TÉCNICO LISBOA

Robust Thrust Vector Control of Launch Vehicles

Simão Lavarinhas Amaro

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors: Prof. Rita Maria Mendes de Almeida Correia da Cunha
Dr. Francisco Câmara

Examination Committee

Chairperson: Prof. Paulo Jorge Coelho Ramalho Oliveira
Supervisor: Prof. Rita Maria Mendes de Almeida Correia da Cunha
Member of the Committee: Prof. Pedro Daniel Graça Casau

November 2022

To my dear girlfriend, Margarida, who I met at the very beginning of my university trajectory and who constantly motivated me to be a better person; to her family that took me in like I was one of their own; to my family that, although lives far away from Lisbon, was always there for me.

Acknowledgments

I would like to firstly thank Spin.Works, namely Francisco Câmara and David Esteves, for the opportunity to work in the VIRIATO project and for providing me with all the necessary tools and insightful advice.

Additionally, I would like to express my gratitude for the guidance given by Professor Rita Cunha, who I had met in another course and promptly accepted my request to be my supervisor.

Likewise, I thank Técnico Solar Boat (TSB) for letting me pursue my passion in control theory and for all the amazing team members.

Resumo

Viriato é um micro lançador de propulsão vetorial, cuja saída da atmosfera passa por várias condições de voo, desde regime subsónico a supersónico, complicando o desenho de um controlador clássico com o desempenho desejado em toda a trajetória. Deste modo, um controlador Robusto é desenvolvido que garante uma resposta rápida e robusta a variações de massa, atrasos na atuação e rajadas de vento. Como termo de comparação, desenvolveram-se controladores PD, LQG e NMPC.

Com o objetivo de construir e testar os algoritmos, a Spin.Works desenvolveu o VIRIATO Test Platform, um protótipo semelhante a VIRIATO, com cerca de 1 metro de altura. De seguida, um ambiente de simulação em *MATLAB/Simulink* é desenvolvido, utilizando as principais equações cinemáticas e dinâmicas, os sensores e as suas especificações de ruído. Depois, o sistema é linearizado e as funções de transferência são utilizadas para construir o controlador PD e as matrizes em espaço de estados os controladores LQG e Robusto. Adicionalmente, por via das equações não lineares, formula-se um controlador NMPC.

Os resultados das simulações indicam que o controlador H_∞ é o mais rápido (juntamente com o NMPCEKF) e é robusto a variações de massa de 20%, atrasos na atuação de 0.1 s e rajadas de vento de 5 m/s. Verificou-se que consegue seguir trajetórias quadradas, em anel e em formato de oito, enquanto aponta para o centro, com erro de posição inferior a 0.4 m e um atraso de 1 s, sendo que o erro aumenta quando as referências de guinada variam abruptamente.

Palavras-chave: Controlo Por Empuxo Vetorial, Controlo Robusto, Filtro de Kalman, Robustez

Abstract

VIRIATO is a thrust vectored launch vehicle, whose flight mode while leaving the atmosphere ranges from subsonic to supersonic, such that it would be challenging to ensure a suitable performance throughout all its trajectory using classical controllers. To overcome this difficulty, a Robust controller is developed that ensures a fast response, robustness to mass variations, actuation delays and wind gusts. To benchmark the H_∞ controller, PD, LQG and NMPC controllers are also derived.

In order to build and test these algorithms, Spin.Works developed the VIRIATO Test Platform, a small scaled prototype of VIRIATO with 1 m height. A simulation environment in *MATLAB/Simulink* is built, comprising the most significant kinematic and dynamic equations, the available sensors and their noise specification. Then the system is linearized, the transfer functions are used to construct the PD controller and the state space matrices the LQG and Robust controllers. Additionally, using the nonlinear equations, the NMPCEKF controller is formulated.

The simulation results comparing the LQG, NMPCEKF and H_∞ controllers indicate that the latter is the fastest alongside the NMPCEKF and is robust to a mass variation of 20%, actuation delays of 0.1 s and wind gusts of 5 m/s. The H_∞ was able to follow square, loop and eight shaped trajectories, while pointing at the center of the loops, with a position error below 0.5 m and a delay of 1 s. The position error increased slightly when the yaw reference changed abruptly.

Keywords: Thrust Vector Control, Robust Control, Kalman Filter, H_∞

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

δ	Angle of the outer gimbal.
ϵ	Angle of the inner gimbal.
λ	Euler angles.
$\mathbf{\Omega}$	Angular velocity vector of the centre of mass with respect to the body frame, expressed in the body frame.
τ	Torque produced by the jet engine.
ω	Frequency in rad/s
ϕ	Euler Roll angle, defining a rotation around the x axis of the body frame.
ψ	Euler Yaw angle, defining a rotation around the z axis of the body frame.
θ	Euler Pitch angle, defining a rotation around the y axis of the body frame.

Roman symbols

τ	Input torque
$\mathbf{F}$	Input forces
$\mathbf{p}$	Position of the centre of mass.
$\mathbf{v}$	Velocity of the centre of mass
C	Centre
e	Unit vector
g	Gravitational acceleration vector expressed in the inertial frame
J	Inertia tensor of the centre of mass with respect to the body frame, expressed in the body frame
l	Distance between the centre of thrust and mass
m	Mass of Viriato

$Q(\lambda)$ Transformation that transforms angular velocities $\mathbf{\Omega}$ to Euler rates $\dot{\lambda}$

R Rotation matrix

T Thrust produced by the jet engine

t Time constant of a first order system

w Angular velocity

Subscripts

s, r, m Servos of the gimbals, rotors, motor, respectively

u Inputs

x, y, z Cartesian components

Superscripts

B Body frame

I Inertial frame

T Transpose

Glossary

CG	Centre of Mass.
DM	Disk Margin.
GM	Gain Margin.
GNC	Guidance Navigation and Control.
LQG	Linear Quadratic Gaussian.
LQR	Linear Quadratic Regulator.
MPC	Model Predictive Control.
NMPCEKF	Nonlinear Model Predictive Control with an Extended Kalman Filter.
NMPC	Nonlinear Model Predictive Control.
PD	Proportional-Derivative Control.
PM	Phase Margin.
PSD	Power Spectral Density.
RMS	Root Mean Square value.
GPS	Global Positioning System.

Chapter 1

Introduction

In this chapter the motivation, a brief overview and an outline of the thesis are described.

1.1 Motivation

SpaceX has sparked the interest of private companies in space exploration and launchers, having done, as of this moment, 187 launches, 149 landings and 124 total reflights [1]. The aforementioned success caused a considerable number of startups in this field of to appear over the years and more precisely, since 2017, this trend has seen an even bigger increase [2].

VIRIATO is a project that aims to develop, integrate and operate a sub-orbital vehicle to validate and test key technologies to the future development of a Portuguese microsatellite launcher, being launched preferably in the planned space port in Santa Maria, Açores [3].

In the scope of VIRIATO, Spin.Works is responsible for the development of the GNC subsystem [4]. To implement and test the algorithms, a smaller scaled VIRIATO, the VIRIATO Test Platform, with an height of 1 m was built. This thesis focus on the design and computational simulations of the GNC system in *MATLAB/Simulink*, namely the development of Robust position controllers and guidance algorithms that enable the vehicle to follow square, loop or eight shaped trajectories.

1.2 Topic Overview

In this Thesis, control algorithms ranging from PD controllers with more than one hundred years to more recent H_2/H_∞ are derived. The assumption is that PD controllers can deal with the problem, but their performance might not be satisfactory enough, which is why more advanced controller structures are also tested. As will be seen, the Robust and nonlinear controllers perform significantly better than the PD controllers; however, the latter are useful due to their well studied behaviour. Having derived all the controllers, the physical tests would be carried out starting with the simpler PD controllers, tuning them properly and only then advancing to the more complex H_∞ or NMPCEKF controllers.

A very similar project to the VIRIATO Test Platform, the EAGLE (Environment for Autonomous GNC

Landing Experiments) project, from the German Aerospace Center has been able to successfully fly its launch vehicle, having used algorithms as simple as PD controllers but also more sophisticated methods, namely sliding-mode-controllers [5]. In this thesis, sliding mode controllers are not developed due to the recurring chattering problems they have, which limit their performance and robustness. As was the case for EAGLE, in the end the team opted for simpler PID controllers, which were able to make it fly. Building on this knowledge, the first attempt to control the VIRIATO Test Platform is made with PD controllers and the sliding mode controllers are not tested at all. Instead, LQR, robust H_2/H_∞ and a nonlinear model predictive controller are designed and tested.

1.3 Objectives

The main objectives in this thesis are to design control laws that achieve certain performance specifications and a guidance algorithm that enables tracking square, loop and eight shaped trajectories. To be more specific, after deriving the dynamics, defining the actuator and sensor specifications, different types of control laws are designed, namely linear, robust and nonlinear. Having obtained the design results, the most promising controller of each mentioned category is tested in a nonlinear simulation in *SIMULINK* against mass variations, actuation delays and wind gusts. Upon selection of the best performing controller in the nonlinear simulations, a guidance algorithm is added that feeds position references to the inner control loop.

This thesis will focus only on simulations, leaving work to be done in the implementation of the algorithms in a microcontroller inside the VIRIATO Test Platform. Furthermore, the developed *MATLAB* code split into the definition of the dynamics of the VIRIATO Test Platform and the different control and guidance algorithms would allow to tune and build upon the developed controllers, ensuring the scalability of this work.

1.4 Literature review

Several small scale launch vehicles project, similar to the VIRIATO Test Platform, have been developed and some even tested, such as EAGLE. The flight controllers developed in this thesis were motivated to some extent by the aforementioned project. To decide what approaches to take, 3 publications were studied [6], [7] and [8] in years 2017, 2018 and 2019, respectively, which allowed to have a better understanding of the advantages and disadvantages of each control or estimation technique and also their limitations.

In the first year in [6], to estimate the position, velocity, Euler angles and angular velocity, a modular approach was taken, having 2 separate filters working together, in which a faster, Strapdown filter, ran at 100 Hz, performed the integration from accelerometers and gyroscopes, obtaining a state prediction and a slower Kalman filter, ran at 10 Hz, did the error correction, using measurements from GPS, altimeter and magnetometer. The control task was split into 4 categories, namely system identification, yaw control (the rotation around the vertical axis), thrust vector control and position control. The thrust

vector control is responsible for the attitude and altitude, receiving references from the outer loop position control. External to these control loops, a guidance system is responsible for generating position and velocity references. With this architecture, hand tuned PD controllers were used in the first place, but were not further discussed and replaced by sliding mode controllers. Additionally, a model-reference adaptive scheme, which feeds a nominal plant with the same inputs as the real plant, during flight, computes the difference in inputs required to compensate for the verified output variations. Both the inner and outer loops use sliding mode controllers (designed independently due to the difference in order of magnitude of the closed loop bandwidth of each loop). The altitude controller produced a response with a settling time of around 5-10 seconds and there is a constant 20 cm of error, which is expected given that the sliding mode controllers had their integral component disabled to prevent the excessive accumulation of error (windup). The position controllers have a settling time of about 10 seconds and also show some static error. The yaw controller works quite well as expected, having a fast response and very little error.

In the next year in [7], the same setup was used, with the addition of a saturation function replacing the sign function, which is a very common technique to avoid chattering in sliding mode control. Additionally, the model-reference adaptive scheme was removed. This time, it appears that the altitude simulations were the same as in [6], having changed only the position controllers, which is understandable because these had the worst results. In this work, the attitude controller was tested independently and exhibited 2° oscillations, which were attributed to inertial matrix uncertainties, wind gusts and sloshing. The results of the position controller were very similar to [6] and the authors considered them to be conservative and that more aggressive gains could be tested.

In the last studied attempt in [8], the sliding mode controllers were abandoned and replaced by traditional PID controllers. The results were again similar to the ones obtained in the previous years. The altitude controller had an offset of about 10%, which increased over time and was attributed to not having real-time mass estimation and using a constant nominal feedforward thrust input, while the mass of the vehicle was decreasing due to fuel consumption. The position controller had a settling time of about 15 s and afterwards, while hovering, a maximum deviation of 0.2 m, under mild winds speeds up to 5 m/s. In this work, a guidance system was designed, using a polynomial scheme to generate a continuous reference for velocity and position. This includes the possibility of limiting the maximum velocity and acceleration to reach the desired position. The results, including the guidance controller, prove that tracking is being done and the controller is able, although with error oscillations of 0.2 m, to track the position references.

From these 3 publications, some key ideas were derived, namely: a 2 phase prediction and correction estimator should be derived, sliding mode controllers do not exhibit better performance than simpler PID controllers and the control architecture of a guidance system, an outer and an inner loop, should be adopted. In fact, to validate the assumption that the sliding mode controllers did not lead to exceptional results, some exploratory simulations for VIRIATO were done using them, but the chattering problem was so persistent that to avoid it, the performance was severely compromised.

1.5 Thesis Outline

This thesis follows an intuitive approach to define the control and guidance algorithms for the VIRIATO Test Platform. In chapter 2 the Newton-Euler equations are written, the inertial and body frames are defined and a thorough description of the 4 inputs of the system is given. In the end, decoupled state space representations for the longitudinal, lateral and vertical movements and transfer functions are derived.

In chapter 3, linear control techniques are applied to control the VIRIATO Test Platform, namely, PD, cascaded, LQR and MPC controllers. Firstly, trade-off requirements for bandwidth, damping and overshoot are specified, which will serve as tuning guidelines. To obtain the controllers, time and frequency analysis are carried out with the help of several *MATLAB* functions. In addition to this, the robustness of the closed loop system with each controller is also studied by using at first classical gain margins. Then, in section 3.3.3 a detailed description of disk-margins is presented, which enables to take a much more comprehensive approach for multi-input-multi-output MIMO systems. Lastly, in tables 3.2 to 3.5 an overview of the design results is given.

In chapter 4 robust control methods are studied. First, a very brief explanation of the generalized plant structure is given, which is the ground for the H_2, H_∞ and μ control problems. Then, the weighting functions that enable tuning of the control design goals of the controllers are presented. Then, the most important conditions, (A1) to (A5), for the correct computation of the robust controllers are defined, the H_2 and H_∞ norms are shown and a brief description of the singular values is given. With the theory settled, making use of engineering heuristics the weighting functions are defined and again using *MATLAB* functions the H_2, H_∞ and μ controllers are computed.

In chapter 5 a nonlinear Model Predictive Controller is derived. The internal dynamical model of the VIRIATO Test Platform is presented, augmented with the input dynamics and a mass state. The control and prediction horizons, the cost function and the constraints are stated which comprise most of the problem definition. Then, an extended Kalman filter is derived, explaining all its steps in detail and also the definitions of the most important variables.

In chapter 6, the control algorithms are tested thoroughly. The most promising controllers from the design in the previous sections, namely the LQG, H_∞ and NMPCEKF controllers are tested extensively in performance, disturbance rejection and robustness.

In chapter 7 a simple guidance algorithm is derived. The goal is to have an outer loop, external to the control algorithms previously derived, which would feed position and yaw references to them, contributing to a smoother flight. The results of the complete control and guidance algorithms are shown for square, loop and one resembling the infinity symbol ∞ trajectories.

Chapter 2

Dynamics

In this chapter the dynamics of the VIRIATO Test Platform, followed by linearization and representation in state space and transfer function are presented. In the end, the noise of the sensors is quantified.

2.1 Vehicle

The complete system is shown in figure 2.1.

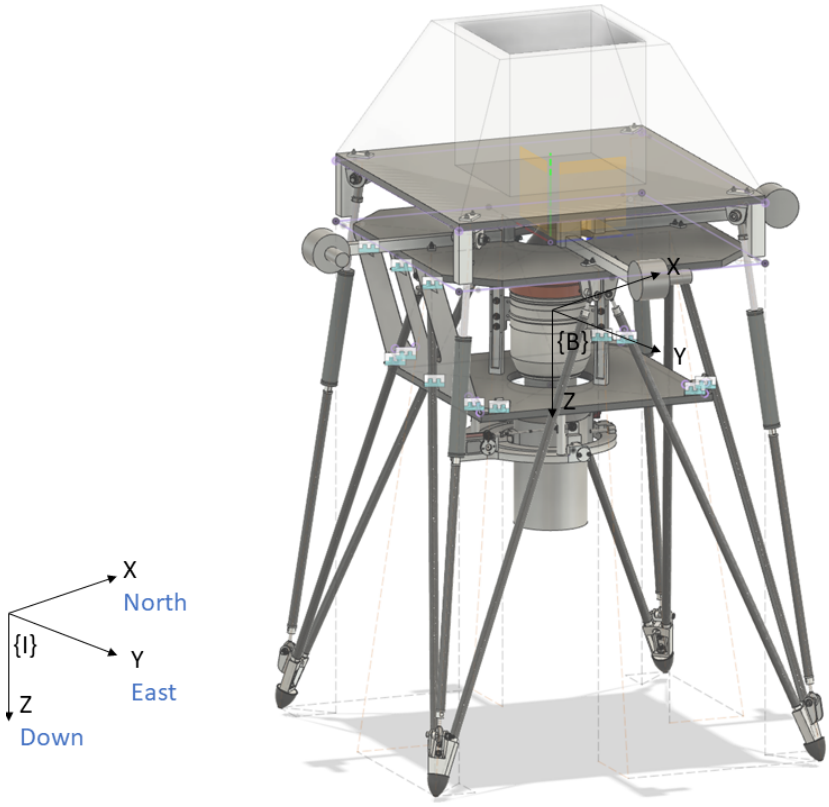


Figure 2.1: VIRIATO Test Platform [4].

The system has 4 actuators, thrust T from the jet engine, the outer and inner gimbals, δ and ϵ respectively and rotors that produce a torque τ_r, enabling to control 4 outputs simultaneously. In total,

as it will be shown later, there are 12 states, which means it is an under-actuated system. To describe the vehicle with Newton-Euler equations, two reference frames are defined: the inertial and body frames, presented in figure 2.1. These allow for a complete description of the most important dynamics, without too much error given that the majority of the mass is fixed when seen from the body frame. Another more precise approach would be to use Denavit-Hartenberg parameters, defining more reference frames to capture all the dynamics (for instance, the rotation of the gimbals).

The transformation from the inertial frame to the body frame is given by the rotation matrix

$$R_I^B = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \cos\phi\sin\psi & \cos\phi\cos\psi + \sin\theta\sin\phi\sin\psi & \cos\theta\sin\phi \\ \sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta & \cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \tag{2.1}$$

2.2 Actuators

Here a thorough description of the actuators is given.

2.2.1 Gimbals and jet engine

The gimbals and thrust outlet are shown in figure 2.2.

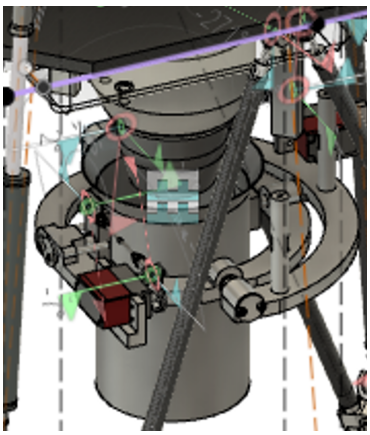


Figure 2.2: Gimbals and thrust outlet system [4].

In figure 2.3, the distance l between the centre of mass CG and the centre of thrust C_t is displayed. The rotation of the outer and inner gimbals δ, ϵ, respectively, can be seen in figure 2.4. The outer gimbal rotates the motor along the roll axis (x) of the body frame, producing a roll moment and a force in the transversal axis (y). The inner gimbal rotates the motor along the pitch axis (y) of the body, producing a pitching moment and a force in the longitudinal axis (x).

