for positive ω_y	$\frac{N}{(rad/s)^2}$
k_{Tyn_2}	Thrust force coefficients of the tail rotor for positive ω_y	$\frac{N}{rad/s}$
k_{Tpp_1}	Cross-thrust force coefficient of the tail rotor for positive ω_y	$\frac{N}{(rad/s)^2}$
k_{Tpp_2}	Cross-thrust force coefficient of the tail rotor for positive ω_y	$\frac{N}{rad/s}$
k_{Tpn_1}	Cross-thrust force coefficient of the tail rotor for negative ω_y	$\frac{N}{(rad/s)^2}$
k_{Tpn_2}	Cross-thrust force coefficient of the tail rotor for negative ω_y	$\frac{N}{rad/s}$
k_{Dp_1}	Viscous damping coefficient for the pitch axis	$\frac{Nm}{(rad/s)^2}$
k_{Dp_2}	Viscous damping coefficient for the pitch axis	$\frac{Nm}{rad/s}$
k_{Dy_1}	Viscous damping coefficient for the yaw axis	$\frac{Nm}{(rad/s)^2}$
k_{Dy_2}	Viscous damping coefficient for the yaw axis	$\frac{Nm}{rad/s}$
k_{d_1}	Drag and air resistance coefficient for the main and tail propellers	$\frac{Nm}{(rad/s)^2}$
k_{d_2}	Drag and air resistance coefficient for the main and tail propellers	$\frac{Nm}{rad/s}$
k_{Fy_p}	Friction coefficient for positive Ω_y	$\frac{Nm}{rad/s}$
k_{Fy_n}	Friction coefficient for negatative Ω_y	$\frac{Nm}{rad/s}$
r_t	Radius of rotation about the yaw axis	m
R_a	Armature resistance for the DC motors	Ω
T_m	Tension force exerted on the horizontal tube by the main rotor	N
T_t	Tension force exerted on the horizontal tube by the tail rotor	N
T_{mp}	Vertical component of the tension force T_m	N
T_{tp}	Vertical component of the tension force T_p	N

List of Figures

List of Tables

Chapter 1

Introduction

The rapid development of electronics and microcontrollers over the last decade has led to an increased interest in unmanned air vehicles (UAVs), such as quadcopters and other drones. UAVs have proved to be helpful in a variety of fields, including military operations, goods transportation, and photography. In order for these vehicles to operate safely in challenging flight conditions, advanced control systems are often necessary. In this project a control structure is developed for a dual-motor lab experiment, called the Aero. This device resembles the behavior of a helicopter while remaining stationary.

1.1 Motivation

When developing complex control systems a mathematical model is very useful. With a reliable mathematical model, simulations, stability analysis, and sophisticated controller design are possible. However, most real-life physical systems have properties that result in a nonlinear model. Due to the difficulty of solving nonlinear dynamical equations, nonlinear models are often linearized for the system to be analysed. Nevertheless, a linerazied nonlinear system is only valid in a small region around the point where it was linerazied. In strong nonlinear systems with multiple operating points, this may be problematic. Throughout this thesis, a modeling technique using linear parameter varying systems (LPV) is used. LPV systems are a paradigm that falls between linear and nonlinear systems, by embedding the nonlinearities in parameters whose values vary based on the state of the system. Thus, a strongly nonlinear system can be represented in a linear framework without the need to introduce linearization.

1.2 Project Description

The main objectives of this thesis have been to:

- Perform a literature review regarding previous research related to the Quanser Aero and similar devices.
- Derive a nonlinear model that describes the dynamics of the 2 DOF Aero configuration.

- Learn the fundamental concepts of LPV theory and apply the principles to obtain a quasi-LPV model of the 2 DOF Aero configuration.
- Design a robust LQR controller for the Aero.
- Develop and evaluate different polytopic representations of the quasi-LPV model
- Analyze the stability of the robust control system based on linear matrix inequality theory.

1.3 Previous Work

The Quanser Aero and similar helicopter devices have been addressed in many research papers and theses, as such devices present an interesting and challenging control problem. In a master thesis by Gabrielsen and Frasik [7], a nonlinear model of the Aero is derived using an experiential identification approach. The resulting model is then used to develop a number of different controllers including cascade P-PI, linear quadratic regulator (LQR) state-feedback, model predictive control (MPC) and model reference adaptive control (MRAC). In a research paper by Kumar and Dewan [11] a nonliear model of the Aero is derived from applying the Euler–Lagrange equation, in order to develop and compare two controllers based on LQR control and sliding mode control (SMC). The paper also conducted a controllability and observability analysis where it was concluded that the Aero system is completely controllable and completely observable. In a research paper by Rotondo, Nejjari and Puig [22], an LPV state observer and state-feedback controller was implemented on a dual-rotor device which is similar to the Quanser Aero. In this study, a nonlinear model of the device, called a twin rotor multiple-input multiple-output system (TRMS), was first identified and then represented as a quasi-LPV system in a polytopic way. The polytopic representation was obtained using a method called the bounding box method. This method resulted in an efficient controller design with satisfactory results both in simulation and with the real device. However, this study did not investigate if a more advanced polytopic representation could lead to a better closed-loop performance.

1.4 Outline

This thesis is structured as follows:

- Chapter 2 describes the Quanser Aero system as well as the nonlinear modeling and identification of the Aero system.
- Chapter 3 describes LPV systems and how they may be analysed and controlled.
- Chapter 4 introduces two bounding box approaches for generating convex hulls, and shows how a polytopic quasi-LPV representation of the Aero was obtained.
- Chapter 5 presents results from solving LMIs in order to obtain a stable LQR state feedback controller.
- Chapter 6 provides a conclusion and suggestions for future work.

Chapter 2

Description and Non-linear Modeling

2.1 Description of the system

The Aero is a dual-rotor laboratory experiment developed by Quanser Consulting Inc, a company that specializes in lab equipment for education and research in control theory, robotics, and mechatronics. The experiment is reconfigurable, meaning that the rotor assembly can be tilted to desirable angles. The main configurations of the Aero resemble either a half-quadrotor or a helicopter, both simplified with fewer degrees of freedom (DOF) than the real aerospace systems. In mechanics, the term DOF is a measure of the number of unique ways that a body can move or rotate in space. The half-quadrotor has one DOF, while the helicopter configuration has two DOF. In this project, the Aero will be used as a two DOF helicopter, where the rotors are perpendicular, as seen in Figure 2.1. This setup is recognized as a challenging problem in the field of control engineering due to its high non-linearity and cross-coupling effects.

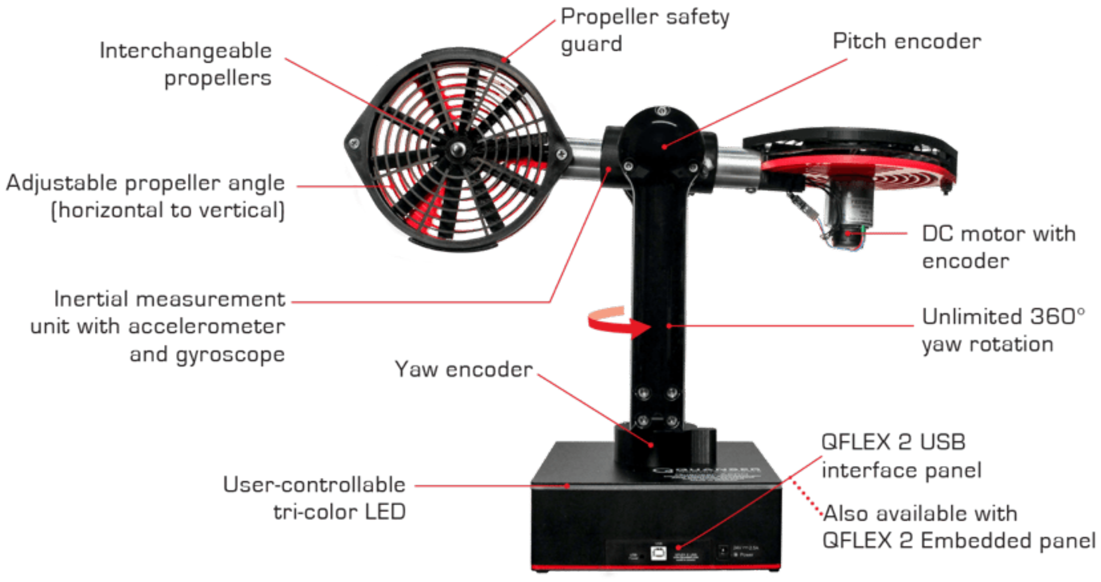

Interchangeable propellers
Propeller safety guard
Pitch encoder
Adjustable propeller angle (horizontal to vertical)
DC motor with encoder
Inertial measurement unit with accelerometer and gyroscope
Unlimited 360° yaw rotation
Yaw encoder
QFLEX 2 USB interface panel
User-controllable tri-color LED
Also available with QFLEX 2 Embedded panel

Figure 2.1: Components of the Quanser Aero [21]

The main parts of the Aero are divided into three categories:

1. *The base* is a stationary box at the bottom of the Aero. It contains electric and electronic system components such as PWM amplifiers, I/O-modules, embedded USB and SPI interfaces, and a data acquisition device (DAQ). The Aero is controlled externally by either a PC or a microcontroller, where the DAQ serves as an interface between the external controller and the internal components.

2. *The support-yoke* has the shape of a fork and stands vertically on the Aero base. Its purpose is to raise the Aero body from the base and enable it to rotate around its vertical axis. A rotational joint between the base and the bottom of the support yoke allows for a 360° unlimited rotation.

3. *The Aero body* is the part that resembles the body of a helicopter. It comprises two coupled metal tubes held together by a tube clamp with a pair of identical rotor blades at both ends. The Aero body is connected to the support-yoke through a rotational joint placed in the midpoint between the two rotors. This connection allows the inclination of the pitch body to be rotated between + 54° and - 62° relative to the support yoke.

The Quanser Aero comes with two pairs of interchangeable propellers, one pair of low efficiency and one pair of high efficiency. The low-efficiency propellers are used in this project as they produce a significantly larger cross-torque than the ones of high efficiency.

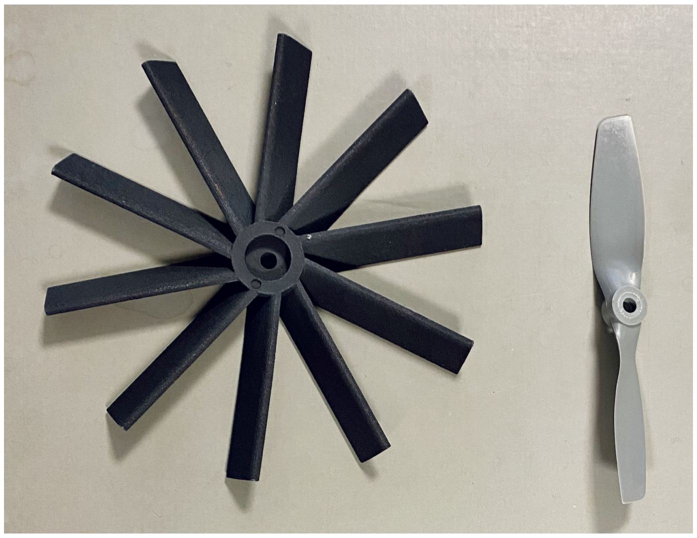


Figure 2.2: 3d-printed 8-bladed low-efficiency propellers (left), high-efficiency propellers produced by Advanced Precision Composites (right)

The helicopter configuration of the Aero has two rotational DOF and zero transnational DOF, meaning that it can rotate in two unique ways while the base remains stationary. However, a real helicopter has three rotational DOF and three transnational DOF, where the transnational DOF includes up-down, front-back, and sideways movement. The terms *pitch*, *yaw*, and *roll* are commonly used in aerospace engineering to describe the rotational DOF of any given aircraft. A pitch rotation is a change in the inclination of an aircraft´s body relative to a defined steady equilibrium state. A yaw rotation occurs when the aircraft rotates around its vertical axis, while a roll is a rotation around the horizontal axis. Figure 2.3 shows the three rotational DOF of a traditional helicopter. The Aero has the ability to pitch and yaw but not roll.

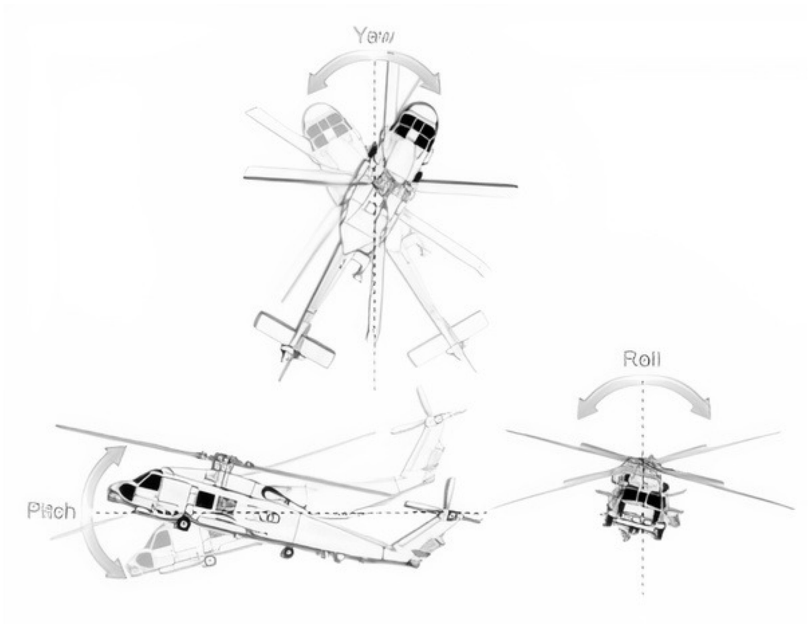


Figure 2.3: Pitch, yaw, and roll, which make up the three rotational degrees of freedom (DOF) of a traditional helicopter [2]

2.1.1 Measurement instrumentation

The Aero is equipped with four optical rotary encoders that measure the angular position of each DC-motor, in addition to the pitch and yaw angle of the Aero body. The encoders are incremental, meaning that they measure the change in angular position rather than the absolute position. The consequence of this property is that the zero angle points of the different positions are given by their initial position when the Aero is powered on. This effect does not present an issue for the pitch position as the Aero is construed so that it always returns to a horizontal equilibrium when the rotors are shut off. However, the yaw position must be set manually to a desired starting point before powering the Aero.

The use of encoders for measuring pitch and yaw is not feasible in a real aerospace system. This is because an encoder has a stationary and a moving part. The stationary part has to be fixed to a non-moving base in order to measure the angular position of a moving object. Therefore, the Aero is also equipped with a more realistic device called an inertial measurement unit (IMU). The IMU used on the Aero consist of a combination of an accelerometer and a gyroscope and is located in the center of the Aero body. The accelerometer measures acceleration along the three principal axes x, y, and z, while the gyroscope measures angular velocities around the principal axes. Methods for estimating the pitch and yaw positions of the Aero body are given in the Quanser lab documentation [20]. The pitch angle is estimated solely through acceleration measurements from the accelerometer. Quanser assumes that the gravitational pull is the only linear acceleration acting on the IMU as the Aero is immobile. This means that the acceleration along the y axis can be neglected as it will always be perpendicular to the acceleration of gravity. The acceleration caused by gravity along the x and z axes at arbitrary pitch angles can then be expressed as:

$$a_x(t) = a_g(t)\sin(\theta_p(t)) \tag{2.1}$$

$$a_y(t) = a_g(t)\cos(\theta_p(t)) \tag{2.2}$$

where a_g is acceleration caused by gravity, a_x is the gravitational acceleration along the x axis, a_y is the gravitational acceleration along the y axis and $\theta_p(t)$ is the pitch angle of the Aero body.

The pitch angle is then be found by dividing a_x by a_y and solving for $\theta_p(t)$:

$$\theta_p(t) = \tan^{-1}\left(\frac{a_x(t)}{a_y(t)}\right) \tag{2.3}$$

The accelerometer can not be used to measure the yaw of a body. Therefore, yaw position is often estimated by integrating the angular yaw velocity measured by the gyroscope. Like the incremental encoder, this method only estimates the change in angular position from an initial angular point $\theta_y(0)$. The yaw angle $\theta_y(t)$ of the Aero can be estimated as:

$$\theta_y(t) = \theta_y(0) + \int_0^t \Omega_y(t)dt \tag{2.4}$$

where $\Omega_y(t)$ is the angular yaw velocity.

As the main focus of this thesis lies in the modeling and control of the Aero, the more precise encoder signals will be used for parameter estimation and online measurement of the state variables. The different angular velocities is found by taking the time derivative of the measured angular positions.

2.2 Modeling

The first step when building an LPV model is to obtain a mathematical representation of the system of interest. In this section, six differential equations describing the dynamic of the Aero are derived.

2.2.1 DC motors

The Quanser Aero is equipped with a main rotor and a tail rotor, each of which is driven by a brushed DC motor. By applying Kirchhoff's voltage law, the motor armature circuits can be modeled as two linear first-order differential equations:

$$\frac{di_{ap/y}(t)}{dt} = \frac{R_a}{L_a} i_{ap/y}(t) - \frac{K_E}{L_a}\omega_{p/y}(t) + v_{p/y}(t) \tag{2.5}$$

The torque produced by each DC motor $\mathcal{T}_{mp/y}$ is proportional with the armature current $i_{ap/y}(t)$,

$$\mathcal{T}_{mp/y} = K_{\mathcal{T}} i_{ap/y}(t) \tag{2.6}$$

where $K_{\mathcal{T}}$ is a torque constant. All of the motor and propeller parameters are identical on the two rotors.

2.2.2 Motor shaft and propeller dynamics

The Quaenser lab guides suggest a linear function for the torque caused by drag and air resistance:

$$\mathcal{T}_d(t) = k_d \omega_m(t) \tag{2.7}$$

where $\mathcal{T}_d(t)$ is the drag torque which opposes the rotational motion of the rotor, k_d is a drag/air resistance coefficient and ω_m is the rotor speed [20]. However, online testing on the Aero's main rotor revealed that the following non-linear function offers a more accurate representation of the real propeller dynamics:

$$f_a(\omega_{p/y}(t)) = sign(\omega_{y/p}(t))k_{d1}\omega_{p/y}(t)^2 + k_{d2}\omega_{y/p}(t) \tag{2.8}$$

Although data from the main rotor was used to obtain f_a, the function can be applied on both rotors given the high degree of similarity of the two components.

Numerical values for the moments of inertia of the set of propeller hubs J_{hub}, the motors J_{motor}, and propellers J_{prop}, are provided by Quanser [20]. The total moment of inertia acting on the rotor shaft is:

$$J_r = J_{motor} + J_{hub} + J_{prop} \tag{2.9}$$

By applying Newton's second law for rotation, the rotor shaft and propeller dynamics can be expressed as:

$$J_r\ddot{\theta} = \sum_i \mathcal{T}_i \tag{2.10}$$

$$\frac{d\omega_{p/y}(t)}{dt} = \frac{\mathcal{T}_m - \mathcal{T}_d}{J_r} = \frac{K_{\mathcal{T}}}{J_r} i_{ap/y}(t) - \frac{f_d(\omega_{p_y}(t))}{J_r} \tag{2.11}$$

Then, by neglecting the motor inductance L_a in Eq. (2.5) the motor current can be found as:

$$i_{ap/y}(t) = \frac{v_{p/y}(t) - K_E\omega_{p/y}(t)}{R_a} \tag{2.12}$$

Eq. (2.11) can then be rewritten as:

$$\frac{d\omega_{p/y}(t)}{dt} = \frac{K_{\mathcal{T}}}{R_a J_r} v_{p/y}(t) - \frac{K_{\mathcal{T}} K_E}{R_a J_r}\omega_{p/y}(t) - \frac{f_d(\omega_{p/y}(t))}{J_r} \tag{2.13}$$

2.2.3 Kinematic equations for the Aero body

By considering the Aero as a rigid body, Newtons' laws and Euler's rotational dynamics can be applied to obtain the set of nonlinear equations describing the system's motions. The main rotor of the Aero is mounted horizontally. It produces a vertical thrust force F_{Mp}, that enables the Aero to pitch, which is a rotation in the vertical plane around the horizontal axes. However, Newton's third law states that there is an equal and opposite reaction to every action. Therefore, an additional perpendicular force F_{My} is produced, causing a cross-torque and thus a negative yaw rotation, which is a rotaion in the horizontal plane around the vertical axis. This cross-torque property is the main reason why traditional helicopters are constructed with a vertical tail rotor, also called an anti-torque rotor. In addition to counteracting the cross torque, the tail rotor is used to control the yaw angle. The Aero tail rotor is mounted vertically and produces a thrust force F_{Ty} causing a positive rotation about the yaw axes. Similar to the main rotor, an additional perpendicular force F_{Tp} is produced, causing a positive rotation about the pitch axis. The four thrust forces and the gravitational pull are illustrated in a free body diagram in Figure 2.4.

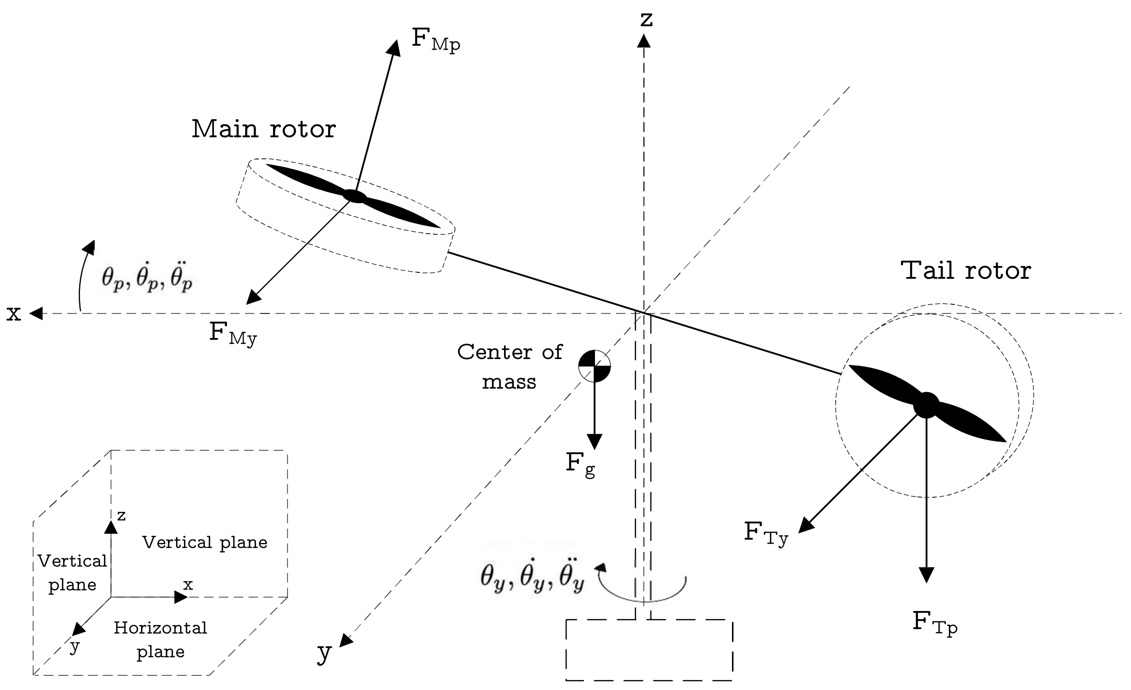


Figure 2.4: Free body diagram of the Aero body

Torques causing the Aero to pitch

- The torque $\mathcal{T}_g$ caused by the gravitational pull on the rigid-body
- The torque $\mathcal{T}_{Mp}$ produced by the thrust force from the main rotor
- The cross-torque $\mathcal{T}_{Tp}$ produced by the tail rotor

- The torque $\mathcal{T}_{Dp}$ caused by viscous damping
- The torque $\mathcal{T}_{Cp}$ caused by the centripetal force on the Aero body

Gravitational Torque

The center of mass of the Aero pitch body is slightly offset from the pivot point. This displacement changes with how the rotors are angled. When the Aero is used as a 1 DOF system, meaning that both the rotors are in a horizontal position, the center of mass for each rotor is below the pivot point. The offset length for the 1 DOF configuration is given in the Quanser documentation. However, when the Aero is used as a 2 DOF helicopter system, only the center of mass of the main rotor is below the pivot point while the tail rotor's center of mass is flush with the pivot point. Therefore, the displacement d_m for the 2 DOF configuration is less than for the 1 DOF, and is not provided by Quanser. In [7], a test where tiny weights where suspended from the pitch body at different distances from the pivot point was used to estimate the displacement of $d_m = 2.9\ mm$. The center of mass displacement is illustrated exaggeratedly in Figure 2.5.

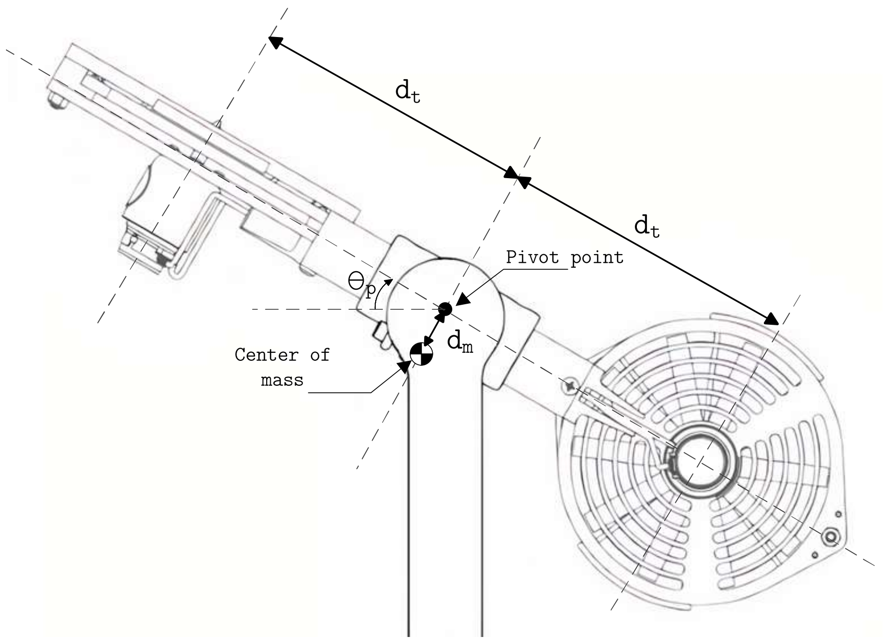


Figure 2.5: Center of mass and thrust displacement. Adapted from [16]

As the center of mass is below the pivot point, the rigid body of the Aero behaves like a pendulum, meaning that it will return to a horizontal orientation with a pitch angle of approximately 0 degrees if no forces are applied. Figure 2.6 illustrates how the center of mass moves with different pitch angles. The gravitational torque is a product of the gravitational force F_g and the moment arm $d_t \sin(\theta_p)$:

$$\mathcal{T}_g(t) = F_g d_m \sin(\theta_p(t)) = m_b g d_m \sin(\theta_p(t)) \tag{2.14}$$

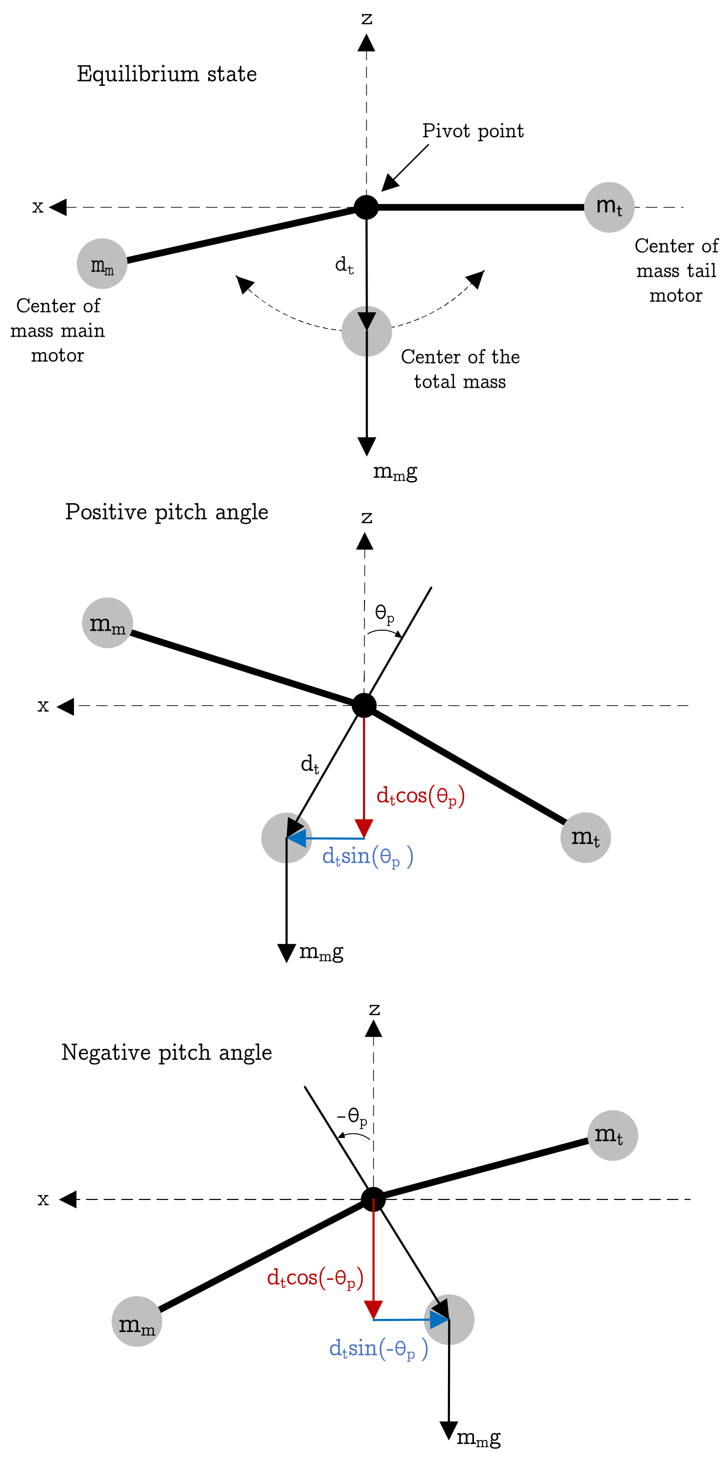


Figure 2.6: Illustration of the pendulum-like behavior of the center of mass of the pitch body

Vertical trust torques

Both the vertical thrust torque form the main rotor and the cross-torque from the tail rotor is obtained by multiplying the force by its moment arm. The arm-length on the Aero d_t, is the distance from the pivot point to the center of the propellers, which is identical for each of the rotors. The two vertical thrust torques are expressed as:

$$\begin{aligned} \mathcal{T}_{Mp}(t) &= F_{Mp}(t)d_t \\ \mathcal{T}_{Tp}(t) &= F_{Tp}(t)d_t \end{aligned} \tag{2.15}$$

where the forces $F_{Mp}(t)$ and $F_{Tp}(t)$ can be described as functions of the propeller velocities:

$$F_{Mp}(t) = f_{Mp}(\omega_p(t)) = \begin{cases} k_{Mpp_1}\omega_p^2(t) + k_{Mpp_2}\omega_p(t) & \text{if } \omega_p(t) \geq 0 \\ k_{Mpn_1}\omega_p^2(t) + k_{Mpn_2}\omega_p(t) & \text{if } \omega_p(t) < 0 \end{cases} \tag{2.16}$$

$$F_{Tp}(t) = f_{Tp}(\omega_y(t)) = \begin{cases} k_{Tpp_1}\omega_y^2(t) + k_{Tpp_2}\omega_y(t) & \text{if } \omega_y(t) \geq 0 \\ k_{Tpn_1}\omega_y^2(t) + k_{Tpn_2}\omega_y(t) & \text{if } \omega_y(t) < 0 \end{cases} \tag{2.17}$$

The fact that the amount of thrust force produced by each rotors changes according to the direction of rotation of the propellers is taken into account by implementing piecewise polynomial functions. The main rotor thrust function $f_{Mp}(\omega_p(t))$ uses the thrust gains $k_{Mpp_{1/2}}$ for positive rotor velocities and $k_{Mpn_{1/2}}$ for negative rotor velocities. Likewise, the tail rotor thrust function $f_{Tp}(\omega_y(t))$ uses the thrust gains coefficients $k_{Tpp_{1/2}}$ for positive rotor velocities and $k_{Tpn_{1/2}}$ for negative rotor velocities.

Vertical damping torque

The vertical damping torque is identified as a non-linear function of the pitch velocity :

$$\mathcal{T}_{Dp}(t) = f_{Dp}(\Omega_p(t)) = sign(\Omega_p(t))k_{Dp_1}\Omega_p(t)^2 + k_{Dp_2}\Omega_p(t) \tag{2.18}$$

where k_{Dp_1} and k_{Dp_2} are damping coefficients. The friction in the pitch joint is neglected.

Centripetal torque

As the Aero rotates about the yaw axis, a force directed toward the center of curvature of the rotation called the centripetal force will drive the pitch body towards its horizontal equilibrium.

The general equation for the centripetal force is defined as,

$$F_c = \frac{mv^2}{r} \tag{2.19}$$

where m is the mass of the moving body, v is the linear velocity, and r is the radius of curvature.

The relationship between linear velocity and angular velocity is $v = \omega r$:

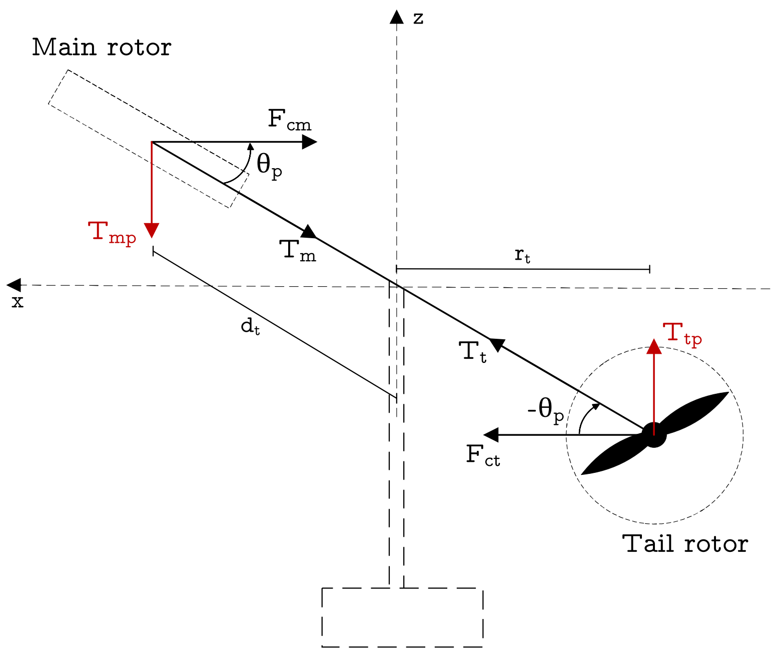


Figure 2.7: The centripetal force acting on the main rotor F_{cm} and the tail rotor F_{cm}.

Using the general formula in Eq. (2.19), the centripetal force on the main rotor can be expressed as:

$$F_{cm}(t) = (\frac{m_{tc}}{2} + m_{tm} + m_{em} + m_{rm})\frac{\Omega_y^2(t)r_t^2}{r_t} = m_A\Omega_y^2(t)r_t \tag{2.20}$$

where $r_t = d_t\cos(\theta_p)$ is the radius of the rotation and $m_A = \frac{m_{tc}}{2} + m_{tm} + m_{em} + m_{rm}$.

The tension force exerted on the horizontal tube by the main motor is then expressed using the centripetal force from Eq.(2.20):

$$T_m(t) = \frac{F_{cm}(t)}{\cos(\theta_p(t))} = \frac{m_A\Omega_y^2(t)d_t\cos(\theta_p(t))}{\cos(\theta_p(t))} = m_A\Omega_y^2(t)d_t \tag{2.21}$$

The vertical component of the main tension force causes a negative rotation about the pitch axis as the Aero rotates about the yaw axis and is found by multiplying by the sin of θ_p:

$$T_{mp}(t) = -T_m(t)\sin(\theta_p(t)) = -m_A\Omega_y^2(t)d_t\sin(\theta_p(t)) \tag{2.22}$$

The torque is derived by multiplying the vertical tension force by its moment arm, r_t:

$$\mathcal{T}_{cm}(t) = -m_A\Omega_y^2(t)d_t\sin(\theta_p(t))r_t = -m_A\Omega_y^2(t)d_t^2\cos(\theta_p(t))\sin(\theta_p(t)) \tag{2.23}$$

The centripetal force acting on the tail rotor is equally obtained as for the main rotor:

$$F_{ct}(t) = (\frac{m_{tc}}{2} + m_{tt} + m_{et} + m_{rt})\frac{\Omega_y^2(t)r_t^2}{r_t} = m_B\Omega_y^2(t)r_t \tag{2.24}$$

where $m_B = \frac{m_{tc}}{2} + m_{tt} + m_{et} + m_{rt}$.

The tension force on the tail tube is:

$$T_t(t) = \frac{F_{ct}(t)}{\cos(-\theta_p(t))} = \frac{m_B\Omega_y^2(t)d_t\cos(-\theta_p(t))}{\cos(-\theta_p(t))} = m_B\Omega_y^2(t)d_t \tag{2.25}$$

The vertical component of the tail tension force also drives the pitch body towards its horizontal equilibrium and is found by multiplying by $\sin(-\theta_p)$:

$$T_{tp}(t) = T_t(t)\sin(-\theta_p(t)) = m_B\Omega_y^2(t)d_t\sin(-\theta_p(t)) \tag{2.26}$$

The torque is then obtained by multiplying the force by its moment arm r_t:

$$\mathcal{T}_{ct}(t) = m_B\Omega_y^2(t)d_t\sin(-\theta_p(t))r_t = m_B\Omega_y^2(t)d_t^2\cos(-\theta_p(t))\sin(-\theta_p(t)) \tag{2.27}$$

The total centripetal torque is found by adding equations (2.23) and (2.27):

$$\begin{aligned}\mathcal{T}_{Cp} &= -m_A\Omega_y^2(t)d_t^2\cos(\theta_p(t))\sin(\theta_p(t)) + m_B\Omega_y^2(t)d_t^2\cos(-\theta_p(t))\sin(-\theta_p(t))\\ &= -2\Omega_y^2(t)d_t^2(m_A+m_B)\cos(\theta_p(t))\sin(\theta_p(t))\\ &= -\Omega_y^2(t)d_t^2(m_A+m_B)\sin(2\theta_p(t))\end{aligned} \tag{2.28}$$

Net torque about the pitch axis

The net torque acting on the pitch axis is:

$$\begin{aligned}\sum_y \mathcal{T}_y &= \mathcal{T}_{Mp}(t) + \mathcal{T}_{Tp}(t) - \mathcal{T}_g(t) - \mathcal{T}_{Dp}(t) - \mathcal{T}_{Cp}(t)\\ &= F_{Mp}(\omega_p(t))d_t + F_{Tp}(\omega_y(t))d_t - m_b g d_m\sin(\theta_p(t)(t)) - f_{Dp}(\Omega_p(t))\\ &\quad - \Omega_y(t)^2 d_t^2(m_A+m_B)\sin(2\theta_p(t))\end{aligned} \tag{2.29}$$

Newton's second law for rotation is then applied to obtain the equation describing the motion about the pitch axis:

$$\begin{aligned}\frac{d\Omega_p(t)}{dt} &= \frac{1}{J_p}(f_{Mp}(\omega_p(t))d_t + f_{Tp}(\omega_y(t))d_t - m_b g d_m\sin(\theta_p(t)) - f_{Dp}(\Omega_p(t))\\ &\quad - \Omega_y(t)^2 d_t^2(m_A+m_B)\sin(2\theta_p(t))\end{aligned} \tag{2.30}$$

where J_p is the total moment of inertia of the pitch body.

Torques causing the Aero to yaw

- The torque $\mathcal{T}_{Ty}$ produced by the thrust force from the tail rotor
- The cross-torque $\mathcal{T}_{M_y}$ from the main rotor
- The torque $\mathcal{T}_{Dy}$ caused by viscous damping
- The torque $\mathcal{T}_{Ry}$ caused by Coulomb friction of the yaw joint.

Horizontal trust torques

The horizontal thrust torque from the tail rotor and the cross-torque from the main rotor is modeled comparably to the vertical thrust torques. However, in this case, the moment arms changes with the pitch angle. A cosine term is therefore included in the torque equations:

$$\begin{aligned} \mathcal{T}_{My}(t) &= F_{My}(t)\cos(\theta_p(t))d_t \\ \mathcal{T}_{Ty}(t) &= F_{Ty}(t)\cos(\theta_p(t))d_t \end{aligned} \tag{2.31}$$

Also in this case, he thrust forces are modeled as nonlinear piecewise functions of the rotor velocities:

$$F_{My}(t) = f_{My}(\omega_p(t)) = \begin{cases} k_{Myp1}\omega_p^2(t) + k_{Myp2}\omega_p(t) & \text{if } \omega_p(t) \geq 0 \\ k_{Myn1}\omega_p^2(t) + k_{Myn2}\omega_p(t) & \text{if } \omega_p(t) < 0 \end{cases} \tag{2.32}$$

$$F_{Ty}(t) = f_{Ty}(\omega_y(t)) = \begin{cases} k_{Typ1}\omega_y^2(t) + k_{Typ2}\omega_y(t) & \text{if } \omega_y(t) \geq 0 \\ k_{Tyn1}\omega_y^2(t) + k_{Tyn2}\omega_y(t) & \text{if } \omega_y(t) < 0 \end{cases} \tag{2.33}$$

where the main rotor thrust function $f_{My}(\omega_p(t))$ uses the thrust gains $k_{Myp_{1/2}}$ for positive rotor velocities and $k_{Myn_{1/2}}$ for negative rotor velocities. Likewise, the tail rotor thrust function $f_{Ty}(\omega_p(t))$ uses the thrust gains $k_{Typ_{1/2}}$ for positive rotor velocities and $k_{Tyn_{1/2}}$ for negative rotor velocities.

Yaw damping and Coulomb friction torque

A model of the damping and Coulomb friction for the yaw motion is proposed in [7], where the static friction is expressed as a constant. In contrast, the damping and viscous friction are combined in a nonlinear function of the angular yaw velocity. A slight improvement on the model has been made by identifying different damping coefficients depending on the direction of rotation:

$$\mathcal{T}_{Dy}(t) = f_{Dy}(t) = \begin{cases} f_{Dyp}(\Omega_y(t)) = k_{Dyp_1}\Omega_y^2(t) + k_{Dyp_2}\Omega_y(t) & \text{if } \Omega_y(t) \geq 0 \\ f_{Dyn}(\Omega_y(t)) = k_{Dyn_1}\Omega_y^2(t) + k_{Dyn_2}\Omega_y(t) & \text{if } \Omega_y(t) < 0 \end{cases} \tag{2.34}$$

where the coefficients $K_{Dyp_{1/2}}$ applies to positive values of the yaw velocity $\Omega_y(t)$, while $K_{Dyn_{1/2}}$ applies to negative values of $\Omega_y(t)$.

The static friction force $F_{Fy}(t)$ is also expressed with different constants depending of the sign of the yaw velocity:

$$F_{Fy}(t) = \begin{cases} k_{Fyp} & \text{if } \Omega_y(t) \geq 0 \\ k_{Fyn} & \text{if } \Omega_y(t) < 0 \end{cases} \tag{2.35}$$

The friction constant k_{Fyp} is used for positive yaw velocities, while the friction constant k_{Fyp} is used for negative yaw velocities.

A final improvement on the static friction is made to ensure that the modeled yaw velocity remains equal to zero as long as the combined rotor thrust force is less in magnitude than the static friction force. This is accomplished by implementing Karnopp's friction model which extends the basic Coulomb fiction model [6]. The torque caused by static friction is expressed as:

$$\mathcal{T}_{Fy}(t) = f_{Fy}(t) = \begin{cases} sat(F_{My}(t) - F_{Ty}(t), F_{Fy}(t)) & \text{when } \Omega_y(t) = 0 \\ F_{Fy}(t) & \text{else} \end{cases} \tag{2.36}$$

where the Karnopp's model uses a *saturation function*[1] to keep the sum of the magnitude off all forces equal to zero until the applied forces are strong enough to overcome the friction force. The static friction force $F_{Fy}(t)$ is defined in Eq. (2.35).

Net torque about the yaw axis

The net torque acting on the yaw axis are:

$$\begin{aligned}\sum_y \mathcal{T}_y(t) &= \mathcal{T}_{Ty}(t) - \mathcal{T}_{My}(t) - \mathcal{T}_{Dy}(t) - \mathcal{T}_{Fy}(t) \\ &= f_{Ty}(\omega_y(t))\cos(\theta_p(t))d_t - f_{My}(\omega_p(t))\cos(\theta_p(t))d_t - f_{Dy}(\Omega_y(t)) - f_{Fy}(t)\end{aligned} \tag{2.37}$$

Newton's second law for rotation is then applied to obtain the equation describing the yaw motion of the Aero:

$$\frac{d\Omega_y(t)}{dt} = \frac{1}{J_y(\theta_p)}(f_{Ty}(\omega_y(t))\cos(\theta_p(t))d_t - f_{My}(\omega_p(t))\cos(\theta_p(t))d_t - f_{Dy}(\Omega_y(t)) - f_{Fy}(t) \tag{2.38}$$

where $J_y(\theta_p)$ is the total moment of inertia of the yaw body as a function of the pitch angle.

2.2.4 Moment of Inertia of the Aero

The moments of inertia acting on the Aero are modeled as proposed by Quanser in their lab documentation [18]. However, the masses of the Aero have been divided into smaller fractions than what Quanser suggests. This resulted in slightly lower numerical values for both the pitch and yaw inertia, which gave better results when the model was tested against the actual plant. In addition, the solution provided by Quanser does not take into account that yaw inertia decreases as the inclination of the pitch body increases, which is a result of the mass being more concentrated along the vertical rotation axis. Therefore, the yaw inertia is modeled as a nonlinear function of the pitch angle, where $J_y(\theta_p = 0)$ is similar to the estimate proposed by Quanser. Figure 2.8 shows the main parts of the Aero pitch and yaw bodies.

[1] "*The saturation function $sat(x, S)$ is defined so that $sat(x, S) = x$ when $|x| \leq S$ and $sat(x, S) = Ssgn(x)$ when $|x| \geq S$*" [6]

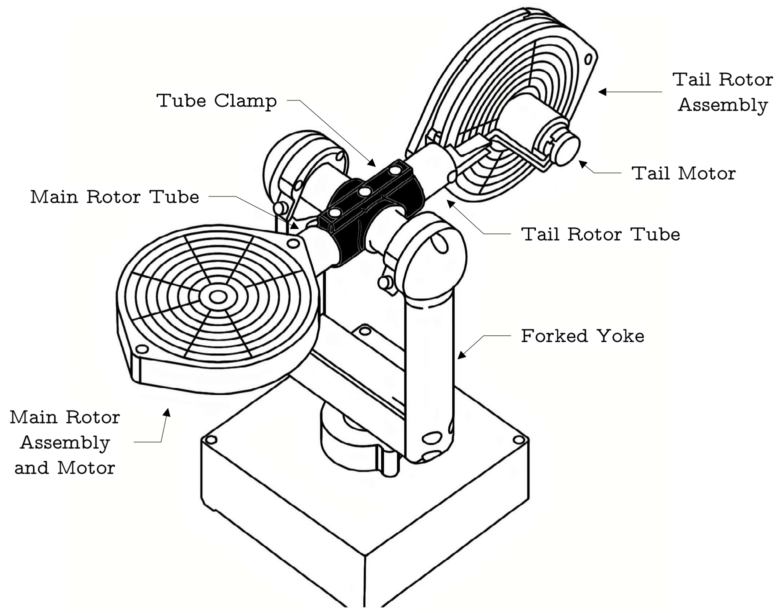


Figure 2.8: General component arrangement of the mechanical parts on the Aero. Adapted from [19]

The main and tail tubes, including the tube clamp, are modeled as a single cylinder rotating about its center with inertia,

$$\begin{aligned} J_{cylinder} &= \frac{1}{12}(m_{tm} + m_{tt} + m_{tc})l_t^2 \\ &= \frac{1}{12}(2 \cdot 0.089kg + 0.280kg) \cdot 0.165^2 m^2 = 1.04 \times 10^{-3}\ kgm^2 \end{aligned} \tag{2.39}$$

where l_t is the total length of tubes and clamp when assembled.

Each of the two rotor assemblies, including the DC motors, are considered as single-point masses rotating at a distance d_t from the pivot point resulting in the inertia:

$$\begin{aligned} J_{rotor} &= (m_{rm/t} + m_{em/t})d_t^2 \\ &= (0.146kg + 0.200kg) \cdot 0.158^2 m^2 = 8.64 \times 10^{-3}\ kgm^2 \end{aligned} \tag{2.40}$$

The forked yoke is approximated as a cylinder rotating about its center which leads to the expression:

$$J_{yoke} = \frac{1}{2}m_y r_y^2 = \frac{1}{2}0.526kg \cdot 0.02^2 m^2 = 1.05 \times 10^{-4}\ kgm^2 \tag{2.41}$$

The total moment of inertia acting about the pitch and yaw axes are:

$$\begin{aligned} J_p &= J_{cylinder} + 2J_{rotor} = 0.0183kgm^2 \\ J_y(\theta_p = 0) &= J_{cylinder} + 2J_{rotor} + J_{yoke} = 0.0184kgm^2 \end{aligned} \tag{2.42}$$

2.3 Nonlinear model of the Aero

By taking into account Eqs. (2.13), (2.30) and (2.38), the resulting model describing the Aero motion consists of a set of six first order non-linear differential equations:

$$\begin{bmatrix} \dot{\omega_p}(t) \\ \dot{\Omega}_p(t) \\ \\ \dot{\theta}_p(t) \\ \dot{\omega_y}(t) \\ \dot{\Omega}_y(t) \\ \\ \dot{\theta}_y(t) \end{bmatrix} = \begin{bmatrix} \frac{K_\tau}{R_a J_r} v_p(t) - \frac{K_\tau K_E}{R_a J_r} \omega_p(t) - \frac{1}{J_r} f_a(\omega_p(t)) \\ \frac{1}{J_p}(f_{Mp}(\omega_p(t))d_t + f_{Tp}(\omega_y(t))d_t - m_b g d_m \sin(\theta_p(t)) - f_{Dp}(\Omega_p(t))) \\ -\Omega_y(t)^2 d_t^2 (m_A + m_B) \sin(2\theta_p(t)) \\ \Omega_p(t) \\ \frac{K_\tau}{R_a J_r} v_y(t) - \frac{K_\tau K_E}{R_a J_r} \omega_y(t) - \frac{1}{J_r} f_a(\omega_y(t)) \\ \frac{1}{J_y(\theta_p(t))}(f_{Ty}(\omega_y(t))\cos(\theta_p(t))d_t \\ -f_{My}(\omega_p(t))\cos(\theta_p(t))d_t - f_{Dy}(\Omega_y(t)) - f_{Fy}(t) \\ \Omega_y(t) \end{bmatrix} \tag{2.43}$$

where the electrical dynamics from Eq. (2.5) has been neglected, as the electrical circuits have a significantly faster response than the mechanical aspects of the Aero. This non-linear model will later be converted into a linear parameter varying (LPV) framework.

2.4 Parameter estimation

Some of the parameters in the non-linear model are given by Quanser. Still, most have been estimated through identification procedures like regression analysis using the method of least squares and iterative tuning. The Coulomb friction and viscous yaw damping was estimated in a similar manner as described in [7]. A list of the parameter values that Quaser has provided in their courseware resources is presented in table 2.1:

Parameter	Value	Parameter	Value
K_τ	0.042 Nm/a	K_E	0.042 $V/rad \cdot s$
R_a	8.4 Ω	d_t	0.158 m
$m_{rm/t}$	0.146 kg	$m_{tm/t}$	0.089 kg
$m_{em/t}$	0.200 kg	m_{tc}	0.280 kg
m_b	1.15 kg	m_y	0.526 kg
m_b	1.15 kg	m_y	0.526 kg
J_{prop}	5.6×10^{-5} kgm^2	J_{motor}	2.0×10^{-6} kgm^2
J_{hub}	5.0×10^{-8} kgm^2	J_{motor}	0.526 kgm^2

Table 2.1: Known parameter provided by Quanser

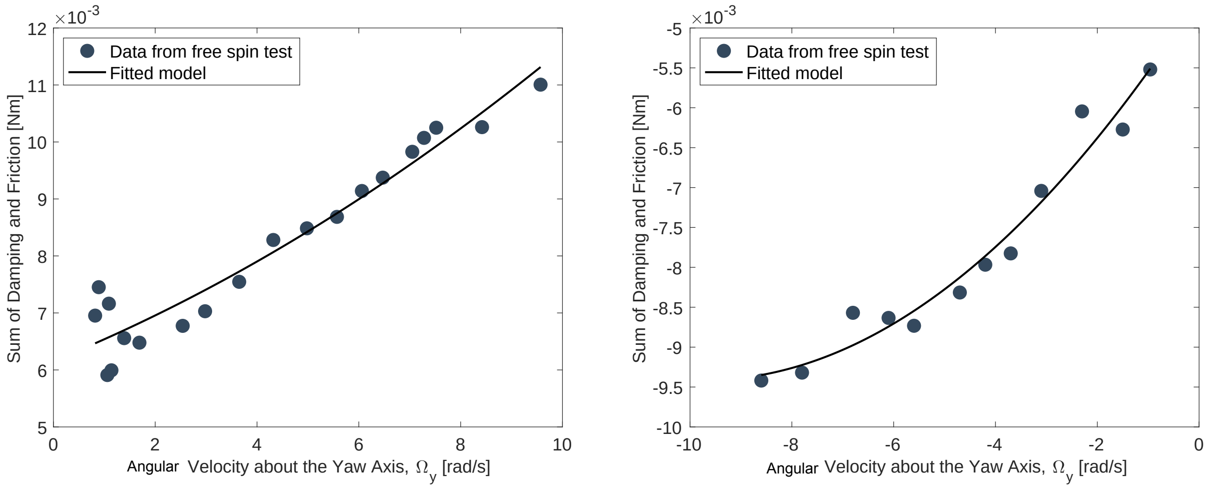
2.4.1 Estimating Coulomb friction and viscous damping

Yaw axis

A free spin experiment was performed to estimate f_{Dy} and F_{fy}. The Aero was decelerated from various positive and negative yaw velocities with both motors disengaged and the pitch axis locked in its horizontal equilibrium. The equation of motion in (2.38) was then reduced to:

$$J_y(\theta_p = 0)\dot{\Omega}_y(t) = -f_{Dy}(\Omega_y(t))) - F_{fy} \tag{2.44}$$

where the angular yaw deceleration was approximated as $\dot{\Omega}_y(t) = \frac{\Delta\Omega_y}{\Delta t}$ and $J_y(\theta_p = 0) = 0.0184$ from Eq. (2.42). The results were plotted against the measured angular yaw deceleration equaling the sum of damping and friction in two separate plots, shown in Figure 2.9. A second-order regression for each plot gave an estimated relationship between angular velocity and the sum of damping and friction. The damping and friction were then distinguished by modeling the viscous damping using the regression coefficients and the Coulomb friction as the regression constant.



(a) $\Omega_y \geq 0$, $R^2 = 93.22\%$ (b) $\Omega_y < 0$, $R^2 = 95.87\%$

Figure 2.9: Test results from free spin experiment conducted to estimate yaw friction and damping

$$\begin{aligned} f_{Dy} &= \begin{cases} k_{Dyp_1}\Omega_y^2 + k_{Dyp_2}\Omega_y & \text{if } \Omega_y(t) \geq 0 \\ k_{Dyn_1}\Omega_y^2 + k_{Dyn_2}\Omega_y & \text{if } \Omega_y(t) < 0 \end{cases} \\ &= \begin{cases} 1.84 \times 10^{-5}\Omega_y^2 + 3.64 \times 10^{-4}\Omega_y & \text{if } \Omega_y(t) \geq 0 \\ -5.05 \times 10^{-5}\Omega_y^2 + 9.86 \times 10^{-4}\Omega_y & \text{if } \Omega_y(t) < 0 \end{cases} \end{aligned} \tag{2.45}$$

The second order regression resulted in the friction coefficients $k_{Fyp} = 0.00616$ and $k_{Fyn} = 0.00461$. However, online test on the Aero indicated that these values where slightly to large. The coefficients have therefore been tuned down to:

$$F_{fy} = \begin{cases} k_{Fyp} & \text{if } \Omega_y(t) \geq 0 \\ k_{Fyn} & \text{if } \Omega_y(t) < 0 \end{cases} = \begin{cases} 0.00586 & \text{if } \Omega_y(t) \geq 0 \\ 0.00341 & \text{if } \Omega_y(t) < 0 \end{cases} \tag{2.46}$$

Pitch axis

Estimating the Coulomb friction and viscous damping in the pitch joint is more challenging than for the yaw joint, given the presence of the gravitational force. Therefore, an assumption that the sum of viscous friction and damping in the pitch joint is similar to the yaw joint is made, while the static friction is neglected. The parameters in f_{Dp} were then found through iterative tuning.

$$\begin{aligned} f_{Dp}(\Omega_p(t)) &= sign(\Omega_p(t))k_{Dp_1}\Omega_p(t)^2 + k_{Dp_2}\Omega_p(t) \\ &= sign(\Omega_p(t))7.6 \times 10^{-6}\Omega_y^2(t) + 6.1 \times 10^{-3}\Omega_y(t) \end{aligned} \tag{2.47}$$

2.4.2 Estimating Yaw inertia

An expression for the yaw inertia was found by rewriting Eq. (2.44) and applying the regression results from Eq. (2.45)

$$J_y(\theta_p = 0) = \frac{-f_{Dy} - F_{fy}}{\dot{\Omega}_y} = \frac{1.84 \times 10^{-5}\Omega_y^2 + 3.63 \times 10^{-4}\Omega}{\frac{\Delta\Omega_y}{\Delta t}} \tag{2.48}$$

The angular yaw deceleration was approximated for different pitch angles. Figure 2.10 shows the relationship between the estimated yaw inertia and the pitch angle. A sum of least square regression was then used to fit a second order curve. However, as suggested in [7], the regression results can be simplified into a cosine function of the pitch angle:

$$J_y(\theta_p) = k_{J_y}\cos(\theta_p) = 0.017\cos(\theta_p) \tag{2.49}$$

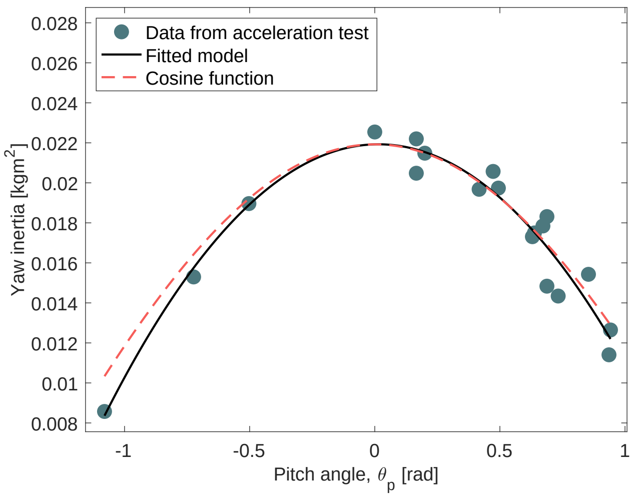


Figure 2.10: Test data showing the relationship between pitch angle and yaw inertia

2.4.3 Estimating vertical cross-thrust force functions

The vertical thrust force functions $f_{Mp}(\omega_p(t))$ and $f_{Tp}(\omega_p(t))$ were estimated with the yaw axis locked, limiting the Aero only to rotate about the pitch axis, thus acting like a 1 DOF system. Then data from each rotor was collected separately by energizing one motor at a time.

Vertical thrust from main rotor

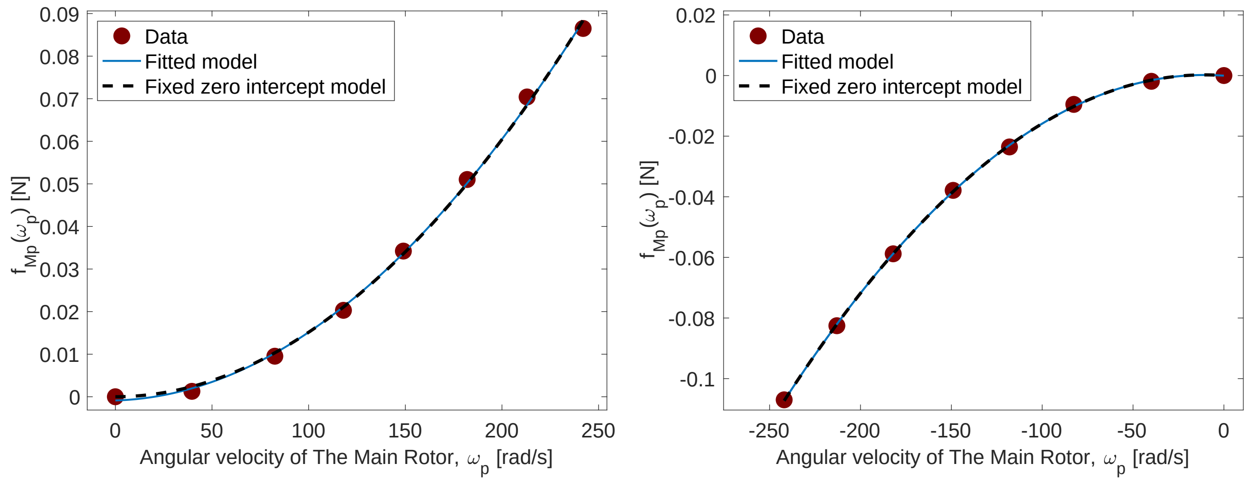
The main rotor was accelerated to different angular velocities by applying various main motor input voltages v_p. The resulting pitch angles at a steady-state were then measured and used to calculate the corresponding magnitudes of the thrust force. When the 1 DOF pitch system is at rest with only the main rotor running, and zero angular velocity about the pitch axis, the equation of motion about the pitch axis (2.30) is reduced to:

$$0 = f_{Mp}(\omega_p(t))d_t - m_b g d_m \sin(\theta_p(t)) \tag{2.50}$$

which can be used to solve for the main thrust force:

$$f_{Mp}(\omega_p(t)) = \frac{m_b g d_m \sin(\theta_p(t))}{d_t} \tag{2.51}$$

The positive and negative magnitudes of the thrust force are shown in separate plots in Figure 2.11. A second-order least square regression curve has been fitted to each plot. Although the resulting regression analysis indicates a satisfactory correlation of 99.86% for positive pitch velocities and 99.99% for negative pitch velocities, both the regression equations contain a constant term. The constant term -8.72×10^{-4} is included in the positive velocity regression equation, while the negative velocity equation contains the constant term -4.79×10^{-5}. It's not desirable to include these terms in the non-linear model as the rotors do not produce any thrust force when the propellers do not rotate. As the constant terms are close to zero, an alternative regression analysis has been applied where the y-intercept of the regression curves has been forced to zero [14]. The correlation obtained with this analysis was $99,85\%$ for positive values of $\omega_p(t)$ and 99.99% for negative values. The two zero intercept regression equations make up the piecewise function in Eq. (2.52).



(a) $\omega_p \geq 0$ (b) $\omega_p < 0$

Figure 2.11: Estimated thrust force-speed curve for positive and negative main motor velocities using second-order polynomial curve fitting. The blue lines represent the resulting regression function from the standard least square method, while the dashed lines represent the least square method with forced zero y-intercept.

$$\begin{aligned} f_{Mp}(\omega_p(t)) &= \begin{cases} k_{Mpp_1}\omega_p^2(t) + k_{Mpp_2}\omega_p(t) & \text{if } \omega_p(t) \geq 0 \\ k_{Mpn_1}\omega_p^2(t) + k_{Mpn_2}\omega_p(t) & \text{if } \omega_p(t) < 0 \end{cases} \\ &= \begin{cases} 1.51 \times 10^{-6}\omega_p^2(t) + 8.61 \times 10^{-7}\omega_p(t) & \text{if } \omega_p(t) \geq 0 \\ -2.01 \times 10^{-6}\omega_p^2(t) - 4.18 \times 10^{-5}\omega_p(t) & \text{if } \omega_p(t) < 0 \end{cases} \end{aligned} \tag{2.52}$$

Vertical thrust from tail rotor

Experimental data for the vertical thrust force form the tail rotor was collected analogously to the main rotor. The Aero body was pitched to various angles by applying different tail motor voltages. When the pitch angle is at steady state with only the tail rotor running, Eq. (2.30) is reduced to:

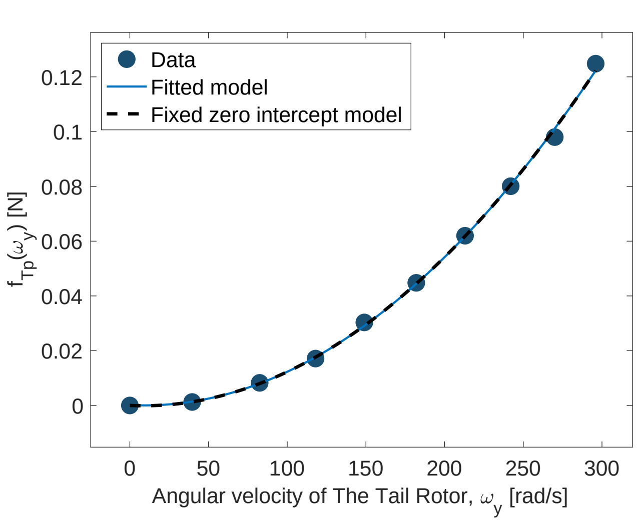
$$0 = f_{Tp}(\omega_p(t))d_t - m_b g d_m \sin(\theta_p(t)) \tag{2.53}$$

where solving for $f_{Tp}(\omega_p(t))$ results in:

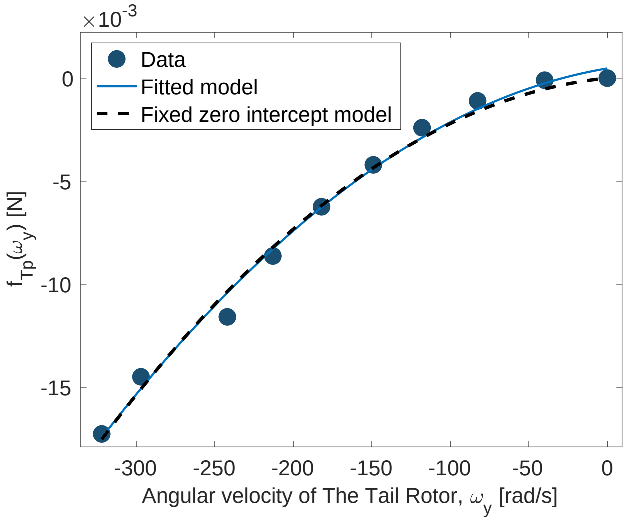
$$f_{Tp}(\omega_p(t)) = \frac{m_b g d_m \sin(\theta_p(t))}{d_t} \tag{2.54}$$

The tail rotor thrust force data are shown in two separate plots in Figure 2.12. As for the main rotor, a second-order least square regression was applied to both the positive and negative tail rotor thrust force data. Also in this case, the resulting regression equations included non-zero constant terms of 1.53×10^{-4} for positive values of $\omega_p(t)$, and 4.72×10^{-4} for negative values of $\omega_p(t)$. As the constant terms was close to zero, the fixed zero intercept regression analysis was applied [14], which gave the expressions that define the thrust function $f_{Ty}(\omega_p(t))$ in Eq. (2.55). The correlation obtained with the standard least square regression analysis was 99.89% for positive values of $\omega_p(t)$ and 99.27% for negative values of $\omega_p(t)$. Applying the fixed zero intercept regression analysis did not result in a significant change in correlation for positive values of $\omega_p(t)$. In contrast, the correlation dropped to 99.17% for negative values of $\omega_p(t)$.

$$\begin{aligned} f_{Ty}(\omega_p(t)) &= \begin{cases} k_{Tpp_1}\omega_p^2(t) + k_{Tpp_2}\omega_p(t) & \text{if } \omega_p(t) \geq 0 \\ k_{Tpn_1}\omega_p^2(t) + k_{Tpn_2}\omega_p(t) & \text{if } \omega_p(t) < 0 \end{cases} \\ &= \begin{cases} 1.49 \times 10^{-6}\omega_p^2(t) - 2.52 \times 10^{-5}\omega_p(t) & \text{if } \omega_p(t) \geq 0 \\ -1.46 \times 10^{-7}\omega_p^2(t) + 7.47 \times 10^{-6}\omega_p(t) & \text{if } \omega_p(t) < 0 \end{cases} \end{aligned} \tag{2.55}$$



(a) $\omega_p \geq 0$



(b) $\omega_p < 0$

Figure 2.12: Estimated thrust force-speed curve for positive and negative main motor velocities using second-order polynomial curve fitting. The blue lines represent the resulting regression function from the standard least square method, while the dashed lines represent the least square method with forced zero y-intercept.