The rotation matrix of the outer gimbal to the body frame B is

$$R_\delta^B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\delta & -\sin\delta \\ 0 & \sin\delta & \cos\delta \end{bmatrix} \tag{2.2}$$

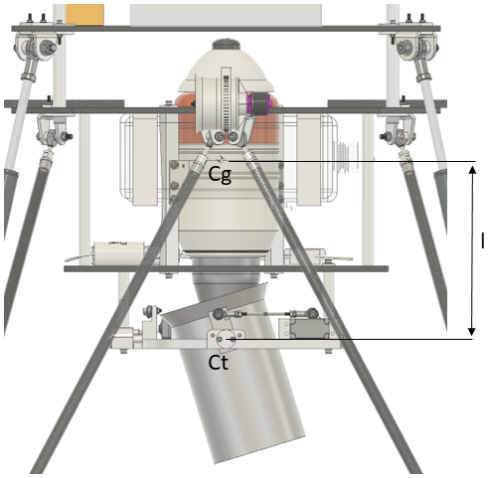
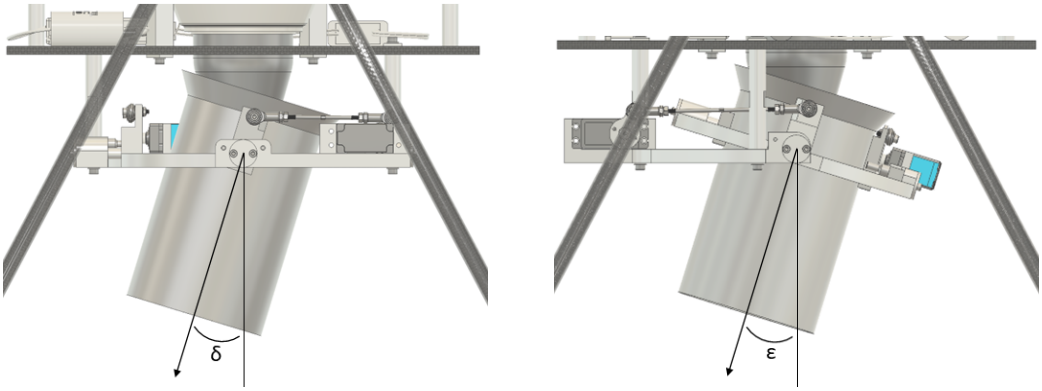


Figure 2.3: Gimbals and thrust outlet lateral view [4].



(a) Outer gimbal δ (b) Inner gimbal ϵ

Figure 2.4: Outer, inner gimbals δ and ϵ [4].

and of the inner gimbal

$$R_{\epsilon}^{B} = \begin{bmatrix} \cos\epsilon & 0 & \sin\epsilon \\ 0 & 1 & 0 \\ -\sin\epsilon & 0 & \cos\epsilon \end{bmatrix}. \tag{2.3}$$

Thus, the force due to the jet engine expressed in the body frame is

$$\mathbf{T}_m = -R_{\delta}^{B} R_{\epsilon}^{B} T \mathbf{e}_z = \begin{bmatrix} -T\sin\epsilon \\ T\cos\epsilon\sin\delta \\ -T\cos\epsilon\cos\delta \end{bmatrix} \tag{2.4}$$

and the resulting motor torque τ_m

$$\tau_m = l\mathbf{e}_z \times \mathbf{T} = \begin{bmatrix} -Tl\cos\epsilon\sin\delta \\ -Tl\sin\epsilon \\ 0 \end{bmatrix}. \tag{2.5}$$

2.2.2 Rotors

One of the 4 rotors around the VIRIATO Test Platform can be seen in figure 2.5.

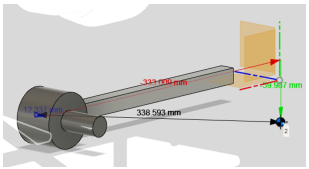


Figure 2.5: One of the 4 rotors of the VIRIATO Test Platform [4].

The total force generated by these 4 rotors when paired is null, resulting only in a torque $\tau_r \mathbf{e}_z$. Therefore, the total force of the inputs expressed in the body frame is $\mathbf{F}_B = \mathbf{T}$ and the torque in the body frame τ^B is

$$\tau^B = \tau_m + \tau_r \mathbf{e}_z = \begin{bmatrix} -Tl\cos\epsilon\sin\delta \\ -Tl\sin\epsilon \\ \tau_r \end{bmatrix} \tag{2.6}$$

Finally, the dynamics of the actuators are approximated by first order systems. This approximation is common for gimbals (controlled by servos) or rotors; however, the engine could benefit from proper system identification. Nonetheless, choosing a first order system is an usual first approach.

$$h(s) = \frac{1}{st_u + 1}. \tag{2.7}$$

2.3 Newton-Euler equations of motion

The equations of motion are derived using the Newton-Euler equations , which combine the translational and rotational dynamics of a rigid body [9].

$$\frac{d}{dt}\begin{bmatrix} \mathbf{p}^B \\ \mathbf{v}^B \\ \lambda \\ \mathbf{\Omega} \end{bmatrix} = \begin{bmatrix} -\mathbf{\Omega}\times\mathbf{p} + \mathbf{v} \\ -\mathbf{\Omega}\times\mathbf{v} + \frac{\mathbf{F}_B}{m} + R_I^B g\mathbf{e}_z \\ Q(\lambda)\mathbf{\Omega} \\ J^{-1}(\tau_B - \mathbf{\Omega}\times J\mathbf{\Omega}) \end{bmatrix} \tag{2.8}$$

with $\boldsymbol{\lambda}$ the Euler angles and $Q(\boldsymbol{\lambda})$ the transformation from angular velocities to Euler rates

$$Q(\lambda) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \tag{2.9}$$

As previously stated, these comprise twelve (12) equations, as following

$$\frac{d}{dt}\begin{bmatrix} p_x^B \\ p_y^B \\ p_z^B \\ v_x^B \\ v_y^B \\ v_z^B \\ \phi \\ \theta \\ \psi \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} v_x^B + p_y^B w_z - p_z^B w_y \\ v_y^B - p_x^B w_z + p_z^B w_x \\ v_z^B + p_x^B w_y - p_y^B w_x \\ v_y^B w_z - v_z^B w_y - g\sin\theta - \frac{T\sin\epsilon}{m} \\ v_z^B w_x - v_x^B w_z + g\cos\theta\sin\phi + \frac{T\cos\epsilon\sin\delta}{m} \\ v_x^B w_y - v_y^B w_x + g\cos\theta\cos\phi - \frac{T\cos\epsilon\cos\delta}{m} \\ w_x + w_z\cos\phi\tan\theta + w_y\tan\theta\sin\phi \\ w_y\cos\phi - w_z\sin\phi \\ w_z\cos\phi\sec\theta + w_y\sin\phi\sec\theta \\ -\frac{J_{zz}w_y w_z - J_{yy}w_y w_z + Tl\cos\epsilon\sin\delta}{J_{xx}} \\ -\frac{J_{xx}w_x w_z - J_{zz}w_x w_z + Tl\sin\epsilon}{J_{yy}} \\ \frac{\tau_r + J_{xx}w_x w_y - J_{yy}w_x w_y}{Jzz} \end{bmatrix} \tag{2.10}$$

2.4 Linearization

Linearization of nonlinear systems is a powerful to gain insight about the system. It allows for time and frequency analysis, which in most cases are enough to produce a fast and robust controller.

This system is linearized around the hovering point with $\mathbf{p}^B$, $\mathbf{v}^B$, λ, $\mathbf{\Omega}$ = 0.

2.4.1 State space

The linearized model is

$$f(x) = A\mathbf{x} + B\mathbf{u} \tag{2.11}$$

When evaluating at the hovering point the matrices A and B are

$$\mathsf{A} = \begin{array}{c} \\ \dot{p_x^B} \\ \dot{p_y^B} \\ \dot{p_z^B} \\ \dot{v_x^B} \\ \dot{v_y^B} \\ \dot{v_z^B} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{w_x} \\ \dot{w_y} \\ \dot{w_z} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} p_x^B & p_y^B & p_z^B & v_x^B & v_y^B & v_z^B & \phi & \theta & \psi & w_x & w_y & w_z \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \tag{2.12}$$

and

$$\mathsf{B} = \begin{array}{c} \\ \dot{p_x^B} \\ \dot{p_y^B} \\ \dot{p_z^B} \\ \dot{v_x^B} \\ \dot{v_y^B} \\ \dot{v_z^B} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{w_x} \\ \dot{w_y} \\ \dot{w_z} \end{array} \begin{array}{c} \begin{array}{cccc} T & \delta & \epsilon & \tau_r \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -g & 0 \\ 0 & g & 0 & 0 \\ -\frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{-mgl}{J_{xx}} & 0 & 0 \\ 0 & 0 & \frac{-mgl}{J_{yy}} & 0 \\ 0 & 0 & 0 & \frac{1}{J_{zz}} \end{bmatrix} \end{array} \tag{2.13}$$

Furthermore, the system can be decoupled into the longitudinal (x), lateral (y) and vertical (z) linear and angular movements. However, the decoupled state space matrices will only be derived for the longitudinal (x) and lateral (y) movements (A_x, B_x and A_y, B_y, respectively). In the case of the vertical movement, with respect to the z position p_z^B and yaw ψ the system is a double integrator, so it is easier to work with a transfer function model.

Longitudinal (x) movement

$$A_x = \begin{array}{c} \\ \dot{p_x^B} \\ \dot{v_x^B} \\ \dot{\theta} \\ \dot{w_y} \end{array} \begin{array}{c} \begin{array}{cccc} p_x^B & v_x^B & \theta & w_y \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, B_x = \begin{array}{c} \\ \dot{p_x^B} \\ \dot{v_x^B} \\ \dot{\theta} \\ \dot{w_y} \end{array} \begin{array}{c} \epsilon \\ \begin{bmatrix} 0 \\ -g \\ 0 \\ \frac{-mgl}{J_{yy}} \end{bmatrix} \end{array}$$

Lateral (y) movement

$$A_y = \begin{array}{c} \\ \dot{p_y^B} \\ \dot{v_y^B} \\ \dot{\phi} \\ \dot{w_x} \end{array} \begin{array}{c} \begin{array}{cccc} p_y^B & v_y^B & \phi & w_x \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, B_y = \begin{array}{c} \\ \dot{p_y^B} \\ \dot{v_y^B} \\ \dot{\phi} \\ \dot{w_x} \end{array} \begin{array}{c} \delta \\ \begin{bmatrix} 0 \\ g \\ 0 \\ \frac{-mgl}{J_{xx}} \end{bmatrix} \end{array}$$

2.4.2 Transfer function

From the linearized system in 2.11, the transfer functions are obtained.

$$p_x^B = \frac{v_x^B}{s} \tag{2.14}$$

$$p_y^B = \frac{v_y^B}{s} \tag{2.15}$$

$$p_z^B = \frac{v_z^B}{s} \tag{2.16}$$

$$v_x^B = -g\frac{\theta}{s} - g\frac{1}{s(t_s s+1)}\epsilon \tag{2.17}$$

$$v_y^B = g\frac{\phi}{s} + g\frac{1}{s(t_s s+1)}\delta \tag{2.18}$$

$$v_z^B = -\frac{1}{m}\frac{1}{s(t_m s+1)}T \tag{2.19}$$

$$\phi = \frac{w_x}{s} \tag{2.20}$$

$$\theta = \frac{w_y}{s} \tag{2.21}$$

$$\psi = \frac{w_z}{s} \tag{2.22}$$

$$w_x = -\frac{mgl}{J_{xx}}\frac{1}{s(t_s s+1)}\delta \tag{2.23}$$

$$w_y = -\frac{mgl}{J_{yy}}\frac{1}{s(t_s s+1)}\epsilon \tag{2.24}$$

$$w_z = \frac{1}{J_{zz}}\frac{1}{s(t_r s+1)}\tau_r \tag{2.25}$$

2.5 Sensors

To observe the position, Euler angles and angular velocity states, the Xsens mti-680 was selected [10]. This unit has a sensor suite that provides inertial position, velocity, orientation and angular velocity estimates. The horizontal position is obtained from a GPS antenna, the vertical position a barometer, the velocities from GPS, the Euler angles from combining the gyroscope and the accelerometers and the angular velocities are obtained mainly from the gyroscopes.

Although this unit can provide velocity measurements, in this work it is considered that they are not available, so their noise is not going to be quantified. From the datasheet [11], the sensor specifications are shown in table 2.1. The acceleration rms values were included because, as it can be seen, the position measurements are too noisy, specially the vertical one, so 3 kalman filter to combine these signals are going to be derived. The values with asterisks (*) in the aforementioned table indicate that the root mean square values (RMS or standard deviation σ) were obtained from the power spectral density (PSD) of the noise, assuming measurements at a frequency of 100 Hz.

$$RMS(\sigma) = PSD\sqrt{frequency} \tag{2.26}$$

For example, in the case of the angular velocity,

$$
\begin{aligned}
PSD &= 0.007\,º/s/\sqrt{Hz} = 0.00012\,rad/s/\sqrt{Hz} \\
RMS &= 0.00012\sqrt{100} = 0.0012\,rad/s
\end{aligned}
\tag{2.27}
$$

Table 2.1: Sensor specification of the Xsens mti-680 in SI units.

	Horizontal Pos.	Vertical Pos.	Roll/Pitch	Yaw	Angular Vel.	Acceleration
RMS	1	2	0.0035	0.0175	0.0012*	0.0059*

*Obtained from the PSD of the noise at 100 Hz.

In the simulations in chapters 6 and 7, the sensors are approximated by unbiased measurements with noise variance specified in table 2.1.

2.6 Pre Kalman filtering

To overcome these noisy signals, a Kalman filter is derived, combining the inertial positions and accelerations using kinematic laws. The Kalman filter is analogous to the LQR controller, which will be presented and derived in section 3.3.1, and uses the same matrices and cost function, namely

$$
\begin{aligned}
\dot{\boldsymbol{x}} &= A\mathbf{x} + B\mathbf{u} \\
\boldsymbol{z} &= C\mathbf{x},
\end{aligned}
\tag{2.28}
$$

with **x** the states, **u** the known inputs and **z** the outputs. For a steady state Kalman filter, the gain can be computed once by solving the associated Ricatti equation [12] with the aforementioned matrices and the costs Q and R.

The cost Q refers to the process noise, so the bigger it is, the less we trust the model. On the contrary, the R matrix corresponds to the measurement noise, being usual to make experimental tests to quantify sensor noise variance to populate the elements of the R matrix. Just as the Q matrix, the bigger the R, the less we trust the measurements and so their noise is reduced.

The state space matrices for the position Kalman filter KFP are

$$
A_{KFP} = \begin{matrix} \\ \dot{p} \\ \dot{v} \end{matrix} \begin{matrix} \begin{matrix} p & v \end{matrix} \\ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \end{matrix}, \; B_{KFP} = \begin{matrix} \\ \dot{p} \\ \dot{v} \end{matrix} \begin{matrix} \begin{matrix} acc \end{matrix} \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{matrix}, \; C_{KFP} = \begin{matrix} \\ y \end{matrix} \begin{matrix} \begin{matrix} p & v \end{matrix} \\ \begin{bmatrix} 1 & 0 \end{bmatrix} \end{matrix}
\tag{2.29}
$$

with p the position, v the velocity and acc the known acceleration. The output position and the input acceleration are obtained directly from the Xsens mti-680 unit.

Computing the Kalman filter with the function *kalman* [13] having

$$
Q = 1 \times 10^{-8} and\, R = 1
\tag{2.30}
$$

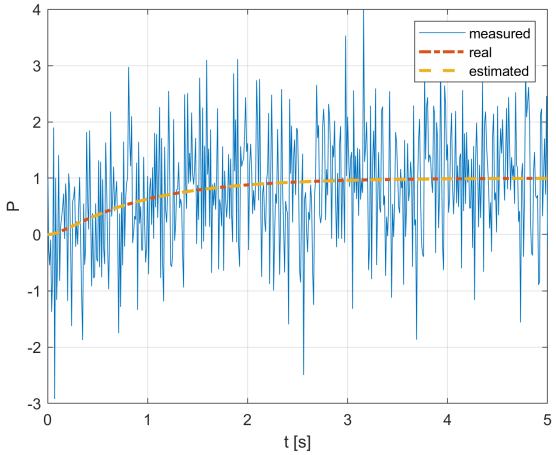
leads to the gain

$$L = \begin{bmatrix} 0.0811 & 0.0033 \\ 0.0033 & 0.0003 \end{bmatrix}. \tag{2.31}$$

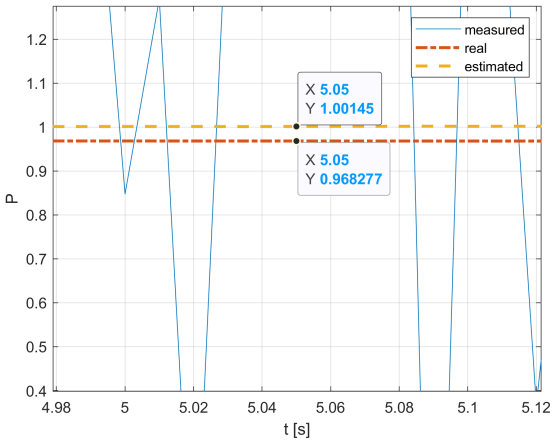
The Q and the gains are very small, which is expected because the position measurements are very noisy and by having a small Q compared to R there is a small trust in these measurements.

With these specifications, a setup of a PD controller to control the KFP system was developed, so the filtering could be tested and the results are presented in figure 2.6. In short, the cost Q should not be increased because that would lead to a higher reliance in the position measurements and thus more noise; however, it could also not be decreased due to the increase in drift.

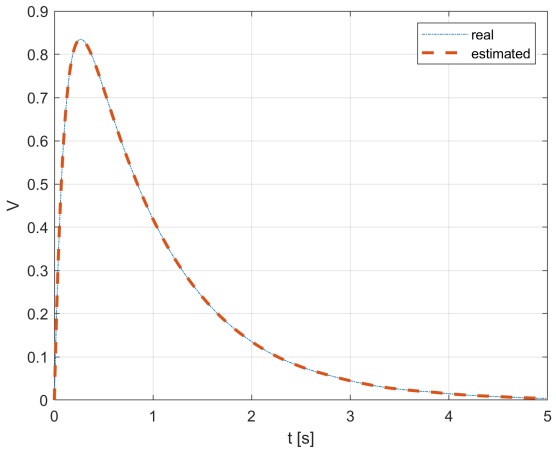
The position and velocity errors are on average around 10^{-3} m and 10^{-5} m/s in the beginning of the simulation, but could increase up to 0.035 m and 10^{-4} m/s due to drift, respectively.



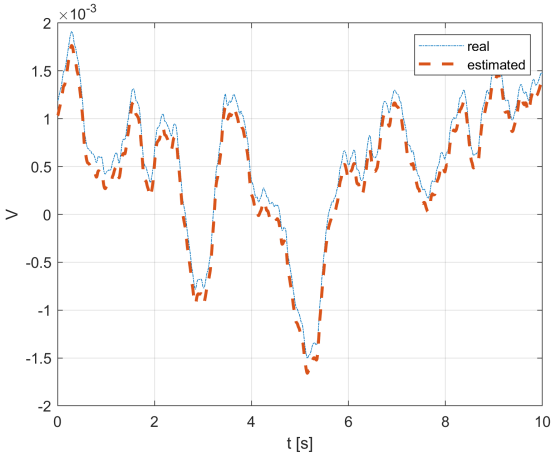
(a) Position transient of the step.



(b) Long simulation time drift of the position.



(c) Velocity transient of the step.



(d) Long simulation time drift of the velocity.

Figure 2.6: Position filtering results of the KFP.

Chapter 3

Linear Control and Estimation

In this section several linear control laws are derived. As a guideline in the control design, design specifications obtained empirically were defined and are presented in table 3.1.

Table 3.1: Design specifications,

	p_z^B	v_z^B	p_x^B, p_y^B	v_x^B, v_y^B	λ	$\mathbf{\Omega}$
Bandwidth [Hz]	1	1 to 2	1	1 to 2	5 to 10	20 to 30
ξ	1	1	0.707 to 1	1	0.707 to 1	0.707 to 1
Overshoot [%]	$<$20	$<$20	$<$20	$<$20	$<$20	$<$20

3.1 PD control

In this section PD (proportional derivative) controllers are derived for the positions x, y, z in the body frame, p_x^B , p_y^B, p_z^B and the Euler angles roll, pitch, yaw ϕ, θ, ψ, respectively. There are 4 loops, the longitudinal p_x^B and θ, the lateral p_y^B and ϕ, the vertical p_z^B and the yaw ψ loops. In the longitudinal loop, the outer PD controller will control the position x in the body frame p_x^B, feeding pitch θ references to the inner pitch PD controller. The lateral loop is analogous to the longitudinal and the vertical position and yaw loops are independent.

The derivation in the horizontal plane is only presented for the x, pitch and angular velocity in the y axis, w_y because the other loop of the position y, roll and w_x is analogous.

3.1.1 Position Along z p_z^B control

Starting with the vertical loop, the transfer function between the input thrust T and the z position is

$$p_z^B = -\frac{1}{m}\frac{T}{s^2(t_m s+1)} = -\frac{0.0727}{s^2(0.1s+1)} \tag{3.1}$$

The pole at $s=-10$ is due to the time constant of the motor, the s^2 is the double integration from vertical force to position and the negative gain is due to the upward direction of the thrust along the

negative z axis (the z axis points downwards). Using the function *pidTuner* of MATLAB it is possible to define the gains according to the given specifications.

The PD controller structure is:

$$PD_z = Kp_z + \frac{Kd_z s}{a_z s + 1} \tag{3.2}$$

with $Kp_z = -30, a_z = 0.00139, Kd_z = -50$.

The closed loop transfer function is:

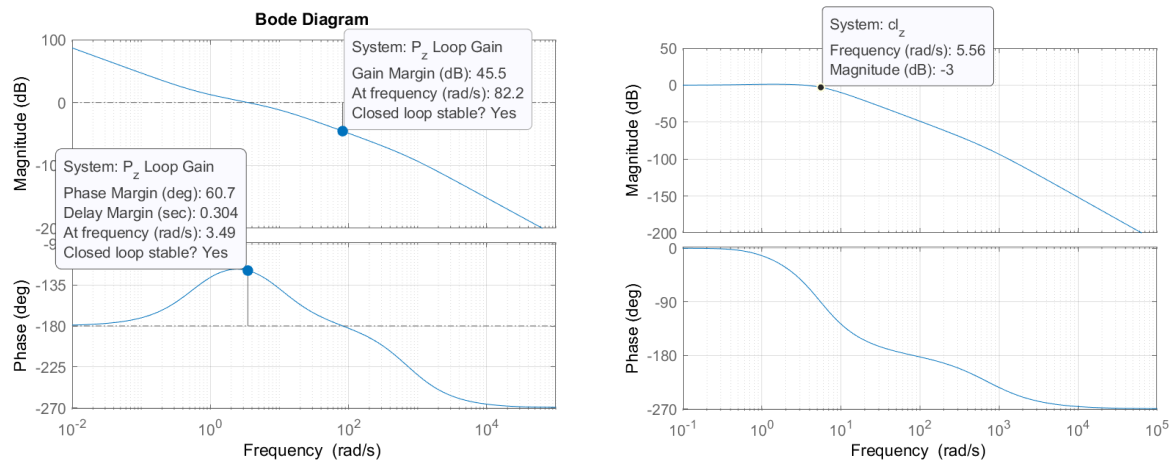
$$cl_z = \frac{2.811s + 1.685}{0.000139s^4 + 0.1014s^3 + s^2 + 2.811s + 1.685} \tag{3.3}$$

with poles at $s = -719.4641, -5.1048, -4.0381, -0.8176$.

The bode plot of the loop gain is presented in figure 3.1 a), displaying gain and phase margins of 45.5 dB and 60.7° , respectively.

$$\begin{aligned} GM_{PD_z} &= 45.5dB \\ PM_{PD_z} &= 60.7^\circ \end{aligned} \tag{3.4}$$

The closed loop bode plot is shown in figure 3.1 b), displaying a closed loop bandwidth of 5.56 rad/s. The step response of the closed loop system is available in figure 3.2.



(a) Loop gain. (b) Closed loop.

Figure 3.1: Bode plots with PD_z.

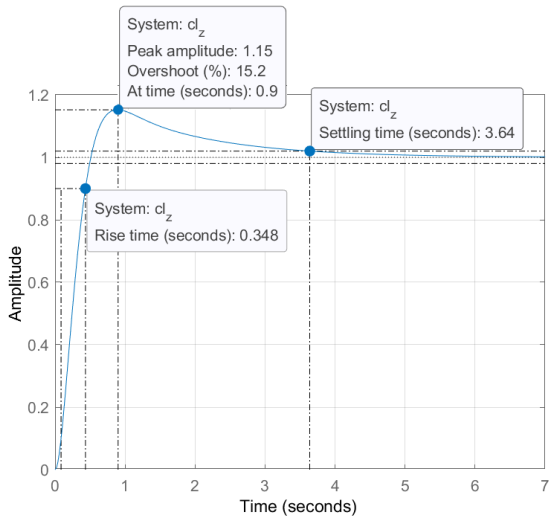


Figure 3.2: Step response with PD_z.

3.1.2 Pitch θ control

The transfer function between the inner gimbal ϵ and pitch is

$$\theta = -\frac{mgl}{J_{yy}} \frac{\epsilon}{(t_s s + 1)s^2} = \frac{-33.06}{s^2(0.005s + 1)}\epsilon \tag{3.5}$$

The PD controller is:

$$PD_\theta = Kp_\theta + \frac{Kd_\theta s}{a_\theta s + 1} \tag{3.6}$$

with $Kp_\theta = -1.24$, $Kd_\theta = -1.24$ and $a_\theta = 0.000139$.

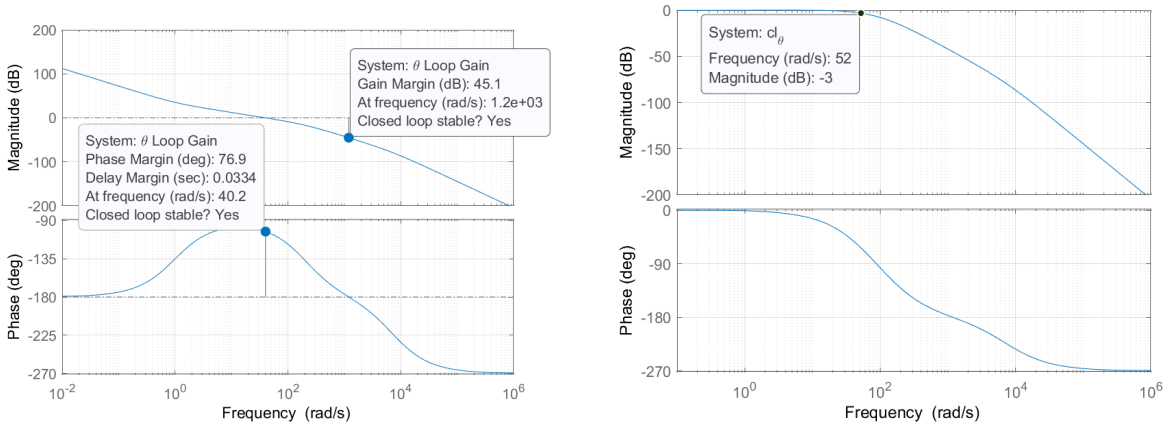
The closed loop transfer function is:

$$cl_\theta = \frac{48.2s + 48.19}{6.95e^{-7}s^4 + 0.005139s^3 + s^2 + 48.2s + 48.19} \tag{3.7}$$

The bode plot of the loop gain is presented in figure 3.3 a). According to this figure, the gain and phase margins are 45.1 dB and 76.9° , respectively.

$$\begin{aligned} GM_{PD_\theta} &= 45.1dB \\ PM_{PD_\theta} &= 76.9^\circ \end{aligned} \tag{3.8}$$

The closed loop bode plot is shown in figure 3.3 b), showing a closed loop bandwidth of 52 rad/s. The step response of the closed loop system is available in figure 3.4.



(a) Loop gain. (b) Closed loop.

Figure 3.3: Bode plots with PD_θ.

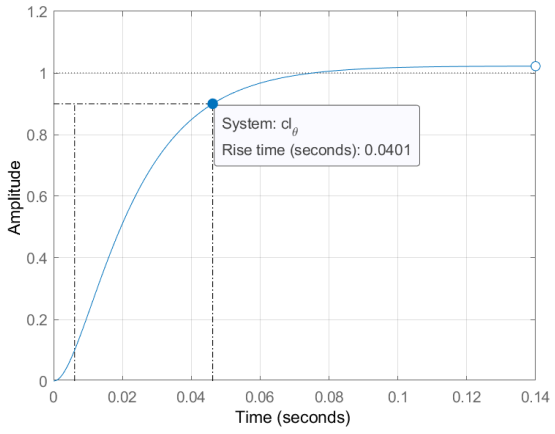


Figure 3.4: Step response with PD_θ.

3.1.3 Position Along x P_x^B control

Having closed the pitch control loop, it is now possible to control the position in the x axis by manipulating the pitch of the vehicle. Thus, the input is the pitch reference to the inner controller developed previously.

The following equations are to be combined to derive the relationship between the pitch reference θ_{ref} and the position in the x axis P_x.

$$\begin{cases} p_x = \frac{v_x}{s} \\ v_x = -g\frac{\theta}{s} - g\frac{\epsilon}{s(t_s s+1)} \\ \theta = \frac{w_y}{s} \\ w_y = -\frac{mgl}{J_{yy}}\frac{\epsilon}{s(t_s s+1)} \end{cases} \tag{3.9}$$

Rearranging the second and fourth equations

$$\begin{cases} v_x = -g\frac{\theta}{s} - \frac{mgl}{J_{yy}}\frac{\epsilon}{s(t_s s+1)}\frac{J_{yy}}{ml} \\ s\theta = -\frac{mgl}{J_{yy}}\frac{\epsilon}{s(t_s s+1)} \end{cases} \tag{3.10}$$

leads to

$$v_x = -g\frac{\theta}{s} + s\theta\frac{J_{yy}}{ml} = (s\frac{J_{yy}}{ml} - g\frac{1}{s})\theta. \tag{3.11}$$

The inner pitch controller has the following transfer function

$$\frac{\theta}{\theta_{ref}} = cl_\theta(=)\theta = cl_\theta\theta_{ref} \tag{3.12}$$

arriving at

$$v_x = (s\frac{J_{yy}}{ml} - g\frac{1}{s})cl_\theta\theta_{ref} \tag{3.13}$$

and finally

$$\frac{P_x}{\theta_{ref}} = (s\frac{J_{yy}}{ml} - g\frac{1}{s})\frac{1}{s}cl_\theta = \frac{12.17s^3 + 12.16s^2 - 472.8s - 472.8}{6.95\times10^{-7}s^6 + 0.005139s^5 + s^4 + 48.2s^3 + 48.19s^2}. \tag{3.14}$$

Once again, choosing the same PD controller as previously

$$PD_{p_x} = Kp_{p_x} + \frac{Kd_{p_x}s}{a_{p_x}s+1} \tag{3.15}$$

with $Kp_{p_x} = -0.036$, $Kd_{p_x} = -0.14$ and $a_{p_x} = 0.164$. Notice the negative gains due to the non minimal phase zero in that complicates the control task.

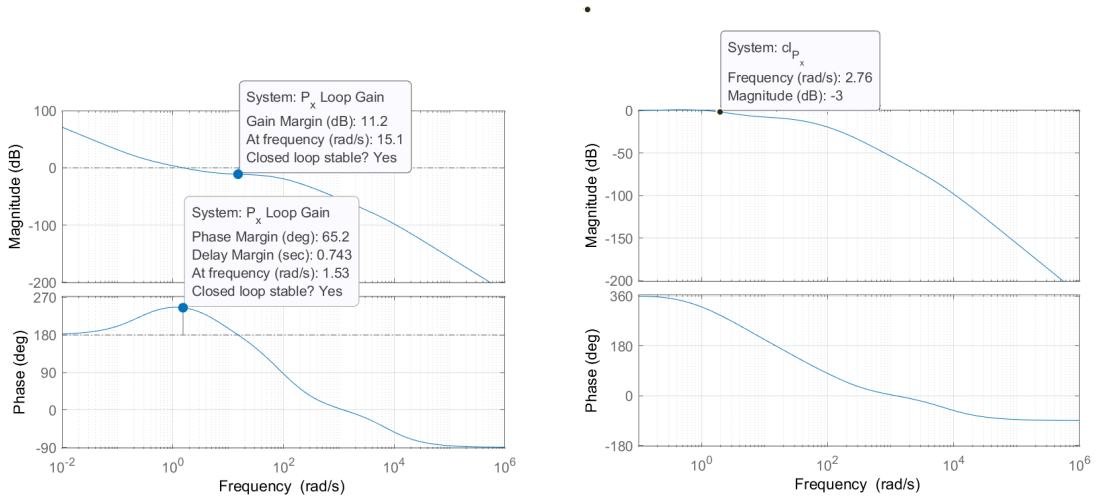
The closed loop transfer function is:

$$cl_{p_x} = \frac{-1.775s^4 - 2.213s^3 + 68.55s^2 + 86s + 17.02}{1.14 \times 10^{-7}s^7 + 0.0008435s^6 + 0.1691s^5 + 7.13s^4 + 53.89s^3 + 116.7s^2 + 86s + 17.02} \tag{3.16}$$

The bode plot of the loop gain is available in figure 3.5 with gain and phase margins of 15.1 and 65.2, respectively.

$$\begin{aligned} GM_{PD_{p_x}} &= 15.1dB \\ PM_{PD_{p_x}} &= 65.2^\circ \end{aligned} \tag{3.17}$$

The closed loop bode plot in 3.5 b), displaying a closed loop bandwidth of 2.76 rad/s. The step response is shown in 3.6.



(a) Loop gain. (b) Closed loop.

Figure 3.5: Bode plots with PD_{p_x}.

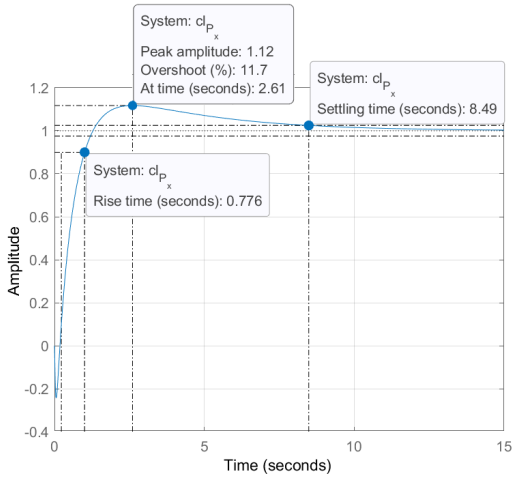


Figure 3.6: Step response with PD_{p_x}.

3.1.4 Yaw ψ control

The transfer function between the input torque τ_r and yaw is

$$\psi = \frac{1}{J_{zz}} \frac{\tau_r}{(t_r s + 1)s^2} = \frac{1.601}{s^2(0.005s + 1)} \tau_r \tag{3.18}$$

The PD controller is:

$$PD_\psi = Kp_\psi + \frac{Kd_\psi s}{a_\psi s + 1} \tag{3.19}$$

with $Kp_\psi = 113$, $Kd_\psi = 21.2$ and $a_\psi = 0.00269$.

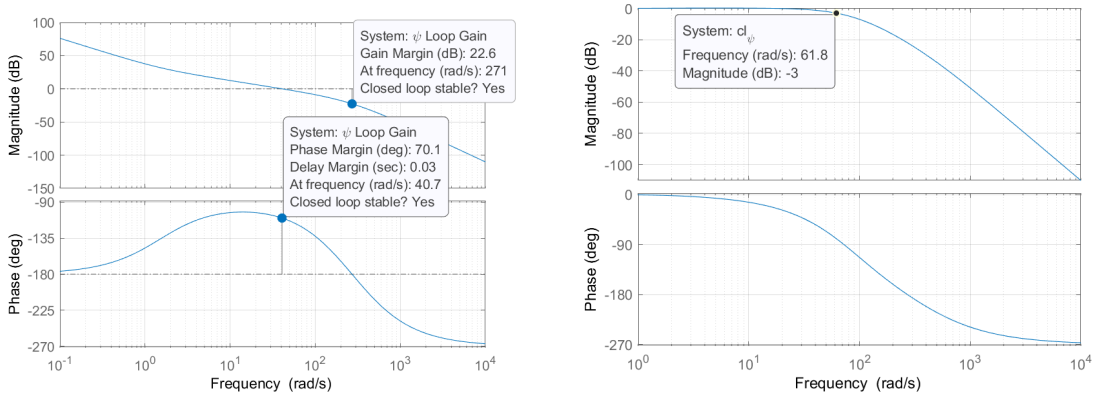
The closed loop transfer function is:

$$cl_\psi = \frac{32.1s + 168.7}{1.345e-05s^4 + 0.00769s^3 + s^2 + 32.1s + 168.7} \tag{3.20}$$

The bode plot of the loop gain is presented in figure 3.7 a). According to this figure, the gain and phase margins are 22.6 dB and 70.1° , respectively.

$$\begin{aligned} GM_{PD_\psi} &= 22.6dB \\ PM_{PD_\psi} &= 70.1^\circ \end{aligned} \tag{3.21}$$

The closed loop bode plot is shown in figure 3.7 b), with a closed loop bandwidth of 61.8 rad/s. The step response of the closed loop system is available in figure 3.8.



(a) Loop gain. (b) Closed loop.

Figure 3.7: Bode plots with PD_ψ.

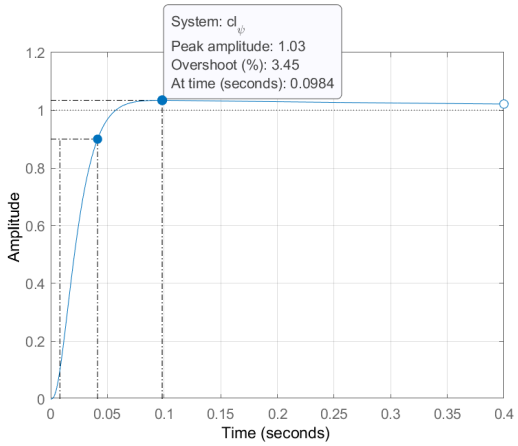


Figure 3.8: Step response with PD_ψ

3.2 Cascaded Position Control

3.2.1 Position Along x P_x^B control

Using the results from 3.1.3, the open loop transfer function is

$$v_x = (s\frac{J_{yy}}{ml} - g\frac{1}{s})cl_\theta\theta_{ref} \tag{3.22}$$

when substituting,

$$\frac{V_x}{\theta_{ref}} = \frac{12.17s^3 + 12.16s^2 - 472.8s - 472.8}{6.95 \times 10^{-7}s^5 + 0.005139s^4 + s^3 + 48.2s^2 + 48.19s} \tag{3.23}$$

with the help of the *ControlSystemDesigner* of MATLAB, it is possible to infer that a controller with a gain and a real pole is enough to ensure the required performance

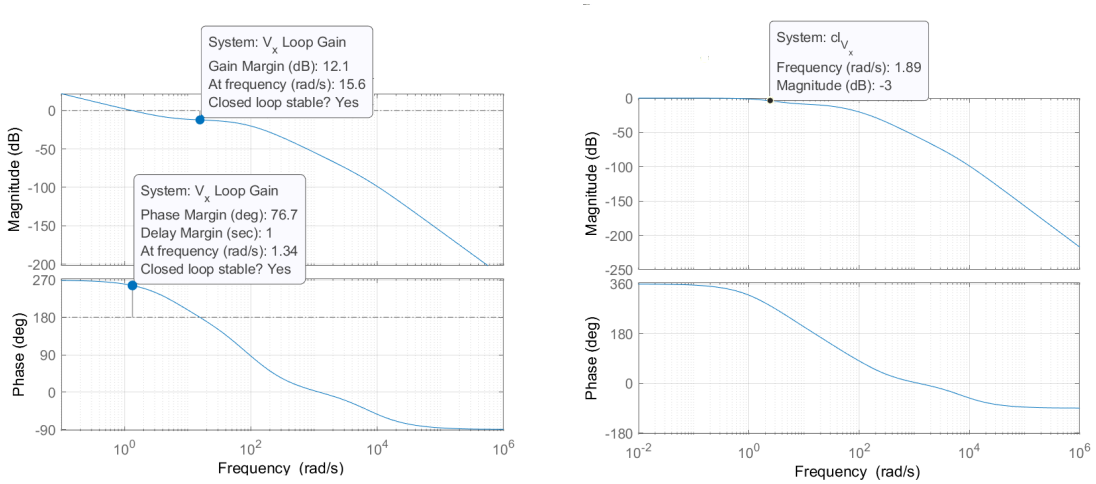
$$K_{v_x} = \frac{-0.13}{0.16s + 1}. \tag{3.24}$$

Notice again the negative gain to compensate for the non minimum phase zero in 3.22. The closed loop transfer function is

$$cl_{v_x} = \frac{-1.582s^3 - 1.581s^2 + 61.47s + 61.46}{1.1 \times 10^{-7}s^6 + 0.0008229s^5 + 0.1651s^4 + 7.13s^3 + 54.33s^2 + 109.7s + 61.46}. \tag{3.25}$$

The bode plot of the loop gain, the bode plot of the closed loop and the step response are presented in figures 3.9 a), b) and 3.10, respectively. The gain and phase margins are shown in equation 3.26.

$$\begin{aligned} GM_{C_{v_x}} &= 12.1dB \\ PM_{C_{v_x}} &= 76.7^\circ \end{aligned} \tag{3.26}$$



(a) Loop gain. (b) Closed loop.