2.4.4 Estimating Horizontal thrust force functions

As the main and tail rotor are identical, it can be assumed that the same function can describe the horizontal thrust from the tail rotor as the vertical thrust from the main rotor and that the same function can describe the horizontal cross-thrust from the main rotor as the vertical cross-thrust from the tail rotor. However, online simulations indicated that the vertical thrust functions resulted in more thrust than needed for the yaw movement.

This inconsistency is likely due to unmodeled uncertainties. Slightly down-tuned versions of the vertical thrust force functions are therefore implemented for the horizontal thrust force functions $f_{My}(\omega_p(t))$ and $f_{Ty}(\omega_y(t))$:

$$\begin{aligned} f_{Ty}(\omega_y(t)) &= \begin{cases} k_{Typ1}\omega_y^2(t) + k_{Typ2}\omega_y(t) & \text{if } \omega_y(t) \geq 0 \\ k_{Tyn1}\omega_y^2(t) + k_{Tyn2}\omega_y(t) & \text{if } \omega_y(t) < 0 \end{cases} \\ &= \begin{cases} 1.06 \times 10^{-6}\omega_p^2(t) + 1.17 \times 10^{-5}\omega_p(t) & \text{if } \omega_p(t) \geq 0 \\ -1.41 \times 10^{-6}\omega_p^2(t) - 4.16 \times 10^{-5}\omega_p(t) & \text{if } \omega_p(t) < 0 \end{cases} \end{aligned} \tag{2.56}$$

$$\begin{aligned} f_{My}(\omega_y(t)) &= \begin{cases} k_{Myp1}\omega_y^2(t) + k_{Myp2}\omega_y(t) & \text{if } \omega_y(t) \geq 0 \\ k_{Myn1}\omega_y^2(t) + k_{Myn2}\omega_y(t) & \text{if } \omega_y(t) < 0 \end{cases} \\ &= \begin{cases} 1.04 \times 10^{-6}\omega_p^2(t) - 1.91 \times 10^{-5}\omega_p(t) & \text{if } \omega_p(t) \geq 0 \\ -6.43 \times 10^{-7}\omega_p^2(t) + 3.28 \times 10^{-5}\omega_p(t) & \text{if } \omega_p(t) < 0 \end{cases} \end{aligned} \tag{2.57}$$

2.5 Model validation

Figure 2.13 shows a model validation experiment comparing the nonlinear model to the real Aero system. The experiment was performed by introducing two sets of shifted impulse signals on the input voltages. The first set led to low rotor velocities, while the second set led to high rotor velocities. The results show that there are some unmodeled dynamics in the yaw motion, where the model behaves worse for low velocities. This is likely linked to the Coulomb friction in the yaw joint. It can further be observed that the pitch motion resembles the real system best for low pitch angular velocities. This could be a indication that the pitch damping term has been overfitted. Nevertheless, the model still provide a satisfactory approximation of the real system.

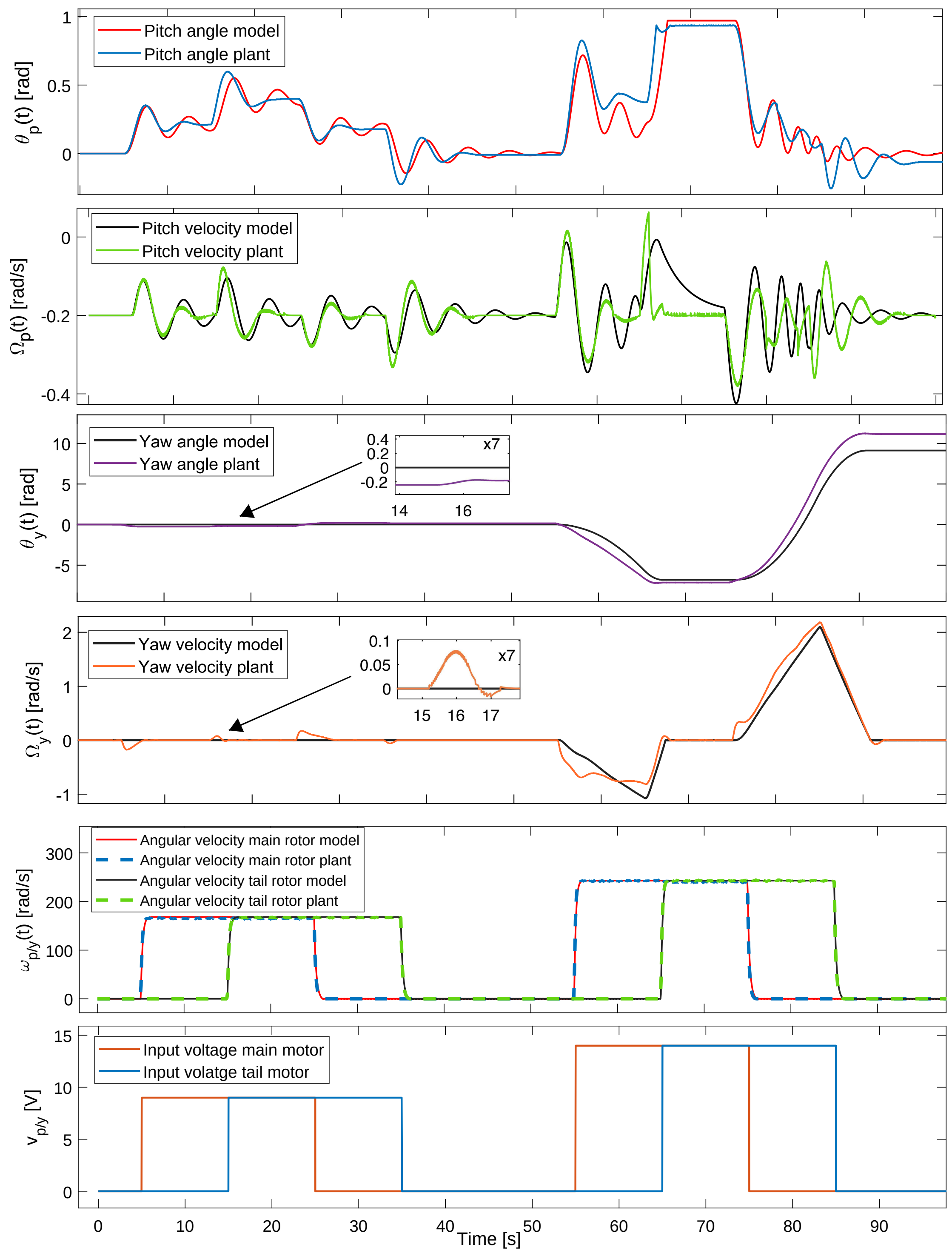


Figure 2.13: Nonlinear model validation

Chapter 3

Modeling and control of LPV systems

A linear parameter varying (LPV) system is a subset of the larger category of linear time-variant (LTV) systems, a type of dynamical system described by a single or a set of differential equations. The linear property of an LTV system implies a linear relationship between the system's input and output variables. However, the time-variant aspect means that the resulting outputs depend on at what time the input was given. LPV systems are time-variant as they contain parameters that depend on measured variables that vary with time [23]. These parameters are often referred to as varying parameters or scheduling parameters. The theory of LPV systems was first described in a Ph.D. thesis by Jeff S. Shamma at the Massachusetts Institute of Technology in 1988 [24]. LPV systems have since then become an attractive tool in modeling and control applications, as they allow for strongly non-linear systems to be transformed into a linear framework. The main advantage of this approach is that powerful linear tools for analysis and control can be applied once the non-linear system is transformed into the LPV framework [22]. There are several different ways to represent systems within the LPV framework. In this project, a quasi-LPV model is used, where the word *quasi* comes from the fact that the measured varying parameters are *endogenous*, meaning that they are functions of the system's state variables. In LPV systems where the varying parameters are independent of the states, the parameters are referred to as *exogenous* [15].

3.1 State-space representation

The main strength of the LPV framework lies in the ability to represent complex non-linear systems in a compact matrix form. One way to express the behavior of a dynamical system is through a so-called state-space representation, which is a well-established method in the field of control theory. A state-space representation is a mathematical method that involves rewriting one or more differential equations of higher order into a system of first-order differential equations. This system of equations is described through a set of endogenous variables called *state variables* $x_i(t), i = 1, ..., n$, where the number n corresponds to the order of the system. However, simplifications can be made that reduce the number of states in relation to the original order. The system input signals are denoted as $u_1(t), ..., u_m(t)$, where the number m corresponds to the number of inputs. The linear first-order equations are then expressed in a set of n functions called *state functions*, that has the general form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{3.1}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ and $\mathbf{u}(t) \in \mathbb{R}^m$ are column vectors containing the states and inputs, and $\dot{\mathbf{x}}(t)$ is the time derivative of the vector function $\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$. The set of first order differential equations contained in state vector function can then be expressed as:

$$\begin{cases} \dot{x}_1(t) & = a_{11}(t)x_1(t) + \ldots + a_{1n}(t)x_n(t) + b_{11}(t)u_1(t) + \ldots + b_{1m}(t)u_m(t) \\ \dot{x}_2(t) & = a_{21}(t)x_1(t) + \ldots + a_{2n}(t)x_n(t) + b_{21}(t)u_1(t) + \ldots + b_{2m}(t)u_m(t) \\ & \vdots \\ \dot{x}_n(t) & = a_{n1}(t)x_1(t) + \ldots + a_{nn}(t)x_n(t) + b_{n1}(t)u_1(t) + \ldots + b_{nm}(t)u_m(t) \end{cases} \tag{3.2}$$

where $a_{ij}(t)$ are time-dependent parameters that characterize the dynamics of the system. If the time dependency of these parameters can be neglected or does not exist, the system is referred to as a linear time-invariant (LTI) system.

Eq. (3.2) can be expressed in a compact matrix form:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_n(t) \end{bmatrix} = \begin{bmatrix} a_{11}(t) & \ldots & a_{1n}(t) \\ a_{21}(t) & \ldots & a_{2n}(t) \\ \vdots & & \vdots \\ a_{n1}(t) & \ldots & a_{nn}(t) \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} + \begin{bmatrix} b_{11}(t) & \ldots & b_{1n}(t) \\ b_{21}(t) & \ldots & b_{2m}(t) \\ \vdots & & \vdots \\ b_{n1}(t) & \ldots & b_{nm}(t) \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix} \tag{3.3}$$

Then by defining a matrix $\mathbf{A}(t)$ containing the state parameters $a_{ij}(t)$ and a matrix $\mathbf{B}(t)$ containing the input parameters $b_{ij}(t)$, the system can be represented in the form:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) \tag{3.4}$$

Similar to the state function, the state-space approach also includes an *output function*:

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{3.5}$$

where $\mathbf{y}(t) \in \mathbb{R}^p$ is a column vectors containing the system outputs and $\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t)$ is a vector function. The output equation is required as the signals of interest in a given application are not necessarily the exact values of the state variables. In a system with m number of defined output signals the vector function in Eq. (3.5) has the form:

$$\begin{cases} y_1(t) & = c_{11}(t)x_1(t) + \ldots + c_{1n}(t)x_n(t) + d_{11}(t)u_1(t) + \ldots + d_{1r}(t)u_m(t) \\ y_2(t) & = c_{21}(t)x_1(t) + \ldots + c_{2n}(t)x_n(t) + d_{21}(t)u_1(t) + \ldots + d_{2r}(t)u_m(t) \\ & \vdots \\ y_p(t) & = c_{p1}(t)x_1(t) + \ldots + c_{pn}(t)x_n(t) + d_{p1}(t)u_1(t) + \ldots + d_{pr}(t)u_m(t) \end{cases} \tag{3.6}$$

where $c_{ij}(t)$ are time-dependent parameters that describe the relationship between the states and the outputs, while $d_{ij}(t)$ are so-called *feedthrough* parameters that enable the outputs to be directly related to the inputs. However, the use of feedthrough parameters is rarely needed in control theory.

Analogously to the state equations, the output equations can be put in the compact matrix form:

$$\begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_p(t) \end{bmatrix} = \begin{bmatrix} c_{11}(t) & \dots & c_{1n}(t) \\ c_{21}(t) & \dots & c_{2n}(t) \\ \vdots & & \vdots \\ c_{p1}(t) & \dots & c_{pn}(t) \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} + \begin{bmatrix} d_{11}(t) & \dots & d_{1m}(t) \\ d_{21}(t) & \dots & d_{2m}(t) \\ \vdots & & \vdots \\ d_{r1}(t) & \dots & d_{rm}(t) \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix} \tag{3.7}$$

where the matrix $\mathbf{C}(t)$ holds the parameters $c_{ij}(t)$ and the matrix $\mathbf{D}(t)$ holds the parameters $d_{ij}(t)$.

The matrix form of the state equations together with the matrix form of the output equations make up the full state space representation of an arbitrary LTV system:

$$\begin{cases} \dot{\mathbf{x}}(t) & = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) \\ \mathbf{y}(t) & = \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t) \end{cases} \tag{3.8}$$

A graphical representation of a general LTV system in state space is presented through a block diagram in Figure 3.1.

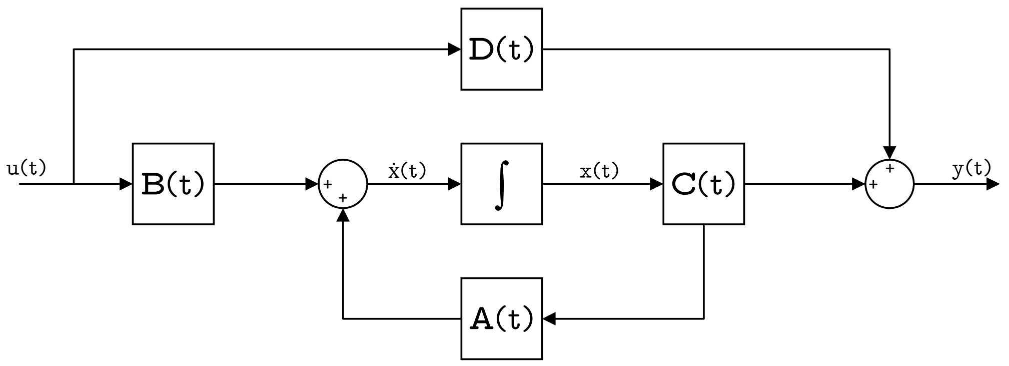


Figure 3.1: A mathematical block diagram of a dynamical system represented in state space

3.2 Overview of quasi-LPV modeling

The main goal in quasi-LPV modeling is to transform nonlinear systems into the compact state-space form:

$$\begin{cases} \dot{\mathbf{x}}(t) & = \mathbf{A}(\boldsymbol{\psi}(t))\mathbf{x}(t) + \mathbf{B}(\boldsymbol{\psi}(t))\mathbf{u}(t) \\ \mathbf{y}(t) & = \mathbf{C}(\boldsymbol{\psi}(t))\mathbf{x}(t) + \mathbf{D}(\boldsymbol{\psi}(t))\mathbf{u}(t) \end{cases} \tag{3.9}$$

There exist several approaches that can achieve such transformations, where the existing approaches can be categorized into three types: *linearization-based*, *state-transformation*, and function *substitution-based* [1]. The linearization-based approach involves linearizing a system at several operating points followed by an interpolating between the operating points to obtain an LPV representation. However, since this method involves linearizing, the resulting LPV system only presents an approximation of the original nonlinear system. The state-transformation approach aims to apply a coordinate change in the nonlinear system to obtain an LPV representation. Although effective, this method is only applicable for a limited type of nonlinear system. The interested reader can find a description of these limitations in [5]. The substitution-based approach is the technique that will be used to obtain an LPV representation of the Aero system. This method, along with the state-transformation approach, has the advantage of potentially producing a LPV representation that is exactly equivalent to the original nonlinear system [1]. The substitution technique is based on obtaining decomposition functions that embed the nonlinear characteristics of the system. These functions are what is referred to as the scheduling- or varying parameters. In this project, a substitution-based approach called the sector nonlinearity technique is applied [25]. In this method, the goal is to find a global or local sector such that $\dot{x}(t) = f(x(t)) \in [\psi_1(t),\ \ldots, \psi_j(t)]$, where $\psi(t)$ are varying parameters.

Example 1. In order to demonstrate how the sector nonlinearity technique is used to obtain a quai-LPV system in state-space form, let us consider an exemplified nonlinear system from [23]:

$$\begin{cases} \dot{x}_1(t) &= \sin(x_1(t)) + x_1(t)x_2(t) + u(t) \\ \dot{x}_2(t) &= x_1^2(t) + x_2^2(t) \end{cases} \tag{3.10}$$

with two state variables $\mathbf{x}(t) = [x_1(t)\ x_2(t)]^T$ and a single input $u(t)$.

By defining the three varying parameters:

$$\psi_1(t) = \frac{\sin(x_1(t))}{x_1(t)} \tag{3.11}$$

$$\psi_2(t) = x_1(t) \tag{3.12}$$

$$\psi_3(t) = x_2(t) \tag{3.13}$$

which embed all the nonlinearities in the system equations (3.10), the nonlinear system can be represented in a quasi-LPV form which is completely equivalent to original system:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} \psi_1(t) & \psi_2(t) \\ \psi_2(t) & \psi_3(t) \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) \tag{3.14}$$

Then, by defining the input function:

$$y(t) = x_1(t) \tag{3.15}$$

and the state-space matrices:

$$\mathbf{A}(\boldsymbol{\psi}(t)) = \begin{bmatrix} \psi_1(t) & \psi_2(t) \\ \psi_2(t) & \psi_3(t) \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.16}$$

The system can be represented in the state-space form:

$$\begin{cases} \dot{\mathbf{x}}(t) &= \mathbf{A}(\boldsymbol{\psi}(t))\mathbf{x}(t) + \mathbf{B}u(t) \\ y(t) &= \mathbf{C}\mathbf{x}(t) \end{cases} \tag{3.17}$$

3.3 Stability analysis of LPV systems

Once a nonlinear system is represented as a quasi-LPV model, powerful linear tools can be used to analyze stability and design a stable closed loop control system. The purpose of this section is to introduce the concept of stability, as well as how stability may be analyzed. As a first step, the simpler theory of autonomous[1] LTI and LPV systems is explored. This is in order to provide a baseline that can be extended to the case of designing a stable closed loop controller.

3.3.1 The definition of stability

Let's consider the autonomous system:

$$\dot{\mathbf{x}}(t) = f(x(t)),\ f : \mathbb{D} \subset \mathbb{R}^n \to \mathbb{R}^n \tag{3.18}$$

where f is a vector function that maps states in the domain $\mathbb{D}$ to a n-dimensional vector space, $\mathbb{R}^n$, and has one or more equilibrium points, $\bar{\mathbf{x}} \in \mathbb{D}$. The stability of the system around a given equilibrium point can then be defined as:

Definition 1 (Stability [10])**.** The equilibrium point $\bar{\mathbf{x}} = 0$ is

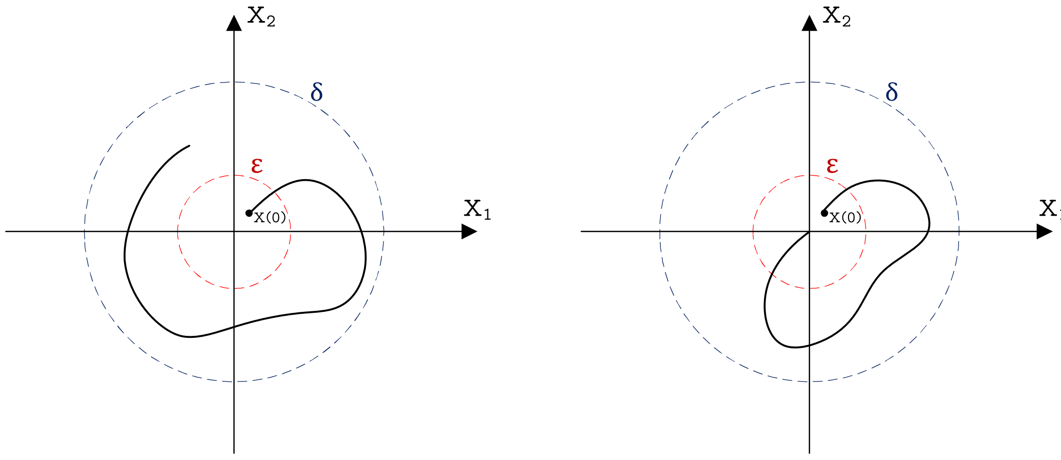
- stable if for each $\epsilon > 0$, there is $\delta = \delta(\epsilon)$ such that:

$$\parallel \mathbf{x}(0) \parallel < \delta \Rightarrow \parallel \mathbf{x}(0) \parallel < \epsilon, \forall t \geq 0 \tag{3.19}$$

- unstable if it's not stable.

Definition 1 only applies to an equilibrium point that lies in the origin of $\mathbb{R}^n$. It is therefore essential that the domain $\mathbb{D}$ contains the origin of $\mathbb{R}^n$. However, this does not lead to a loss of generality as non-zero equilibrium points can be shifted to the origin through a variable change: $\mathbf{y} = \mathbf{x} - \bar{\mathbf{x}}$. The condition in Eq. (3.19) states that a region in $\mathbb{R}^n$ around the origin with radius ϵ should always contain a smaller region with radius δ, where if the initial state of the system lays within δ, the system trajectory should never leave ϵ. This property needs to hold for all values of ϵ, for the system to be stable [6]. In simpler terms, this means that when a stable system is initiated close to an equilibrium point, it stays close to the equilibrium point for all future time. A visual illustration of the stability definition is shown in Figure 3.2a.

[1]The term *autonomous* refers to a system without input signals.



(a) Stability (b) Asymptotic stability

Figure 3.2: A visual illustration of the concept of stability for a two-dimensional system.

Definition 2 (Asymptotic stability [10])**.** The equilibrium point $\bar{\mathbf{x}} = 0$ is asymptotically stable if its stable and δ can be chosen such that:

$$\| \mathbf{x}(0) \| < \delta \Rightarrow \lim_{t\to\infty} \mathbf{x}(t) = 0 \tag{3.20}$$

Definition 2 states that a system is asymptotically stable if its stable and there exists a region with radius δ, where if the system is initiated within δ its state trajectory will converge to the equilibrium point. This system property is illustrated in Figure 3.2b.

3.3.2 Lyapunov stability criterion

A powerful method of determining stability around an equilibrium point in a nonlinear system is to use *Lyapunov stability criteria.* This method was first described by the Russian mathematician Aleksandr Mikhailovich Lyapunov in 1892, and involves searching for a function of the system states called the *Lyapunov function*, $V(\mathbf{x})$.

Theorem 1 (Lyapunov stability criterion [10])**.** *Let $\bar{\boldsymbol{x}} = 0$ be an equilibrium point for the autonomous system in Eq. (3.18) and $\mathbb{D} \subset \mathbb{R}^n$ be a domain containing $\boldsymbol{x} = 0$. If and only if there exist a function $V(\boldsymbol{x})$ such that:*

$$1. \quad V(\boldsymbol{x}) = 0, \quad \textit{when } \boldsymbol{x} = \bar{\boldsymbol{x}} \tag{3.21}$$

$$2. \quad V(\boldsymbol{x}) > 0, \quad \forall \boldsymbol{x} \in \mathbb{D} \backslash \{0\} \tag{3.22}$$

$$3. \quad \dot{V}(\boldsymbol{x}) \leq 0, \quad \forall \boldsymbol{x} \in \mathbb{D} \tag{3.23}$$

then, $\bar{\boldsymbol{x}}$ is stable. Furthermore, if

$$\dot{V}(\boldsymbol{x}) < 0, \quad \forall \boldsymbol{x} \in \mathbb{D} \backslash \{0\} \tag{3.24}$$

then $\bar{x}$ is a asymptotically stable equilibrium point.

The criteria in Theorem 1 states that the Lyapunov function has to be a decreasing function that has a minimum point in origin of the state space. This function can be thought of as a general energy function in mechanical systems. Identifying such a function implies that the system is always moving towards a region of lower energy, and therefore is a stable system. An example of a Lyapunov function for a two-dimensional system is shown in Figure 3.3.

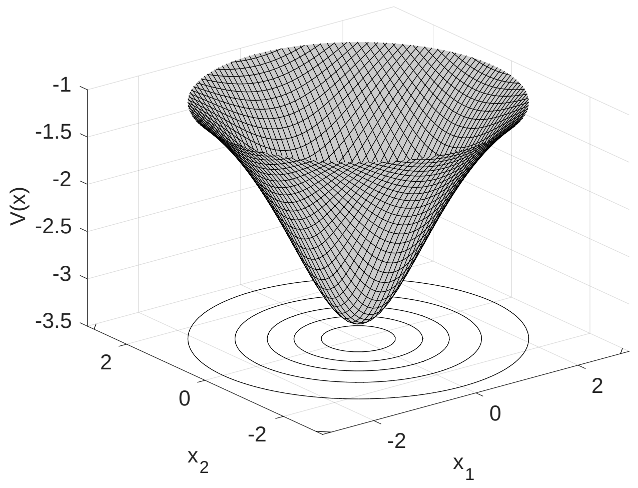


Figure 3.3: Example of a Lyapunov function for a two-dimensional system

3.3.3 LMI-based stability analysis

In most cases, finding Lyapunov functions analytically is considered a difficult task. The Lyapunov criteria are, therefore, more commonly rewritten as *convex optimization problems* in control theory. A convex optimization problem is a problem that involves minimizing a convex objective function or maximizing a concave objective function over constrained sets that are also convex[2]. When a problem is presented as a convex optimization problem, a solution can typically be found quickly by using sophisticated computer algorithms.

By choosing the Lyapunov function as a quadratic function of the form :

$$V(\mathbf{x}) = \mathbf{x}(t)^T \mathbf{P} \mathbf{x}(t) \tag{3.25}$$

where $\mathbf{P}$ is a symmetric matrix, the Lyapunov criteria for an autonomous system can be expressed as a convex optimization problem [23]. The objective of the convex optimization is then to find a matrix $\mathbf{P}$ that satisfies conditions (3.21) -(3.23), and (3.24) when analysing asymptotic stability.

[2]A set is defined as convex if, for any given pair of points within the set, the line segment between the points stays within the set.

Condition (3.21) and (3.22) in Theorem 1 can then be checked by introducing the following linear matrix inequality (LMI):

$$\mathbf{P} \succ 0 \tag{3.26}$$

which states that the matrix $\mathbf{P}$ has to be positive definite, meaning that all the eigenvalues has to be positive. Applying a matrix $\mathbf{P}$ that is positive definite in Eq. (3.25) will ensure that $V(\mathbf{x}) > 0$ for all $\mathbf{x} \neq 0$ and that $V(\mathbf{x}) = 0$.

In order to check condition (3.23) and (3.24) in Theorem 1, an expression of $\dot{V}(x)$ is needed. For an autonomous LTI sytem that has the form:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) \tag{3.27}$$

the derivative of Eq. (3.25) can be derived using the chain rule:

$$\begin{aligned} \dot{V}(\mathbf{x}(t)) &= \dot{\mathbf{x}}(t)^T\mathbf{P}\mathbf{x}(t) + \mathbf{x}(t)^T\mathbf{P}\dot{\mathbf{x}}(t) \\ &= \mathbf{x}(t)^T\mathbf{A}^T\mathbf{P}\mathbf{x}(t) + \mathbf{x}(t)^T\mathbf{P}\mathbf{x}(t)\mathbf{A} \\ &= \mathbf{x}(t)^T(\mathbf{A}^T\mathbf{P} + \mathbf{P}A)\mathbf{x}(t) \\ &= \mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} \end{aligned} \tag{3.28}$$

The asymptotic stability condition (3.24) then leads to the LMI:

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} \prec 0 \tag{3.29}$$

By choosing the Lyapunov function in Eq. (3.25) for an autonomous LPV system of the form:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(\boldsymbol{\psi}(t))\mathbf{x}(t), \quad \boldsymbol{\psi} \in \boldsymbol{\Psi} \subset \mathbb{R}^n \tag{3.30}$$

where $\boldsymbol{\Psi}$ represents a closed set which the parameters $\boldsymbol{\psi}(t)$ vary within, condition (3.23) in Theorem 1 can be satisfied by solving the LMI:

$$\mathbf{A}(\boldsymbol{\psi})^T\mathbf{P} + \mathbf{P}\mathbf{A}(\boldsymbol{\psi}) \prec 0, \quad \forall\boldsymbol{\psi} \in \boldsymbol{\Psi} \tag{3.31}$$

However, this results in a computational problem as the continuous variability of the parameters in $\mathbf{A}(\boldsymbol{\psi})$ leads to an infinite number of inequalities. In [23], a solution is proposed that involves obtaining a polytopic representation of the state matrix $\mathbf{A}(\boldsymbol{\psi})$. This means expressing $\mathbf{A}(\boldsymbol{\psi})$ as a weighted sum of the form:

$$\mathbf{A}(\boldsymbol{\psi}(t)) = \sum_{i=1}^{N} \mu_i(\boldsymbol{\psi}(t))\mathbf{A}_i \tag{3.32}$$

where the number N corresponds to the number of vertices in the polytope such that $\mathbf{A}_i$ becomes so-called vertex matrices. $\mu_i(\psi(t))$ are coefficients of the polytopic presentation that has to satisfy the following condition:

$$\sum_{i=1}^{N} \mu_i(\boldsymbol{\psi}(t)) = 1, \quad \mu_i(\boldsymbol{\psi}(t)) \geq 0 \quad \forall 1, \ldots, N \tag{3.33}$$

If such a polytopic representation can be obtained, then the LMI in Eq. (3.31) can be rewritten as a finite set of inequalities corresponding to the number of vertices in the polytope:

$$\mathbf{A}_i^T\mathbf{P} + \mathbf{P}\mathbf{A}_i \prec 0, \quad \forall 1, \ldots, N \tag{3.34}$$

where the set of inequalities can be solved together trough convex optimization. Then, if a positive definite matrix $\mathbf{P}$ is obtained for all the N number of inequalities, the system is stable.

There exist several numerical computing platforms that can solve LMIs. In this project, the Matlab toolbox `Yalmip` is used. This toolbox acts as an interface between Matlab and different solvers that numerically solve optimization problems. The code example below show how the N number of LMIs is solved, for an arbitrary n-dimensional autonomous LPV system, in the `Yalmip` framework. The different solvers are installed manually and applied by replacing the text string *'thissolver'* with the solver name. In this project, the solvers `Sedumi`, and `SDPT3` and `MOSEK` is used.

Code 3.1: Yalmip Example

```
P = sdpvar(n,n);
F = [P ≥ 1e-09];
% The vertex matrices are stored in the cell array A
for i = 1:N
    F = [F, A{i}'*P+P*A{i} ≤ 1e-09];
end
optimize(F,sdpsettings('solver','thissolver'))
```

Strict inequalities with Yalmip

When using linear programming techniques such as Yalmip to solve optimization problems, strict inequalities cannot be employed. To demonstrate this, let us consider the following simple optimization problem [13]:

$$\begin{array}{cc} \min & x \\ \text{s.t.} & x > 0 \end{array} \tag{3.35}$$

A computer would not be able to solve this problem as there always exist a lower limit for a digital number of any kind. For example, the lowest number that Matlab can produce is 2.2251×10^{-308} [13]. However, presenting this as a solution for the problem in Eq. (3.35) is not correct, as this corresponds to the solution for the non-strict equality problem:

$$\begin{array}{cc} \min & x \\ \text{s.t.} & x \geq 2.2251 \times 10^{-308} \end{array} \tag{3.36}$$

As a consequence of this, Yalmip cannot directly prove asymptotic stability by solving the strict inequalities presented in this section. Therefore, as a solution, the use non-strict inequality with a tolerance value is instead used. In the code example 3.1 a tolerance value of 1×10^-9 have been implemented. However, determining the right tolerance margin can be challenging as a too small margin is equivalent to using the value 0, while a too large margin might lead to a less optimal solution.

3.4 State feedback control design

The 2DOF configuration of the Aero system is a *multiple input multiple output* (MIMO) system, with two input signals: $[v_p(t)\ v_y(t)]^T$ and six output signals: $[\omega_p(t)\ \Omega_p(t)\ \theta_p(t)\ \omega_y(t)\ \Omega_y(t)\ \theta_y(t)]^T$. Controlling MIMO systems can be challenging, as cross-coupling effects often exist between inputs and outputs, and multiple outputs can be affected by changing a single input.

A state-feedback controller is a common approach when dealing with MIMO systems that can be put in the compact matrix form of a state-space model. It involves multiplying the feedback of all state variables with a predetermined gain matrix $\mathbf{K}$, where the matrix multiplication result is the input values that will drive the system to the desired reference value. In the simplest case, where the aim is to drive the states to zero from an arbitrary initial state, the control law can be formulated as:

$$\mathbf{u}(t) = \mathbf{K}\mathbf{x}(t) \tag{3.37}$$

where $\mathbf{u}(t)$ is the system input vector, $\mathbf{K}$ is the gain and $\mathbf{x}(t)$ is the state vector. Figure 3.4 shows a block diagram of how the control law in Eq. (3.37) is implemented on the Aero plant.

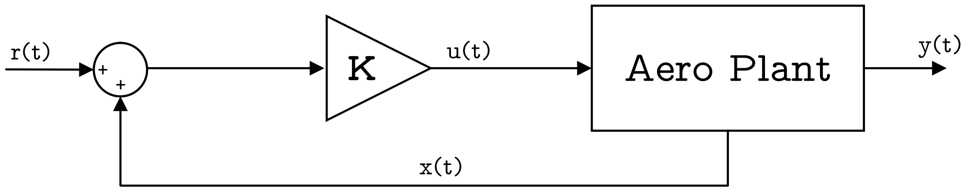


Figure 3.4: A block diagram of the state feedback controller implemented on the Aero system

3.4.1 LMI-based robust control

The implementation of a state feedback controller is, in most cases, trivial, but determining the gain matrix $\mathbf{K}$ that ensures a stable closed loop system is not. A common approach for ensuring closed loop stability when designing controllers for systems based on the LPV framework is to apply the LMI theory described in the previous chapter. This section presents how the LMIs in Eq. (3.26) and (3.30) can be extended to obtain a *robust* controller gain $\mathbf{K}$ that stabilizes the Aero. The term robust refers to the fact that the constant gain matrix $\mathbf{K}$ needs to stabilize the system for all values of $\mathbf{A}(\boldsymbol{\psi}(t))$.

As stated in Chapter 3, the open-loop quasi-LPV model that describes the Aero is expressed as:

$$\begin{cases} \dot{\mathbf{x}}(t) & = \mathbf{A}(\boldsymbol{\psi}(t))\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) & = \mathbf{C}\mathbf{x}(t) \end{cases} \tag{3.38}$$

where the open-loop dynamics and stability is detriment by the properties of the state matrix $\mathbf{A}(\boldsymbol{\psi}(t))$. By implementing the control law in Eq. (3.37) and thereby creating a closed feedback loop, the state space equations are rewritten as:

$$\begin{cases} \dot{\mathbf{x}}(t) &= (\mathbf{A}(\boldsymbol{\psi}(t)) + \mathbf{BK})\mathbf{x}(t) \\ \mathbf{y}(t) &= \mathbf{Cx}(t) \end{cases} \tag{3.39}$$

Now the stability of the closed loop system is not only determined by the state matrix but rather on the matrix $(\mathbf{A}(\boldsymbol{\psi}(t)) + \mathbf{BK})$. This implies that the gain matrix $\mathbf{K}$ can be used to alter the system dynamics and thus stabilize an originally unstable system or alter a system to behave in a desired way. A concept called $\mathcal{D}$-stability [23] allows the poles of the closed loop system to be placed in a desired region in the complex plane, such that a desired transient response can be ordained. However, with regard to the Aero control design, the only closed loop requirement is that the controller is able to stabilize all states to the equilibrium point $\bar{\mathbf{x}} = [0\ 0\ 0\ 0\ 0\ 0]^T$ from an arbitrary initial state. This requirement is akin to finding a gain matrix $\mathbf{K}$ which ensures that all the poles of the matrix $(\mathbf{A}(\boldsymbol{\psi}(t)) + \mathbf{BK})$ have a strictly negative real part for all values of $\boldsymbol{\psi} \in \boldsymbol{\Psi}$. This close loop stability criteria can be formulated as a set of LMIs by substituting $\mathbf{A}(\boldsymbol{\psi}(t))$ with $(\mathbf{A}(\boldsymbol{\psi}(t)) + \mathbf{BK})$ in the LMI in Eq. (3.30):

$$(\mathbf{A}(\boldsymbol{\psi}(t)) + \mathbf{BK})^T\mathbf{P} + \mathbf{P}(\mathbf{A}(\boldsymbol{\psi}(t)) + \mathbf{BK}) \prec 0 \tag{3.40}$$

which can be expanded to:

$$\mathbf{A}(\boldsymbol{\psi}(t))^T\mathbf{P} + \mathbf{PA}(\boldsymbol{\psi}(t)) + \mathbf{K}^T\mathbf{B}^T\mathbf{P} + \mathbf{PBK} \prec 0 \tag{3.41}$$

The resulting inequality in Eq.(3.41), that is obtained by closing the loop, is actually known as a bilinear matrix inequality (BMI) [23]. The reason for this is that the two unknown variables $\mathbf{K}$ and $\mathbf{P}$ appear in the same matrix product. BMIs are not convex and therefore not a desired way of formulating optimization problems. It is possible to solve BMIs, but since they are not convex, there is no guarantee that there exists a global minimum. However, as shown in [23], it is possible to rewrite the BMI in Eq.(3.41) as an LMI by introducing the change of variable $\boldsymbol{\Gamma} = \mathbf{KP}$. In order to easily apply the change of variable , the BMI in Eq.(3.41) is first rewritten into a so-called dual form, which is an completely equivalent representation [23]. This is done by pre- and post-multiplying Eq. (3.41) by $\mathbf{P}^{-1}$:

$$\begin{gathered} \mathbf{A}(\boldsymbol{\psi}(t))^T\mathbf{P} + \mathbf{PA}(\boldsymbol{\psi}(t)) + \mathbf{K}^T\mathbf{B}^T\mathbf{P} + \mathbf{PBK} \prec 0 \\ \mathbf{P}^{-1}\mathbf{A}(\boldsymbol{\psi}(t))^T\mathbf{PP}^{-1} + \mathbf{P}^{-1}\mathbf{PA}(\boldsymbol{\psi}(t))\mathbf{P}^{-1} + \mathbf{P}^{-1}\mathbf{K}^T\mathbf{B}^T\mathbf{PP}^{-1} + \mathbf{P}^{-1}\mathbf{PBKP}^{-1} \prec 0 \\ \mathbf{P}^{-1}\mathbf{A}(\boldsymbol{\psi}(t))^T + \mathbf{A}(\boldsymbol{\psi}(t))\mathbf{P}^{-1} + \mathbf{P}^{-1}\mathbf{K}^T\mathbf{B}^T + \mathbf{BKP}^{-1} \prec 0 \end{gathered} \tag{3.42}$$

Then, as stated in section 3.3.3 the matrix $\mathbf{P}$ is a symmetric matrix. Where the inverse of a symmetric matrix, if it exists, is also a symmetric matrix. Therefor the change $\mathbf{P}^{-1} \rightarrow \mathbf{P}$ can be done to obtain the BMI:

$$\mathbf{A}(\boldsymbol{\psi}(t))\mathbf{P} + \mathbf{PA}(\boldsymbol{\psi}(t))^T + \mathbf{BKP} + \mathbf{PK}^T\mathbf{B}^T \prec 0 \tag{3.43}$$

Now, the change of variable $\boldsymbol{\Gamma} = \mathbf{KP}$ can be preformed to obtain:

$$\mathbf{A}(\boldsymbol{\psi}(t))\mathbf{P} + \mathbf{PA}(\boldsymbol{\psi}(t))^T + \mathbf{B}\boldsymbol{\Gamma} + \boldsymbol{\Gamma}^T\mathbf{B}^T \prec 0 \tag{3.44}$$

which is an LMI.

Finally, in order to make the LMI in Eq. (3.44) computationally solvable the the state matrix $\mathbf{A}(\boldsymbol{\psi}(t))$ is expressed trough the polytopic representation described in Eq. (3.32). This results in an N number of LMIs that is solved together to obtain the two matrices $\mathbf{P}$ and $\mathbf{\Gamma}$:

$$\mathbf{P} \succ 0 \tag{3.45}$$

$$\mathbf{A}_i\mathbf{P} + \mathbf{B\Gamma} + \mathbf{PA}_i^T + \mathbf{\Gamma}^T\mathbf{B}^T \prec 0 \quad \forall i = 1, \ldots, N \tag{3.46}$$

Then, if a feasible solution exists, the robust controller gain $\mathbf{K}$ is obtained by preforming:

$$\mathbf{K} = \mathbf{\Gamma P}^{-1} \tag{3.47}$$

3.4.2 Linear quadratic regulator (LQR)

The LMI-based controller design derived in the previous section can be further improved by incorporating the concept of the linear quadratic regulator (LQR).

Robust LQR control of LTI systems

When implementing the control law in Eq. (3.37) on an LTI system in the form:

$$\dot{\mathbf{x}}(t) = \mathbf{Ax}(t) + \mathbf{Bu}(t) \tag{3.48}$$

the LQR approach allows obtaining an optimal control gain $\mathbf{K}$ by minimizing the cost function:

$$J = \int_0^\infty \left(\mathbf{x}(t)^T\mathbf{Qx}(t) + \mathbf{u}(t)^T\mathbf{Ru}(t)\right) dt \tag{3.49}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are diagonal and positive definite weight matrices that multiply the system states and outputs, respectively. $\mathbf{Q}$ assigns different levels of importance to each state variable. When using the control law (3.37), which stabilizes a system to the state space origin, the matrix $\mathbf{Q}$ is used to prioritize which states should move to zero the fastest. The matrix $\mathbf{R}$ determines the relative importance of the energy consumption in the system actuators and is thus used to assess the aggressiveness of the controller. The importance of each state and input variable is determined by the corresponding diagonal element in $\mathbf{Q}$ and $\mathbf{R}$. As the LQR approach seeks to minimize the value of J, larger values in the weighting matrices correspond to a high level of importance. In contrast, low values correspond to a low level of importance [17].

The strength of the LQR approach lies in the ability to obtain a stable controller gain while systematically placing the poles of the closed loop system:

$$\dot{\mathbf{x}}(t) = (\mathbf{A} + \mathbf{BK})\mathbf{x}(t) \tag{3.50}$$

such that the resulting response has a desired trade-off between controller effort and control quality. For an LTI system, the gain matrix $\mathbf{K}$ can easily be obtained by using the Matlab function `K = lqr(A, B, Q, R)`.

Robust LQR control of LPV systems

The LQR approach cannot be directly applied to case of an LPV system by simply minimizing the value of the cost function J. This is because the final value of J is dependent not only on $\mathbf{Q}$ and $\mathbf{R}$, but also on the trajectory of the varying parameters within $\mathbf{A}(\boldsymbol{\psi}(t))$ [23]. A common way to overcome this problem is to instead minimize the *trace* of the Lyapunov matrix $\mathbf{P}$, where the term *trace* means to take the sum of all the elements in the main diagonal of a matrix and is denoted as $tr(\mathbf{P})$. In order to demonstrate this, let us recall the Lyapunov function introduced in Chapter 3.3:

$$V(\mathbf{x}(t)) = \mathbf{x(t)}^T\mathbf{P}\mathbf{x}(t) \tag{3.51}$$

where $\mathbf{P}$ is a positive definite matrix:

$$\mathbf{P} \succ 0 \tag{3.52}$$

Then, it can be shown that minimizing $tr(\mathbf{P})$ subject to:

$$\dot{V}(\mathbf{x}(t)) + \mathbf{x}(t)^T\mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)^T\mathbf{R}\mathbf{u}(t) < 0 \tag{3.53}$$

will lead to a minimization of J and thus result in an optimal value of $\mathbf{K}$. This is proven by integrating Eq.(3.53) from 0 to ∞:

$$\int_0^\infty \left(\dot{V}(\mathbf{x}(t)) + \mathbf{x}(t)^T\mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)^T\mathbf{R}\ \mathbf{u}(t)\right) dt < 0 \tag{3.54}$$

which, by taking into account Eq. (3.49), results in:

$$V(\mathbf{x}_\infty) - V(\mathbf{x}_0) + J < 0 \tag{3.55}$$

where $\mathbf{x}_\infty$ is equal to zero as all states will converge to the origin of the state space, which means that $V(\mathbf{x}_\infty) = 0$. The vector $\mathbf{x}_0$ refers to the initial condition of the state variables.

Now, by substituting $V(\mathbf{x}_0) = \mathbf{x}_0^T\mathbf{P}\mathbf{x}_0$ in Eq. (3.55):

$$-\mathbf{x}_0^T\mathbf{P}\mathbf{x}_0 + J < 0 \tag{3.56}$$

$$J < \mathbf{x}_0^T\mathbf{P}\mathbf{x}_0 \tag{3.57}$$

it is clear to see that minimizing $tr(\mathbf{P})$ will lead to a minimization of J.

The expression in Eq. (3.55) can then by formulated as a BMI by recalling that $\dot{V}(\mathbf{x}(t))$ is equal to the expression on the left-hand side of Eq. (3.41):

$$\mathbf{A}(\boldsymbol{\psi}(t))^T\mathbf{P} + \mathbf{P}\mathbf{A}(\boldsymbol{\psi}(t)) + \mathbf{K}^T\mathbf{B}^T\mathbf{P} + \mathbf{P}\mathbf{B}\mathbf{K} + \mathbf{x}(t)^T\mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)^T\mathbf{R}\mathbf{u}(t) \prec 0 \tag{3.58}$$

which, by taking into account the control law in Eq. (3.37), becomes:

$$\mathbf{A}(\boldsymbol{\psi}(t))^T\mathbf{P} + \mathbf{P}\mathbf{A}(\boldsymbol{\psi}(t)) + \mathbf{K}^T\mathbf{B}^T\mathbf{P} + \mathbf{P}\mathbf{B}\mathbf{K} + \mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K} \prec 0 \tag{3.59}$$

Then, by multiplying each side of the BMI by -1:

$$-\mathbf{A}(\boldsymbol{\psi}(t))^T\mathbf{P} - \mathbf{P}\mathbf{A}(\boldsymbol{\psi}(t)) - \mathbf{K}^T\mathbf{B}^T\mathbf{P} - \mathbf{P}\mathbf{B}\mathbf{K} - \mathbf{Q} - \mathbf{K}^T\mathbf{R}\mathbf{K} \succ 0 \tag{3.60}$$

and expanding the term $-\mathbf{Q} - \mathbf{K}^T\mathbf{R}\mathbf{K}$ intro three matrices:

$$-\mathbf{A}(\boldsymbol{\psi}(t))^T\mathbf{P} - \mathbf{P}\mathbf{A}(\boldsymbol{\psi}(t)) - \mathbf{K}^T\mathbf{B}^T\mathbf{P} - \mathbf{P}\mathbf{B}\mathbf{K} - \begin{bmatrix}\mathbf{I} & \mathbf{K}^T\end{bmatrix}\begin{bmatrix}\mathbf{Q} & 0\\ 0 & \mathbf{R}\end{bmatrix}\begin{bmatrix}\mathbf{I}\\ \mathbf{K}\end{bmatrix} \succ 0 \tag{3.61}$$

a matrix inequality is obtained, which has a form that allows for a *Schur complement* [4] to be applied. Applying the Schur complement to Eq. (3.61) results in:

$$\begin{bmatrix}-\mathbf{A}(\boldsymbol{\psi}(t))^T\mathbf{P} - \mathbf{P}\mathbf{A}(\boldsymbol{\psi}(t)) - \mathbf{K}^T\mathbf{B}^T\mathbf{P} - \mathbf{P}\mathbf{B}\mathbf{K} & \mathbf{I} & \mathbf{K}^T\\ \mathbf{I} & \mathbf{Q}^{-1} & 0\\ \mathbf{K} & 0 & \mathbf{R}^{-1}\end{bmatrix} \succ 0 \tag{3.62}$$

The use of schur complements is a standard method for expressing matrix inequalities as convex optimizations problems. However, the matrix inequality in Eq. (3.62) is still a BMI, where in order to obtain a LMI, the change of variable $\boldsymbol{\Gamma} = \mathbf{K}\mathbf{P}$ can be performed equivalently to Eq. 3.44. Although to perform the variable change, Eq. (3.62) first needs to be represented it's dual form. This can done as follows:

$$\begin{bmatrix}\mathbf{P}^{-1} & 0 & 0\\ 0 & I & 0\\ 0 & 0 & I\end{bmatrix}\begin{bmatrix}-\mathbf{A}(\boldsymbol{\psi}(t))^T\mathbf{P} - \mathbf{P}\mathbf{A}(\boldsymbol{\psi}(t)) - \mathbf{K}^T\mathbf{B}^T\mathbf{P} - \mathbf{P}\mathbf{B}\mathbf{K} & \mathbf{I} & \mathbf{K}^T\\ \mathbf{I} & \mathbf{Q}^{-1} & 0\\ \mathbf{K} & 0 & \mathbf{R}^{-1}\end{bmatrix}\begin{bmatrix}\mathbf{P}^{-1} & 0 & 0\\ 0 & I & 0\\ 0 & 0 & I\end{bmatrix} \succ 0$$
$$\begin{bmatrix}\mathbf{P}^{-1}(-\mathbf{A}(\boldsymbol{\psi}(t))^T\mathbf{P} - \mathbf{P}\mathbf{A}(\boldsymbol{\psi}(t)) - \mathbf{K}^T\mathbf{B}^T\mathbf{P} - \mathbf{P}\mathbf{B}\mathbf{K})\mathbf{P}^{-1} & \mathbf{P}^{-1} & \mathbf{P}^{-1}\mathbf{K}^T\\ \mathbf{P}^{-1} & \mathbf{Q}^{-1} & 0\\ \mathbf{K}\mathbf{P}^{-1} & 0 & \mathbf{R}^{-1}\end{bmatrix} \succ 0 \tag{3.63}$$

Now, the top left entry of resulting matrix in Eq. (3.63) can be rewritten exactly as in Eq. (3.42), resulting in:

$$\begin{bmatrix}-\mathbf{P}^{-1}\mathbf{A}(\boldsymbol{\psi}(t))^T - \mathbf{A}(\boldsymbol{\psi}(t))\mathbf{P}^{-1} - \mathbf{P}^{-1}\mathbf{K}^T\mathbf{B}^T - \mathbf{B}\mathbf{K}\mathbf{P}^{-1} & \mathbf{P}^{-1} & \mathbf{P}^{-1}\mathbf{K}^T\\ \mathbf{P}^{-1} & \mathbf{Q}^{-1} & 0\\ \mathbf{K}\mathbf{P}^{-1} & 0 & \mathbf{R}^{-1}\end{bmatrix} \succ 0 \tag{3.64}$$

Then, by performing the change $\mathbf{P}^{-1} \to \mathbf{P}$ and the variable change $\boldsymbol{\Gamma} = \mathbf{K}\mathbf{P}$ the following LMI is obtained:

$$\begin{bmatrix}-\mathbf{A}(\boldsymbol{\psi}(t))\mathbf{P} - \mathbf{P}\mathbf{A}(\boldsymbol{\psi}(t))^T - \mathbf{B}\boldsymbol{\Gamma} - \boldsymbol{\Gamma}^T\mathbf{B}^T & \mathbf{P} & \boldsymbol{\Gamma}^T\\ \mathbf{P} & \mathbf{Q}^{-1} & 0\\ \boldsymbol{\Gamma} & 0 & \mathbf{R}^{-1}\end{bmatrix} \succ 0 \tag{3.65}$$

Note that introducing the change $\mathbf{P}^{-1} \to \mathbf{P}$ in Eq. (3.65) leads to the change $\mathbf{P} \to \mathbf{P}^{-1}$ in the lyapunov function $V(\mathbf{x})$:

$$V(\mathbf{x}(t)) = \mathbf{x}(t)^T\mathbf{P}^{-1}\mathbf{x}(t) \tag{3.66}$$

As a consequence, in order to minimize the value of J, the trace of $\mathbf{P}$ should instead be maximized. The reason for this is that taking the inverse of a matrix with high valued entries results in a matrix with low valued entries.

Finally, by representing $\mathbf{A}(\boldsymbol{\psi}(t))$ using the polytopic representation from Eq.(3.32), the convex optimization problem that must be solved in order to obtain a stable controller gain is given as follows [12]:

$$\begin{aligned} \max_{\mathbf{P},\,\boldsymbol{\Gamma}} \quad & tr(\mathbf{P}) \\ \text{s.t.} \quad & \begin{bmatrix} -\mathbf{A}_i\mathbf{P} - \mathbf{P}\mathbf{A}_i^T - \mathbf{B}\boldsymbol{\Gamma} - \boldsymbol{\Gamma}^T\mathbf{B}^T & \mathbf{P} & \boldsymbol{\Gamma}^T \\ \mathbf{P} & \mathbf{Q}^{-1} & 0 \\ \boldsymbol{\Gamma} & 0 & \mathbf{R}^{-1} \end{bmatrix} \succ 0 \quad \forall i = 1, \dots, N \\ \text{and} \quad & \mathbf{P} \succ 0 \end{aligned} \tag{3.67}$$

Once an optimal solution is obtained, the controller gain is simply found by using the resulting values for $\mathbf{P}$ and $\boldsymbol{\Gamma}$:

$$\mathbf{K} = \boldsymbol{\Gamma}\mathbf{P}^{-1} \tag{3.68}$$

The code listing below shows how the optimization problem in Eq. (3.67) is implemented using the Yalmip toolbox in Matlab for a an arbitrary LPV sytem with an time-invariant input matrix $\mathbf{B} \in \mathbb{R}^{n\times m}$ and a time-varying state matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$. In this example, the matrix $\boldsymbol{\Gamma}$ is represented as the variable $\mathbf{L}$. The lines $19-23$ check that the feasible solution returned by a solver is, in fact, feasible by checking the eigenvalues of $\mathbf{P}$ and the matrix expression in Equation (3.67).

Code 3.2: State feedback design using Yalmip

```
1  Q = eye(n);
2  R = eye(m);
3  P = sdpvar(n,n);
4  L = sdpvar(m,n,'full');
5  F = [P ≥ 1e-09];
6  % The vertex matrices are stored in the cell array A
7  for i = 1:N
8      F = [F, [-A{i}*P - P*A{i}' - B*L -L'*B' P L';P inv(Q) zeros(n,m);L ...
            zeros(n,m) inv(R)] ≤ 1e-09];
9  end
10 optimize(F,sdpsettings('solver','thissolver'))
11 K = value(L)*inv(value(Y));
12 
13 E = zeros(N,1);
14 for e =1:nLMI
15     E(e,1) = min(eig([-A{e}*Y_value-B*L_value + ...
            (-A{e}*Y_value-B*L_value)' Y_value L_value';Y_value inv(Q) ...
            zeros(6,2);L_value zeros(2,6) inv(R)]));
16 end
17 
18 %Checking eigenvalues
19 if min(E) ≥ 1e-09 && min(eig(Y_value)) ≥ 1e-09
20    fprintf("No feasible solution");
21 else
22    fprintf("Feasible solution found");
23 end
```

Chapter 4

Quasi-LPV modeling of the Aero

This chapter describes how a quasi-LPV representation of the nonlinear Aero model in Eq. (2.43) has been obtained and then approximated by the use of two different polytopic representations.

4.1 Quasi-LPV representation of the Aero

By applying the substitution-based method introduced in Chapter 3, known as the sector nonlinearity technique [23], the nonlinear Aero model (2.43) can be represented in the quasi-LPV form:

$$\begin{cases} \dot{\mathbf{x}}(t) & = \mathbf{A}(\boldsymbol{\psi}(t))\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) & = \mathbf{C}\mathbf{x}(t) \end{cases} \tag{4.1}$$

where the state vector $\mathbf{x}(t)$ is defined as:

$$\mathbf{x}(t) = [\omega_p(t)\ \Omega_p(t)\ \theta_p(t)\ \omega_y(t)\ \Omega_y(t)\ \theta_y(t)]^T \tag{4.2}$$

and the input vector $\mathbf{u}(t)$ is defined as:

$$\mathbf{u}(t) = [v_p(t)\ v_y(t)]^T \tag{4.3}$$

The parameter vector $\boldsymbol{\psi}(t)$ contains ten varying parameter that embed the nonlinearities in the nonlinear Aero model (2.43):

$$\boldsymbol{\psi}(t) = [\psi_1(t)\ \psi_2(t)\ \psi_3(t)\ \psi_4(t)\ \psi_5(t)\ \psi_6(t)\ \psi_7(t)\ \psi_8(t)\ \psi_9(t)\ \psi_{10}(t)]^T \tag{4.4}$$

These varying parameters are scheduled by all the states in Eq. (4.2) except $\theta_y(t)$, as there are no nonlinearities associated with this state. The decomposition functions that make up the varying parameters are shown in Eq. (4.5)- (4.14):

$$\psi_1(t) = a_{11}(t) = -\frac{K_{\mathcal{T}} K_E}{R_a J_r \omega_p(t)}\omega_p(t) - \frac{f_a(\omega_p(t))}{J_r \omega_p(t)} = -\frac{K_{\mathcal{T}} K_E}{R_a J_r} - \frac{sign(\omega_p(t))k_{a_1}\omega_p(t) + k_{a_2}}{J_r} \tag{4.5}$$

$$\psi_2(t) = a_{21}(t) = \frac{f_{M_p}(\omega_p(t))dt}{J_p\omega_p(t)} = \begin{cases} \frac{d_t}{J_p}(k_{Mpp_1}\omega_p(t) + k_{Mpp_2}) & \text{if } \omega_p(t) \geq 0 \\ \frac{d_t}{J_p}(k_{Mpn_1}\omega_p(t) + k_{Mpn_2}) & \text{if } \omega_p(t) < 0 \end{cases} \tag{4.6}$$

$$\psi_3(t) = a_{22}(t) = -\frac{f_{D_p}(\Omega_p(t))}{\Omega_p(t)} = -sign(\Omega_p(t))k_{Dp_1}\Omega_p(t) - k_{Dp_2} \tag{4.7}$$

$$\psi_4(t) = a_{23}(t) = -\frac{m_b g d_m \sin(\theta_p(t))}{\theta_p(t)} \tag{4.8}$$

$$\psi_5(t) = a_{24}(t) = \frac{f_{T_p}(\omega_y(t))d_t}{\omega_y(t)} = \begin{cases} d_t(k_{Tpp_1}\omega_y(t) + k_{Tpp_2}) & \text{if } \omega_y(t) \geq 0 \\ d_t(k_{Tpn_1}\omega_y(t) + k_{Tpn_2}) & \text{if } \omega_y(t) < 0 \end{cases} \tag{4.9}$$

$$\begin{aligned} \psi_6(t) = a_{25}(t) &= -\frac{\Omega_y(t)^2 d_t^2(m_A + m_B)\sin(2\theta_p(t))}{\Omega_y} \\ &= -\Omega_y(t)d_t^2(m_A + m_B)\sin(2\theta_p(t)) \end{aligned} \tag{4.10}$$

$$\begin{aligned} \psi_7(t) = a_{34}(t) &= -\frac{K_{\mathcal{T}}K_E}{R_a J_r \omega_y(t)}\omega_y(t) - \frac{f_a(\omega_y(t))}{J_r\omega_y(t)} \\ &= -\frac{K_{\mathcal{T}}K_E}{R_a J_r} - \frac{sign(\omega_y(t))k_{a_1}\omega_y(t) + k_{a_2}}{J_r} \end{aligned} \tag{4.11}$$

$$\begin{aligned} \psi_8(t) = a_{41}(t) &= -\frac{f_{M_y}(\omega_p(t))\cos(\theta_p(t))d_t}{J_y(\theta_p)\omega_p(t)} \\ &= \begin{cases} -\frac{\cos(\theta_p(t))d_t}{k_{J_y}\cos(\theta_p(t))\omega_p(t)}(k_{Myp1}\omega_p^2(t) + k_{Myp2}\omega_p(t)) & \text{if } \omega_p(t) \geq 0 \\ -\frac{\cos(\theta_p(t))d_t}{k_{J_y}\cos(\theta_p(t))\omega_p(t)}(k_{Myn1}\omega_p^2(t) + k_{Myn2}\omega_p(t)) & \text{if } \omega_p(t) < 0 \end{cases} \\ &= \begin{cases} -\frac{d_t}{k_{J_y}}(k_{Myp1}\omega_p(t) + k_{Myp2}) & \text{if } \omega_p(t) \geq 0 \\ -\frac{d_t}{k_{J_y}}(k_{Myn1}\omega_p(t) + k_{Myn2}) & \text{if } \omega_p(t) < 0 \end{cases} \end{aligned} \tag{4.12}$$

$$\begin{aligned} \psi_9(t) = a_{44}(t) &= \frac{f_{T_y}(\omega_y(t))\cos(\theta_p(t))d_t}{J_y(\theta_p)\omega_y(t)} \\ &= \begin{cases} \frac{\cos(\theta_p(t))d_t}{k_{J_y}\cos(\theta_p(t))\omega_y(t)}(k_{Typ1}\omega_y^2(t) + k_{Typ2}\omega_y(t)) & \text{if } \omega_y(t) \geq 0 \\ \frac{\cos(\theta_p(t))d_t}{k_{J_y}\cos(\theta_p(t))\omega_y(t}(k_{Tyn1}\omega_y^2(t) + k_{Tyn2}\omega_y(t)) & \text{if } \omega_y(t) < 0 \end{cases} \\ &= \begin{cases} \frac{d_t}{k_{J_y}}(k_{Typ1}\omega_y(t) + k_{Typ2}) & \text{if } \omega_y(t) \geq 0 \\ \frac{d_t}{k_{J_y}}(k_{Tyn1}\omega_y(t) + k_{Tyn2}) & \text{if } \omega_y(t) < 0 \end{cases} \end{aligned} \tag{4.13}$$

$$\begin{aligned}\psi_{10}(t) = a_{45}(t) &= \frac{-f_{D_y}(\Omega_y(t)) - f_{F_y}}{J_y(\theta_p)\Omega_y(t)} \\ &= \begin{cases} \dfrac{-k_{Dyp_1}\Omega_y(t) - k_{Dyp_2} - k_{Fyp}}{k_{J_y}\cos(\theta_p(t))} & \text{if } \Omega_y(t) > 0 \\ \dfrac{-k_{Dyp_2} - \ sat(F_{Ty}(t) \ - F_{My}(t), F_{Fy}(t))}{k_{J_y}\cos(\theta_p(t))} & \text{if } \Omega_y(t) = 0 \\ \dfrac{-k_{Dyn_1}\Omega_y(t) - k_{Dyn_2} - k_{Fyn}}{k_{J_y}\cos(\theta_p(t))} & \text{if } \Omega_y(t) < 0 \end{cases}\end{aligned} \tag{4.14}$$

Placing the varying parameters in the state matrix $\mathbf{A}(\boldsymbol{\psi}(t))$ results in:

$$\mathbf{A}(\boldsymbol{\psi}(t)) = \begin{bmatrix} \psi_1(t) & 0 & 0 & 0 & 0 & 0 \\ \psi_2(t) & \psi_3(t) & \psi_4(t) & \psi_5(t) & \psi_6(t) & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \psi_7(t) & 0 & 0 \\ \psi_8(t) & 0 & 0 & \psi_9(t) & \psi_{10}(t) & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.15}$$

The input matrix elements b_{ij} are time invariant and thus referred to as nonscheduling parameters. As the main rotor and tail rotor are identical, the input parameters are also identical:

$$b_{11} = b_{24} = \frac{K_{\mathcal{T}}}{R_a J_r} \tag{4.16}$$

The input matrix $\mathbf{B}$ is defined as:

$$\mathbf{B} = \begin{bmatrix} b_{11} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & b_{24} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.17}$$

As described in Chapter 2, the Aero is equipped with rotary encoders that enables online measurements of all the state variables in $\mathbf{x}(t)$. The output matrix C is therefore time invariant and equal to the identity matrix I_6:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.18}$$

and the output vector $\mathbf{y}(t)$ is an exact reproduction of the state vector $\mathbf{x}(t)$:

$$\mathbf{y}(t) = [\omega_p(t)\ \Omega_p(t)\ \theta_p(t)\ \omega_y(t)\ \Omega_y(t)\ \theta_y(t)]^T \tag{4.19}$$

The feedthrough matrix $\mathbf{D}$ is equal to zero as there are exist no direct link between the input and output signals.

4.2 Methods for generating polytopic LPV systems

As discussed in the previous chapter, a polytopic representation of the Aero quasi-LPV model (4.1) is needed in order to obtain a stable controller gain. The process of obtaining the polytopic LPV representation expressed in Eq. (3.32) can in simpler terms be described as building a *convex polytope* that encloses a cloud of points. Where the points represent all the possible values of the parameter vector $\boldsymbol{\psi}(t) = [\psi_1(t),\ \ldots, \psi_j(t)]^T$. There exists a number of ways to define the term *convex polytope*. According to a definition known as the vertex representation, a convex polytope is a convex hull of a finite set of points [28], where a convex hull is defined as the intersection of all the convex sets comprising the set of points [26]. Furthermore, a set is defined as convex if, for any given pair of points within the set, the line segment between the points stays within the set [29].

For a given LPV-system, an infinite number of convex figures will allow the model to be represented in a polytopic way. In this section, two bounding box approaches are considered.

4.2.1 Bounding box method

The first approach, called the bounding box method [22], is a simple but efficient technique that involves building a rectangular or a hyper-rectangular box when working in higher dimensions. The obtained box, which satisfies the conditions for being a convex polytope, is constructed to enclose all possible values of $\boldsymbol{\psi}(t)$.

Prior to applying this method, the range of variation for each varying parameter must be known, which can be expressed as:

$$\underline{\psi}_j \leq \psi_j \leq \overline{\psi_j} \tag{4.20}$$

where $\underline{\psi}_j$ and ψ_j denotes the lower and upper bounds. A set of $N = 2^{n_\psi}$ vectors $\boldsymbol{\omega}_i$ are then created, where each vector contains a unique permutation of upper and lower bounds. Then, by combining the vectors $\boldsymbol{\omega}_i$, a convex hull which comprises all values of $\boldsymbol{\psi}(t)$ is formed, where each vector corresponds to a vertex of the resulting polytope. Mathematically, this corresponds to:

$$\boldsymbol{\psi} \in Co\{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_{\boldsymbol{N}}\} \tag{4.21}$$

where Co refers to the term *convex combination* [27]. Then, by using the vectors $\boldsymbol{\omega}_i$, the parameter vector $\boldsymbol{\psi}(t)$ can put in a polytopic representation:

$$\boldsymbol{\psi}(t) \in \left\{ \sum_{i=1}^{N} \mu_i \boldsymbol{\omega}_i, \quad \mu_i \geq 0, \quad \sum_{i=1}^{N} \mu_i = 1 \right\} \tag{4.22}$$

In Figure 4.1, the bounding box method has been applied to an exemplified system consisting of two varying parameters $\boldsymbol{\alpha}(t) = [\alpha_1(t)\ \alpha_2(t)]^T$. For the purposes of avoiding confusion with the Aero model, the varying parameters in the example are denoted as $\alpha(t)$. The resulting convex shape, depicted by the blue line, demonstrates that the bounding box method indeed produces a rectangular box that encompasses all values of $\boldsymbol{\alpha}$.