Figure 3.9: Bode plots with K_{v_x}.

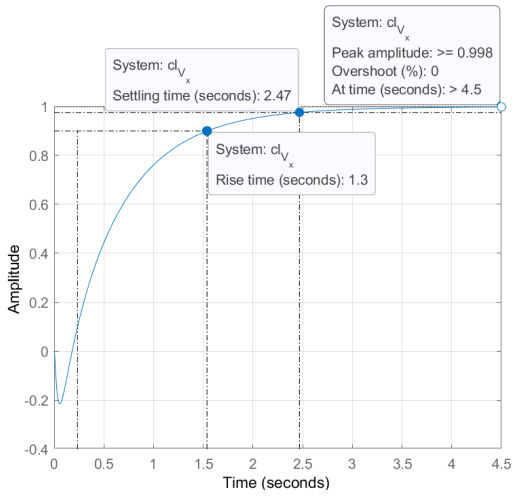


Figure 3.10: Step response with K_{v_x}.

Having defined the inner loop of the cascaded position control p_x, the open loop transfer function of the outer loop can be approximated by

$$\frac{v_x}{v_{x_{ref}}} = cl_{v_x}(=)\frac{sp_x}{v_{x_{ref}}} = cl_{v_x}(=)\frac{p_x}{v_{x_{ref}}} = \frac{cl_{v_x}}{s} \tag{3.27}$$

substituting,

$$\frac{p_x}{v_{x_{ref}}} = \frac{-1.582s^3 - 1.581s^2 + 61.47s + 61.46}{1.1 \times 10^{-7}s^7 + 0.0008229s^6 + 0.1651s^5 + 7.13s^4 + 54.33s^3 + 109.7s^2 + 61.46s} \tag{3.28}$$

using only a proportional controller

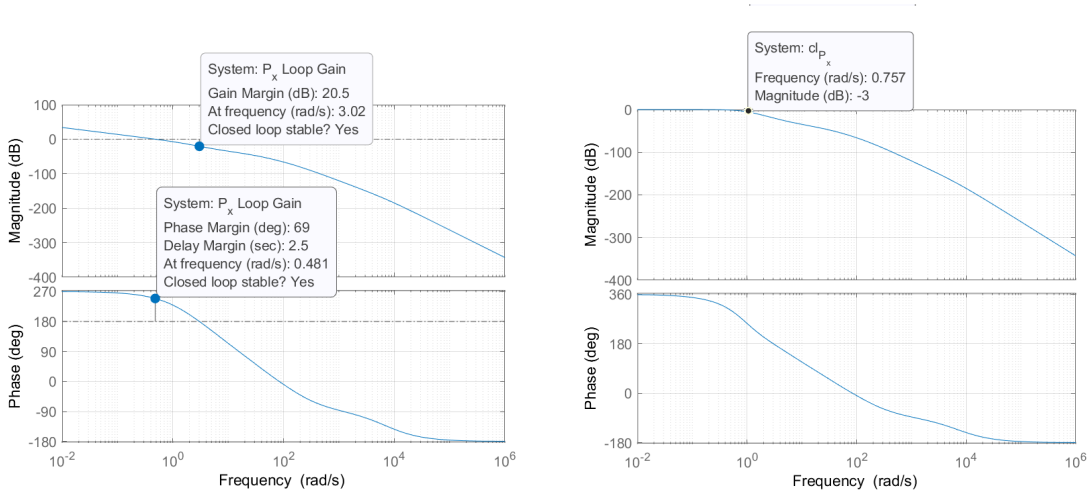
$$K_{p_x} = 0.5 \tag{3.29}$$

the closed loop transfer function is

$$cl_{p_x} = \frac{-0.7908s^3 - 0.7907s^2 + 30.73s + 30.73}{1.1 \times 10^{-7}s^7 + 0.0008229s^6 + 0.1651s^5 + 7.13s^4 + 53.54s^3 + 108.9s^2 + 92.2s + 30.73} \tag{3.30}$$

Again, the bode plot of the loop gain, the bode plot of the closed loop and the step response are presented in figures 3.11 a), b) and 3.12, respectively. The gain and phase margins are presented in equation 3.31.

$$\begin{aligned} GM_{C_{p_x}} &= 20.5dB \\ PM_{C_{p_x}} &= 69^\circ \end{aligned} \tag{3.31}$$



(a) Loop gain.

(b) Closed loop.

Figure 3.11: Bode plots with K_{p_x}.

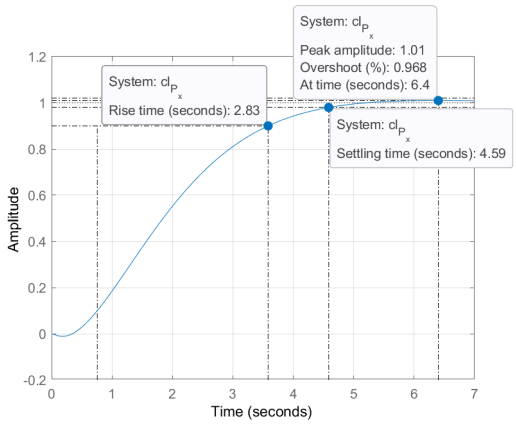


Figure 3.12: Step response with K_{p_x}.

3.3 Linear Quadratic Gaussian

A LQG controller is going to be designed, combining the inner angular position loop and the outer position loop in one full state feedback loop [14]. Assuming no velocity measurements, a Kalman filter is used to estimate them. Note that all the classical gain and phase margins presented in the next sections are measured at the outputs, given that we are dealing with a MIMO system. Considering that the gain and phase margins proved to be much higher in the 3 last outputs (velocity, Euler angles and angular velocity) than in the position loop, only the latter is shown throughout the rest of this thesis. Therefore, the quantities such as GM_{LQR} or PM_{LQR} correspond to the gain and phase margins, respectively, measured at the outputs, of the position loop.

Again, only the derivation for the x position, velocity, pitch and W_y LQR is shown.

3.3.1 LQR

The dynamics can be described by

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\mathbf{x} + B\mathbf{u} \\ \boldsymbol{z} &= C\mathbf{x}, \end{aligned} \tag{3.32}$$

with **x** the states, **u** the inputs and **z** the outputs. The cost function of the system is

$$J = \int_0^\infty \mathbf{x}^T Q\mathbf{x} + \mathbf{u}^T R\mathbf{u} \quad dt, \tag{3.33}$$

and the algorithm attempts to minimize this cost by solving the corresponding Riccati equation. It can also be proven that the resulting gain from this solution leads to an asymptotically stable closed loop [14].

The real task in LQR design is to choose suitable Q and R matrices such that the system has the required performance in terms of response and robustness. In short, a bigger Q leads to a higher cost on the states and as such the resulting gain will attempt to keep them low, whereas a high R matrix means that the control actuation is very expensive and the gain will try to minimize the control effort. A starting point can be settled using the Bryson's rule [14], defining Q and R as diagonal matrices and assigning each entry the inverse of the maximum acceptable variation of the corresponding state or control action squared.

3.3.2 Position Along x P_x^B control

Recalling the decoupled A_x and B_x matrices derived in section 2.4.1

$$A_x = \begin{array}{c} \\ \dot{p_x^B} \\ \dot{v_x^B} \\ \dot{\theta} \\ \dot{w_y} \end{array} \begin{array}{c} \begin{array}{cccc} p_x^B & v_x^B & \theta & w_y \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, B_x = \begin{array}{c} \\ \dot{p_x^B} \\ \dot{v_x^B} \\ \dot{\theta} \\ \dot{w_y} \end{array} \begin{array}{c} \begin{array}{c} \epsilon \end{array} \\ \begin{bmatrix} 0 \\ -g \\ 0 \\ \frac{-mgl}{J_{yy}} \end{bmatrix} \end{array}$$

Having calculated the A and B matrices, one can define the Q and R matrices to obtain the optimal gain, computed in this case with MATLAB's lqr function

$$\text{Q} = \begin{bmatrix} \frac{1}{0.1^2} & 0 & 0 & 0 \\ 0 & \frac{1}{0.1^2} & 0 & 0 \\ 0 & 0 & \frac{1}{(1.2\pi/180)^2} & 0 \\ 0 & 0 & 0 & \frac{1}{(1\pi/180)^2} \end{bmatrix} \text{ and R} = \frac{1}{12 \times \pi/180} = 11.36 \tag{3.34}$$

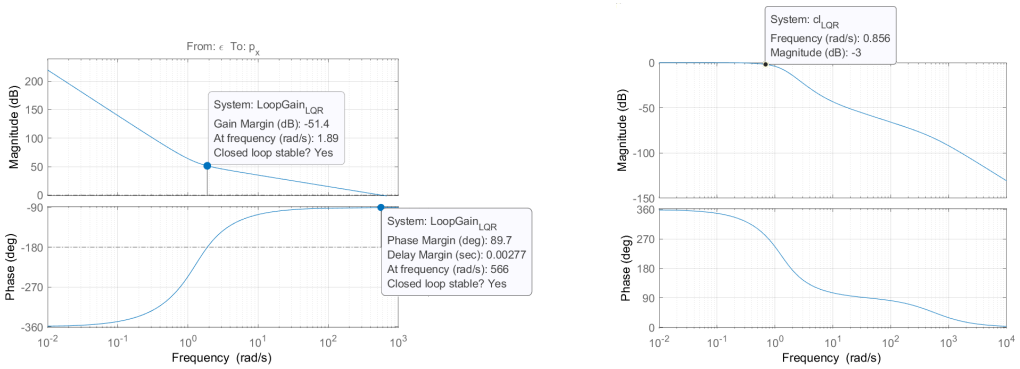
which, when computed using the lqr function of MATLAB, arrived at the following gains

$$K_{LQR_{p_x}} = \begin{bmatrix} 2.9671 & 6.2105 & -49.2085 & -18.9511 \end{bmatrix} \tag{3.35}$$

The bode plot of the loop gain is presented in figure 3.13 a). In this case, it does not make sense to compare the margins present in the aforementioned figure because it does not have the same physical meaning (notice the negative sign in the gain margin). The gain margin is simply displaying the gain in dB that the system has when the phase is $180°$, which for complex systems such as this one (4 poles at the origin of the plant, from input ϵ to output P_x^B) might not be useful. This time, using the function *allmargin* [15] of MATLAB, the classical gain and phase margins can be computed from the loop gain of the *MIMO* system consisting of A_{p_x} and $K_{LQR_{p_x}}$, resulting in the gain and phase margins in equation 3.36.

$$\begin{aligned} GM_{LQR} &= 14.68dB \\ PM_{LQR} &= 67.72° \end{aligned} \tag{3.36}$$

The closed loop bode plot is shown in figure 3.13 b), with a closed loop bandwidth of 0.86 rad/s and the step response of the closed loop system is available in figure 3.14.



(a) Loop gain. (b) Closed loop.

Figure 3.13: Bode plots with LQR_{p_x}.

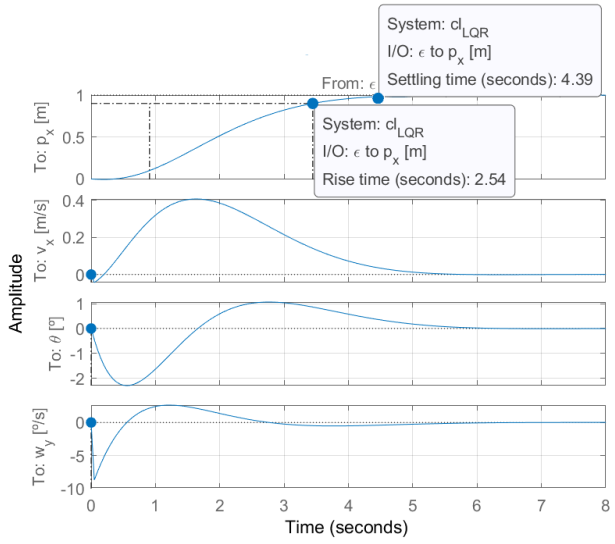


Figure 3.14: Step response with LQR_{p_x}.

3.3.3 Disk margins

Disk margins are a more comprehensive approach to stability margins than classical gain and phase margins, namely in the domain of multiple input multiple output *MIMO* systems. Contrary to classical margins, disk margins compare variations in gain and phase simultaneously, leading to a disk in a plane having as y axis the phase margin and x axis the gain margin. Additionally, it takes into account multi loop variations (in this system, there are 4 loops, so it would vary the gain and phase of them individually simultaneously) at the inputs and outputs (it can be shown that perturbations at the inputs can affect stability differently than at the outputs) [16].

To calculate the disk margin, a complex valued perturbation f is added to the loop, comprising a disturbed loop $L_f = fL$, as depicted in figure 3.15.

Each set of perturbations, denoted f or $D(\alpha, \sigma)$, is a disk parameterized by a size α and skew σ. Given a skew σ, the disk margin is the largest size α for which the closed loop remains stable for all perturbations in $D(\alpha, \sigma)$, whose domain is given by

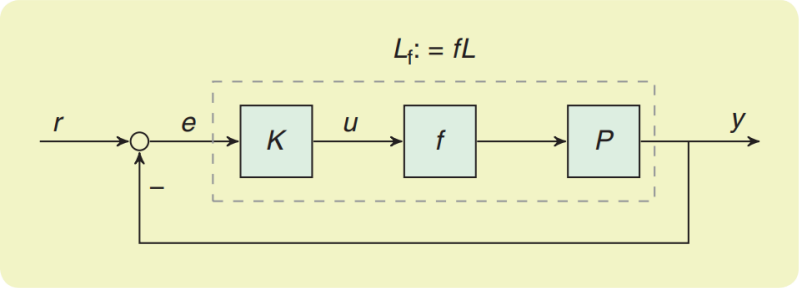


Figure 3.15: Disturbed feedback loop with perturbation f [16].

$$f \in D(\alpha, \sigma) = \left\{ \frac{1 + \frac{1-\sigma}{2}\delta}{1 - \frac{1+\sigma}{2}\delta} : \delta \in \mathbb{C} \quad \text{with } |\delta| < \alpha \right. \quad . \tag{3.37}$$

In short, the value of skew, σ specifies if the gain is expected to increase or decrease, having chosen, for simplicity, a value of $\sigma = 0$.

An example of variations $D(\alpha, \sigma)$ having $\sigma = 0.2$ and $\alpha = 0.75$ is displayed in figure 3.16. From these perturbations, a minimum and maximum gain of γ_{min} and γ_{max}, respectively, can be multiplied to the system, assuming a null variation in phase. On the contrary, a maximum phase of ϕ_{max} can be added, assuming a null variation in gain.

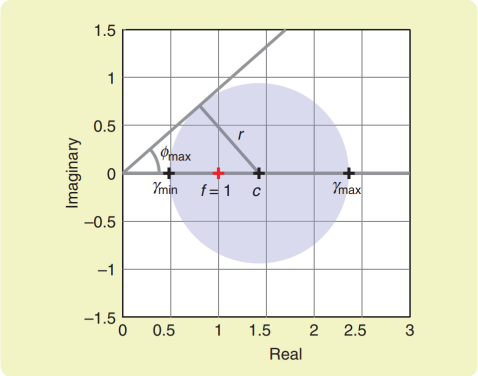


Figure 3.16: The set of variations $D(\alpha, \sigma)$ having $\sigma = 0.2$ and $\alpha = 0.75$ [16].

Lastly, the function *diskmargin* of MATLAB was used to perform the calculations [17], leading to the following disk gain and phase margins (considering multi-loop variations), DGM_{LQR} and DPM_{LQR}, respectively.

$$\begin{aligned} DGM_{LQR} &= 2.5dB \\ DPM_{LQR} &= 16.3^\circ \\ DM_{LQR} &= 0.29 \end{aligned} \tag{3.38}$$

To illustrate these margins, the disk can be visualized in figure 3.17. As expected, disk margin is much smaller than the classical margins or the individual gain or phase margins.

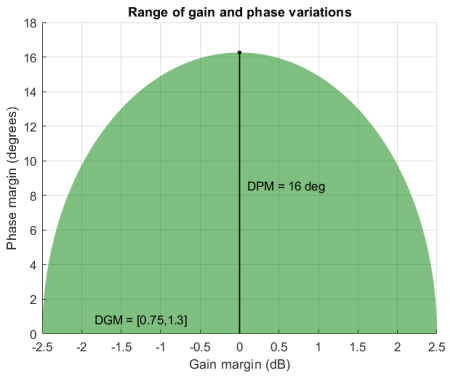


Figure 3.17: Maximum simultaneous gain and phase variations using the LQR controller.

3.3.4 LQG

Having designed the LQR, now the Kalman filter is built. In fact, according to the separation theorem, the controller and filter can be designed independently and the closed poles are the sum of their poles individually [18].

In this work, the loop transfer recovery (LTR) technique is used, which aims to recover the properties of a LQR controller, even when not having access to some states [19].

The covariance of the error could be computed at each time step to calculate a variable gain, but this time only the optimal gain computed by the Riccati equation is going to be used, formulating a steady state Kalman filter.

The algorithm is split into two phases, prediction and correction, and can be described by the following equations

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u} + L[\mathbf{y} - C\mathbf{x}] \\ \mathbf{y} &= C\mathbf{x} \end{aligned} . \tag{3.39}$$

According to the LTR technique, in the design of the Kalman filter, the perturbations are introduced as

$$\dot{x} = A\mathbf{x} + B\mathbf{u} + B\mathbf{d} \tag{3.40}$$

which means that the perturbations are weighted by the input matrix. Then, the Q and R matrices are chosen

The chosen Q and R values are

$$\mathrm{Q} = 1 \quad \text{and} \quad \mathrm{R} = \sigma = 10 \tag{3.41}$$

and using the function *kalman* of MATLAB resulted in the Kalman gain [13]

$$L = \begin{bmatrix} 3.7834 & -0.6640 & 0.2662 \\ 7.4131 & -3.0149 & 6.5834 \\ -0.6640 & 0.7473 & 0.9764 \\ 0.2662 & 0.9764 & 23.3847 \end{bmatrix}. \tag{3.42}$$

To validate these results, the loop gain for different values of σ is shown in figure 3.18.

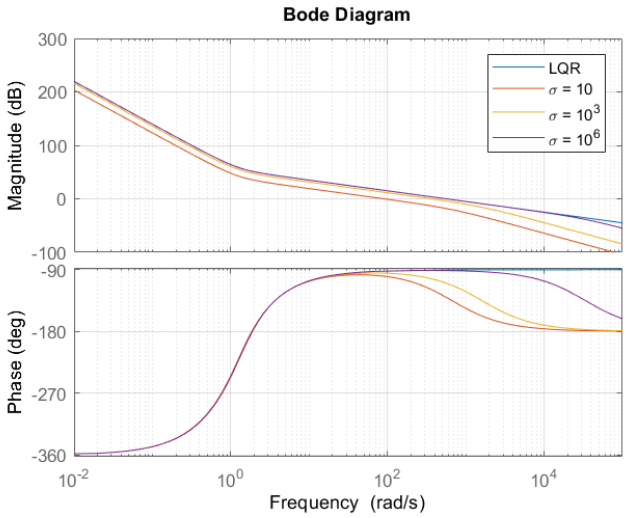


Figure 3.18: Loop gain using several σ values.