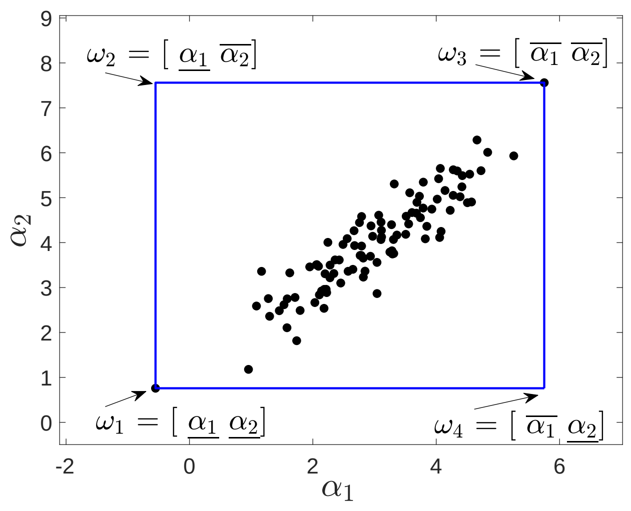


Figure 4.1: Example of the bounding box method applied to a random cloud of correlated points.

As the state matrix $\mathbf{A}(\boldsymbol{\psi}(t))$ depends on the varying parameters, it also will vary within a polytope of matrices:

$$\mathbf{A}(\boldsymbol{\psi}(\boldsymbol{t}) \in Co\{\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_N\} \tag{4.23}$$

where the vertices correspond to the vectors $\boldsymbol{\omega}_i$ [22].

Consequently, the state matrix $\mathbf{A}(\boldsymbol{\psi}(t))$ can be represented as described in chapter 3.3:

$$\mathbf{A}(\boldsymbol{\psi}(t)) = \sum_{i=1}^{N} \mu_i(\boldsymbol{\psi}(t))\mathbf{A}_i \tag{4.24}$$

with:

$$\sum_{i=1}^{N} \mu_i(\boldsymbol{\psi}(t)) = 1, \quad \mu_i(\boldsymbol{\psi}(t)) \geq 0 \quad \forall 1, \ldots, N \tag{4.25}$$

4.2.2 SVD-based boxing

A disadvantage of the bounding box approach is that it commonly produces polytopes encompassing a broader space than the cloud of points formed by the varying parameters. This can be seen in the example in Figure 4.1, where the bounding box method resulted in a relatively loose enclosing of the cloud of points. According to [3] the control performance of LPV systems may be strongly influenced by how the polytopic representation is obtained. In addition, [3] states that when describing LPV systems with a too simple polytopic representation, it may lead to an infeasible solution in the LMI-based control design, even if the system itself can be controllable. However, finding a tighter polytopic representation for a complex quasi-LPV system can be challenging. In [23], the authors state that the number of varying parameters increases exponentially with the number of nonlinearities when using the sector nonlinearity embedding technique. This means that the number of vertices quickly can become significant. In [3], a solution is proposed which involves building a bounding box in a new orthonormal coordinate system by the use of the matrix factorization known as a *singular value decomposition*. This method, which is referred to as SVD-based boxing in [3], may lead to a tighter polytopic representation than the standard bounding box method without increasing the number of vertices.

The singular value decomposition (SVD) is a powerful factorization tool used across a variety of areas, including data processing and machine learning. It involves decomposing a real or complex $n \times m$ matrix $\mathbf{X}$ into the matrix product: $\mathbf{U\Sigma V}^T$. For the purposes of this thesis, let us only consider real matrices.

Theorem 2 (The SVD existence for general real matrices [8])**.** *For any real matrix $\boldsymbol{X} \in \mathbb{R}^{n\times m}$, there exist a decomposition such that:*

$$\boldsymbol{X} = \boldsymbol{U\Sigma}\,\boldsymbol{V}^T \tag{4.26}$$

where $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{V} \in \mathbb{R}^{m\times m}$ are orthogonal matrices, and $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times m}$ is a diagonal matrix with non-negative entries, such that when n < m:

$$\boldsymbol{X} = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \dots & u_m \\ | & | & & | \end{bmatrix} \left[\begin{array}{cccc|ccc} \sigma_1 & & & 0 & 0 & \dots & 0 \\ & \sigma_2 & & & \vdots & & \vdots \\ & & \ddots & & \vdots & & \vdots \\ 0 & & & \sigma_n & 0 & \dots & 0 \end{array}\right] \begin{bmatrix} — & v_1^T & — \\ — & v_2^T & — \\ & \vdots & \\ — & v_n^T & — \end{bmatrix} \tag{4.27}$$

and when n > m:

$$\boldsymbol{X} = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \dots & u_m \\ | & | & & | \end{bmatrix} \left[\begin{array}{cccc} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ 0 & & & \sigma_n \\ \hline 0 & \dots & \dots & 0 \\ \vdots & & & \vdots \\ 0 & \dots & \dots & 0 \end{array}\right] \begin{bmatrix} — & v_1^T & — \\ — & v_2^T & — \\ & \vdots & \\ — & v_n^T & — \end{bmatrix} \tag{4.28}$$

Remark 1. The diagonal entries $\sigma_1, \sigma_2 \dots \sigma_n$ are referred to as singular values of $\mathbf{X}$, while the columns of $\mathbf{U}$ and $\mathbf{V}$ are referred to as left and right singular vectors of $\mathbf{X}$, respectively [30].

The SVD may be interpreted in a variety of ways depending on the practical application. For a matrix $\mathbf{X}$ composed of n number of column vectors containing data points corresponding to an n-dimensional dataset, the SVD presents a tool for identifying the dimensions which exhibit the most variation in the data. In the case of building a convex hull for a set of data points, this property can be advantageously utilized to rotate the data set into a new orthonormal coordinate system in which the axes align with the directions of most variation. Following this, a standard bounding box method can be applied in the new coordinate system, and the vertices of this box can then be rotated back to the original coordinate system. A noteworthy point is that the new orthonormal coordinate frame obtained by the SVD and the original coordinate frame will have the same origin. Therefore, in [3], it is recommended that the data points should be centered at the origin prior to applying the SVD-based boxing. This is achieved by removing the mean value from the set of data.

Example 2. For the purpose of demonstrating the SVD based boxing method, let us consider the LPV system from the example presented in Figure 4:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(\boldsymbol{\alpha}(t)) + \mathbf{B}\mathbf{u}(t) \tag{4.29}$$

where $\boldsymbol{\alpha}(t) = [\alpha_1(t)\ \alpha_2(t)]$ is system's vector of varying parameters.

First, each varying parameter needs to be represented as discrete variables:

$$\boldsymbol{\alpha}_d^i(n) = [\alpha_{1d}^i(n)\ \alpha^i 2d(n)] \tag{4.30}$$

where the superscript i represents an initial coordinate frame and the subscript d indicates that the parameters are discrete.

Then, once discretized, the mean value of each varying parameters is subtracted:

$$\hat{\alpha}_{1d}^i(n) = \alpha_{1d}^i(n) - \bar{\alpha}_{1d}^i \tag{4.31}$$

$$\hat{\alpha}_{2d}^i(n) = \alpha_{2d}^i(n) - \bar{\alpha}_{2d}^i \tag{4.32}$$

where $\bar{\alpha}_{jd}$ indicates the mean value and $\hat{\alpha}_{jd}$ refers to the data being centered.

Next, each varying parameter is expressed as a column vector and combined to form the matrix $\mathbf{M}^i \in \mathbb{R}^{n\times m}$. In this example, with two varying parameters, the matrix $\mathbf{M}^i$ has the size 100×2. The number n corresponds to the length of the column vectors, which is detriment by the number of samples used to describe the discrete varying parameters.

$$\mathbf{M}^i = \begin{bmatrix} \hat{\alpha}_{1d}^i(1) & \hat{\alpha}_{2d}^i(1) \\ \hat{\alpha}_{1d}^i(2) & \hat{\alpha}_{2d}^i(2) \\ \vdots & \vdots \\ \hat{\alpha}_{1d}^i(100) & \hat{\alpha}_{2d}^i(100) \end{bmatrix} \tag{4.33}$$

Then, by applying the SVD, the matrix $\mathbf{M}^i$ is decomposed into:

$$\mathbf{M}^i = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \tag{4.34}$$

where $\mathbf{U} \in \mathbb{R}^{100\times 100}$, $\mathbf{V} \in \mathbb{R}^{2\times 2}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{100\times 2}$.

Now, the dimensions which exhibit the most variation can be identified by examining the matrix $\mathbf{V}^T$:

$$\mathbf{V}^T = \begin{bmatrix} v_{1,1}^T & v_{1,2}^T \\ v_{2,1}^T & v_{2,2}^T \end{bmatrix} \tag{4.35}$$

The rows in $\mathbf{V}^T$ represent vector coordinates for vectors that point in the direction of the largest variation. These row vectors are also sorted in decreasing order from the direction of most variation to the direction of least variation. Figure 4.2 illustrates this property, where $\mathbf{v}_1 = (v_{11}^T \; v_{12}^T)$ is depicted by a red line and $\mathbf{v}_2 = (v_{21}^T \; v_{22}^T)$ appears as a green line. Clearly, $\mathbf{v}_1$ lies in the dimension with the most variation, while $\mathbf{v}_2$ lies in the dimension with the second most variation. Also, when compared to Figure 4.1, Figure 4.2 demonstrates that by removing the mean value, the cloud of points is centered in the origin.

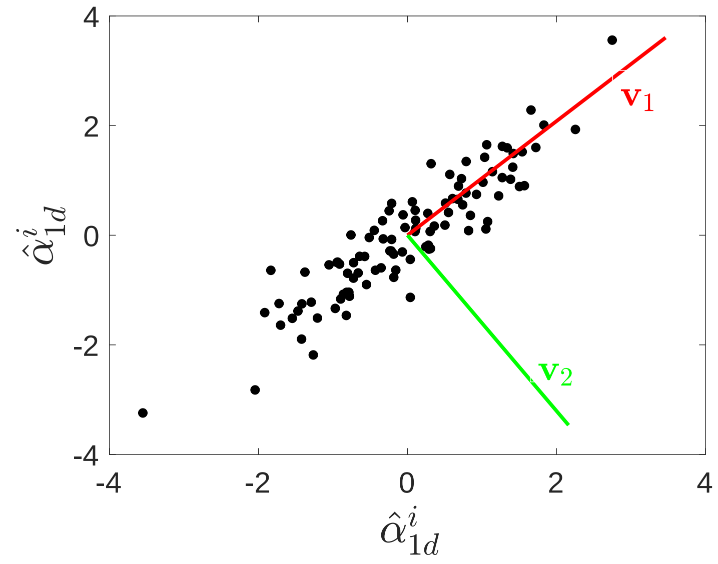


Figure 4.2: Using the SVD to identify the dimensions which exhibit the most variation in a random cloud of correlated points

In addition to indicating the dimensions which exhibit the most variation, the matrix $\mathbf{V}^T$ also acts as a rotation matrix that can rotate the data set into a new coordinate system in which the axes align with the directions of most variation. A rotation matrix that performs a counter-clockwise rotation of a set of points in a two-dimensional plane by an angle of ϕ is defined as:

$$\mathbf{R} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{4.36}$$

Rotation matrices are known as orthogonal matrices, meaning that $\mathbf{R}^T = \mathbf{R}^{-1}$. Thus, the transposed rotation matrix will perform an inverse rotation, that is, a clockwise rotation. This effect can be observed in Figure 4.3.

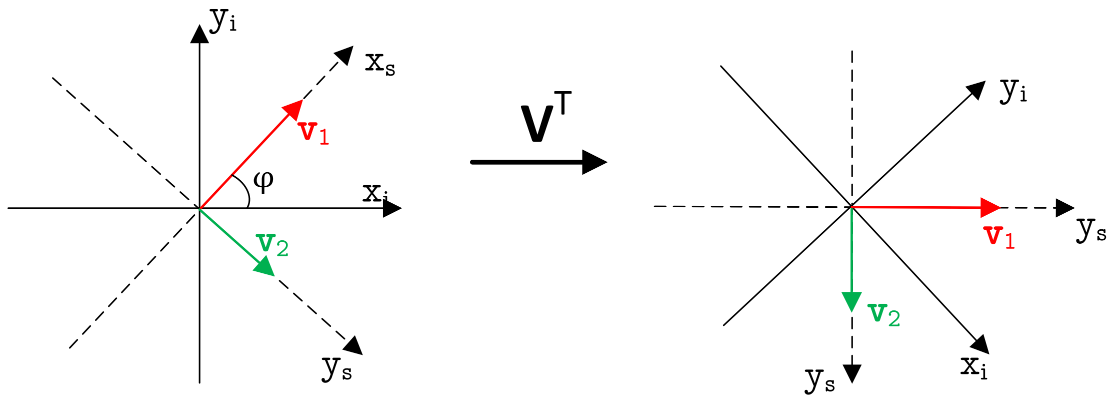


Figure 4.3: Illustration of how $\mathbf{V}^T$ acts as a rotation matrix.

Rotating the cloud of points by using $\mathbf{V}^T$ can then be expressed as:

$$\mathbf{M}^s = (\mathbf{V}^T \mathbf{M}^{iT})^T = \mathbf{M}^i \mathbf{V} \tag{4.37}$$

which corresponds to:

$$\begin{bmatrix} \hat{\alpha}^s_{1d}(1) & \hat{\alpha}^s_{2d}(1) \\ \hat{\alpha}^s_{1d}(2) & \hat{\alpha}^s_{2d}(2) \\ \vdots & \vdots \\ \hat{\alpha}^s_{1d}(100) & \hat{\alpha}^s_{2d}(100) \end{bmatrix} = \begin{bmatrix} \hat{\alpha}^i_{1d}(1) & \hat{\alpha}^i_{2d}(1) \\ \hat{\alpha}^i_{1d}(2) & \hat{\alpha}^i_{2d}(2) \\ \vdots & \vdots \\ \hat{\alpha}^i_{1d}(100) & \hat{\alpha}^i_{2d}(100) \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix} \tag{4.38}$$

where the superscript s refers to the new orthonormal coordinate frame.

Now, the standard bounding box approach can be applied in the new coordinate system as previously described. That is, identifying the lower and upper bounds of the transformed varying parameters $\hat{\alpha}^s_{1d}(n)$ and $\hat{\alpha}^s_{2d}(n)$:

$$\underline{\hat{\alpha}^s_{1d}} \leq \hat{\alpha}^s_{1d}(n) \leq \overline{\hat{\alpha}^s_{1d}} \tag{4.39}$$

$$\underline{\hat{\alpha}^s_{2d}} \leq \hat{\alpha}^s_{2d}(n) \leq \overline{\hat{\alpha}^s{}_{2d}} \tag{4.40}$$

and building vertex vectors containing all possible permutation of upper and lower bounds:

$$\hat{\boldsymbol{\omega}}^s_1 = [\underline{\hat{\alpha}^s_{1d}} \quad \underline{\hat{\alpha}^s_{2dr}}] \tag{4.41}$$

$$\hat{\boldsymbol{\omega}}^s_2 = [\underline{\hat{\alpha}^s_{1d}} \quad \overline{\hat{\alpha}^s_{2dr}}] \tag{4.42}$$

$$\hat{\boldsymbol{\omega}}^s_3 = [\overline{\hat{\alpha}^s_{1d}} \quad \overline{\hat{\alpha}^s_{2d}}] \tag{4.43}$$

$$\hat{\boldsymbol{\omega}}^s_4 = [\overline{\hat{\alpha}^s_{1d}} \quad \underline{\hat{\alpha}^s_{2d}}] \tag{4.44}$$

such that:

$$\hat{\boldsymbol{\alpha}}^s \in Co\{\hat{\boldsymbol{\omega}}^s_1,\ \hat{\boldsymbol{\omega}}^s_2,\ \hat{\boldsymbol{\omega}}^s_3,\ \hat{\boldsymbol{\omega}}^s_4\} \tag{4.45}$$

Figure 4.4 shows the result from applying the bounding box method in the rotated coordinate frame, where the y-axis has been inverted.

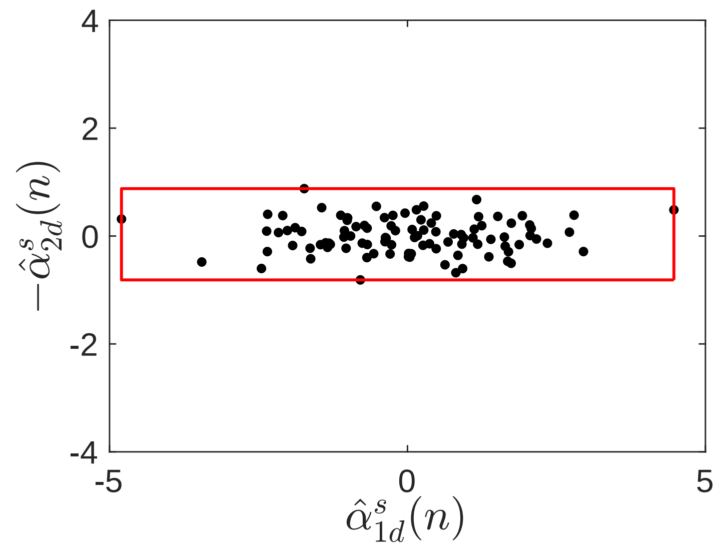


Figure 4.4: Bounding box method applied in a rotated coordinate frame on a random cloud of correlated points

Then, by defining a matrix containing the vertex vectors as column vectors:

$$\hat{\mathbf{\Omega}}^s = \begin{bmatrix}\hat{\boldsymbol{\omega}}^s_1 & \hat{\boldsymbol{\omega}}^s_2 & \hat{\boldsymbol{\omega}}^s_3 & \hat{\boldsymbol{\omega}}^s_4\end{bmatrix} \tag{4.46}$$

the vertex coordinates can be rotated back to the original coordinate frame by performing an inverse linear transformation. That is pre-multiplying $\hat{\mathbf{\Omega}}$ by $(\mathbf{V}^T)^{-1}$ which, since a rotation matrix performers an orthogonal transformation, equals $(\mathbf{V}^T)^T = \mathbf{V}$. The inverse rotation of the vertex vectors can be expressed as:

$$\begin{gathered}\hat{\mathbf{\Omega}}^i = \mathbf{V}\hat{\mathbf{\Omega}}^s \\ \begin{bmatrix}\hat{\boldsymbol{\omega}}^i_1 & \hat{\boldsymbol{\omega}}^i_2 & \hat{\boldsymbol{\omega}}^i_3 & \hat{\boldsymbol{\omega}}^i_4\end{bmatrix} = \begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{\omega}}^s_1 & \hat{\boldsymbol{\omega}}^s_2 & \hat{\boldsymbol{\omega}}^s_3 & \hat{\boldsymbol{\omega}}^s_4\end{bmatrix} \\ \begin{bmatrix}\hat{\omega}^i_{11} & \hat{\omega}^i_{12} & \hat{\omega}^i_{13} & \hat{\omega}^i_{14} \\ \hat{\omega}^i_{21} & \hat{\omega}^i_{22} & \hat{\omega}^i_{23} & \hat{\omega}^i_{24}\end{bmatrix} = \begin{bmatrix}v_{11} & v_{12} \\ v_{21} & v_{22}\end{bmatrix}\left[\begin{array}{cccc}\underline{\hat{\alpha}^s_{1d}} & \underline{\hat{\alpha}^s_{1d}} & \overline{\hat{\alpha}^s_{1d}} & \overline{\hat{\alpha}^s_{1d}} \\ \underline{\hat{\alpha}^s_{2d}} & \overline{\hat{\alpha}^s_{2d}} & \overline{\hat{\alpha}^s_{2d}} & \underline{\hat{\alpha}^s_{2d}}\end{array}\right]\end{gathered} \tag{4.47}$$

Now, since a rotation matrix preserves the ratio of distance between points, the inverse-rotated vertices will encompass the centred cloud of points in the initial coordinate frame:

$$\hat{\boldsymbol{\alpha}}^i \in Co\{\hat{\boldsymbol{\omega}}^i_1,\ \hat{\boldsymbol{\omega}}^i_2,\ \hat{\boldsymbol{\omega}}^i_3,\ \hat{\boldsymbol{\omega}}^i_4\} \tag{4.48}$$

Finally, the mean value of $\alpha_1 d(n)$ and $\alpha_2 d(n)$ is added to first and second row of $\hat{\mathbf{\Omega}}^i$ in order to obtain $\mathbf{\Omega}^i$:

$$\begin{aligned}\mathbf{\Omega}^i &= \begin{bmatrix}\hat{\omega}^i_{11} + \bar{\alpha}^i_{1d} & \hat{\omega}^i_{12} + \bar{\alpha}^i_{1d} & \hat{\omega}^i_{13} + \bar{\alpha}^i_{1d} & \hat{\omega}^i_{14} + \bar{\alpha}^i_{1d} \\ \hat{\omega}^i_{21} + \bar{\alpha}^i_{12d} & \hat{\omega}^i_{22} + \bar{\alpha}^i_{2d} & \hat{\omega}^i_{23} + \bar{\alpha}^i_{2d} & \hat{\omega}^i_{24} + \bar{\alpha}^i_{2d}\end{bmatrix} \\ &= \begin{bmatrix}\boldsymbol{\omega}^i_1 & \boldsymbol{\omega}^i_2 & \boldsymbol{\omega}^i_3 & \boldsymbol{\omega}^i_4\end{bmatrix}\end{aligned} \tag{4.49}$$

such that:

$$\boldsymbol{\alpha}^i \in Co\{\boldsymbol{\omega}_1^i,\ \boldsymbol{\omega}_2^i,\ \boldsymbol{\omega}_3^i,\ \boldsymbol{\omega}_4^i\} \tag{4.50}$$

Figure 4.5 illustrates the resultant SVD boxes for the centered and original points. It is evident that both the centered and the original points are enclosed by the SVD generated polytopes. Hence, the vertex coordinates that encompasses the original points can be used to obtain a polytopic LPV representation of $\mathbf{A}(\boldsymbol{\alpha}(t))$ by following the steps in Eq. (4.21), (4.22), (4.23), (4.24) and (4.25).

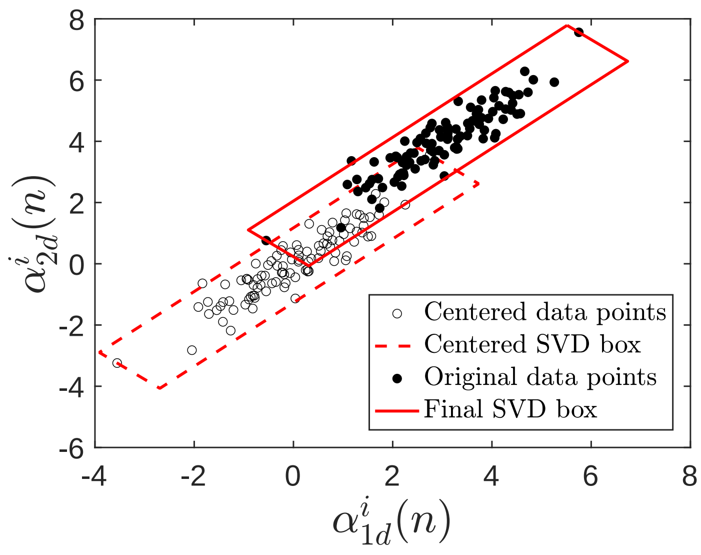


Figure 4.5: Illustration on how the obtained vertex coordinates are translated in order to fit the original set of points.

In Figure 4.6, the standard bounding box is shown in blue while the SVD box is shown in red. Clearly, the SVD-based bounding box encompasses the points more tightly than the original bounding box. In fact, applying the SVD-based boxing resulted in the enclosed area being reduced by 170%. Such a reduction may be crucial when solving LMIs in order to prove stability. As discussed in [3], if there exists an infeasible region close to the cloud of points, the SVD-based boxing might serve as a tool to exclude this region. This scenario is illustrated in Figure 4.6 by a black area which represents a non-controllable region. However, the SVD-based boxing method does not guarantee a tighter enclosing than the standard bounding box approach. Figure 4.7 shows a set of data points in a xy-coordinate system, which is distributed in two different directions. When a standard bounding box and an SVD-based box are applied to these points, the SVD-based box will result in an area that is 6% larger than the standard bounding box.

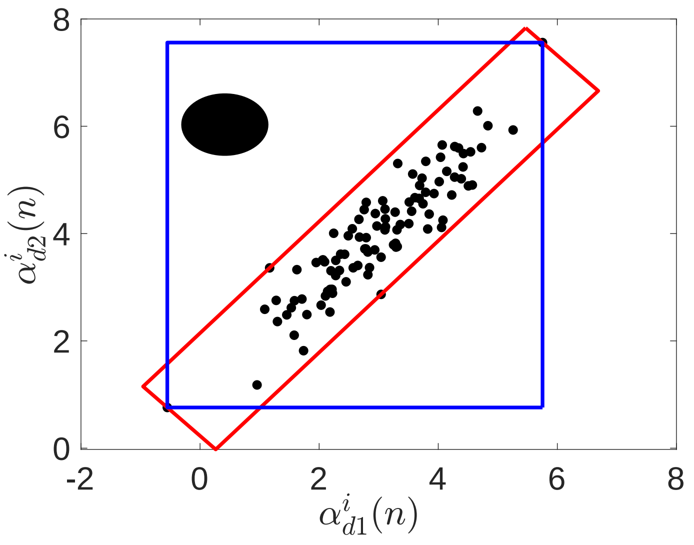


Figure 4.6: Illustrative comparison of the standard and SVD-based bounding box method, where the black area represents an infeasible region.

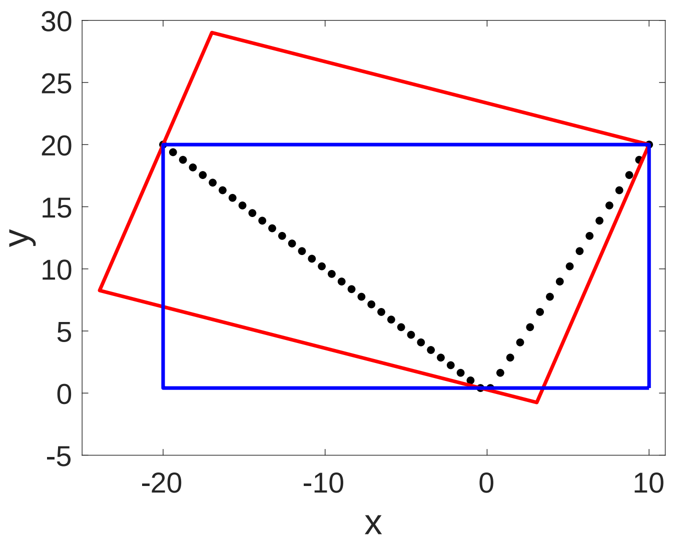


Figure 4.7: An example of a set of data points where the SVD-based method does not result in an improved encompassing

4.3 Polytopic quasi-LPV Aero models

Mathematically, the polytopic quasi-LPV model of the Aero can be expressed as:

$$\begin{cases} \dot{\mathbf{x}}(t) &= \sum_{i=1}^{N} \big(\mu_i \boldsymbol{\psi}(t)\mathbf{A}_i\mathbf{x}(t)\big) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) \end{cases} \tag{4.51}$$

where the non-negative coefficients μ_i follows the condition in (4.25). This section describes how the standard bounding box method and the SVD-based boxing method have been applied in order to obtain the vertex matrices $\mathbf{A}_1, \mathbf{A}_2 \dots \mathbf{A}_i$.

4.3.1 Bounding box method applied to the Aero model

In accordance with section 4.2.1, the first step in building a bounding box is identifying the upper and lower bounds of the system's varying parameters. For the Aero model, the boundary on each varying parameter is determined by the physical limitations of the state variables they depend on. The physical limitations of the state variables $\omega_p(t)$, $\Omega_p(t)$, $\omega_y(t)$, and $\Omega_y(t)$ are are determined by the voltage saturation on the main and tail motor, which are both $\pm 24V$. The construction of the Aero creates a physical limitation for the pitch angle $\theta_p(t)$, while the yaw angle $\theta_y(t)$ is unlimited. Nevertheless, this unlimited yaw rotation does not lead to an issue since there are no varying parameters dependent on $\theta_y(t)$. Table 4.1 shows the physical limitations of the state variables, which were identified by online testing on the Aero experiment setup.

State variable	Minimum	Maximum
$\omega_p(t)$	$-280\ rad/s$	$320\ rad/s$
$\Omega_p(t)$	$-1.2\ rad/s$	$1.5\ rad/s$
$\theta_p(t)$	$-1.065\ rad$	$0.970\ rad$
$\omega_y(t)$	$-280\ rad/s$	$320\ rad/s$
$\Omega_y(t)$	$-8.80\ rad/s$	$8.68\ rad/s$
$\theta_y(p)$	$-\infty\ rad$	$+\infty\ rad$

Table 4.1: limitations on the state variables

A Matlab script attached in Appendix A.2 was used to calculate the local minima and maxima for each of the varying parameters. The resulting lower and upper bounds are presented in table 4.2. As can be seen from the table, the parameter $\psi_3(t)$ is close to constant with the small range of only 1.0×10^{-6}. This is due to the constant K_{Dp_2} being significantly larger than K_{Dp_1} in Eq. (4.7). The state matrix element a_{22} is therefore approximated as the constant value of $-K_{Dp_2}$. This means that there is 9 varying parameters that result in a vertex system with $2^9 = 512$ vertices and $2^9 + 1 = 513$ LMIs that need to be solved.

Parameter, $\psi_j(t)$	Lower Bound, $\underline{\psi_j}$	Upper Bound, $\overline{\psi_j}$
$\psi_1(t)$	-5.29	-3.70
$\psi_2(t)$	-3.40×10^{-4}	4.50×10^{-3}
$\psi_3(t)$	-6.09×10^{-3}	-6.10×10^{-3}
$\psi_4(t)$	-3.27×10^{-2}	-2.69×10^{-2}
$\psi_5(t)$	-2.84×10^{-6}	7.14×10^{-5}
$\psi_6(t)$	-3.81×10^{-1}	3.82×10^{-1}
$\psi_7(t)$	-5.29	-3.70
$\psi_8(t)$	3.70×10^{-4}	8.41×10^{-2}
$\psi_9(t)$	4.13×10^{-4}	9.78×10^{-2}
$\psi_{10}(t)$	-8.11×10^{-1}	3.76×10^{-1}

Table 4.2: Lower and upper bounds of the varying parameters

In Figure 4.8, the range of variation for each varying parameter is graphically presented by a bar graph. This representation makes it clear that the variation of the parameters $\psi_1(t)$, $\psi_6(t)$, $\psi_7(t)$ and $\psi_{10}(t)$ are dominate. This means that the variation in the least varying parameters could be approximated with a constant value. If this were to be done with for example $\psi_2(t)$, $\psi_4(t)$ and $\psi_5(t)$, the number of vertices would be reduce from 512 to $2^6 = 64$. In [22], such an approximation was found through a least-squares algorithm. However, due to the time constraints of this thesis, such an approximation has not been included.

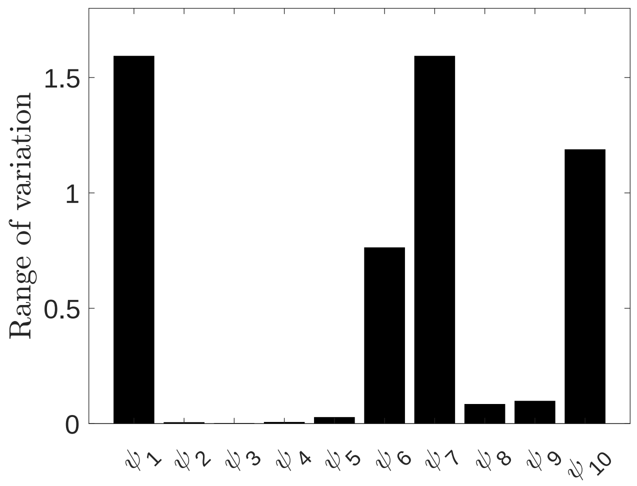
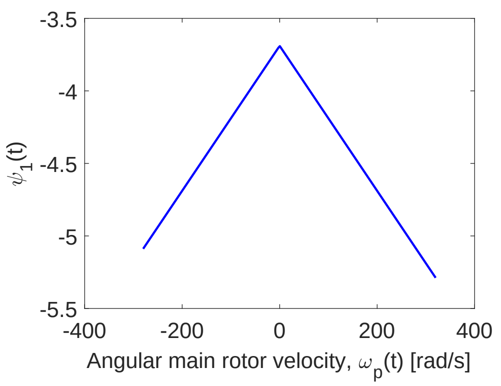
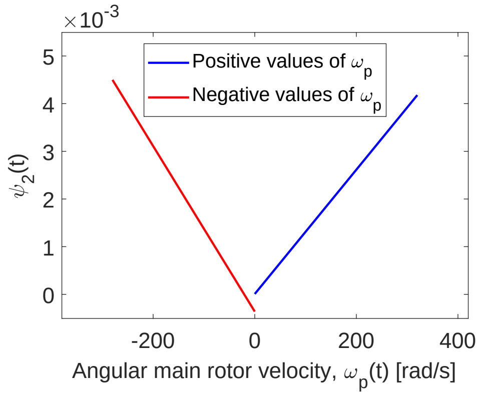


Figure 4.8: A bar chart showing the range of variation for each varying parameter

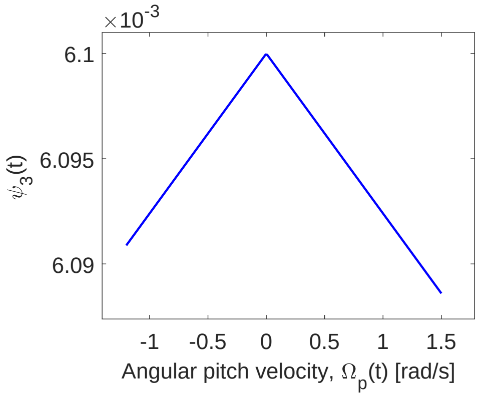
The functions that make up the varying parameters are graphically presented in Figure 4.9, with a plot for each parameter.



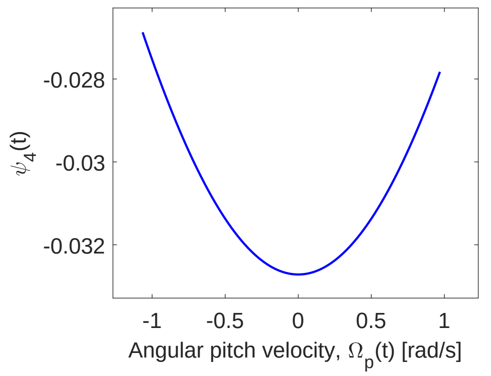
(a) $\psi_1(\omega_p(t))$



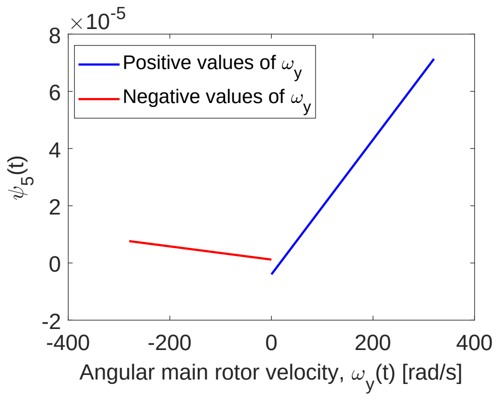
(b) $\psi_2(\omega_p(t))$



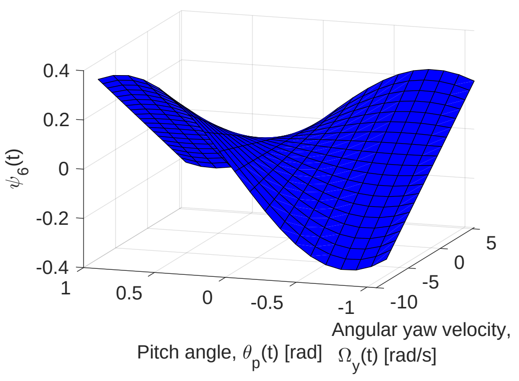
(c) $\psi_3(\Omega_p(t))$



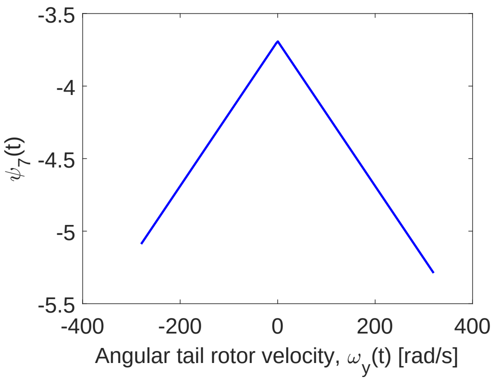
(d) $\psi_4(\Omega_p(t))$



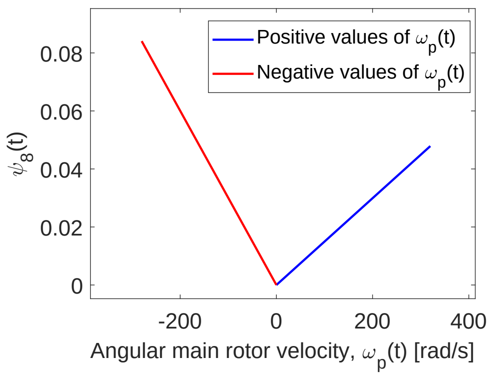
(e) $\psi_5(\omega_y(t))$



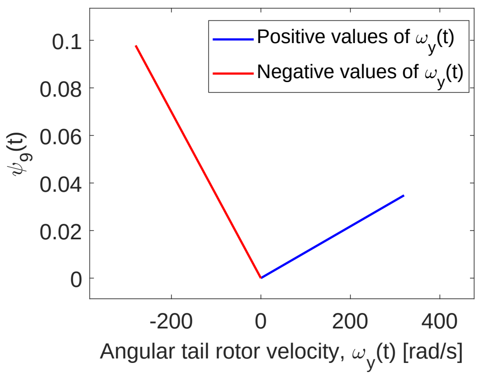
(f) $\psi_6(\Omega_y(t), \theta_p(t))$



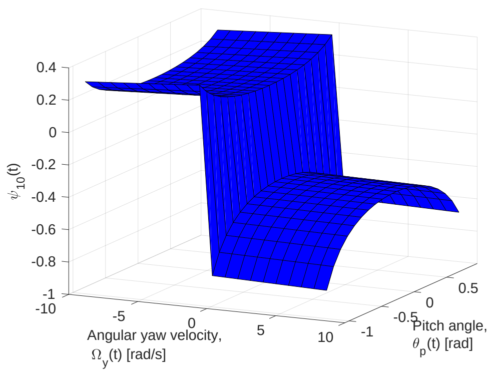
(g) $\psi_7(\omega_y(t))$



(h) $\psi_8(\omega_y(t))$



(i) $\psi_9(\omega_y(t))$



(j) $\psi_{10}(\Omega_y(t), \theta_p(t))$

Figure 4.9: Plots showing how the scheduling parameters $\psi_1(t)$ through $\psi_{10}(t)$ vary within their lower and upper bounds.

The next step in the development of a polytopic Aero quasi-LPV model is placing every possible permutation of the upper and lower bounds from table 4.2 into $N = 512$ vertex matrices. Rather than having to do this manually, a Matlab code has been developed where this is performed automatically through the use of for-loops. This code can be found in Appendix A.2. The generated vertex matrices are then stored in a cell array such that the LMIs for the LQR control design can be solved using Yalmip as shown in the code listing 3.2. The resulting polytopic consisting of 512 LTI systems, can then be expressed as:

$$\begin{cases} \dot{\mathbf{x}}(t) & = \mathbf{A}_{Bbox_1}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \dot{\mathbf{x}}(t) & = \mathbf{A}_{Bbox_2}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \vdots & = \quad \vdots \\ \dot{\mathbf{x}}(t) & = \mathbf{A}_{Bbox_{512}}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \end{cases} \tag{4.52}$$

where the *1st* and the *100th* vertex matrix is defined as:

$$\mathbf{A}_{Bbox_1} = \begin{bmatrix} \overline{\psi}_1 & 0 & 0 & 0 & 0 & 0 \\ \overline{\psi}_2 & K_{Dp2} & \overline{\psi}_4 & \overline{\psi}_5 & \overline{\psi}_6 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \overline{\psi}_7 & 0 & 0 \\ \overline{\psi}_8 & 0 & 0 & \overline{\psi}_9 & \overline{\psi}_{10} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -3.7 & 0 & 0 & 0 & 0 & 0 \\ 0.0045 & -0.0061 & -0.0269 & 7.14e{-5} & 0.382 & 0 \\ 0 & 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -3.7 & 0 & 0 \\ 0.0841 & 0 & 0 & 0.0978 & 0.376 & 0 \\ 0 & 0 & 0 & 0 & 1.0 & 0 \end{bmatrix} \tag{4.53}$$

$$\mathbf{A}_{Bbox_{100}} = \begin{bmatrix} \overline{\psi}_1 & 0 & 0 & 0 & 0 & 0 \\ \overline{\psi}_2 & K_{Dp2} & \underline{\psi}_4 & \underline{\psi}_5 & \overline{\psi}_6 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \overline{\psi}_7 & 0 & 0 \\ \overline{\psi}_8 & 0 & 0 & \underline{\psi}_9 & \underline{\psi}_{10} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -3.7 & 0 & 0 & 0 & 0 & 0 \\ 0.0045 & -0.0061 & -0.0327 & -2.84e{-6} & 0.382 & 0 \\ 0 & 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -3.7 & 0 & 0 \\ 0.0841 & 0 & 0 & 4.13e-4 & -0.811 & 0 \\ 0 & 0 & 0 & 0 & 1.0 & 0 \end{bmatrix} \tag{4.54}$$

4.3.2 SVD-based boxing method applied to the Aero model

To develop a second polytopic representation of the Aero quasi-LPV model based on the SVD boxing method, the varying parameters have been categorized according to which state variables they depend upon. A visual illustration of this categorization can be found in Figure 4.11. The varying parameter ψ_3 is excluded since it is regarded as constant. The SVD-based boxing method was performed separately on each of the three groups following the approach description from section 4.2.2, resulting in three smaller vertex systems, each with the size of $N = 2^3 = 8$. Figure 4.10 shows the resulting three-dimensional SVD-based polytope for ψ_1, ψ_2, and ψ_8 compared to a standard bounding box. In this case, the SVD-based approach resulted in a polytope with 2430 % less volume than a standard box.

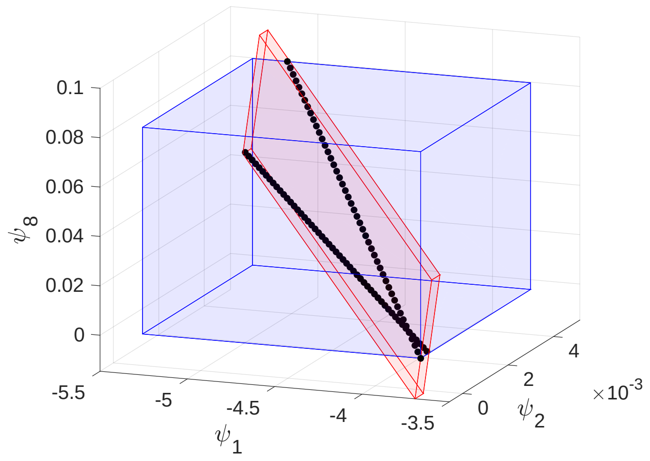


Figure 4.10: SVD-based bounding box (red) compared to a standard bounding box (blue) applied to ψ_1, ψ_2 and ψ_8

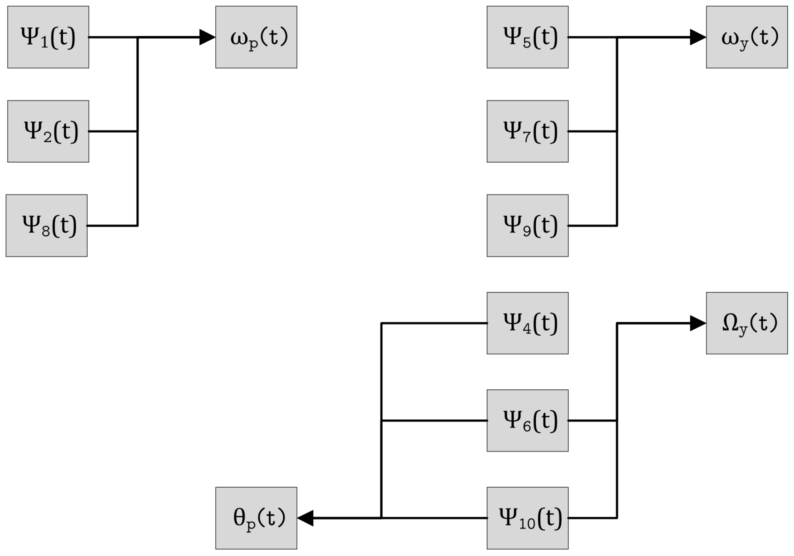


Figure 4.11: Systematized scheme of the state-dependent varying parameters.

Also for ψ_5, ψ_7, and ψ_9, building a polytope based on the SVD theory resulted in a tighter enclosing than with the use of a standard box. This can be seen in Figure 4.12, where in this case, the SVD-based approach resulted in a polytope with 1920 % less volume than a standard box. The fact that the reduction was greater for the results in Figure 4.10 than for the results in Figure 4.12 is linked to the fact that the parameters ψ_5 and ψ_9 has a greater range of variation than ψ_2 and ψ_8, which can be seen in Figure 4.8.

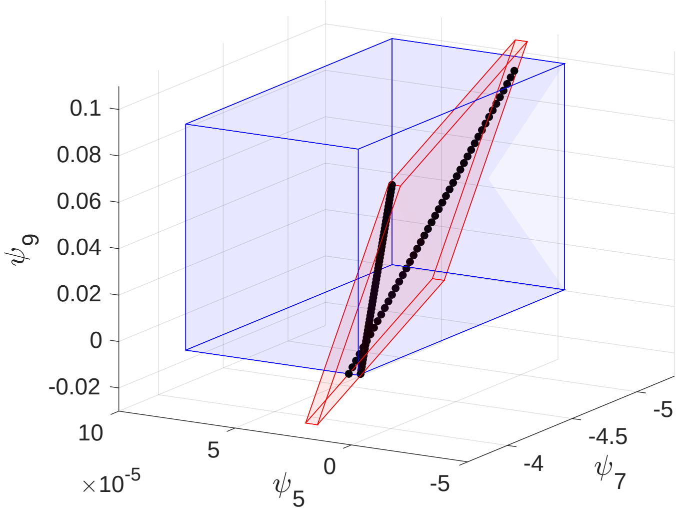


Figure 4.12: SVD-based bounding box (red) compared to a standard bounding box (blue) applied to ψ_5, ψ_7 and ψ_9

For the group containing the parameters ψ_4, ψ_6 and ψ_{10}, applying the SVD-based boxing method resulted in a larger polytope than by using the standard bounding box method. The resulting polytopes are shown in Figure 4.13. The red box corresponding to the SVD-based polytope has a volume that is 8.70% greater than the blue standard bounding box. From the plot in Figure 4.13, it can be observed that the variation in the data points move in two distinct directions. This formation is a result of the piecewise function in Eq. (4.14) which takes different values depending on the sign of $\Omega_y(t)$. Hence, the varying parameters ψ_4, ψ_6, and ψ_{10} falls under the category discussed in section 4.2.2 and illustrated in Figure 4.7, where the SVD-based method does not result in an improved polytopic representation.

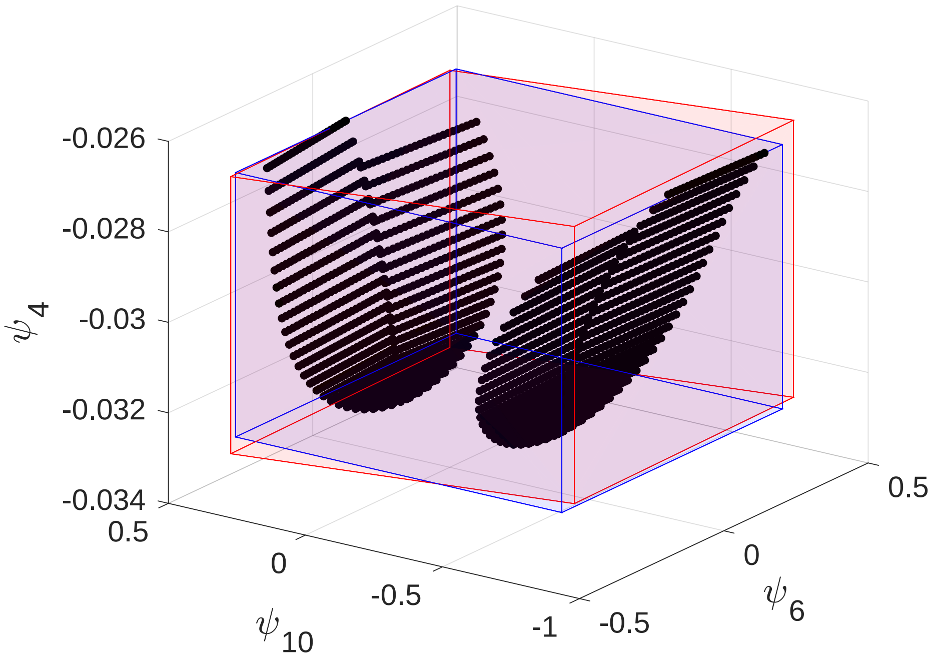


Figure 4.13: SVD-based bounding box (red) compared to a standard bounding box (blue) applied to ψ_4, ψ_6 and ψ_{10}

The three sets of vertex systems of size $N = 8$ are then combined into one larger system with $N = 512$ vertices, which can be expressed as:

$$\begin{cases} \dot{\mathbf{x}}(t) & = \mathbf{A}_{SVD_1}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \dot{\mathbf{x}}(t) & = \mathbf{A}_{SVD_2}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \vdots & = \quad \vdots \\ \dot{\mathbf{x}}(t) & = \mathbf{A}_{SVD_{512}}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \end{cases} \tag{4.55}$$

where the *1st* and the *100th* vertex matrix is defined as:

$$\mathbf{A}_{SVD_1} = \begin{bmatrix} -5.29 & 0 & 0 & 0 & 0 & 0 \\ 0.00444 & -0.0061 & 0.0479 & 0.0479 & -3.7 & 0 \\ 0 & 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.0235 & 0 & 0 \\ 0.424 & 0 & 0 & -0.807 & -0.0265 & 0 \\ 0 & 0 & 0 & 0 & 1.0 & 0 \end{bmatrix} \tag{4.56}$$

$$\mathbf{A}_{SVD_{100}} = \begin{bmatrix} -3.7 & 0 & 0 & 0 & 0 & 0 \\ -2.22e-4 & -0.0061 & -0.0146 & -0.0146 & -3.7 & 0 \\ 0 & 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.0235 & 0 & 0 \\ -0.366 & 0 & 0 & -0.84 & -0.0264 & 0 \\ 0 & 0 & 0 & 0 & 1.0 & 0 \end{bmatrix} \tag{4.57}$$

Chapter 5

Results

The polytopic representations of the Aero quasi-LPV model derived in the previous chapter have been used to solve the robust LQR controller LMIs derived in Chapter 3.3. However, neither the standard bounding box method nor the SVD-based box solution was able to provide feasible solutions due to a too large variation in the varying parameters. In other words, it has not been possible to find a Lyapunov matrix $\mathbf{P}$ and a controller gain $\mathbf{K}$ that satisfy all 512 + 1 LMIs by using the obtained representations. An LQR controller has therefore not been successfully implemented on the Aero. However, s simulation scheme attached in Appendix B.2 shows how an LQR controller could be implemented on the Aero if a feasible controller gain were obtained. This Simulink logic employs an LQR controller developed by Quanser to drive the Aero to different initial positions and then switches to the robust LPV-based LQR controller derived in Chapter 3. The LQR controller designed by Quaser is based on a linearized state space model containing the states: $\mathbf{x}(t) = [\theta_p(t)\ \theta_y(t)\ \Omega_p(t)\ \Omega_y(t)]^T$ [18].

5.1 LMI results for robust control design

Although the obtained polytopic quasi-LPV models did not lead to a feasible controller design, a comparative analysis of the two polytopic representations has been conducted. This has been done by considering two cases, hereafter referred to as scenarios 1 and 2. In these cases, the state variables limits have been further constrained in order to arrive at feasible solutions.

5.1.1 Scenario 1

The state variable constraints used in scenario 1, can be found in the top part of Table 5.1. The lower part of Table 5.1 shows the results form solving the LMIs presented in Eq. (3.67) using the `Yalmip` tool box. Additionally, the table presents a comparison between the polytopic representations obtained and the three different solvers `MOSEK`, `SeDuMi` and `SDPT3`. Each LMI solution is assessed based on the massage returned by the solver, the feasibility of the solution, the resulting value of the trace of $\mathbf{P}$, and the execution time of the code. As the objective in the LMI-based optimization problem in Eq. (3.67) is to maximize the value of the trace of $\mathbf{P}$, a large value of $tr(\mathbf{P})$ is desired.

State variables limits					
State	**Minimum**	**Maximum**	**State**	**Minimum**	**Maximum**
$\omega_p(t)$	80 rad/s	320 rad/s	$\omega_y(t)$	50 rad/s	320 rad/s
$\theta_p(t)$	-1.065 rad	0.970 rad	$\Omega_y(t)$	0 rad/s	8.68 rad/s

LMI results					
Solver	**Polytopic representation**	**Problem status**	**Feasibility**	$tr(\mathbf{P})$	**Execution time** $[s]$
`MOSEK`	Bounding box	*Successfully solved*	Feasible	40.52	2.8 s
	SVD-based box	*Successfully solved*	Feasible	63.01	1.4 s
`SeDuMi`	Bounding box	*Successfully solved*	Feasible	40.27	25.2 s
	SVD-based box	*Numerical problems*	Feasible	63.01	11.4 s
`SDPT3`	Bounding box	*Numerical problems*	Feasible	40.27	12.9 s
	SVD-based box	*Lack of progress*	Feasible	63.01	11.1 s

Table 5.1: Stability assessment, scenario 1

Despite some error messages that solvers `SeDuMi` and `SDPT3` returned, all three solvers were able to determine a feasible solution. It is worth noting that the solver `MOSEK` performed the optimization significantly faster than the other solvers. This is understandable considering `MOSEK` is a commercially licensed product, while `SeDuMi` and `SDPT3` are free. Table 5.1 also shows that using `MOSEK` resulted in a slightly larger value for the trace of $\mathbf{P}$ for the bounding box representation than the other solvers.

By observing the resulting values for the trace of $\mathbf{P}$, it is clear to see that using the SVD-based method results in a more optimal solution. This implies that the SVD-based polytopic representation should have better performance on the Aero than the standard bounding box method. That is, if a method to reduce the variation within the polytope can be obtained such that the LMIs with the original state variable constraints can be solved. This suggestion comes from the fact that the value of $trace(\mathbf{P})$ has a direct link to the value of the linear quadratic cost function J as discussed in Chapter 3.3.

5.1.2 Scenario 2

The second scenario evaluates a case with less constrained state variable limits as seen in the top part of Table 5.2. In this case, only the SVD-base polytopic representation resulted in a feasible solution, which further suggest that this representation should produce a better controller.

State variables limits					
State	**Minimum**	**Maximum**	**State**	**Minimum**	**Maximum**
$\omega_p(t)$	$5\ rad/s$	$320\ rad/s$	$\omega_y(t)$	$-280\ rad/s$	$320\ rad/s$
$\theta_p(t)$	$-1.065\ rad$	$0.970\ rad$	$\Omega_y(t)$	$0\ rad/s$	$8.68\ rad/s$

LMI results					
Solver	**Polytopic representation**	**Problem status**	**Feasibility**	$tr(\mathbf{P})$	**Execution time** $[s]$
`MOSEK`	Bounding box	*Unknown error*	Infeasible	N/A	N/A
	SVD-based box	*Successfully solved*	Feasible	12.04	1.8 s
`SeDuMi`	Bounding box	*Infeasible problem*	Infeasible	N/A	N/A
	SVD-based box	*Numerical problems*	Feasible	12.04	15.1s
`SDPT3`	Bounding box	*Numerical problems*	Infeasible	N/A	N/A
	SVD-based box	*Lack of progress*	Feasible	12.04	11.1 s

Table 5.2: Stability assessment, scenario 2

Figure 5.1 shows the range of variability for the varying parameters for scenarios 1 and 2 compared to the original variation from Figure 4.8. Based on this graph, it is evident that parameter ψ_{10} plays a substantial role in the fact that the LMIs corresponding to the original range of variation are not feasible. Therefore finding a solution to the split formation of points seen in Figure 4.13 could prove to be vital in obtaining a controllable polytopic LPV representation of the Aero. This is further discussed in the next chapter.

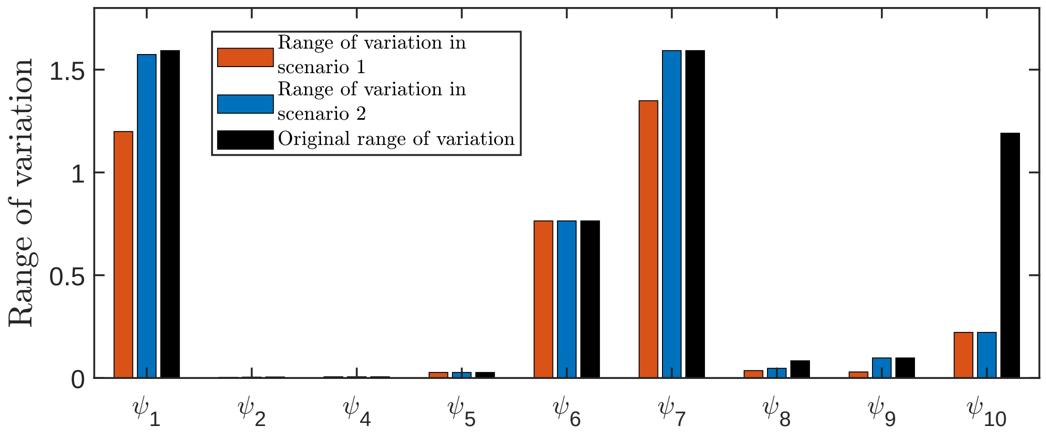


Figure 5.1: Varying parameter variation for scenarios 1 and 2 compared to original variation

Chapter 6

Conclusions and future work

6.1 Conclusions

This thesis has derived and experimentally identified a nonlinear model of the Quanser Aero 2 DOF configuration. Furthermore, the model was proven to represent the real system well. However, through model validation, it was observed that the yaw motion had some unmodeled dynamics and performed more poorly for low angular velocities than for high angular velocities. The nonlinear model was transformed into a quasi-LPV system by introducing ten varying parameters, which later were reduced to nine. Two different polytopic representations of the quasi-LPV model were derived using a bounding box method and an SVD-based approach. This allowed for a finite number of LMIs to be solved in order to obtain a stable, robust LQR state feedback control gain. However, due to too much variation in the varying parameters, a stable controller gain could not be obtained without containing the varying parameters. The control design could therefore not be implemented on the Aero system. Despite this, the two polytopic representations were compared using two different sets of constraints on the system states and thus also the varying parameters. Based on the findings of this project, the following conclusions can be drawn:

- A robust LQR state feedback controller could not be obtained for neither the two polytopic Aero quasi-LPV models because of too much variation in the varying parameters.
- The polytopic representation obtained by using the bounding box method resulted in a polytope encompassing a significantly larger space than the region containing points corresponding to the Aero varying parameters.
- The SVD-based polytopic representation resulted in a smaller convex polytope than the bounding box method for the parameters ψ_1, ψ_2, ψ_8, ψ_5, ψ_7 and ψ_9.
- Due to a split formation of the variation within the parameters ψ_4, ψ_6 and ψ_{10}, the SVD-based polytopic representation did not result in a smaller convex polytope than the bounding box method.
- The SVD-based polytopic representation of the Aero LPV-model resulted in a more optimal LQR controller than the standard bounding box method. In addition, the

SVD-based polytopic representation resulted in a feasible LMI solution for a set of varying parameter constraints where the bounding box method did not.

- In comparison to the other two solvers, the MOSEK solver executes code significantly faster.

6.2 Future work

The following suggests future work that might lead to feasible LMI solutions for the LPV Aero model and further enhance the state feedback controller.

- In order to deal with the fact that the varying parameters $\psi_1(t)$, $\psi_2(t)$, $\psi_5(t)$, $\psi_7(t)$, $\psi_8(t)$, $\psi_9(t)$ and $\psi_{10}(t)$ includes piecewise functions that creates a split in the variation of the parameters, a method known as *control of switched LPV systems* could by applied. This switching effect in the varying parameters in this project was especially seen for the varying parameters in Figure 4.13. However, dealing with the switched variation in all the above mentioned parameters could be vital in order to arrive at a feasible LMI solution. In [9], a switched LPV system is developed by introducing different regions in the state space depending on the values of the system varying parameters. The Lyapunov function $V(\mathbf{x}(t))$ was then made dependent on the varying parameters: $V(\mathbf{x}(t), \boldsymbol{\psi}(t))$ and a family of LPV controllers were designed for each region. This method could be applied on the Aero model by introducing different regions depending on the sign of the states $\omega_{p/y}$ and $\Omega_{p/y}$.
- The state feedback controller could further be improved by introducing gain scheduling control. For an LPV system, this is done by defining a parameter dependent gain matrix $\mathbf{K}(\psi)$ [23]. The control system will then contain a N number of controller gain $\mathbf{K}_i$ corresponding to the N number of vertices of the polytopic representation.
- The state feedback control law could be made more advanced such that the controller would be able to track trajectory references.
- For the SVD-based polytopic representation, the number of vertex systems could be reduced by performing a dimensionality reduction. This is based on the theory known as a principal component analysis (PCA).
- The number of varying parameters could be reduced by neglecting the rotor dynamics and instead creating thrust functions based on different input voltages, as done in [7].
- Other more advanced polytopic representations could be developed.
- The state feedback controller cloud be designed by pole placement methods such as $\mathcal{D}$-stability [23]. This would allow the poles of the closed loop system to be placed in a desired region in the complex plane, such that a desired transient response could be ordained.

The nonlinear Aero model could be further improved by considering the following:

- The model equations for the yaw motion could be improved by incorporation different functions depending on both the magnitude and sign of the yaw velocity. This could resolve the problems shown in Figure 2.13, where the the model performed worse for low angular yaw velocities.
- The damping about the pitch axis could be identified be a sum of least squares algorithm instead of experimental tuning.

References

[1] H. S. Abbas, R. Tóth, M. Petreczky, N. Meskin, and J. Mohammadpour. Embedding of nonlinear systems in a linear parameter-varying representation. *IFAC Proceedings Volumes*, 47(3):6907–6913, 2014.

[2] ACME. *ACME H-60 Dynamic Motion Seat System*, 2022. URL [Online; accessed 06.04.2022].

[3] P. Baranyi. Convex hull generation methods for polytopic representations of lpv models. In *2009 7th International Symposium on Applied Machine Intelligence and Informatics*, pages 69–74. IEEE, 2009.

[4] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[5] F. Casella and M. Lovera. Lpv/lft modelling and identification: overview, synergies and a case study. In *2008 IEEE International Conference on Computer-Aided Control Systems*, pages 852–857. IEEE, 2008.

[6] O. Egeland and J. T. Gravdahl. *Modeling and simulation for automatic control*, volume 76. Marine Cybernetics Trondheim, Norway, 2002.

[7] J. M. Frasik and S. I. L. Gabrielsen. Practical application of advanced control. Master's thesis, University of Agder, 2018.

[8] M. Grasmair. The singular value decomposition. Norwegian University of Science and Technology, 2016. URL [Online; accessed 09.05.2022].

[9] X. He, G. M. Dimirovski, and J. Zhao. Control of switched lpv systems using common lyapunov function method and an f-16 aircraft application. In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 386–392. IEEE, 2010.

[10] H. K. Khalil. Nonlinear systems third edition. *Patience Hall*, 115, 2002.

[11] S. Kumar and L. Dewan. A comparative analysis of lqr and smc for quanser aero. In *Control and Measurement Applications for Smart Grid*, pages 453–463. Springer, 2022.

[12] J. Löfberg. State feedback design for LPV system, 2016. URL [Online; accessed 09.05.2022].

[13] J. Löfberg. Strict inequalities, 2021. URL [Online; accessed 09.05.2022].

[14] Mark Mikofski. *polyfitZero*. MATLAB Central File Exchange, 2022. URL [Online; accessed 02.04.2022].

[15] J. Mohammadpour and C. Scherer. *Control of Linear Parameter Varying Systems with Applications*. Springer New York, 2012. ISBN 9781461418344.

[16] NI. *Quanser AERO: Advanced Controls Applications - Half-Quad*, 2022. URL [Online; accessed 02.04.2022].

[17] K. Ogata et al. *Modern control engineering*, volume 5. Prentice hall Upper Saddle River, NJ, 2010.

[18] *QUANSER AERO - 2DOF Laboratory Guide.* Quanser Inc, 119 Spy Court Markham, Ontario, 2016.

[19] *Quick Start Guide: Quanser AERO USB.* Quanser Inc, 119 Spy Court Markham, Ontario, 2016.

[20] *QUANSER AERO - First Principles Modeling.* Quanser Inc, 119 Spy Court Markham, Ontario, 2020.

[21] Quanser Inc. *Quanser AERO - Features*, 2022. URL [Online; accessed 17.03.2022].

[22] D. Rotondo, F. Nejjari, and V. Puig. Quasi-lpv modeling, identification and control of a twin rotor mimo system. *Control Engineering Practice*, 21(6):829–846, 2013.

[23] D. Rotondo, H. S. Sanchez, F. Nejjari, and V. Puig. Analysis and design of linear parameter varying systems using lmis. *Rev. Iberoam. Autom. Inform. Ind*, 16:1–14, 2019.

[24] J. S. Shamma. *Analysis and design of gain scheduled control systems.* PhD thesis, Massachusetts Institute of Technology, 1988.

[25] H. O. Wang and K. Tanaka. *Fuzzy control systems design and analysis: A linear matrix inequality approach.* John Wiley & Sons, 2004.

[26] Weisstein, Eric W. *Convex Hull.* MathWorld - A Wolfram Web Resource, 2022. URL [Online; accessed 19.04.2022].

[27] Wikipedia. Convex combination — Wikipedia, the free encyclopedia, 2022. URL [Online; accessed 09.05.2022].

[28] Wikipedia. Convex polytope — Wikipedia, the free encyclopedia, 2022. URL [Online; accessed 08.05.2022].

[29] Wikipedia. Convex set — Wikipedia, the free encyclopedia, 2022. URL [Online; accessed 08.05.2022].

[30] Wikipedia. Singular value decomposition — Wikipedia, the free encyclopedia, 2022. URL [Online; accessed 09.05.2022].