The results having the different values of σ were very similar, so in the end $\sigma = 10$ was chosen because it is enough to give a clear estimate of the states and would lead to the most noise reduction. After closing the feedback loop of the open loop system with the coupled LQR and Kalman filter, the step response to a position reference of 1 is presented in figure 3.19. The kalman filter is able to estimate precisely the velocity, as can be seen in the velocity graph. The time response of the LQG is very similar to the LQR alone, but the gain and phase margins are significantly lower and, as will be seen in the nonlinear simulations, will cause some oscillation at the input (the inner gimbal ϵ). As was the case for the LQR controller, it is not possible to estimate stability margins from the loop gain bode plot, so it is not presented. In addition to this, the bandwidth of the system is the same (as the settling time is also similar), so the closed loop bode plot is also disregarded. Again, with the functions *allmargin* and *diskmargin*, the classical and disk margins can be computed, arriving at the values in equation 3.43.

$$\begin{aligned} GM_{LQG} &= 4.6dB \\ PM_{LQG} &= 23.8^\circ \\ DM_{LQG} &= 0.16 \end{aligned} \tag{3.43}$$

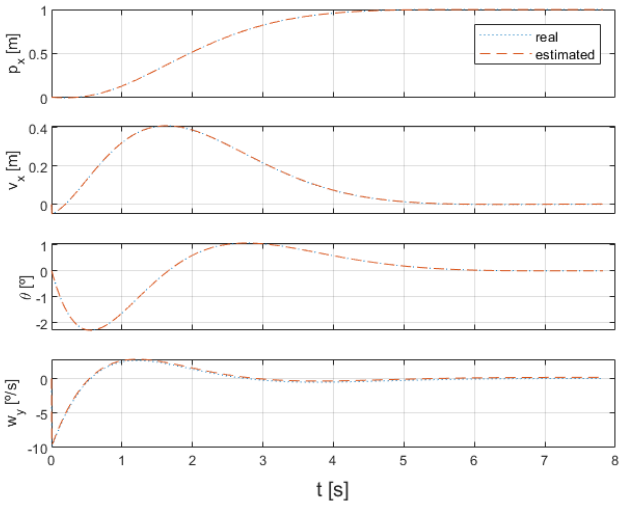


Figure 3.19: Step response with LQG_{p_x}.

3.4 Linear Model Predictive Controller

Considering that the A and B matrices have been developed and MATLAB has a model predictive control toolbox readily available, an attempt to control the VIRIATO Test Platform with a linear mpc is made. To control the positions P_x and P_y, 2 model predictive controllers are used with the same A_x, B_x and A_y, B_y matrices of the 2 previous subsections.

A sample time of

$$h_s = 0.1s \tag{3.44}$$

was chosen after testing several values. Although the sampling rate of the sensors and actuators is bigger (100 Hz), the solver of MATLAB showed some instability if a faster rate was chosen. The plant

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\mathbf{x} + B\mathbf{u} \\ \boldsymbol{y} &= C\mathbf{x}, \end{aligned} \tag{3.45}$$

is fed to the MPC object of MATLAB, and it converts it to a discrete time model with the specified sample time using the function c2d using zero order hold on the inputs [20][21].

The problem formulation is similar to the lqr, except that the model predictive controller tries to minimize the cost function 3.48 in a finite time horizon, instead of infinite time. This implies that a prediction horizon must be chosen, selecting a window to minimize the cost function

$$P_h = 1.5s \tag{3.46}$$

which is equal to 15 time steps. A control horizon should also be defined, being the time up until the control input is changing; after it, the inputs remain the same in the remaining of the prediction horizon.

The control horizon chosen was

$$C_h = 0.2s \tag{3.47}$$

At each time step, the mpc tries to minimize the following cost function

$$J = \int_0^{P_h} \mathbf{x}(t)^T Q \mathbf{x}(t) + \mathbf{u}(t)^T R \mathbf{u}(t) dt, \tag{3.48}$$

noting that the control input u is now changing with time and the upper limit of the integral is no longer infinity but the prediction horizon. In fact, the mpc function of MATLAB is not solving this problem in continuous time, but in discrete. In this case, the plant model is discretized

$$\begin{aligned} \boldsymbol{x}(k+1) &= A\mathbf{x}(k) + B\mathbf{u}(k) \\ \boldsymbol{y}(k) &= C\mathbf{x}(k), \end{aligned} \tag{3.49}$$

and the cost function is

$$J = \sum_{k=1}^{P_{h_n}} (\mathbf{y}(k)^T Q \mathbf{y}(k) + \mathbf{u}(k)^T R \mathbf{u}(k)) \tag{3.50}$$

with P_{h_n} the number of time steps of the prediction horizon. The mpc function of MATLAB solves this problem with a quadratic programming approach, which also allows to define the constraints that this cost function is subject to.

3.4.1 Position Along x P_x^B control

The derivation and results are only shown for the position x P_x control, as for P_y the procedure is the same.

To control the x position P_x^B, the matrices A_x and B_x previously derived in section 2.4.1 are used. The constraints are

$$\begin{aligned} -17\pi/180 <= \epsilon <= 17\pi/180 \\ -15\pi/180 <= \theta <= 15\pi/180, \end{aligned} \tag{3.51}$$

defining a maximum inner gimbal angle ϵ of 17° and a pitch θ of 15 °.

The chosen Q and R weights are

$$\text{Q} = \begin{bmatrix} 1 & 0.1 & 0.1 & 0.1 \end{bmatrix} \text{ and } \text{R} = 10, \tag{3.52}$$

which are arrays and not matrices due to the mpc function having them as arguments in this format.

With these specifications, a linear simulation was carried out in figure 3.20, consisting of a step

response to a P_x^B reference. As it can be seen, the performance is the best so far; however, it is much harder to perform a frequency and robustness analysis and will not be done here.

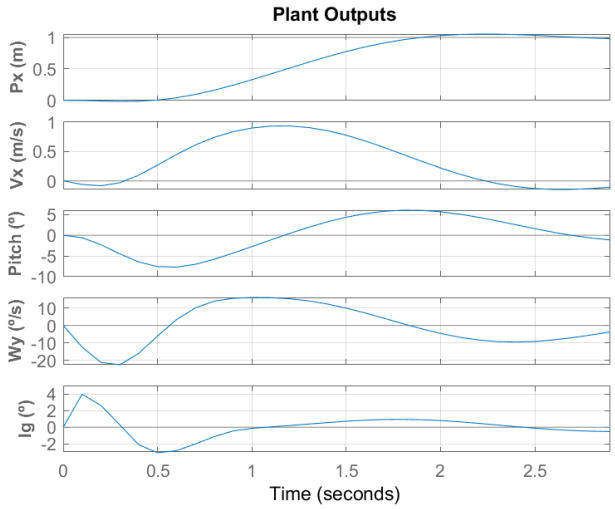


Figure 3.20: Step plot with MPC_{p_x}.

3.5 Final Performance Results

The resulting performance of the linear controllers of the position along x P_x^B can be summarized in table 3.2, with ξ the damping factor, T_R the rise time (defined as the time to reach 90% of the reference level 1), T_S the settling time (time required for the signal to reach 98% of the reference value and does not leave this threshold later), GM the gain margin and PM the phase margin.

Table 3.2: Resulting position x P_x^B performance.

	Bandwidth [Hz]	ξ	Overshoot [%]	T_R [s]	T_S [s]	GM [dB]	PM[º]
PD	0.07	0.564	11.7	0.78	8.49	11.2	65.2
Cascaded	0.12	0.844	1.0	2.83	4.59	20.5	69.0
LQR	0.14	0.702	0.0	2.54	4.39	14.7	67.7
MPC	-	-	5.3	0.95	2.74	-	-

As one would expect the MPC controller showed the best performance, taking only 2.74s to reach the reference. However, in terms of robustness the cascaded controller has the highest stability margins, although close to the PD and LQR controllers (it is possible to calculate these values for the MPC controller, but it is not done in this work). As described previously, a more thorough analysis of robustness for *MIMO* systems can be carried out using the function *diskmargin* of MATLAB, which takes into account multi loop variations at the inputs and outputs simultaneously. These margins computed with the function *diskmargin* are significantly lower (for instance, $DGM_{LQR} = 2.5dB$ as opposed to $GM_{LQR} = 14.7dB$), which means that one should not be overconfident on the SISO results.

Although the performance and robustness of the cascaded controller are lower than the LQR and MPC, it still would be the best first approach to control the VIRIATO Test Platform. The cascaded scheme enables to tune each loop independently which makes the tuning task significantly easier.

In table 3.3 the resulting specifications of the PD pitch controller are presented. It can be seen that they are fulfilled with the simplest controller, so it might not be required to use more complicated structures. The results for the roll ϕ control are identical.

Table 3.3: Resulting pitch θ performance.

	Bandwidth [Hz]	ξ	Overshoot [%]	T_R [s]	T_S [s]	GM [dB]	PM[º]
PD	8.28	1	2.2	0.04	0.26	45.1	76.9

The resulting specifications for the height P_z control can be visualized in table 3.4. It can be seen that the specifications are almost met, being the desired closed loop bandwidth of 1 Hz very close to the obtained one.

Table 3.4: Resulting height P_z performance.

	Bandwidth [Hz]	ξ	Overshoot [%]	T_R [s]	T_S [s]	GM [dB]	PM[º]
PD	0.88	0.847	15.2	0.348	3.64	45.5	60.7

Lastly, the specifications of the yaw ψ control are displayed in table 3.5. The desired performance parameters are once again met.

Table 3.5: Resulting yaw ψ performance.

	Bandwidth [Hz]	ξ	Overshoot [%]	T_R [s]	T_S [s]	GM [dB]	PM[º]
PD	9.84	0.946	3.5	0.03	0.45	22.6	70.1

Chapter 4

Robust Control

In this section H_2, H_∞ and μ synthesis control laws are derived. In figure 4.1 a general feedback control system structure is shown, with $\mathbf{P}$ the augmented plant model and $\mathbf{K}$ the controller model.

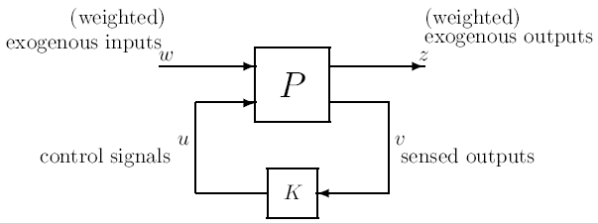


Figure 4.1: General control configuration [22].

From the block diagram,

$$\begin{bmatrix} \mathbf{z} \\ \mathbf{v} \end{bmatrix} = \mathbf{P}(s) \begin{bmatrix} \mathbf{w} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{11}(s) & \mathbf{P}_{12}(s) \\ \mathbf{P}_{21}(s) & \mathbf{P}_{22}(s) \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{u} \end{bmatrix} \tag{4.1}$$

$$\mathbf{u} = \mathbf{K}(s)\mathbf{v} \tag{4.2}$$

and the generalized plant $\mathbf{P}$ is

$$\mathbf{P} = \left[\begin{array}{c|cc} \mathbf{A} & \mathbf{B}_1 & \mathbf{B}_2 \\ \hline \mathbf{C}_1 & \mathbf{D}_{11} & \mathbf{D}_{12} \\ \mathbf{C}_2 & \mathbf{D}_{21} & \mathbf{D}_{22} \end{array}\right] \tag{4.3}$$

The objective of this formulation of the control problem is to find a controller $\mathbf{K}$, which based on the information in $\mathbf{v}$ generates an input $\mathbf{u}$ which counteracts the influence of $\mathbf{w}$ on $\mathbf{z}$ thereby minimizing the norm from $\mathbf{w}$ to $\mathbf{z}$.

The signals are $\mathbf{u}$ the control variables, $\mathbf{v}$ the measured variables, $\mathbf{w}$ the exogenous signals such as disturbances $\mathbf{wd}$ and commands $\mathbf{r}$ and $\mathbf{z}$ the so called error signals which are to be minimized in some sense to meet the control objectives. As shown in [22], the closed loop transfer function from $\mathbf{w}$ to $\mathbf{z}$ is

given by the linear fractional transformation

$$\mathbf{z} = \mathbf{F}_l(\mathbf{P}, \mathbf{K})\mathbf{w} = \mathbf{N}\mathbf{w} \tag{4.4}$$

where

$$\mathbf{F}_l(\mathbf{P}, \mathbf{K}) = \mathbf{P}_{11} + \mathbf{P}_{12}\mathbf{K}(\mathbf{I} - \mathbf{P}_{22}\mathbf{K})^{-1}\mathbf{P}_{12} \tag{4.5}$$

As said previously, the goal of the H_2 and H_∞ problems is to minimize the H_2 and H_∞ norms of $\mathbf{F}_l(\mathbf{P}, \mathbf{K})$, respectively. The most general widely available and widely used algorithms for H_2 and H_∞ control problems are based on the state space solutions in [23] and are the ones used by the *H2syn* and *Hinfsyn* functions of MATLAB.

To specify the performance requirements, the generalized plant $\mathbf{P}$ must have weighting functions $\mathbf{W}_z$ and $\mathbf{W}_\mathbf{w}$. Thus, considering the exogenous inputs $\mathbf{w}$, being disturbances, references or noise and $\mathbf{z}$ the controlled outputs. The weighting matrices are chosen frequency dependent and selected such that weighted signals $\mathbf{w}$ and $\mathbf{z}$ are of magnitude 1 that is, such that the norm from $\mathbf{w}$ to $\mathbf{z}$ should be less than 1. Thus, in most cases only the magnitude of the weights matter and we choosing $\mathbf{W}_w(s)$ and $\mathbf{W}_z(s)$ stable and minimum phase.

In H_2 or H_∞ problems, the problem is formulated by the following specifications

$$\min_{\mathbf{K}} ||\mathbf{N}(k)||_{2\,\text{or}\,\infty}, \quad \mathbf{N} = \begin{bmatrix} \mathbf{W}_u\mathbf{K}\mathbf{S} \\ \mathbf{W}_T\mathbf{T} \\ \mathbf{W}_P\mathbf{S} \end{bmatrix} \tag{4.6}$$

with $\mathbf{S}$ the sensitivity function (for performance), $\mathbf{T}$ the complementary sensitivity function (for robustness and noise attenuation) and the loop transfer $\mathbf{KS}$ (to penalize large inputs) and $\mathbf{K}$ is a stabilizing controller. The corresponding block diagram is presented in 4.2. From it, the following equalities are derived

$$\mathbf{z}_1 = \mathbf{W}_\mathbf{u}\mathbf{u} \tag{4.7}$$

$$\mathbf{z}_2 = \mathbf{W}_T\mathbf{G}\mathbf{u} \tag{4.8}$$

$$\mathbf{z}_3 = \mathbf{W}_P\mathbf{w} + \mathbf{W}_P\mathbf{G}\mathbf{u} \tag{4.9}$$

$$\mathbf{v} = -\mathbf{w} - \mathbf{G}\mathbf{u} \tag{4.10}$$

and the generalized plant $\mathbf{P}$ is

$$\mathbf{P} = \begin{bmatrix} \mathbf{0} & \mathbf{W}_\mathbf{u}\mathbf{I} \\ \mathbf{0} & \mathbf{W}_T\mathbf{G} \\ \mathbf{W}_P\mathbf{I} & \mathbf{W}_P\mathbf{G} \\ -\mathbf{I} & -\mathbf{G} \end{bmatrix} \tag{4.11}$$

having this partition, one obtains

$$\mathbf{P}_{11} = \begin{bmatrix} 0 \\ 0 \\ \mathbf{W}_P\mathbf{I} \end{bmatrix}, \quad \mathbf{P}_{12} = \begin{bmatrix} \mathbf{W_u} \\ \mathbf{W}_T\mathbf{G} \\ \mathbf{W}_P\mathbf{G} \end{bmatrix}, \quad \mathbf{P}_{21} = -\mathbf{I}, \quad \mathbf{P}_{22} = -\mathbf{G} \tag{4.12}$$

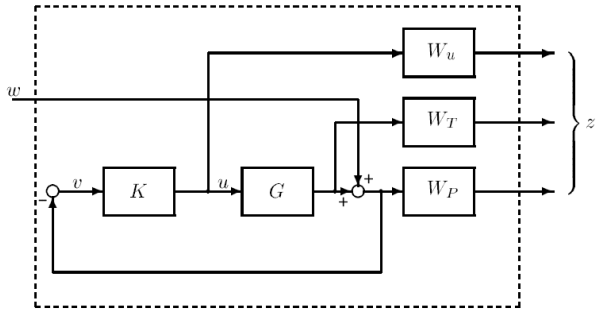


Figure 4.2: Block diagram of $\mathbf{z} = \mathbf{N}\mathbf{w}$ [22].

Lastly, to synthesize the H_2 and H_∞ controllers, the following conditions must be met [22]

(A1) $(\mathbf{A}, \mathbf{B}_2, \mathbf{C}_2)$ is stabilizable and detectable.

(A2) $\mathbf{D}_{11}$ and $\mathbf{D}_{21}$ have full rank.

(A3) $\begin{bmatrix} \mathbf{A} - j\omega\mathbf{I} & \mathbf{B}_2 \\ \mathbf{C}_1 & \mathbf{D}_{12} \end{bmatrix}$ has full column rank for all ω.

(A4) $\begin{bmatrix} \mathbf{A} - j\omega\mathbf{I} & \mathbf{B}_1 \\ \mathbf{C}_2 & \mathbf{D}_{21} \end{bmatrix}$ has full row rank for all ω.

(A5) $\mathbf{D}_{11} = 0$ and $\mathbf{D}_{22} = 0$.

Assumption $(\mathbf{A}_1)$ is required for the existence of stabilizing controllers $\mathbf{K}$ and assumption $(\mathbf{A}_2)$ is sufficient to ensure the controllers are proper and hence realizable. Assumptions $(\mathbf{A}_3)$ and $(\mathbf{A}_4)$ ensure that the optimal controller does not try to cancel poles or zeros on the imaginary axis which would result in closed loop instability. Assumption $(\mathbf{A}_5)$ is conventional in H_2 control. $\mathbf{D}_{11}$ makes $\mathbf{P}_{11}$ strictly proper. It can be shown that H_2 is the set of strictly proper stable transfer functions. $\mathbf{D}_{22} = 0$ makes $\mathbf{P}_{22}$ strictly proper and simplifies the formulae in the H_2 algorithms. In H_∞ neither $D_{11} = 0$ nor $D_{22} = 0$ is required but they do significantly simplify the algorithm formulae.

Sometimes more assumptions are made; however, these are the most important ones. The H_2 and H_∞ are going to be used to control the positions P_x and P_y, where the former has the following state space representation (the same as in 2.4.1).

$$A_x = \begin{array}{c} \\ \dot{p_x^B} \\ \dot{v_x^B} \\ \dot{\theta} \\ \dot{w_y} \end{array} \begin{array}{c} \begin{array}{cccc} p_x^B & v_x^B & \theta & w_y \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}, \quad B_x = \begin{array}{c} \\ \dot{p_x^B} \\ \dot{v_x^B} \\ \dot{\theta} \\ \dot{w_y} \end{array} \begin{array}{c} \epsilon \\ \begin{bmatrix} 0 \\ -g \\ 0 \\ \frac{-mgl}{J_{yy}} \end{bmatrix} \end{array}$$

The state space system comprises 4 poles at the origin, which means that at least conditions $\mathbf{A}_3$ and $\mathbf{A}_4$ are not met. To overcome this issue, the system can be shifted by an arbitrary value α and then shifted back to the original plant [23]. In this case, to achieve a fast performance, an α of 2 was chosen, so the poles move to $s = -2$.

Then, the plant is augmented with weights W_1, W_2 and W_3 [24], corresponding to $\mathbf{W}_P$ (except that MATLAB considers the error feedback and not the state), $\mathbf{W_u}$ and $\mathbf{W}_T$, respectively. W_1, as is the case for $\mathbf{W}_P$, should be high in the control bandwidth to ensure good reference tracking and disturbance rejection (obtaining small sensitivity S), whereas W_3 should be the opposite and is responsible for robustness (in the case of multiplicative disturbance) and noise attenuation (to achieve small complementary sensitivity T). In order to limit control effort in a particular band, W_2 should have a high magnitude in it to obtain small $\mathbf{KS}$.