Appendices

Appendix A

Matlab code

A.1 Regression Analysis

```
%% Yaw Inertia
clear
close all

YawVelocity_p  = [5.39 5.68 2.42 5.62 5.66 5.68 4.17 6.50 6.77 5.13 ...
    1.84 6.88 6.76 2.64 7.14 7.2 4.47 5.8 5.96]; %rad/s
DecelerationTime_p = [10.9 11.1 5.39 10.67 5.69  9.87 7.97 11.16 9.61 ...
    8.19 3.08 10.26 9.90 4.06 7.23 8.86 3.65 7.75 9.77]; % s
PitchAngle = [0 0.166 0.166  0.20 0.936 0.494 0.417 0.473 0.63 0.638 ...
    0.688 0.688 0.672 0.733 0.942 0.854 -1.08 -0.724 -0.503]; % rad

Deceleration_p = YawVelocity_p./DecelerationTime_p; %rad/s^2

J_y = (1.84e-05*YawVelocity_p.^2 +0.000363*YawVelocity_p + ...
    0.00616)./Deceleration_p;

p_p = polyfit(PitchAngle, J_y, 2);
yfit_p = polyval(p_p,PitchAngle);
yresid = J_y - yfit_p;
SSresid_p = sum(yresid.^2);
SStotal_p = (length(J_y)-1) * var(J_y);
rsq_p = 1 - SSresid_p/SStotal_p;

x_pos = linspace(min(PitchAngle), max(PitchAngle), 10000);

% Defining rgb colors
c1 = [76, 120, 126 ]/256;
c2 = [0, 0, 0 ]/256;
c3 = [247, 93, 89]/256;

figure(1)
plot(PitchAngle, J_y, 'o', 'MarkerEdgeColor', ...
    c1,'MarkerFaceColor',c1, ...
     LineWidth=1.2, MarkerSize=10)
hold on
```

```matlab
plot(x_pos, polyval(p_p,x_pos), 'color', c2, LineWidth=1.6)
plot(x_pos,p_p(3)*cos(x_pos),'--','color', c3,LineWidth=1.6)
xlim([-1.2 1])
xlabel('Pitch angle, \theta_p [rad]')
ylabel('Yaw inertia [kgm^2]')
legend('Data from acceleration test', 'Fitted model', 'Cosine ...
    function',Location='northwest')

set(findall(figure(1),'-property','FontSize'),'FontSize',14)

%% Yaw Friction and Damping
clear
J_y = 0.0184;

DecelerationTime_p = [2.17 2.2 3.3 2.8 3.5 3.9 4.8 6.9 7.8 8.9 9.6 ...
    10.8 11.8 12.2 12.7 13.2 13.3 13.5 15.1 16]; % s
YawVelocity_p  = [0.82 0.891 1.06 1.09 1.14 1.39 1.69 2.54 2.98 3.65 ...
    4.32 4.98 5.57 6.06 6.47 7.05 7.28 7.52 8.42 9.57]; %rad/s;
Deceleration_p = YawVelocity_p./DecelerationTime_p;
FrictionDampingSum_p = Deceleration_p .* J_y;

DecelerationTime_n = [7.0  8.1 8.7 9.7 10.4 11.8 13.0 16.8 15.4 14.6 ...
    3.2 4.4]; % s
YawVelocity_n  = [-2.3  -3.1 -3.7 -4.2 -4.7 -5.6 -6.1 -8.6 -7.8 -6.8 ...
    -0.96 -1.5]; %rad/s;
Deceleration_n = YawVelocity_n./DecelerationTime_n;
FrictionDampingSum_n = Deceleration_n .* J_y;

% Defining rgb colors
c1 = [52, 73, 94 ]/256;
c2 = [0, 0, 0 ]/256;

p_p = polyfit(YawVelocity_p, FrictionDampingSum_p, 2);
yfit_p = polyval(p_p,YawVelocity_p);
yresid_p = FrictionDampingSum_p - yfit_p;
SSresid_p = sum(yresid_p.^2);
SStotal_p = (length(FrictionDampingSum_p)-1) * var(FrictionDampingSum_p);
rsq_p = 1 - SSresid_p/SStotal_p;
x_pos_p = linspace(min(YawVelocity_p),max(YawVelocity_p), 10000);

figure(2)
plot(YawVelocity_p, FrictionDampingSum_p, 'o', 'MarkerEdgeColor', ...
     c1, 'MarkerFaceColor',c1, LineWidth=1.2, MarkerSize=10)
hold on
plot(x_pos_p, polyval(p_p,x_pos_p),"Color",c2, LineWidth=1.6, ...
    MarkerSize=8)
xlabel('Angluar Velocity about the Yaw Axis, \Omega_y [rad/s]')
ylabel('Sum of Damping and Friction [Nm]')
legend('Data from free spin test', 'Fitted model', Location='northwest')
xlim([0 10])
set(findall(figure(2),'-property','FontSize'),'FontSize',14)

p_n = polyfit(YawVelocity_n, FrictionDampingSum_n, 2);
yfit_n = polyval(p_n,YawVelocity_n);
yresid_n = FrictionDampingSum_n - yfit_n;
```

```matlab
SSresid_n = sum(yresid_n.^2);
SStotal_n = (length(FrictionDampingSum_n)-1) * var(FrictionDampingSum_n);
rsq_n = 1 - SSresid_n/SStotal_n;
x_pos_n = linspace(min(YawVelocity_n),max(YawVelocity_n), 10000);

figure(12)
plot(YawVelocity_n, FrictionDampingSum_n, 'o', 'MarkerEdgeColor', ...
    c1, 'MarkerFaceColor',c1, LineWidth=1.2, MarkerSize=10)
hold on
plot(x_pos_n, polyval(p_n,x_pos_n),"Color",c2, LineWidth=1.6, ...
    MarkerSize=8)
xlabel('Angluar Velocity about the Yaw Axis, \Omega_y [rad/s]')
ylabel('Sum of Damping and Friction [Nm]')
legend('Data from free spin test', 'Fitted model', Location='northwest')
ylim([-10e-03 -5e-03])
set(findall(figure(12),'-property','FontSize'),'FontSize',14)



%% Cross Torque about Pitch axis
clear
close all

omega_y_pos = [0  39.6    82.5   118    149   182   213    242   270 ...
     296];
theta_p_pos = [0  0.00613 0.0399 0.0828 0.147 0.218 0.304  0.397 ...
    0.493 0.647];

omega_y_neg =[-322   -297   -242   -213   -182   -149   -118     ...
    -82.6   -39.9 0];
theta_p_neg= [-0.556 -0.459 -0.362 -0.267 -0.192 -0.129 -0.0736  ...
    -0.0338 -0.00307 0];

g = 9.81;
D_m = 0.0029;
m = 1.15;
D_t = 0.158; %m

F_g_pos = g*D_m*m*sin(theta_p_pos)/D_t;
F_g_neg = g*D_m*m*sin(theta_p_neg);

p_pos = polyfit(omega_y_pos, F_g_pos, 2);
yfit_pos = polyval(p_pos,omega_y_pos);
yresid_pos = F_g_pos - yfit_pos;
SSresid_pos = sum(yresid_pos.^2);
SStotal_pos = (length(F_g_pos)-1) * var(F_g_pos);
rsq_pos = 1 - SSresid_pos/SStotal_pos

% Regression model with fixt zero intercept
p_pos2 = polyfitB(omega_y_pos, F_g_pos, 2,0)
yfit_pos2 = polyval(p_pos2,omega_y_pos);
yresid_pos2 = F_g_pos - yfit_pos2;
SSresid_pos2 = sum(yresid_pos2.^2);
SStotal_pos2 = (length(F_g_pos)-1) * var(F_g_pos);
rsq_pos2 = 1 - SSresid_pos2/SStotal_pos2
```

```

p_neg = polyfit(omega_y_neg, F_g_neg, 2)
yfit_neg = polyval(p_neg,omega_y_neg);
yresid_neg = F_g_neg - yfit_neg;
SSresid_neg = sum(yresid_neg.^2);
SStotal_neg = (length(F_g_neg)-1) * var(F_g_neg);
rsq_neg = 1 - SSresid_neg/SStotal_neg

p_neg2 = polyfitB(omega_y_neg, F_g_neg, 2,0)
yfit_neg2 = polyval(p_neg2,omega_y_neg);
yresid_neg2 = F_g_neg - yfit_neg2;
SSresid_neg2 = sum(yresid_neg2.^2);
SStotal_neg2 = (length(F_g_neg)-1) * var(F_g_neg);
rsq_neg = 1 - SSresid_neg2/SStotal_neg2

x_pos = linspace(min(omega_y_pos),max(omega_y_pos), 10000);
x_neg = linspace(min(omega_y_neg),max(omega_y_neg), 10000);
% Defining rgb colors
c1 = [27, 79, 114]/256;
c2 =[0 0.4470 0.7410];

figure(3)
plot(omega_y_pos, F_g_pos, 'o', 'MarkerEdgeColor', c1, ...
    'MarkerFaceColor',c1,LineWidth=1.2, MarkerSize=12)
hold on
plot(x_pos, polyval(p_pos,x_pos),'Color',c2, LineWidth=1.6)
plot(x_pos, polyval(p_pos2,x_pos),'k --',  LineWidth=2.5, MarkerSize=12)
xlabel('Angular velocity of The Tail Rotor, \omega_y ...
    [rad/s]',FontSize=16)
ylabel('f_{Tp}(\omega_y) [N]',FontSize=16)
legend('Data', 'Fitted model','Fixed zero intercept ...
    model',Location='northwest')

figure(4)
plot(omega_y_neg, F_g_neg, 'o', 'MarkerEdgeColor', c1, ...
    'MarkerFaceColor',c1,LineWidth=1.2, MarkerSize=12)
hold on
plot(x_neg, polyval(p_neg,x_neg),"Color",c2, LineWidth=1.6)
plot(x_neg, polyval(p_neg2,x_neg),'k --',  LineWidth=2.5, MarkerSize=12)
xlabel('Angular velocity of The Tail Rotor, \omega_y [rad/s]')
ylabel('f_{Tp}(\omega_y) [N]')
legend('Data', 'Fitted model','Fixed zero intercept ...
    model',Location='northwest')

set(findall(figure(3),'-property','FontSize'),'FontSize',16)
set(findall(figure(4),'-property','FontSize'),'FontSize',16)

%% Main Torque About Pitch Axis
clear
close all

theta_p = [-0.543 -0.41 -0.288 -0.184 -0.114 -0.046 -0.0092 ....
    0 0.00614 0.046 0.0982 0.166 0.249 0.347 0.431];

omega_p_pos = [0 39.6    82.5  118    149   182   213   242];
```

```
theta_p_pos = [0 0.00614 0.046 0.0982 0.166 0.249 0.347 0.431];

omega_p_neg = [-242 -213 -182 -149 -118 -82.6 -39.9 0];
theta_p_neg = [-0.543 -0.41 -0.288 -0.184 -0.114 -0.046 -0.0092 0];

g = 9.81;
D_m = 0.0029;
m = 1.15;
D_t = 0.158; %m

F_g_pos = g*D_m*m*sin(theta_p_pos)/D_t;
F_g_neg = g*D_m*m*sin(theta_p_neg)/D_t;

p_pos = polyfit(omega_p_pos, F_g_pos, 2);
yfit_pos = polyval(p_pos,omega_p_pos);
yresid_pos = F_g_pos - yfit_pos;
SSresid_pos = sum(yresid_pos.^2);
SStotal_pos = (length(F_g_pos)-1) * var(F_g_pos);
rsq_pos = 1 - SSresid_pos/SStotal_pos;

% Regression model with fixt zero intercept
p_pos2 = polyfitB(omega_p_pos, F_g_pos, 2,0);
yfit_pos2 = polyval(p_pos2,omega_p_pos);
yresid_pos2 = F_g_pos - yfit_pos2;
SSresid_pos2 = sum(yresid_pos2.^2);
SStotal_pos2 = (length(F_g_pos)-1) * var(F_g_pos);
rsq_pos2 = 1 - SSresid_pos2/SStotal_pos2;

p_neg = polyfit(omega_p_neg, F_g_neg, 2);
yfit_neg = polyval(p_neg,omega_p_neg);
yresid_neg = F_g_neg - yfit_neg;
SSresid_neg = sum(yresid_neg.^2);
SStotal_neg = (length(F_g_neg)-1) * var(F_g_neg);
rsq_neg = 1 - SSresid_neg/SStotal_neg;

% Regression model with fixt zero intercept
p_neg2 = polyfitB(omega_p_neg, F_g_neg, 2,0);
yfit_neg2 = polyval(p_neg2,omega_p_neg);
yresid_neg2 = F_g_neg - yfit_neg2;
SSresid_neg2 = sum(yresid_neg2.^2);
SStotal_neg2 = (length(F_g_neg)-1) * var(F_g_neg);
rsq_neg2 = 1 - SSresid_neg2/SStotal_neg2;

x_pos = linspace(min(omega_p_pos),max(omega_p_pos), 10000);
x_neg = linspace(min(omega_p_neg),max(omega_p_neg), 10000);

% Defining rgb colors
c1 = [128, 0,0 ]/256;
c2 =[0 0.4470 0.7410];

figure(6)
plot(omega_p_pos, F_g_pos, 'o', 'MarkerEdgeColor', c1, ...
    'MarkerFaceColor',c1,LineWidth=1.2, MarkerSize=12)
hold on
plot(x_pos, polyval(p_pos,x_pos),"Color",c2,LineWidth=1.5, MarkerSize=12)
plot(x_pos, polyval(p_pos2,x_pos),'k --',  LineWidth=2.5, MarkerSize=12)
xlabel('Angular velocity of The Main Rotor, \omega_p [rad/s]')
ylabel('f_{1p}(\omega_p) [N]')
legend('Data', 'Fitted model', 'Fixed zero intercept ...
    model',Location='northwest')

figure(7)
plot(omega_p_neg, F_g_neg, 'o', 'MarkerEdgeColor', c1, ...
    'MarkerFaceColor',c1,LineWidth=1.2, MarkerSize=12)
hold on
plot(x_neg, polyval(p_neg,x_neg),"Color",c2, LineWidth=1.6, ...
    MarkerSize=12)
plot(x_neg, polyval(p_neg2,x_neg),'k --',  LineWidth=2.5, MarkerSize=12)
xlabel('Angular velocity of The Main Rotor, \omega_p [rad/s]')
ylabel('f_{2}(\omega_p) [N]')
legend('Data', 'Fitted model','Fixed zero intercept ...
    model',Location='northwest')

set(findall(figure(6),'-property','FontSize'),'FontSize',16)
set(findall(figure(7),'-property','FontSize'),'FontSize',16)
```

A.2 Bounding Box Method

```
clear
format short e
% Constants
Aero_parameters

% State variable limits

op_min = 5;
op_max = 320;
oy_min = -280;
oy_max = 320;
theta_p_max = 0.9695;
theta_p_min = -1.065;
Omega_y_min = -8.8;
Omega_y_max = 0;

op = linspace(op_min, op_max,100);
oy = linspace(oy_min, oy_max,100);
numPoints = 50;
Omega_y_v = linspace(Omega_y_min, Omega_y_max , numPoints);
Omega_y_v2 = [Omega_y_v(1,1:25), 0, Omega_y_v(1,26:end) ];
theta_p_v = linspace(theta_p_min, theta_p_max, numPoints);
theta_p_v2 = [theta_p_v(1,1:26), 0, theta_p_v(1,27:end) ];
[Omega_y, theta_p] = ndgrid(Omega_y_v2, theta_p_v2);

% Calculating the functions that make up the varying parameters
p1 = -((K_tau*K_E)/(R_a*J_r)) - (sign(op)*k_a1.*op + k_a2)/J_r;
p2 = zeros(1,length(op));
p8 = zeros(1,length(op));
```

```
for i = 1:length(op)
    if op(i) ≥ 0
        p2(i) = d_t*(K_Mpp1*op(i) + K_Mpp2)/J_p;
        p8(i) = -(d_t/K_Jy)*(K_Myp2.*op(i) + K_Myp1);
    else
        p2(i) = d_t*(K_Mpn1*op(i) + K_Mpn2)/J_p;
        p8(i) = -(d_t/K_Jy)*(K_Myn2*op(i) + K_Myn1);
    end
end

p7 = -((K_tau*K_E)/(R_a*J_r)) - (sign(oy)*k_a1.*oy + k_a2)/J_r;
p5 = zeros(1,length(oy));
p9 = zeros(1,length(oy));
for i = 1:length(oy)
    if oy(i) ≥ 0
        p5(i) = d_t*(K_Tpp1.*oy(i) + K_Tpp2);
        p9(i) = (d_t/K_Jy)*(K_Typ2.*oy(i) + K_Typ1);
    else
        p5(i) = d_t*(K_Tpn1.*oy(i) + K_Tpn2);
        p9(i) = (d_t/K_Jy)*(K_Tyn2.*oy(i) + K_Tyn1);
    end
end

p6 = -Omega_y.*d_t^2*(m_A + m_B).*sin(2.*theta_p);
p10 = zeros(length(Omega_y));
p4 = zeros(length(theta_p));
n = size(Omega_y);
for i = 1:n(1)^2
    if Omega_y(i)> 0
        p10(i) = (-K_Dyp1.*Omega_y(i) - K_Dyp2 - ...
            K_Fyp)./(K_Jy.*cos(theta_p(i)));
    elseif Omega_y(i)< 0
        p10(i) = (-K_Dyn1.*Omega_y(i) - K_Dyn2 - ...
            K_Fyn)./(K_Jy.*cos(theta_p(i)));
    elseif Omega_y(i) == 0
        p10(i) = (-K_Dyn2 - K_Fyn)./(K_Jy.*cos(theta_p(i)));
    end
    if theta_p(i)== 0
        p4(i) = -m_b*g*d_m; % since lim x-> 0 sin(x)/x = 1
    else
        p4(i) = -(m_b*g*d_m.*sin(theta_p(i)))./theta_p(i);
    end
end

p1_min = min(p1);
p2_min = min(p2);
p4_min = min(p4,[],'all');
p5_min = min(p5);
p6_min = min(p6,[],'all');
p7_min = min(p7);
p8_min = min(p8);
p9_min = min(p9);
p10_min = min(p10,[],'all');

p1_max = max(p1);
```

```
p2_max = max(p2);
p4_max = max(p4,[],'all');
p5_max = max(p5);
p6_max = max(p6,[],'all');
p7_max = max(p7);
p8_max = max(p8);
p9_max = max(p9);
p10_max = max(p10,[],'all');

% Building the vertex matrices
v_min = [p1_min p2_min  p4_min p5_min p6_min p7_min p8_min p9_min ...
    p10_min];
v_max = [p1_max p2_max  p4_max p5_max p6_max p7_max p8_max p9_max ...
    p10_max];
n = length(v_min);
% Creating a matrix that contain all the possible combinations of a ...
    10 bit
% binary code
VertexVectors = (dec2bin(2^n-1:-1:0)-'0');
% Exchanging the number 1 with upper bounds and the number 0 with ...
    lower bounds. Each row of the VertexVectors matrix then becomes a
% row vector containing all the varying parameters with different ...
    combinations of upper and lower bounds.


for i = 1:2^n
    for j = 1:n
        if VertexVectors(i,j) == 1
            VertexVectors(i,j)= v_max(j);
        else
            VertexVectors(i,j)= v_min(j);
        end
    end
end

A = cell(1,n);
% a is used to create the different matrices that will be stored in A
a = (zeros(5,6));
N = size(VertexVectors);


% Loops through all the vertex vectors and builds vertex matrices
for k = 1:N(1)
     a(1,1) = VertexVectors(k,1);
     a(2,1) = VertexVectors(k,2);
     a(2,2) = -K_DP2;
     a(2,3) = VertexVectors(k,3);
     a(2,4) = VertexVectors(k,4);
     a(2,5) = VertexVectors(k,5);
     a(3,2) = 1;
     a(4,4) = VertexVectors(k,6);
     a(5,1) = VertexVectors(k,7);
     a(5,4) = VertexVectors(k,8);
     a(5,5) = VertexVectors(k,9);
     a(6,5) = 1;
     A{1,k} = a;
end

% Solving the LMI's
b11 = K_tau/(R_a*J_r);
b24 = b11;
B = [b11 0; 0 0; 0 0; 0 b24; 0 0; 0 0];
nLMI = N(1);

%robust controller
Q = eye(6);
R = eye(2);
Y = sdpvar(6,6);
L = sdpvar(2,6,'full');
F = [Y ≥ 1e-09];
for x = 1:nLMI
     F = [F, [-A{x}*Y-B*L + (-A{x}*Y-B*L)' Y L';Y inv(Q) zeros(6,2);L ...
         zeros(2,6) inv(R)] ≥1e-09];
end
%optimize(F,-trace(Y), ...
    sdpsettings('solver','sedumi','verbose',0,'debug',1))
optimize(F,-trace(Y),sdpsettings('solver','sdpt3','sdpt3.maxit',500,'verbose',0,'debug',1
Y_value = value(Y);
L_value = value(L);
K = value(L)*inv(value(Y));
E = zeros(nLMI,1);
for e =1:nLMI
   E(e,1) = min(eig([-A{e}*Y_value-B*L_value + ...
       (-A{e}*Y_value-B*L_value)' Y_value L_value';Y_value inv(Q) ...
       zeros(6,2);L_value zeros(2,6) inv(R)]));
end

fprintf('The value of -trace(Y)for the robust control = %e.\n\n', ...
    value(trace(Y)));

%Checking eigenvalues
 if min(E) ≥ 0 && min(eig(Y_value)) ≥ 0
       fprintf('\nok\n');
   else
       fprintf('\nnot ok\n');
 end


%%
%Range
p1_range = abs(p1_max-p1_min);
p1_range_2 = 1.5736;
p2_range = abs(p2_max-p2_min);
p2_range_2 = 4.1067e-03;
p4_range = abs(p4_max-p4_min);
p4_range_2 = 5.8432e-03;
p5_range = abs(p4_max-p5_min);
p5_range_2 =  2.6870e-02;
p6_range = abs(p6_max-p6_min);
p6_range_2 = 7.6416e-01;
```

```
p7_range = abs(p7_max-p7_min);
p7_range_2 = 1.5926e+00;
p8_range = abs(p8_max-p8_min);
p8_range_2 = 4.7135e-02;
p9_range = abs(p9_max-p9_min);
p9_range_2 = 9.7425e-02;
p10_range = abs(p10_max-p10_min);
p10_range_2 = 2.2166e-01;

R = [[p1_range p1_range_2 1.5926e+00] ;[p2_range p2_range_2 ...
    4.8381e-03]; [p4_range p4_range_2 5.8432e-03 ];
    [p5_range p5_range_2 2.6870e-02 ]; [p6_range p6_range_2 ...
        7.6416e-01]; [p7_range p7_range_2 1.5926e+00];
    [p8_range p8_range_2 8.3692e-02]; [p9_range p9_range_2 9.7425e-02 ...
        ]; [p10_range p10_range_2 1.1908e+00];
    ];


X = categorical({'\psi_1','\psi_2','\psi_4','\psi_5','\psi_6', ...
    '\psi_7','\psi_8','\psi_9','\psi_{10}'});
X = reordercats(X,{'\psi_1','\psi_2','\psi_4','\psi_5','\psi_6', ...
    '\psi_7','\psi_8','\psi_9','\psi_{10}'});

figure(11)
bar(X,R)
set(findall(figure(11),'-property','FontSize'),'FontSize',16)
ylabel('Range of variation','interpreter','latex')
ylim([0 1.8])

%
% Plots:
figure(1)
plot(op, p1,'b','LineWidth',1.6)
xlabel('Angular main rotor velocity, \omega_p(t) [rad/s]')
ylabel('\psi_1(t)')
set(findall(figure(1),'-property','FontSize'),'FontSize',16)

figure(2)
plot(op, p2,'b','LineWidth',1.6)
xlabel('Angular main rotor velocity, \omega_p(t) [rad/s]')
ylabel('\psi_2(t)')
set(findall(figure(2),'-property','FontSize'),'FontSize',16)
hold off


figure(4)
plot(theta_p, p4,'b','LineWidth',1.6)
xlabel('Angular pitch velocity, \Omega_p(t) [rad/s]')
ylabel('\psi_4(t)')
set(findall(figure(4),'-property','FontSize'),'FontSize',16)

figure(5)
plot(oy, p5,'b','LineWidth',1.6)
xlabel('Angular main rotor velocity, \omega_p(t) [rad/s]')
ylabel('\psi_5(t)')
```

```matlab
set(findall(figure(5),'-property','FontSize'),'FontSize',16)
hold off

figure(6)
surf(Omega_y, theta_p, p6, 'FaceColor','b')
xlabel('Angular yaw velocity, \newline \Omega_y(t) [rad/s]')
ylabel('Pitch angle, \theta_p(t) [rad]')
zlabel('\psi_6(t)')
set(findall(figure(6),'-property','FontSize'),'FontSize',14)

figure(7)
plot(oy, p7,'b','LineWidth',1.6)
xlabel('Angular tail rotor velocity, \Omega_y(t) [rad/s]')
ylabel('\psi_7(t)')%
set(findall(figure(7),'-property','FontSize'),'FontSize',16)

figure(8)
plot(op, p8,'b','LineWidth',1.6)
xlabel('Angular main rotor velocity, \omega_p(t) [rad/s]')
ylabel('\psi_8(t)')
set(findall(figure(8),'-property','FontSize'),'FontSize',16)

figure(9)
plot(oy, p9,'b','LineWidth',1.6)
xlabel('Angular tail rotor velocity, \Omega_y(t) [rad/s]')
ylabel('\psi_9(t)')
set(findall(figure(9),'-property','FontSize'),'FontSize',16)

figure(10)
surf(Omega_y, theta_p, p10,'FaceColor','b')
xlabel('Angular yaw velocity, \newline \Omega_y(t) [rad/s]')
ylabel('Pitch angle, \theta_p(t) [rad]')
zlabel('\psi_{10}(t)')
set(findall(figure(10),'-property','FontSize'),'FontSize',14)
```

A.3 SVD-based Box Method

```matlab
clear
format short
idx = [4 8 5 1 4; 1 5 6 2 1; 2 6 7 3 2; 3 7 8 4 3; 5 8 7 6 5; 1 4 3 2 ...
    1]';format shortE
close all
%Load Aero constants from file:
Aero_parameters

% State variable limits
op_min = -280;
op_max = 320;
oy_min = -280;
oy_max = 320;
theta_p_max = 0.9695;
theta_p_min = -1.065;
```

```
Omega_y_min = -8.8;
Omega_y_max = 8.68;


op = linspace(op_min, op_max,100);
oy = linspace(oy_min, oy_max,100);
numPoints = 50;
Omega_y_v = linspace(Omega_y_min, Omega_y_max , numPoints);
Omega_y_v2 = [Omega_y_v(1,1:25), 0, Omega_y_v(1,26:end) ];
theta_p_v = linspace(theta_p_min, theta_p_max, numPoints);
theta_p_v2 = [theta_p_v(1,1:26), 0, theta_p_v(1,27:end) ];
[Omega_y, theta_p] = ndgrid(Omega_y_v2, theta_p_v2);

% Calculating the functions that make up the varying parameters
p1 = -((K_tau*K_E)/(R_a*J_r)) - (sign(op)*k_a1.*op + k_a2)/J_r;
p2 = zeros(1,length(op));
p8 = zeros(1,length(op));
for i = 1:length(op)
    if op(i) ≥ 0
        p2(i) = d_t*(K_Mpp1*op(i) + K_Mpp2)/J_p;
        p8(i) = -(d_t/K_Jy)*(K_Myp2.*op(i) + K_Myp1);
    else
        p2(i) = d_t*(K_Mpn1*op(i) + K_Mpn2)/J_p;
        p8(i) = -(d_t/K_Jy)*(K_Myn2*op(i) + K_Myn1);
    end
end

p7 = -((K_tau*K_E)/(R_a*J_r)) - (sign(oy)*k_a1.*oy + k_a2)/J_r;
p5 = zeros(1,length(oy));
p9 = zeros(1,length(oy));
for i = 1:length(oy)
    if oy(i) ≥ 0
        p5(i) = d_t*(K_Tpp1.*oy(i) + K_Tpp2);
        p9(i) = (d_t/K_Jy)*(K_Typ2.*oy(i) + K_Typ1);
    else
        p5(i) = d_t*(K_Tpn1.*oy(i) + K_Tpn2);
        p9(i) = (d_t/K_Jy)*(K_Tyn2.*oy(i) + K_Tyn1);
    end
end

p6 = -Omega_y.*d_t^2*(m_A + m_B).*sin(2.*theta_p);
p10 = zeros(length(Omega_y));
p4 = zeros(length(theta_p));
n = size(Omega_y);
for i = 1:n(1)^2
    if Omega_y(i)> 0
        p10(i) = (-K_Dyp1.*Omega_y(i) - K_Dyp2 - ...
            K_Fyp)./(K_Jy.*cos(theta_p(i)));
    elseif Omega_y(i)< 0
        p10(i) = (-K_Dyn1.*Omega_y(i) - K_Dyn2 - ...
            K_Fyn)./(K_Jy.*cos(theta_p(i)));
    elseif Omega_y(i) == 0
        p10(i) = (-K_Dyn2 - K_Fyn)./(K_Jy.*cos(theta_p(i)));
    end
    if theta_p(i)== 0
```

```
        p4(i) = -m_b*g*d_m; % since lim x-> 0 sin(x)/x = 1
    else
        p4(i) = -(m_b*g*d_m.*sin(theta_p(i)))./theta_p(i);
    end
end

% Building standard bounding boxes
w128_min = [min(p1) min(p2) min(p8)];
w128_max = [max(p1) max(p2) max(p8)];
w128 = vertices(w128_min,w128_max);
w579_min = [min(p5) min(p7) min(p9)];
w579_max = [max(p5) max(p7) max(p9)];
w579 = vertices(w579_min,w579_max);
w6104_min = [min(p6,[],'all') min(p10,[],'all') min(p4,[],'all')];
w6104_max = [max(p6,[],'all') max(p10,[],'all') max(p4,[],'all')];
w6104 = vertices(w6104_min,w6104_max);

% Removing mean value
p1t = p1 - mean(p1);
p2t = p2 - mean(p2);
p8t = p8 - mean(p8);
p5t = p5 - mean(p5);
p7t = p7 - mean(p7);
p9t = p9 - mean(p9);
p6t = p6 - mean(p6,'all');
p10t = p10 - mean(p10,'all');
p4t = p4 - mean(p4,'all');

% Rewriting from array to vector
p6t = p6t(:);
p10t = p10t(:);
p4t = p4t(:);

% Placing the varying parameters as columns in a matrix P
P128 = [p1t' p2t' p8t'];
P579 = [p5t' p7t' p9t'];
P6104 = [p6t p10t p4t];

% Apply singular value decomposition
[U128,S128,V128] = svd(P128);
[U579,S579,V579] = svd(P579);
[U6104,S6104,V6104] = svd(P6104);

% Rotating the P Matrices
Pn128 = (V128'*P128')';
Pn579 = (V579'*P579')';
Pn6104 = (V6104'*P6104')';

% Creating x, y, z vectors
p1n = (Pn128(:,1));
p2n = (Pn128(:,2));
p8n = (Pn128(:,3));
p5n = (Pn579(:,1));
p7n = (Pn579(:,2));
p9n = (Pn579(:,3));
```

```
p6n = (Pn6104(:,1));
p10n = (Pn6104(:,2));
p4n = (Pn6104(:,3));

% Building a new box in the rotated coordinate frame
wn128_min = [min(p1n) min(p2n) min(p8n)];
wn128_max = [max(p1n) max(p2n) max(p8n)];
wn128 = vertices(wn128_min,wn128_max);
wn579_min = [min(p5n) min(p7n) min(p9n)];
wn579_max = [max(p5n) max(p7n) max(p9n)];
wn579 = vertices(wn579_min,wn579_max);
wn6104_min = [min(p6n) min(p10n) min(p4n)];
wn6104_max = [max(p6n) max(p10n) max(p4n)];
wn6104 = vertices(wn6104_min,wn6104_max);

% Rotatating back the vertecies to the original coordinate system
Wr128 = (V128*wn128')';
Wr579 = (V579*wn579')';
Wr6104 = (V6104*wn6104')';

% Adding back the mean
Wo128 = Wr128 + [mean(p1), mean(p2), mean(p8)];
Wo579 = Wr579 + [mean(p5), mean(p7), mean(p9)];
Wo6104 = Wr6104 + [mean(p6,'all'), mean(p10,'all'), mean(p4,'all')];

V = VertexVectors(Wo128,Wo579,Wo6104);

q = size(V);
n = q(2);
N = length(V);
VertexVectors = V;


A = cell(1,n);
% a is used to create the different matrices that will be stored in A
a = (zeros(5,6));

% Loops through all the vertex vectors and build A(psi)
for k = 1:N
     a(1,1) = V(k,1);
     a(2,1) = V(k,2);
     a(2,2) = -K_DP2;
     a(2,3) = V(k,3);
     a(2,4) = V(k,3);
     a(2,5) = V(k,5);
     a(3,2) = 1;
     a(4,4) = V(k,6);
     a(5,1) = V(k,7);
     a(5,4) = V(k,8);
     a(5,5) = V(k,9);
     a(6,5) = 1;
     A{1,k} = a;
end

% Building B
```

```
b11 = K_tau/(R_a*J_r);
b24 = b11;
B = [b11 0; 0 0; 0 0; 0 b24; 0 0; 0 0];
nLMI = N;

robust controller
Q = eye(6);
R = eye(2);
Y = sdpvar(6,6);
L = sdpvar(2,6,'full');
F = [Y ≥ 1e-09];
for x = 1:nLMI
     F = [F, [-A{x}*Y-B*L + (-A{x}*Y-B*L)' Y L';Y inv(Q) zeros(6,2);L ...
         zeros(2,6) inv(R)] ≥1e-09];
end
optimize(F,-trace(Y),sdpsettings('solver','sdpt3','sdpt3.maxit',500,'verbose',0,'debug',1
%optimize(F,-trace(Y), ...
    sdpsettings('solver','sedumi','verbose',0,'debug',1))
Y_value = value(Y);
L_value = value(L);
K = value(L)*inv(value(Y));
E = zeros(nLMI,1);
for e =1:nLMI
    E(e,1) = min(eig([-A{e}*Y_value-B*L_value + ...
        (-A{e}*Y_value-B*L_value)' Y_value L_value';Y_value inv(Q) ...
        zeros(6,2);L_value zeros(2,6) inv(R)]));
end

%Checking eigenvalues
 if min(E) ≥ 0 && min(eig(Y_value)) ≥ 0
        fprintf('\nok\n');
    else
        fprintf('\nnot ok\n');
 end
 fprintf('The value of trace(Y)for the robust control = %f.\n\n', ...
     value(trace(Y)));

%% Calculating Volume

% Psi 128
% Original box:
xcoor128 = w128(:,1);
ycoor128 = w128(:,2);
zcoor128 = w128(:,3);
[K128,Vol128] = boundary(xcoor128,ycoor128, zcoor128);
%SVD box
xcoor128n = Wo128(:,1);
ycoor128n = Wo128(:,2);
zcoor128n = Wo128(:,3);
[Kn128,Voln128] = boundary(xcoor128n,ycoor128n, zcoor128n);

diff128 = Vol128- Voln128;
percentage128 = (diff128/Voln128)*100;

% Psi 579
% Original box:
xcoor579 = w579(:,1);
ycoor579 = w579(:,2);
zcoor579 = w579(:,3);
[K579,Vol579] = boundary(xcoor579,ycoor579, zcoor579);
%SVD box
xcoor579n = Wo579(:,1);
ycoor579n = Wo579(:,2);
zcoor579n = Wo579(:,3);
[Kn579,Voln579] = boundary(xcoor579n,ycoor579n, zcoor579n);

diff579 = Vol579- Voln579;
percentage579 = (diff579/Voln579)*100;

% Psi 6104
% Original box:
xcoor6104 = w6104(:,1);
ycoor6104 = w6104(:,2);
zcoor6104 = w6104(:,3);
[K6104,Vol6104] = boundary(xcoor6104,ycoor6104, zcoor6104);
%SVD box
xcoor6104n = Wo6104(:,1);
ycoor6104n = Wo6104(:,2);
zcoor6104n = Wo6104(:,3);
[Kn6104,Voln6104] = boundary(xcoor6104n,ycoor6104n, zcoor6104n);

diff6104 = Voln6104- Vol6104;
percentage6104 = (diff6104/Voln6104)*100;



%% Plots
figure(1)
scatter3(p1, p2, p8, 30, 'k')
patch('vertices', Wo128, 'faces', idx', 'facecolor', 'r', ...
    'facealpha', 0.05,  'EdgeColor','r');
%patch('vertices', w128, 'faces', idx', 'facecolor', 'b', ...
    'facealpha', 0.05,'EdgeColor','b');
set(findall(figure(1),'-property','FontSize'),'FontSize',16)
xlabel('\psi_1','FontSize',20)
ylabel('\psi_2','FontSize',20)
zlabel('\psi_8','FontSize',20)


figure(2)
scatter3(p5, p7, p9, 30, 'k', 'filled')
patch('vertices', Wo579, 'faces', idx', 'facecolor', 'r', ...
    'facealpha', 0.05,  'EdgeColor','r');
patch('vertices', w579, 'faces', idx', 'facecolor', 'b', 'facealpha', ...
    0.05,'EdgeColor','b');
set(findall(figure(2),'-property','FontSize'),'FontSize',16)
xlabel('\psi_5','FontSize',20)
ylabel('\psi_7','FontSize',20)
zlabel('\psi_9','FontSize',20)
zlim([-0.03 0.11])
```

```


figure(3)
scatter3(p6(:), p10(:), p4(:),20, 'k', 'filled')
patch('vertices', Wo6104, 'faces', idx', 'facecolor', 'r', ...
    'facealpha', 0.05,  'EdgeColor','r');
patch('vertices', w6104, 'faces', idx', 'facecolor', 'b', ...
    'facealpha', 0.05,'EdgeColor','b');
set(findall(figure(3),'-property','FontSize'),'FontSize',16)
xlabel('\psi_6','FontSize',20)
ylabel('\psi_{10}','FontSize',20)
zlabel('\psi_4','FontSize',20)
```