Given that the weighting functions W_1,W_2 and W_3 shape N, in order to minimize its norm, these functions should be chosen accordingly, by following the aforementioned guidelines and with some tuning. In this case, the weighting functions are the same for the 3 algorithms, where W_1 was chosen setting a pole close to the desired bandwidth and a zero a decade faster, with unity static gain, namely

$$W_1 = \frac{0.1(s + 10w)}{s + w}, \tag{4.13}$$

with w the frequency of the pole so that the transfer function has a bandwidth close to its location. The following desired frequencies have been selected

$$\begin{aligned} w_{p_x} &= 1Hz = 6.28rad/s \\ w_{v_x} &= 10Hz = 62.8rad/s \\ w_\theta &= 30Hz = 188.5rad/s \\ w_{w_y} &= 70Hz = 439.8rad/s, \end{aligned} \tag{4.14}$$

with w_{p_x} to w_{w_y} the desired frequencies, leading to the transfer functions

$$\mathbf{W}_1 = \begin{bmatrix} \frac{0.1(s+62.8)}{s+6.28} \\ \frac{0.1(s+628)}{s+62.8} \\ \frac{0.1(s+1885)}{s+188.5} \\ \frac{0.1(s+4398)}{s+439.8} \end{bmatrix}. \tag{4.15}$$

The weighting function $\mathbf{W}_3$ is the inverse of $\mathbf{W}_1$ to ensure robustness,

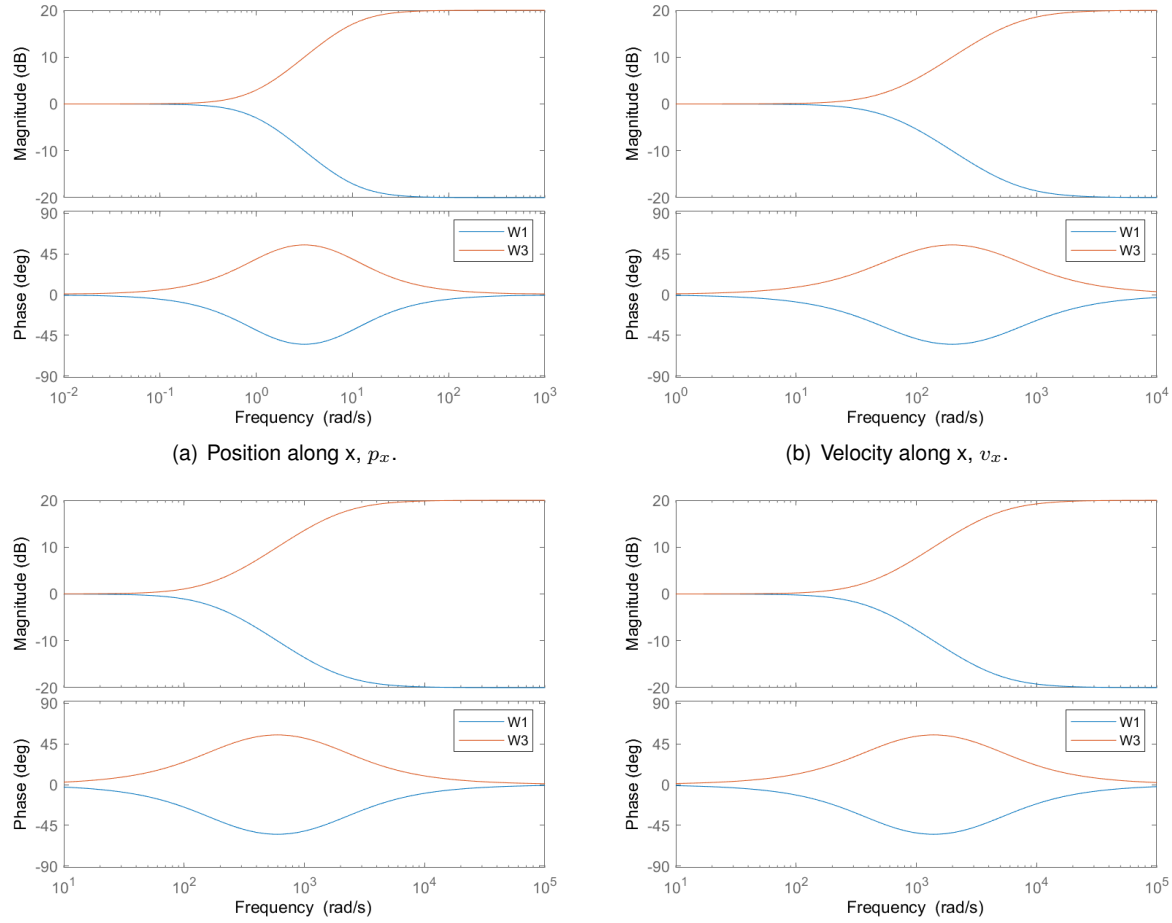
$$\mathbf{W}_3 = \begin{bmatrix} \frac{10(s+6.28)}{s+62.8} \\ \frac{10(s+62.8)}{s+628} \\ \frac{10(s+188.5)}{s+1885} \\ \frac{10(s+439.8)}{s+4398} \end{bmatrix}. \tag{4.16}$$

And finally,

$$W_2 = 1/(17 \times pi/180), \tag{4.17}$$

sets actuation cost.

The $\mathbf{W}_1$ and $\mathbf{W}_3$ weighting functions can be visualized in figure 4.3.



(a) Position along x, p_x.

(b) Velocity along x, v_x.

(c) Pitch, θ.

(d) Angular velocity around y, w_y.

Figure 4.3: Weighting functions for the different outputs.

4.1 H_2 synthesis

The goal of the H_2 control problem is to find a stabilizing controller that minimizes [22]

$$||\mathbf{F}(s)|| = \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty} \mathbf{F}(jw)\mathbf{F}(jw)^T dw; \quad \mathbf{F} = \mathbf{F}_l(\mathbf{P}, \mathbf{K}) = \mathbf{N}} \tag{4.18}$$

where P includes the plant model, the interconnection structure and the weighting functions. It can be proven that by minimizing the H_2 norm, we are minimizing the root mean square rms value of z.

With the above formulation, a step response produced the results available in figure 4.4. In figure 4.5 the bode plot of the closed loop system is shown, displaying a closed loop bandwidth of 1.95 rad/s. The response proves to be faster than the linear controllers, but still slower than the NMPC, which is still

an interesting result given that it is possible to quantify robustness contrary to the NMPC. Moreover, the H_∞ controller does not require full state feedback.

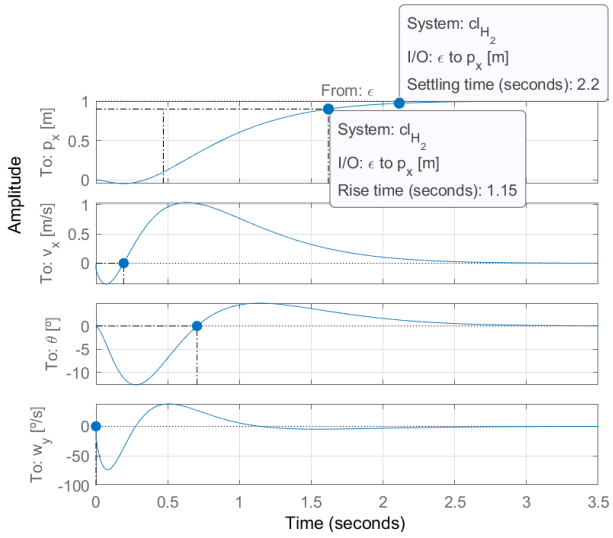


Figure 4.4: Step response of the closed loop system with H_2 controller

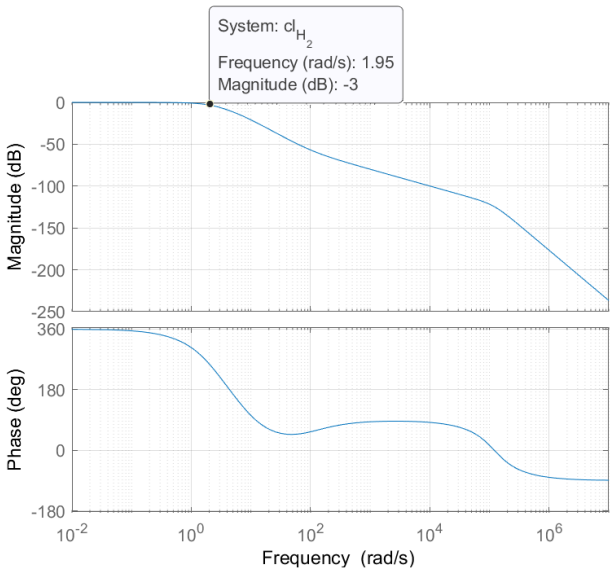


Figure 4.5: Bode plot of the closed loop system with the H_2 controller.

Lasty, the classical and disk gain and phase margins (using the functions *allmargin* and *diskmargin* of MATLAB) of the system with the H_2 controller are lower than the LQR, which is expected given that the response is much faster.

$$\begin{aligned} GM_{H_2} &= 11.0dB \\ PM_{H_2} &= 78.4^\circ \\ DM_{H_2} &= 0.20 \end{aligned} \tag{4.19}$$

4.2 H_∞ synthesis

First the singular values σ must be defined. In short, the eigenvalue analysis for MIMO systems offer some limitations, namely they can only be computed for square matrices and that they only measure the gain for the special case when the inputs and outputs have the same direction (in the eigenvectors). This proves to be only useful for stability analysis, but not performance [22]. To obtain the singular values, the singular value decomposition is usually performed, resulting in

$$\mathbf{G} = \mathbf{U} \sum \mathbf{V}^T \tag{4.20}$$

$\sum$ is an $l \times m$ matrix with $k = \min l, m$ non-negative singular values, σ_i arranged in descending order along its main diagonal; the other entries are zero. The singular values are the square roots of the eigenvalues of $\mathbf{G}^H\mathbf{G}$ where $\mathbf{G}^H$ is the complex conjugate transpose of $\mathbf{G}$.

$$\sigma_i(\mathbf{G}) = \sqrt{\lambda_\mathbf{i}(\mathbf{G}^\mathbf{H}\mathbf{G})} \tag{4.21}$$

$\mathbf{U}$ is an $l \times l$ unitary matrix of output singular values $\mathbf{u}_i$ and $\mathbf{V}$ is an $m \times m$ unitary matrix of input singular values $\mathbf{v}_i$ [22].

From the singular values it is possible to find its maximum (and minimum), defining the largest gain for any input direction, the peak of the singular value $\bar{\sigma}$. Thus, the H_∞ problem is very similar to the H_2 one, but the function to minimize is

$$||\mathbf{F}_l(\mathbf{P}, \mathbf{K})||_\infty = \min_\omega \bar{\sigma}(\mathbf{F}_l(\mathbf{P}, \mathbf{K})(j\omega)). \tag{4.22}$$

In addition to this, for all signals $\boldsymbol{w}$ with finite H_2 norm, the H_∞ norm corresponds to the maximum gain from $\boldsymbol{w}$ to $\boldsymbol{z}$, considering the L_2 norm (this is the L_2-induced norm)

$$||\mathbf{F}_l(\mathbf{P}, \mathbf{K})||_\infty = \sup_{\|\boldsymbol{w}\|_2 < \infty} \frac{\|\boldsymbol{z}\|_2}{\|\boldsymbol{w}\|_2}. \tag{4.23}$$

The *Hinfsyn* function of MATLAB was used to find the controller that minimizes the peak singular value of $\mathbf{F}_l(\mathbf{P}(j\omega), \mathbf{K}(j\omega))$[25].

With these specifications, the step response of the closed loop system can be seen in figure 4.6. Similar to the H_2 controller, the response is fast and smooth, although the pitch angle θ reaches around $-15°$, which may be too much. In figure 4.7 the bode plot of the closed loop system is shown, displaying a closed loop bandwidth of 1.81 rad/s.

The results of the H_∞ seem to be the most promising, when compared to the previous controller and the μ synthesis, considering that it has interesting time and frequency responses.

Finally, the gain and phase margins (using the *diskmargin* function of MATLAB) of the system with

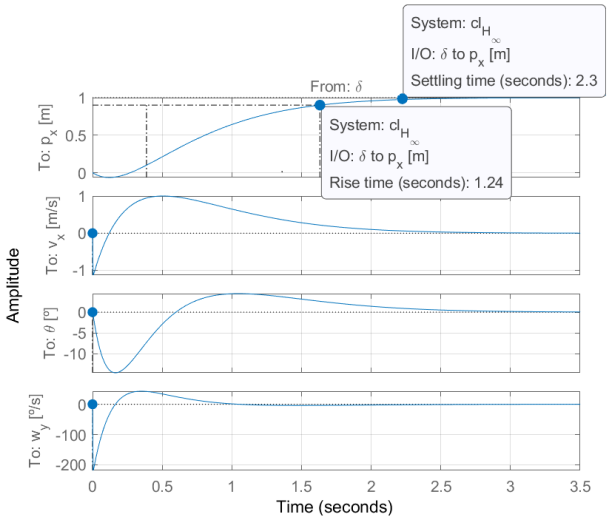


Figure 4.6: Step response of the closed loop system with the H_∞ controller.

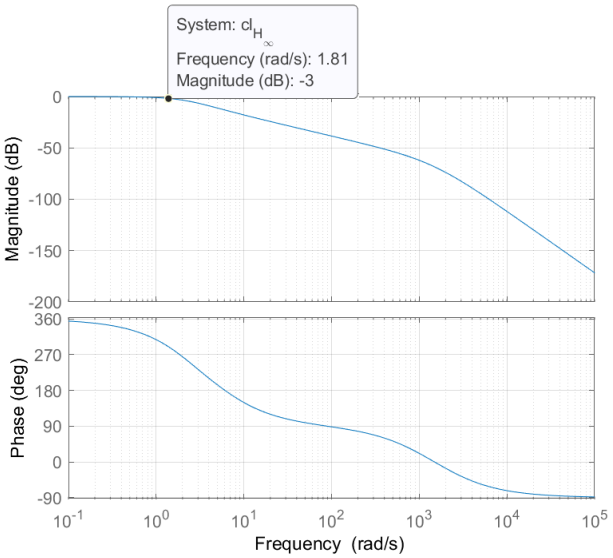


Figure 4.7: Bode plot of the closed loop system with the H_∞ controller.

the H_∞ controller are similar to the H_2 controller.

$$\begin{aligned} GM_{H_\infty} &= 12.2dB \\ PM_{H_\infty} &= 75.9^\circ \\ DM_{H_\infty} &= 0.22 \end{aligned} \tag{4.24}$$

4.3 μ synthesis

The last robust controller to be designed is the μ synthesis controller. The biggest difference to the H_2 and H_∞ techniques is that the uncertainty in the model can be specified, as well as the desired gain and phase margins. By analysing the state space matrices A_x and B_x, one can infer that the last element of the control matrix B_x has the biggest expected variation throughout the flights of the VIRIATO Test Platform. The mass is expect to vary around 20% due to fuel usage, which adding a 10% margin, means that the controller should be robust to the following expected variation

$$-0.7\frac{mgl}{J_{yy}} <= -\frac{mgl}{J_{yy}} <= -1.3\frac{mgl}{J_{yy}}. \tag{4.25}$$

The structured singular value μ is central to the μ synthesizes and is a function that provides a generalization of the singular value σ and the spectral radius ρ. The name structured singular value is used because it generalizes the singular value RS-condition, $\bar{\sigma}(M) <= 1 \quad \forall\omega$, with M the nominal system. To define μ, consider the stability of the $M\Delta - structure$ in figure 4.8 for the case where Δ is a set of norm bounded block diagonal perturbations. From the determinant stability condition, which applies to both real and complex perturbations, one arrives at

$$det(\mathbf{I} - \mathbf{M}\Delta(j\omega)) \neq 0, \quad \forall\omega, \forall\Delta, \bar{\sigma}(\Delta(j\omega)) <= 1 \quad \forall\omega \tag{4.26}$$

As this condition only accepts a qualitative answer, a factor k_m is defined, which quantifies the robustness and stability of the system and the uncertainty Δ is scaled by k_m. The goal is to find the smallest k_m which makes the matrix $\mathbf{I} - k_m\mathbf{M}\Delta$ singular,

$$det(\mathbf{I} - k_m\mathbf{M}\Delta) = 0 \tag{4.27}$$

and the structured singular value, μ is defined as $1/k_m$.

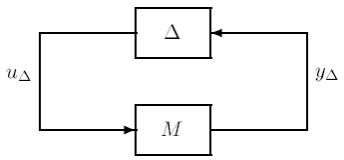


Figure 4.8: $\mathbf{M}\Delta - structure$ for robust stability analysis [22].

To compute the controller, the function *musyn* of MATLAB uses an iterative process called *D-K iteration*, which combines H_∞ synthesis and μ analysis and often yields good results [22]. In short the

process is [26]

(1) Uses H_∞ synthesis to find a controller that minimizes the closed-loop gain of the nominal system.

(2) Performs a robustness analysis to estimate the robust H_∞ performance of the closed-loop system, which is typically larger than the peak value of the structured singular value $\bar{\mu}$. This amount is expressed as a scaled H_∞ norm involving dynamic scalings called the D and G scalings (the D step).

(3) Finds a new controller to minimize the scaled H_∞ norm obtained in step 2 (the K step).

(4) Repeats steps 2 and 3 until the robust performance stops improving.

Having formulated the problem, the μ controller was synthesized with the *musyn* function of MATLAB and produced the closed loop step and bode plots in figures 4.9 and 4.10. The response is slower than the H_2 and H_∞ controllers and similar to the LQG. Furthermore, the closed loop system is robust to variations of up to 30% of the expression in equation 4.25 and has the worst case gain and disk margins in the complete range of mass variation (calculated using the function *wcdiskmargin* of MATLAB [27]) shown in equation 4.28.

The μ synthesis controller is a an interesting example of why it is important to look at disk margins. Although the classical gain and phase margins are bigger than the other robust controllers, the disk margin is significantly lower. The response is not as good as expected, and the controller tuning was challenging, so until better tuning methods appear, the H_2 or H_∞ controllers are more suitable. In fact, some papers such as [28] came up with heuristics to find the desired weighting functions to synthesize the robust controllers; however, when this was experimented, the results were not very satisfactory.

$$\begin{aligned} GM_\mu &= 21.2dB \\ PM_\mu &= 72.5^\circ \\ DM_\mu &= 0.16 \end{aligned} \tag{4.28}$$

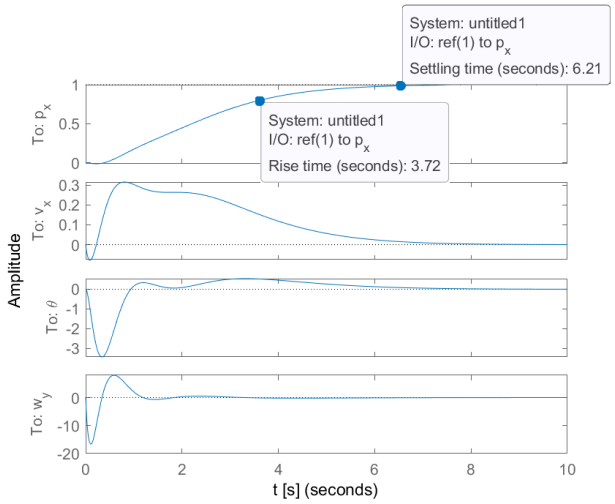


Figure 4.9: Step response of the closed loop system with the μ controller.

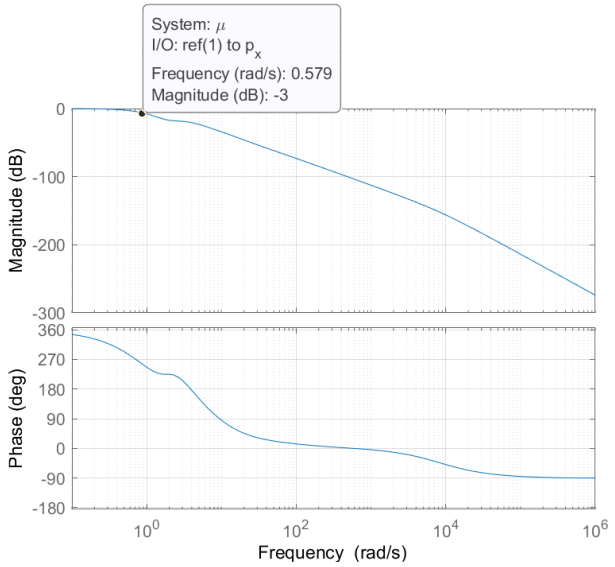


Figure 4.10: Bode plot of the closed loop system with the μ controller.