A.4 Aero Parameters

```
J_p = 0.0183;
J_r = 5.6e-05 + 5.04e-08 + 2.0e-06;
d_t = 0.158;
g = 9.81;
d_m = 0.0029;
m_b = 1.15;
R_a = 8.4;
m_A = 0.87;
m_B = 0.87;
K_tau = 0.042;
K_E = 0.042;
k_a1 = 2.9e-07;
k_a2 = 4.2e-06;
K_Mpp1 =  1.51e-06;
K_Mpp2 =  8.61e-07;
K_Mpn1 = -2.01e-06;
K_Mpn2 = -4.18e-05;
K_DP1 = 7.6e-06;
K_DP2 = 6.1e-03;
K_Tpp1 =  1.49e-06;
K_Tpp2 = -2.52e-05;
K_Tpn1 = -1.46e-07;
K_Tpn2 =  7.47e-06;
K_Myp1 =  7.30e-07;
K_Myp2 = -1.61e-05;
K_Myn1 = -6.11e-07;
K_Myn2 =  3.23e-05;
K_Typ1 =  1.06e-06;
K_Typ2 =  1.17e-05;
K_Tyn1 = -1.11e-06;
K_Tyn2 = -3.76e-05;
K_Dyp1 = 1.84e-05;
K_Dyp2 = 3.64e-04;
K_Dyn1 = -5.05e-05;
K_Dyn2 = 9.86e-04;
K_Jy = 0.017;
K_Fyp = 0.00616;
```

```matlab
K_Fyn = -0.00411;

% State boundaries:
omega_max = 215;
omega_min = -215;
OMEGA_P_max = 0.81;
OMEGA_P_min = -0.72;
theta_p_max = 0.9695;
theta_p_min = -1.065;
OMEGA_Y_max = 4.6;
OMEGA_Y_min = -4.9;
```

A.5 Functions

```matlab
function v = vertices(min, max)
    v = [0 0 0; 1 0 0; 1 1 0; 0 1 0; 0 0 1; 1 0 1; 1 1 1; 0 1 1];
    for i = 1:2^3
        for j = 1:length(min)
            if v(i,j) == 1
                v(i,j)= max(j);
            else
                v(i,j)= min(j);
            end
        end
    end
end
```

```matlab
function V = VertexVectors(V1, V2, V3)
    w = zeros(8,6);
    A = cell(1,8);
    B = cell(1,8);
    C = cell(1,8);

    % Building 8 8x6 matrices containg combinations of psi128 and psi579
    for i = 1:8
        for j = 1:8
           w(j,:) = [V2(i,:), V3(j,:)];
        end
        A{1,i} = w;
    end

    % Building 64 8x9 matrices containg combinations of psi128, ...
       psi579 and psi6104
    for i = 1:8
          b1 = repmat(V1(i,:),8,1);
          for j = 1:8
              z = zeros(8,9);
              z(:,1:3)= b1;
              z(:,4:9)= A{j};
              B{1,j} = z;
```

```
            end
        C{i} = B;
    end

    % stacking the 64 matrices to create one 512x9 matrix containg ...
        all possible
    % permutations of vertecies
    V = [];
    for i = 1:8
        for j = 1:8
            w = C{i}{j};
            V = vertcat(V,w);
        end
    end
end
```

Appendix B

Simulink schemes

B.1 Nonlinear Aero model

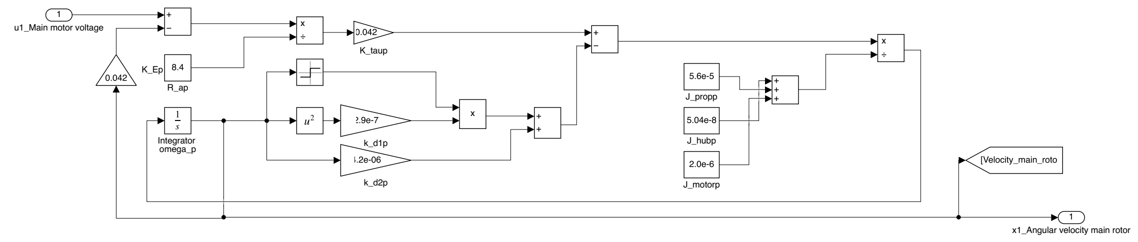


Figure B.1: Simulink scheme representing the equation of motion for the main rotor dynamics described in Eq. (2.13)

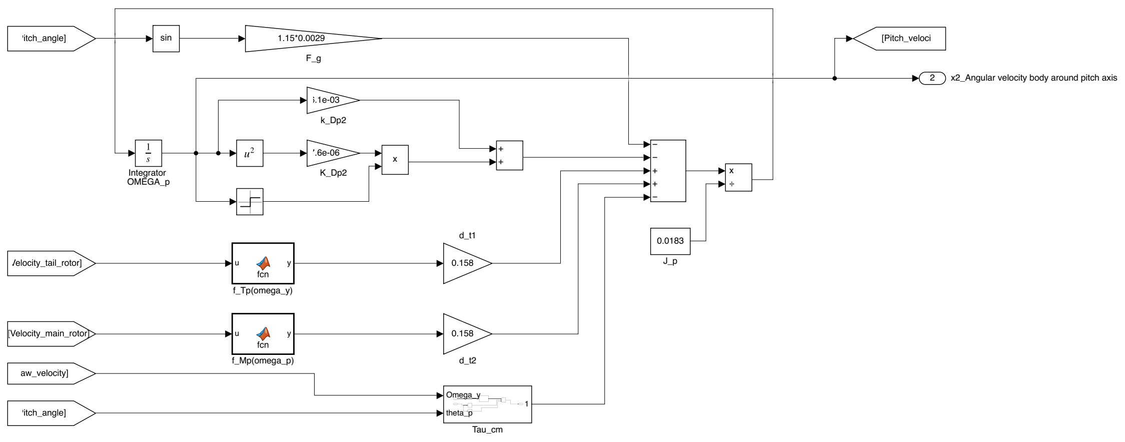


Figure B.2: Simulink scheme representing the equation of motion for the pitch velocity described in Eq. (2.30)

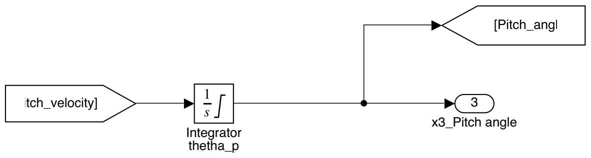


Figure B.3: Simulink scheme which integrates $\Omega_p(t)$ in order to get the $\theta_p(t)$

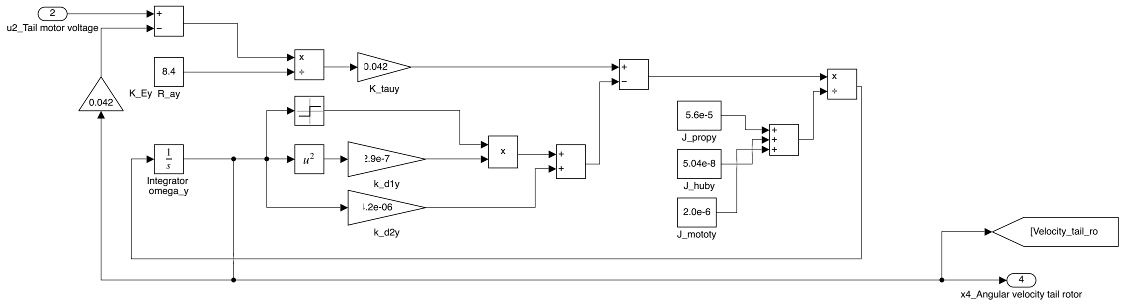


Figure B.4: Simulink scheme representing the equation of motion for the tail rotor dynamics described in Eq. (2.13)

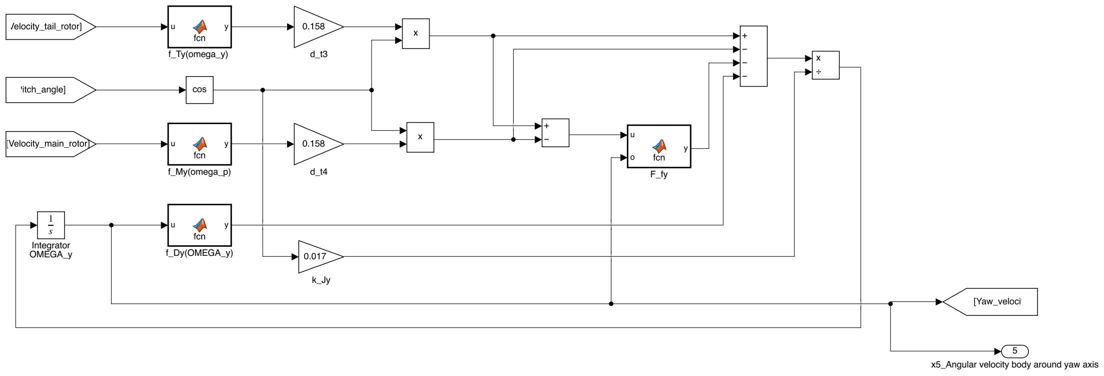


Figure B.5: Simulink scheme representing the equation of motion for the yaw velocity described in Eq. (2.38)

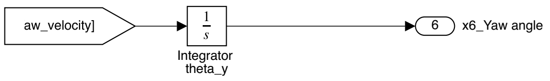


Figure B.6: Simulink scheme which integrates $\Omega_y(t)$ in order to get the $\theta_y(t)$

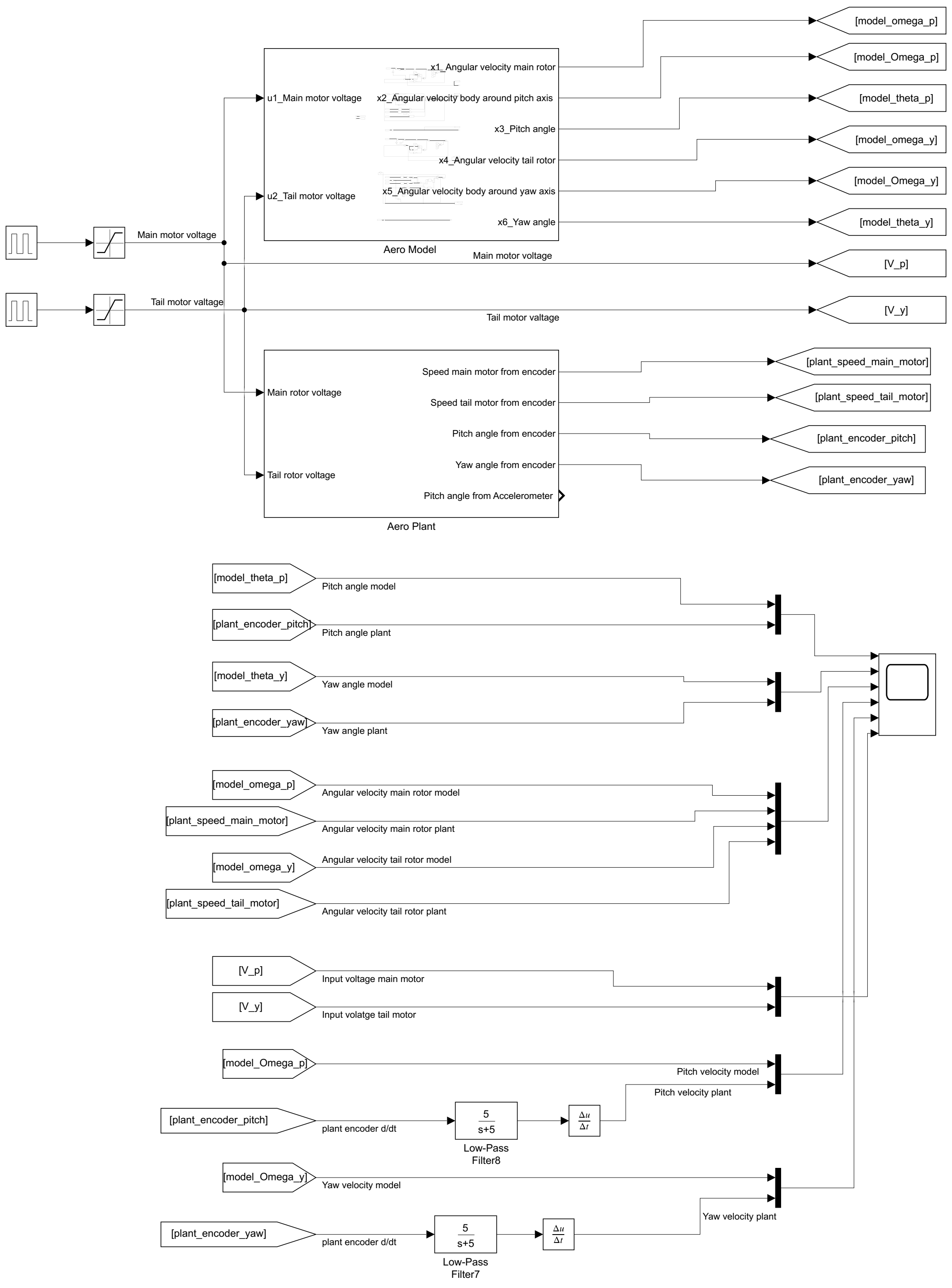


Figure B.7: Simulink scheme where the nonlinear Aero model can be compared to the actual plant

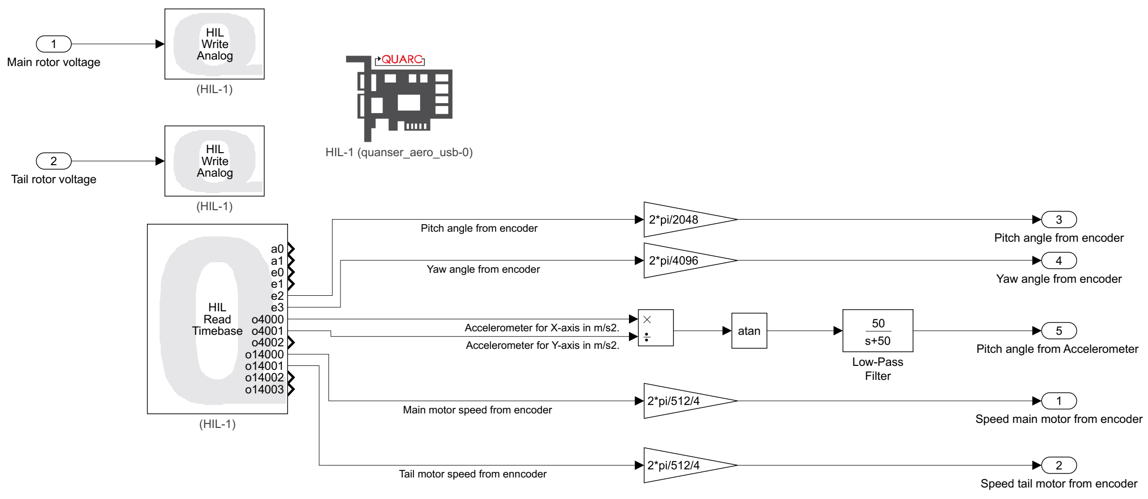
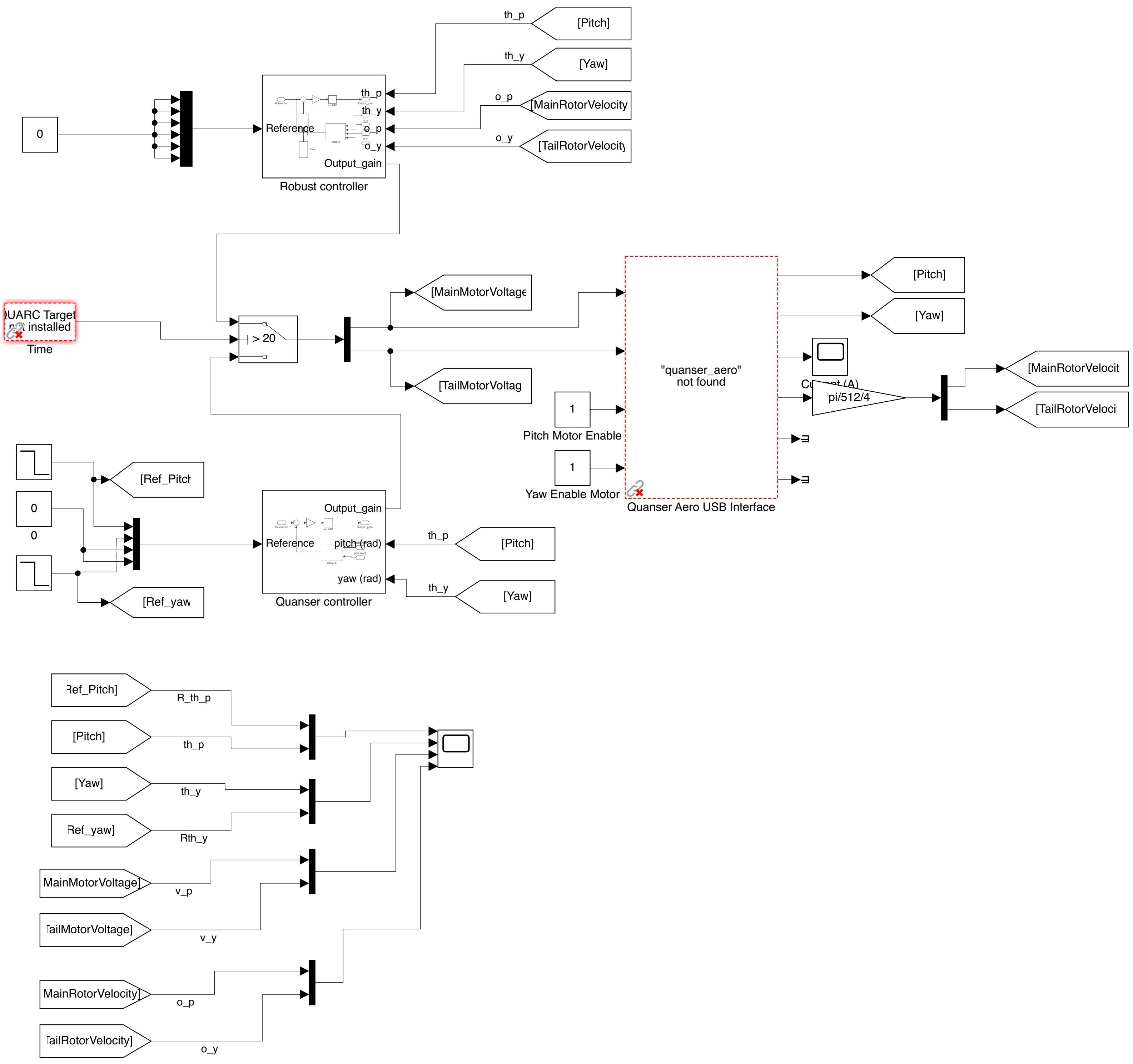
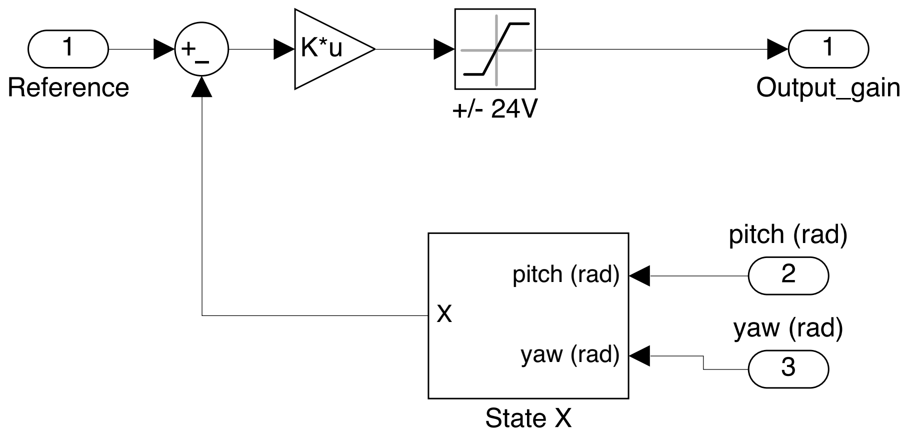
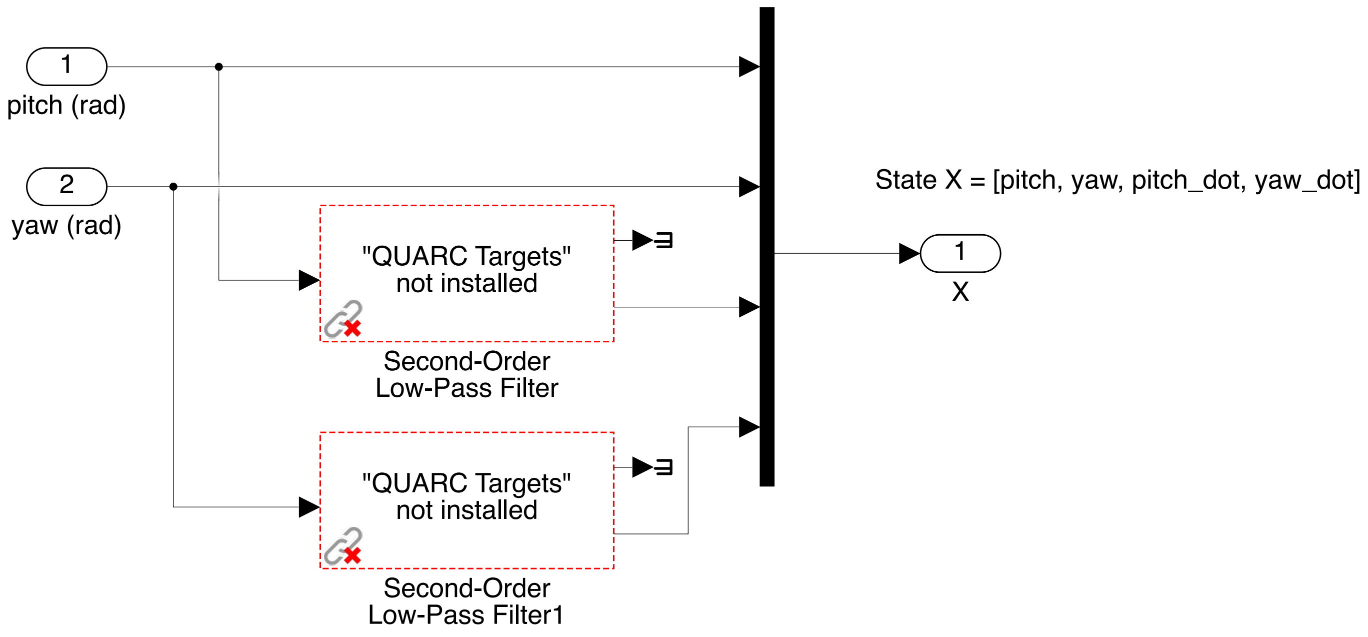


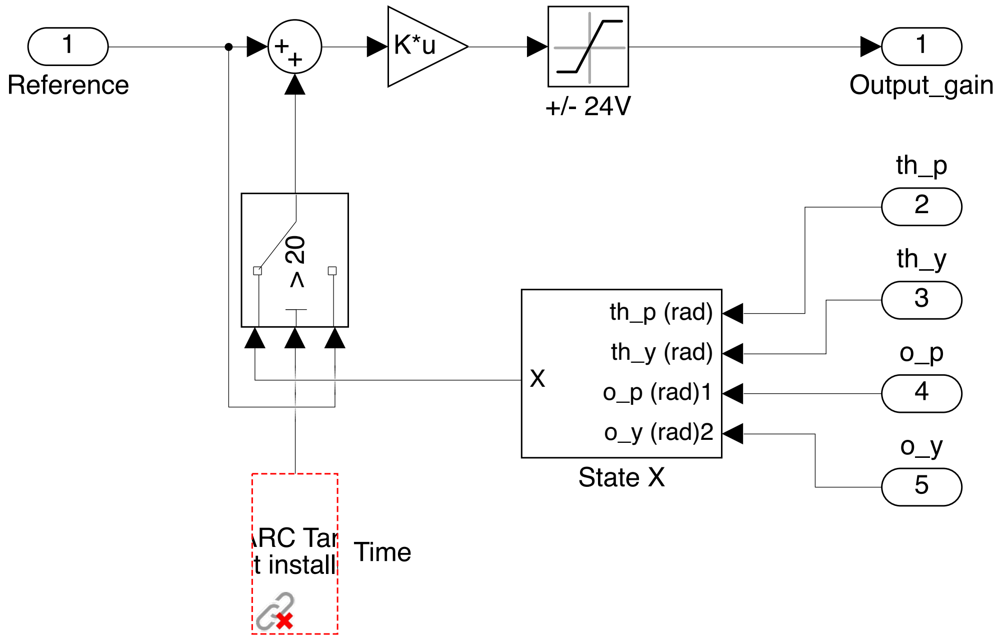
Figure B.8: Simulink scheme including Quanser I/O-blocks

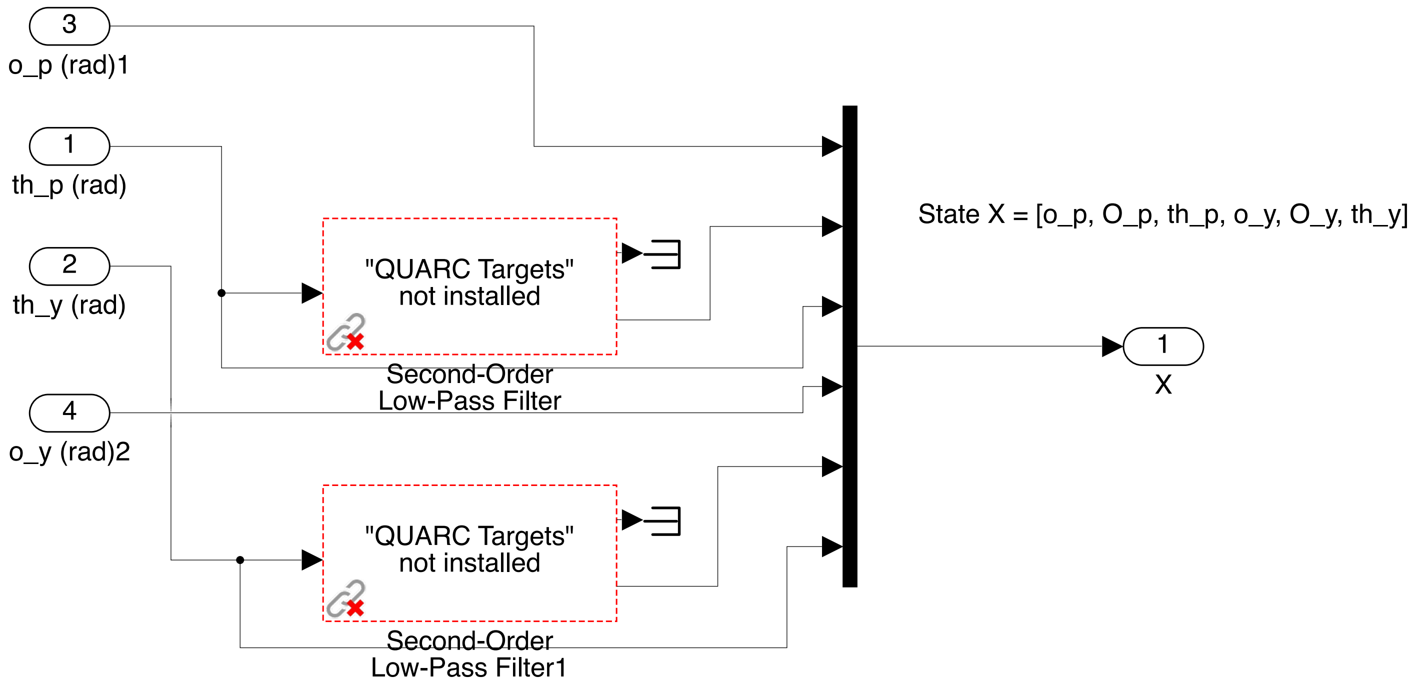
B.2 State feedback controller


th_p
[Pitch]
th_y
[Yaw]
o_p
[MainRotorVelocity
o_y
[TailRotorVelocity
0
Reference
th_p
th_y
o_p
o_y
Output_gain
Robust controller
[MainMotorVoltage
QUARC Target not installed
Time
> 20
[TailMotorVoltag
"quanser_aero" not found
Quanser Aero USB Interface
1
Pitch Motor Enable
1
Yaw Enable Motor
Current (A)
pi/512/4
[MainRotorVelocit
[TailRotorVeloci
[Ref_Pitch
0
0
[Ref_yaw
Output_gain
Reference
pitch (rad)
yaw (rad)
Quanser controller
th_p
[Pitch]
th_y
[Yaw]
Ref_Pitch]
R_th_p
[Pitch]
th_p
[Yaw]
th_y
Ref_yaw]
Rth_y
MainMotorVoltage]
v_p
TailMotorVoltage]
v_y
MainRotorVelocity]
o_p
TailRotorVelocity]
o_y


1
Reference
+_
K*u
+/- 24V
1
Output_gain
pitch (rad)
2
X
yaw (rad)
3
State X


1
pitch (rad)
2
yaw (rad)
"QUARC Targets" not installed
Second-Order Low-Pass Filter
"QUARC Targets" not installed
Second-Order Low-Pass Filter1
State X = [pitch, yaw, pitch_dot, yaw_dot]
1
X


1
Reference
K*u
+/- 24V
1
Output_gain
> 20
th_p
2
th_y
3
o_p
4
o_y
5
th_p (rad)
th_y (rad)
o_p (rad)1
o_y (rad)2
X
State X
RC Tar
t install
Time


3
o_p (rad)1
1
th_p (rad)
2
th_y (rad)
4
o_y (rad)2
"QUARC Targets" not installed
Second-Order Low-Pass Filter
"QUARC Targets" not installed
Second-Order Low-Pass Filter1
State X = [o_p, O_p, th_p, o_y, O_y, th_y]
1
X