Chapter 5

Nonlinear Control and Estimation

In this section a nonlinear model predictive controller NMPC and an extended Kalman filter EKF are derived.

5.1 Nonlinear Model Predictive Controller

The NMPC works similarly to the MPC, with the exception that the model is nonlinear. Due to its ability to deal with nonlinearities, the model used no longer has the position and velocity states in the body frame, but in the inertial one. Thanks to this, no transformation is needed to give position references nor to visualize the resulting coordinates in the inertial frame. As such, the dynamics are

$$\frac{d}{dt}\begin{bmatrix}\mathbf{p}^I\\ \mathbf{v}^I\\ \lambda\\ \mathbf{\Omega}\\ \delta_m\end{bmatrix} = \begin{bmatrix}\mathbf{v}^I\\ R_B^I\frac{\mathbf{F_B}}{m+\delta_m}+ge_z\\ Q(\lambda)\mathbf{\Omega}\\ J^{-1}(\tau_\mathbf{B}-\mathbf{\Omega}\times J\mathbf{\Omega})\\ \mathbf{0}_{1\times n}\end{bmatrix} \tag{5.1}$$

and recalling that the forces in the body frame $\mathbf{F_B}$ are

$$\mathbf{F}_B = -R_\delta^B R_\epsilon^B T\mathbf{e}_z = \begin{bmatrix}-T\sin\epsilon\\ T\cos\epsilon\sin\delta\\ -T\cos\epsilon\cos\delta\end{bmatrix} \tag{5.2}$$

and the torque in the body frame τ_B

$$\tau_B = l\mathbf{e}_z\times\mathbf{T}+\tau_r\mathbf{e}_z = \begin{bmatrix}-T_m l\cos\epsilon\sin\delta\\ -T_m l\sin\epsilon\\ \tau_r\end{bmatrix} \tag{5.3}$$

The mass of the vehicle is added as a state, updating it at each time step with the EKF. The dynamical equation of the mass is $\frac{d\delta_m}{dt}=0$, assuming that in the prediction horizon of the NMPC the mass does

not change (which is a reasonable approximation assuming a short time of a few seconds).

The chosen prediction and control horizons are

$$\begin{aligned} P_h &= 2s \\ C_h &= 2s \end{aligned} \tag{5.4}$$

accounting for 20 time steps, as the sample time is

$$h_s = 0.1s \tag{5.5}$$

The cost function is similar to the linear mpc

$$J = \sum_{k=1}^{P_h} \mathbf{z}(k)^T Q \mathbf{z}(k) + \mathbf{u}(k)^T R \mathbf{u}(k) \tag{5.6}$$

The constraints are

$$\begin{aligned} -17^\circ \ <= \delta <= 17^\circ \\ -17^\circ \ <= \epsilon <= 17^\circ \\ 0N <= T <= 220N \\ -1.1Nm <= \tau_r <= 1.1Nm \end{aligned} \tag{5.7}$$

To formulate the optimization problem, 2 common approaches are available, namely single or multiple shooting. In single shooting, the approach is more intuitive, as the decision variables are only the inputs. In the case of multiple shooting, at each time step in the prediction horizon the next state has to equal the state function, having as arguments the current state and inputs. This leads to the addition of the following constraint for each time step of the prediction horizon

$$\sum_{k=1}^{P_h} \mathbf{x}(k+1) - f(\mathbf{x}(k), \mathbf{u}(k)) = 0 \tag{5.8}$$

and the solver tries not only to find the optimal inputs but also the states. The dimensionality of the problem increases; however it is sparse and more linear, reducing the calculation speed [29].

Lastly, the chosen solver was CasADi [30] using an interior point method [31], which proved to be significantly faster than the solver of MATLAB.

Setting the following reference

$$\begin{aligned} P_{x_{ref}} &= 1 \\ P_{y_{ref}} &= 1 \\ P_{z_{ref}} &= 1 \\ \psi_{ref} &= 150^\circ \end{aligned} \tag{5.9}$$

The response can be seen in figure 5.1. Notice the static error of the Down P_z coordinate due to a mass error introduced of 2.5 Kg. To mitigate the effect of mass error on the NMPC, an extended Kalman filter EKF with the ability to estimate parameters such as the mass will be derived in the following

section. Regardless, the performance is the best so far, even with the mass error. The calculated inputs are shown in figure 5.2.

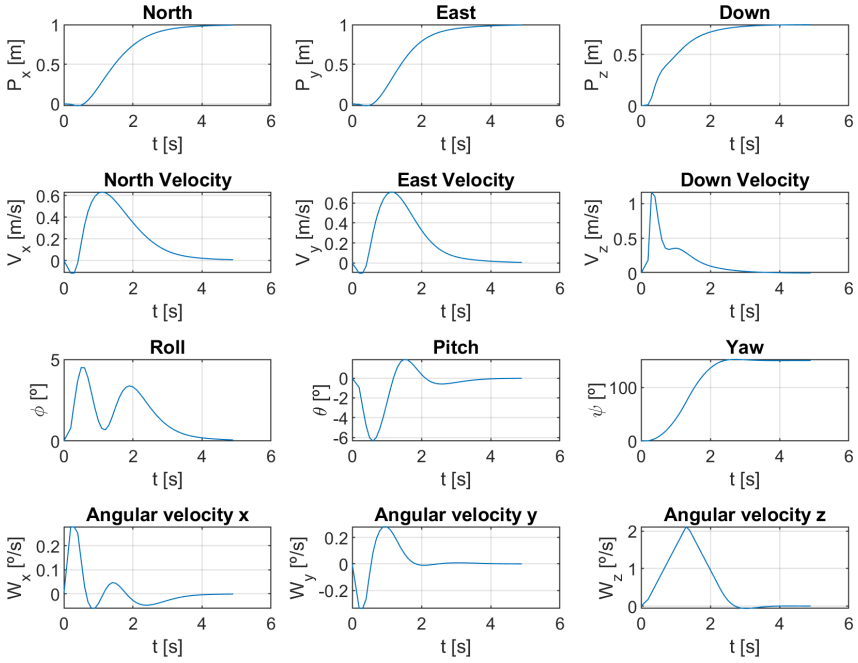


Figure 5.1: Step plot with NMPC

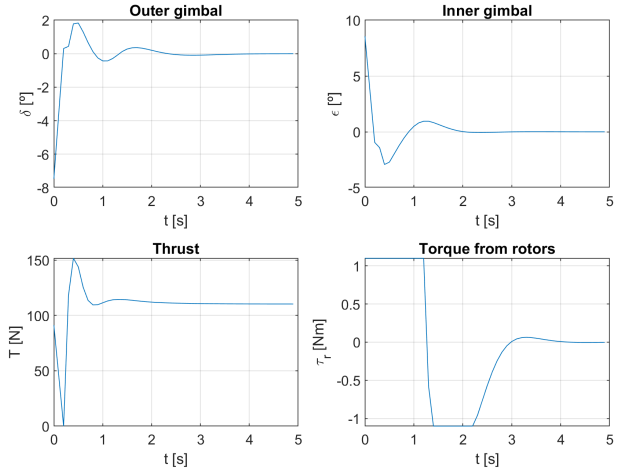


Figure 5.2: Inputs calculated by the NMPC

5.2 Extended Kalman Filter

In this section the derivation of the EKF extended Kalman filter is made. The objective is to estimate the mass of the VIRIATO Test Platform to use in the NMPC model.

The algorithm can be summarized in 2 steps, prediction and correction, as is summarized in equations 5.10 to 5.15.

Prediction of state (5.1), covariance and output (the same as the state, except for the mass variation δ_m) estimates

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}(k)) \tag{5.10}$$

$$\mathbf{P}_{k|k-1} = \mathbf{\Phi}_k \mathbf{P}_{k-1|k-1} \mathbf{\Phi}_k^T + \mathbf{Q}_k \tag{5.11}$$

$$\mathbf{y}_k = h(\hat{\mathbf{x}}_{k|k-1}) \tag{5.12}$$

gain computation

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \left[\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \right]^{-1} \tag{5.13}$$

correction of state and covariance estimates

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k (\mathbf{z}_k - \mathbf{y}_k) \tag{5.14}$$

$$\mathbf{P}_{k|k} = (\boldsymbol{I}_n - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} (\boldsymbol{I}_n - \mathbf{K}_k \mathbf{H}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \tag{5.15}$$

In addition to this, the matrix $\mathbf{Q}_k$ is obtained from [32]

$$\mathbf{Q}_k = \mathbf{\Phi}_k^T \mathbf{Q}_u \mathbf{\Phi}_k \tag{5.16}$$

and $\mathbf{\Phi}_k$ results from the discretization by matrix exponentation of the continuous time matrix A

$$\mathbf{\Phi_k} = e^{A h_s} \tag{5.17}$$

which can be calculated using the *expm* function of MATLAB. The sample time is

$$h_s = 0.1s \tag{5.18}$$

The nonlinear model $f(\mathbf{x}_k, \mathbf{u}_k)$ used in the EKF is in 5.1 and its jacobian $A(\mathbf{x}, \mathbf{u})$, which was obtained symbolically in MATLAB is

$$A(\mathbf{x}, \mathbf{u}) = \frac{f(\mathbf{x}_k, \mathbf{u}_k)}{d\mathbf{x}} \tag{5.19}$$

The observations noise covariance matrix is typically

$$\mathbf{R}_k = \begin{bmatrix} \sigma_1^2 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \sigma_m^2 \end{bmatrix} \tag{5.20}$$

with σ_1 to σ_m the standard deviation of the sensors and m is the number of measurements. In this case, the sensors are considered to produce clean measurements, so the $\mathbf{R}_k$ is chosen very low, such as

10^{-5}, representing the significant trust in the sensors. If the sensors prove to be noisy, inaccurate or biased, the observations noise covariance matrix $\mathbf{R}_k$ should be increased accordingly.

The noise covariance matrix $\mathbf{Q}_k$ represents the trust of the designer in the prediction model. It is computed from $\mathbf{Q}_u$

$$\mathbf{Q}_u = \begin{bmatrix} q_{11} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & q_{nn} \end{bmatrix} \tag{5.21}$$

with $\mathbf{Q}_{11}$ to $\mathbf{Q}_{nn}$ chosen according to the reliance on the prediction model and n the number of states. In the present work, the entry of $\mathbf{Q}_u$ corresponding to δ_m will be set significantly higher than the others (fifty (50) as opposed to 0) to take into account that this parameter has a higher uncertainty.

The observation function $h(\hat{\mathbf{x}})$ receives as argument the current state and returns the outputs. Assuming that the measurements correspond to the states, $h(\hat{\mathbf{x}})$ is

$$h(\hat{\mathbf{x}}_k) = \begin{bmatrix} x_i & & \mathbf{0} & 0 \\ & \ddots & & \mathbf{0} \\ \mathbf{0} & & x_{n-1} & 0 \end{bmatrix} \tag{5.22}$$

where the last column is full of zeros because δ_m is not being measured, having a total of rows of the number of states n minus 1.

The observations matrix of the extended Kalman filter is [32]

$$\mathbf{H}_k = \left[\frac{dh_i(\mathbf{x}_k)}{dx_j}\right]_{(m\times n)} = \begin{bmatrix} 1 & & \mathbf{0} & 0 \\ & \ddots & & \mathbf{0} \\ \mathbf{0} & & 1 & 0 \end{bmatrix}. \tag{5.23}$$

5.3 Formulation without velocity measurements

In this thesis, it is assumed that there are no velocity measurements available and with this slight modification, the matrices chosen are, starting with the cost Q,

$$\mathbf{Q}_u = \begin{bmatrix} \mathbf{I}_{12\times 12} & \mathbf{0}_{12\times 1} \\ \mathbf{0}_{1\times 12} & 200 \end{bmatrix} \tag{5.24}$$

the last element of the previous matrix is significantly higher because it was found empirically after several attempts and it is intuitive to attribute less trust to the mass variation equation (it is assumed that the mass does not vary in a sample which is not true, it is slowly decreasing). Perhaps by measuring the current mass consumption it would be possible to include it in this equation, leading to much better estimates.

The number of measurements available are 9, positions, Euler angles and angular velocities, so the size of the R matrix is 9×9, and its entries were chosen as

$$\mathbf{R} = \begin{bmatrix} 0.01\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{2\times2} & 0.0035^2\mathbf{I}_{2\times2} & \mathbf{0}_{2\times5} \\ \mathbf{0}_{1\times3} & 0.0175^2 & \mathbf{0}_{1\times5} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & 0.0012^2\mathbf{I}_{3\times3} \end{bmatrix} \tag{5.25}$$

where the non null entries for the Euler angles and angular velocity states (rows 3 to 9) were specified as the variance of the sensors and the position values by trial and error (the position already had the pre filtering defined in section 2.

The measurement matrix $\mathbf{H}_k$ is

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} \end{bmatrix} \tag{5.26}$$

which selects the measurements being taken from the available 13 states, in this case, positions, Euler angles and angular velocities.

5.4 Nonlinear MPC and EKF

Having formulated the EKF, it can be coupled together with the NMPC, feeding it the states and the mass error estimate, forming a NMPCEKF. The simulation results are found in 5.3 and the inputs as well as the estimated mass error in 5.4. Due to the exact estimation of δ_m by the EKF, the static error no longer exists.

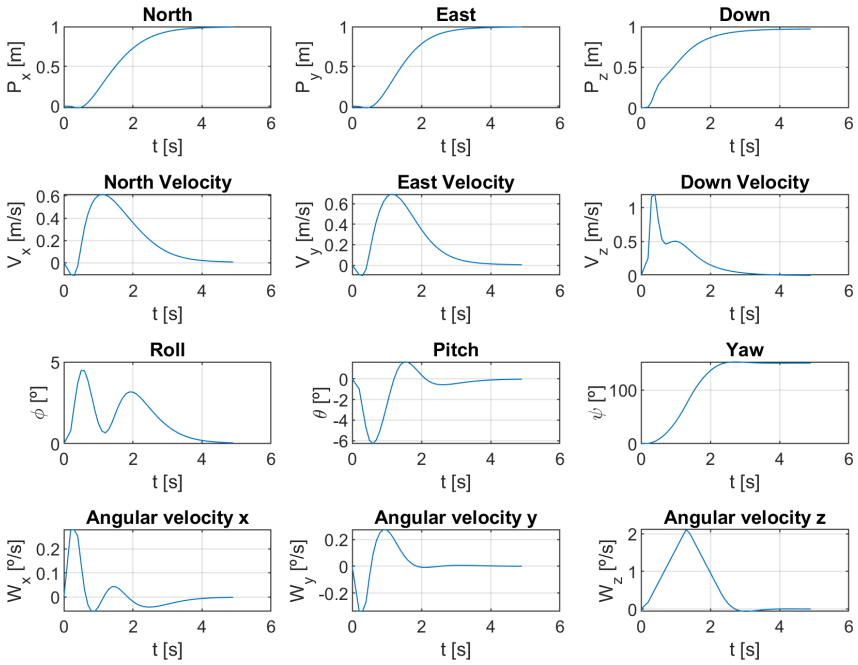


Figure 5.3: Step plot with NMPC with updated mass estimate.

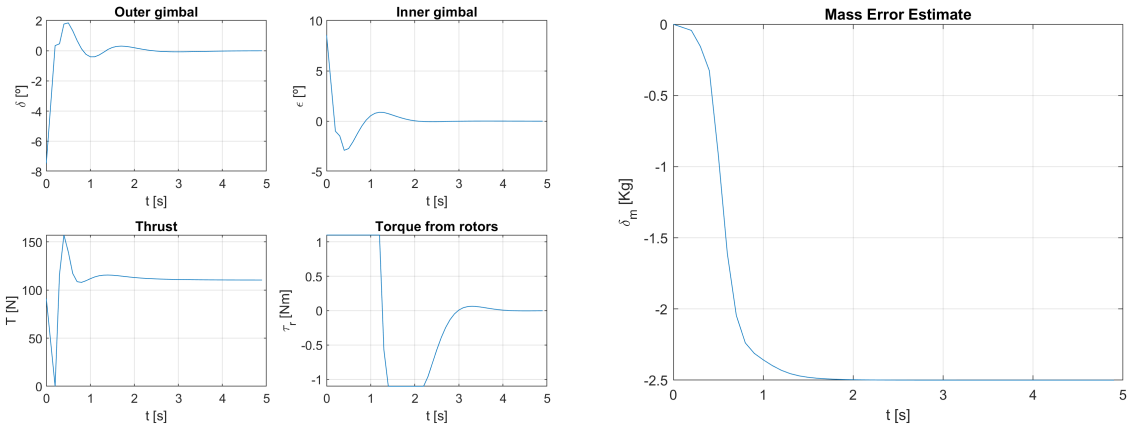


Figure 5.4: Inputs calculated by the NMPC and mass error estimate provived by the EKF

Chapter 6

Nonlinear Simulations

In this chapter, the most promising controllers, one from each category: linear (LQG), nonlinear (NMPCEKF) and robust (H_∞) are tested against mass variation, actuation delay and wind gusts (perturbations). In the case of the mass variation and actuation, the controller is asked to do a complex trajectory, combining a step input of 1 in each position axis (north, east and down) and a yaw reference of 150º, whereas to test it for wind gusts, it is asked to stand still.

All the controllers received filtered position measurements using the Kalman filter derived in section 2. The LQG controller receives these position and velocity estimates and the raw Euler angles and angular velocities. Given that it was troublesome enough to tune the H_∞ controller for control performance, instead of tuning it also to deal with noisy inputs, it was decided to just use the linear Kalman filter of the LQG. The NMPCEKF also receives the pre filtered position and velocity estimates, the raw Euler angles and angular velocity signals. In addition to this, a low gain integrator was added to the height loop of the LQG and H_∞ controllers.

The nominal case for these results is defined as starting with an initial mass of 13.75, a mass variation of -0.008 Kg/s, no wind and no delay. In addition to these, the controllers are given step references of 1 meter in each position axis (north, east, down) and a yaw of 150º.

6.1 Estimation results

The results of the estimation for the linear Kalman filter of the LQG and the extended Kalman filter of the NMPCEKF are presented. The simulation is ran in the nominal case.

6.1.1 Linear Kalman filter

The results of the estimation can be visualized in figure 6.1. The errors of the position and velocity estimates are low, showing a good performance; however, the Euler angles and angular velocities have significant errors. This is mostly due to the fact that when the yaw is varying, the Euler angles and the angular velocities are coupled (2.10) which leads to model mismatch. To prevent this issue, the Q matrix

of the filter could be increased to represent the lack of trust in the model, but this would in turn increase noise.

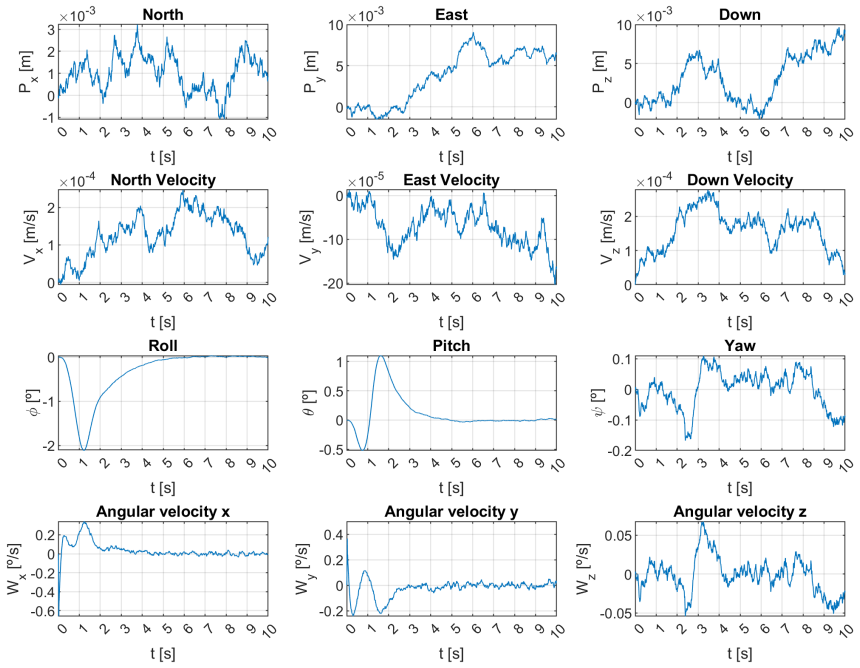


Figure 6.1: Estimation errors of the states of the VIRIATO Test Platform using the linear Kalman filter of the LQG controller.

6.1.2 Extended Kalman filter

The results of the estimation can be visualized in figure 6.3. This time, the errors are low for all states, showing an excellent performance, which is attributed to the ability of the EKF to include the coupled nonlinear dynamics reducing model mismatch. In addition to this, this filter is discrete, contrary to the continuous time linear Kalman filter presented previously, so it would also be ready for implementation.

In figure 6.2 the mass estimate for a longer simulation time is shown. As can be verified, the estimator is able to precisely follow the mass variation, with an average mass error of 0.02 Kg.

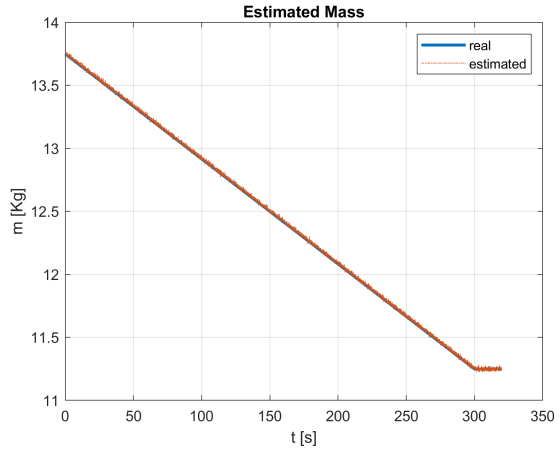


Figure 6.2: Estimation of the mass of the VIRIATO Test Platform using the extended Kalman filter of the NMPCEKF controller.

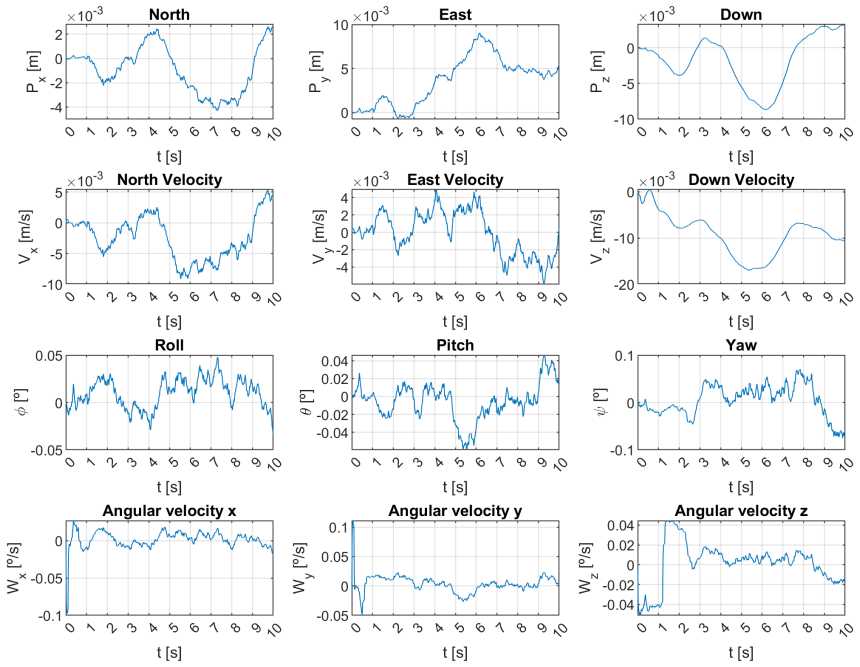


Figure 6.3: Estimation errors of the states of the VIRIATO Test Platform using the extended Kalman filter of the NMPCEKF controller.

6.2 Mass variation

In this section the controllers are compared against a mass variation of $-2.5Kg$ or 20%, which is the available fuel, with the nominal references (again, step references of position of 1 meter in all 3 axis and a yaw of 150º), no wind and no added actuation delay. Each of the following 3 subsections presents the state errors and the inputs calculated by the controllers. The integrator of the height controller PID is assumed to have settled in the correct mass variation of 20%.

6.2.1 LQG

The LQG results can be visualized in figures 6.4 and 6.5. The controller is able to track the references reasonably, having clearly more difficulty in the horizontal position, which is expected because it has the highest order (4 integrations from gimbals to position). Nevertheless, the yaw PD and the height PID controllers are able to track the references with a settling time of around 3 to 4 seconds, which proves that these controllers are suitable enough for these states.

Although the tracking performance is satisfactory, the outer and inner gimbals have very fast dynamics, forcing the gimbals to move abruptly, which could damage them in the long term or lead to instability. In the case of the thrust and the torque from the rotors the actuation inputs are slower and smoother, proving that the PD and PID controllers are competent to control yaw and height, respectively.

Considering that the behaviour of the VIRIATO Test Platform with the LQG controller proves to be similar with or without significant mass changes, it is proven that the controller is able to deal with the expected mass variation. In fact, the limitation of this controller is that it fails to deal with a fast changing

yaw, as can be seen in the North state, where it has a significant overshoot. This can also be attributed to the estimation error of the linear Kalman filters while the yaw is varying.

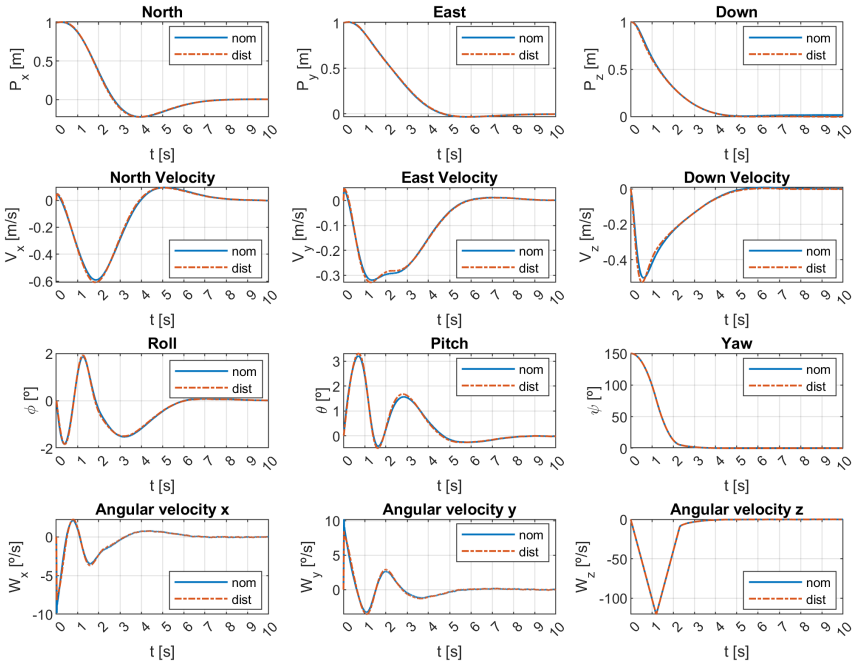


Figure 6.4: State errors of the VIRIATO Test Platform using the LQG controller. In the nominal *nom* and disturbed *dist* cases, $m = 13.8\,Kg$, $\dot{m} = -0.008\,Kg/s$ and $m = 11.3\,Kg$, $\dot{m} = 0\,Kg/s$, respectively.

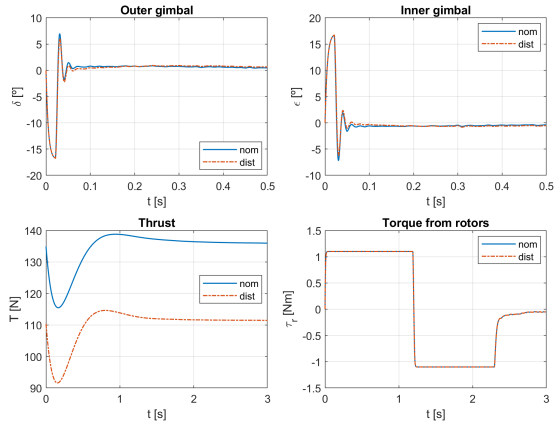


Figure 6.5: Inputs of the VIRIATO Test Platform using the LQG controller. In the nominal *nom* and disturbed *dist* cases, $m = 13.8\,Kg$, $\dot{m} = -0.008\,Kg/s$ and $m = 11.3\,Kg$, $\dot{m} = 0\,Kg/s$, respectively.

6.2.2 NMPCEKF

The NMPCEKF results can be visualized in figures 6.6 and 6.7. The performance of the controller is very promising, being able to deal with the complex step references smoothly. Notice that, contrary to the LQG, the North state does not have an overshoot because the NMPCEKF is able to predict the effect of the fast changing yaw and correct accordingly.

The provided inputs of this controller, namely the gimbals, have a much smoother trajectory, leading to less abrupt movements and to increase the lifetime of the inputs.

Thus, it is proved that the NMPCEKF can deal with the expected mass variation and also any complex reference trajectories.

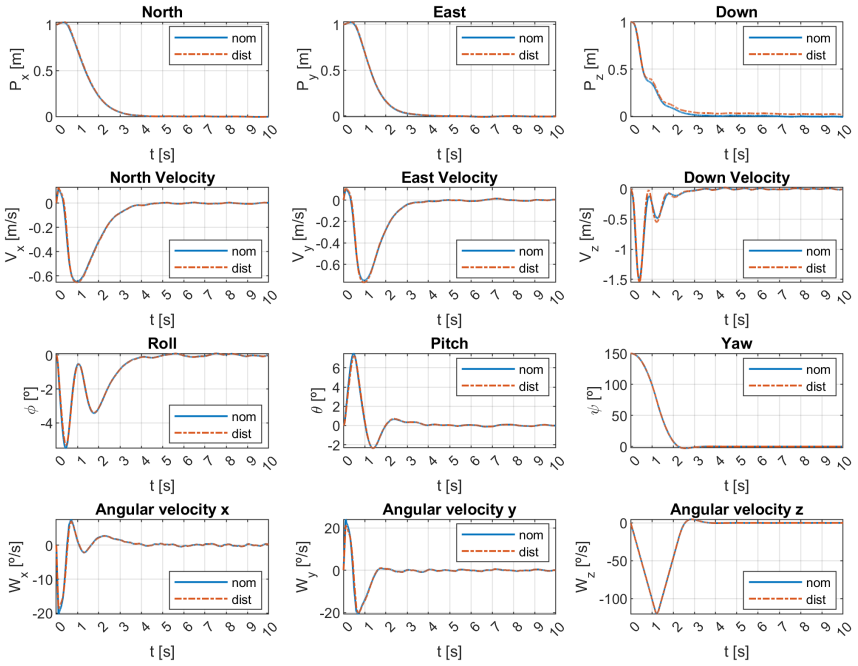


Figure 6.6: State errors of the VIRIATO Test Platform using the NMPCEKF controller. In the nominal *nom* and disturbed *dist* cases, $m = 13.8\,Kg$, $\dot{m} = -0.008\,Kg/s$ and $m = 11.3\,Kg$, $\dot{m} = 0\,Kg/s$, respectively.

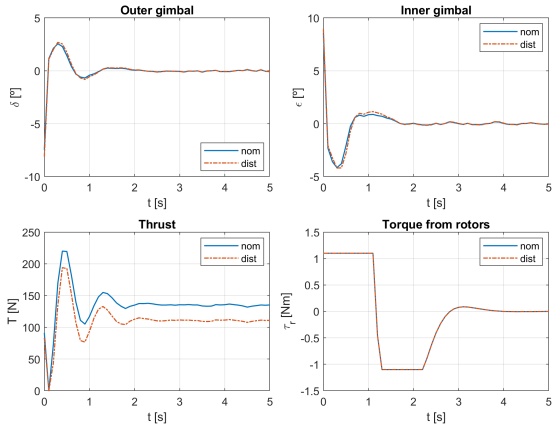


Figure 6.7: Inputs of the VIRIATO Test Platform using the NMPCEKF controller. In the nominal *nom* and disturbed *dist* cases, $m = 13.8\,Kg$, $\dot{m} = -0.008\,Kg/s$ and $m = 11.3\,Kg$, $\dot{m} = 0\,Kg/s$, respectively.

6.2.3 H_∞

The H_∞ results can be visualized in figures 6.8 and 6.9. As can be seen, the controller is faster than the LQG and as fast as the NMPCEKF. Again, it has some overshoot in the North state due to the varying yaw, and could also be attributed to the linear Kalman filter estimation error.

Contrary to the LQG and similarly to the NMPCEKF, the gimbals have acceptable, slow dynamics; however, the asked gimbal angles differ the most among the 3 controllers between the nominal case in

blue and the red case with less 2.5 Kg of mass. This is mostly due to the fact that the tuning of this controller was aggressive and as such the mass variation has a considerable impact in the dynamics.

The H_∞ controller is able to track the references reasonably while under a mass variation of 2.5 Kg, although the LQG and NMPCEKF controllers appear to be more robust to a variation of mass.

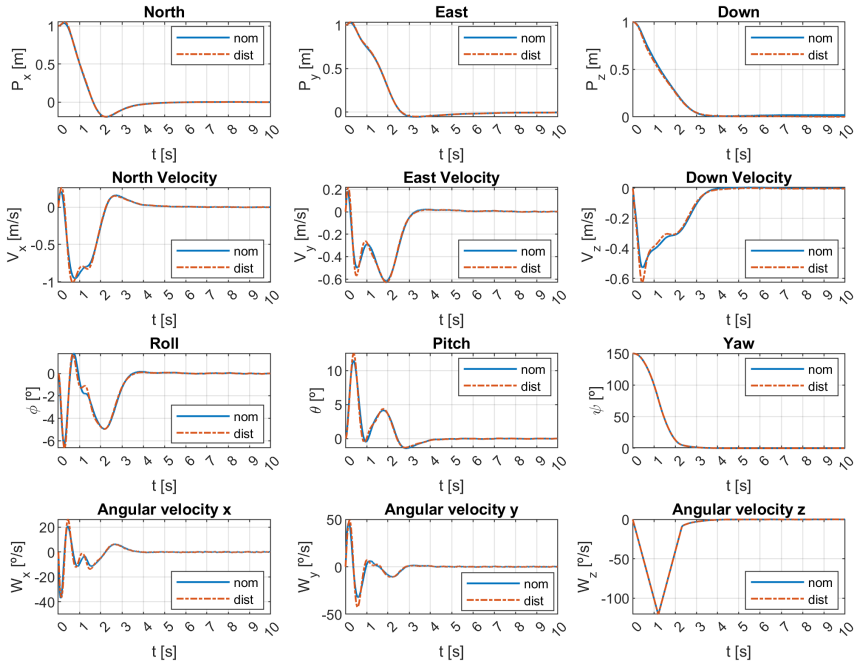


Figure 6.8: State errors of the VIRIATO Test Platform using the H_∞ controller. In the nominal *nom* and disturbed *dist* cases, $m = 13.8\,Kg$, $\dot{m} = -0.008\,Kg/s$ and $m = 11.3\,Kg$, $\dot{m} = 0\,Kg/s$, respectively.

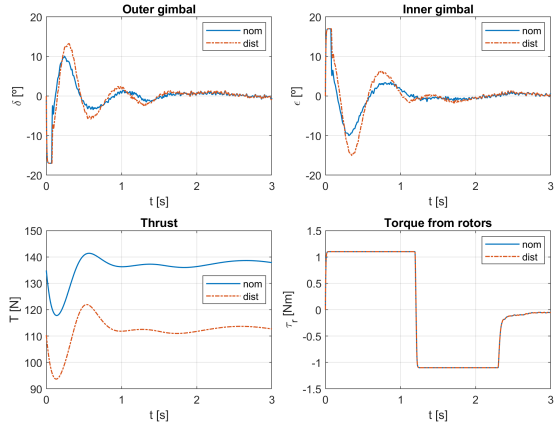


Figure 6.9: Inputs of the VIRIATO Test Platform using the H_∞ controller. In the nominal *nom* and disturbed *dist* cases, $m = 13.8\,Kg$, $\dot{m} = -0.008\,Kg/s$ and $m = 11.3\,Kg$, $\dot{m} = 0\,Kg/s$, respectively.

6.3 Actuation Delay

In this section the controllers are compared against an actuation delay of around 0.1 seconds in the gimbals, with the same step references as in the previous section.

6.3.1 LQG

The LQG results can be visualized in figures 6.10 and 6.11. Although the trajectory followed is very similar with or without delay, from the inputs it is clear that there are dangerous oscillations which could lead to dangerous situations. This is a consequence of the dynamics of the gimbals using the LQG controller being so fast, which cause little resistance to delays.

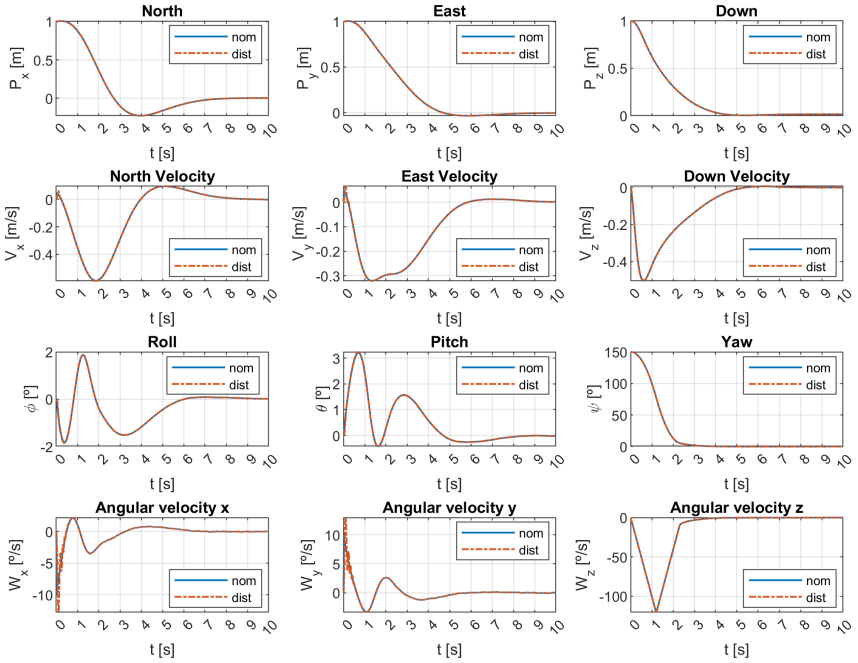


Figure 6.10: State errors of the VIRIATO Test Platform using the LQG controller. In the nominal *nom* and disturbed *dist* cases, delays of 0.005 and 0.01 seconds, respectively.

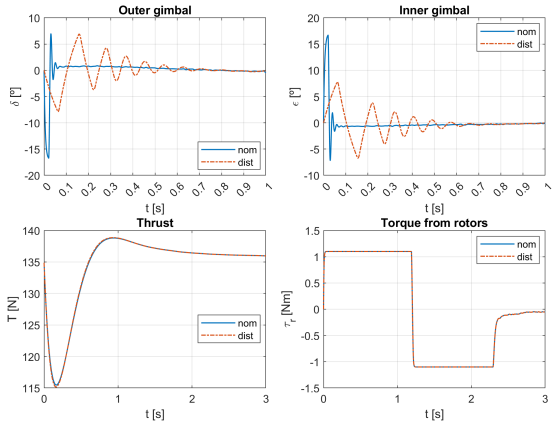


Figure 6.11: Inputs of the VIRIATO Test Platform using the LQG controller. In the nominal *nom* and disturbed *dist* cases, delays of 0.005 and 0.01 seconds, respectively.

6.3.2 NMPCEKF

The NMPCEKF results can be visualized in figures 6.12 and 6.13. The NMPCEKF, similar to the LQG, is able to follow the step references, but this time the orientation of and the input gimbals of the

VIRIATO Test Platform are oscillatory, which could also lead to accidents. This controller also fails to deal with the given delay.

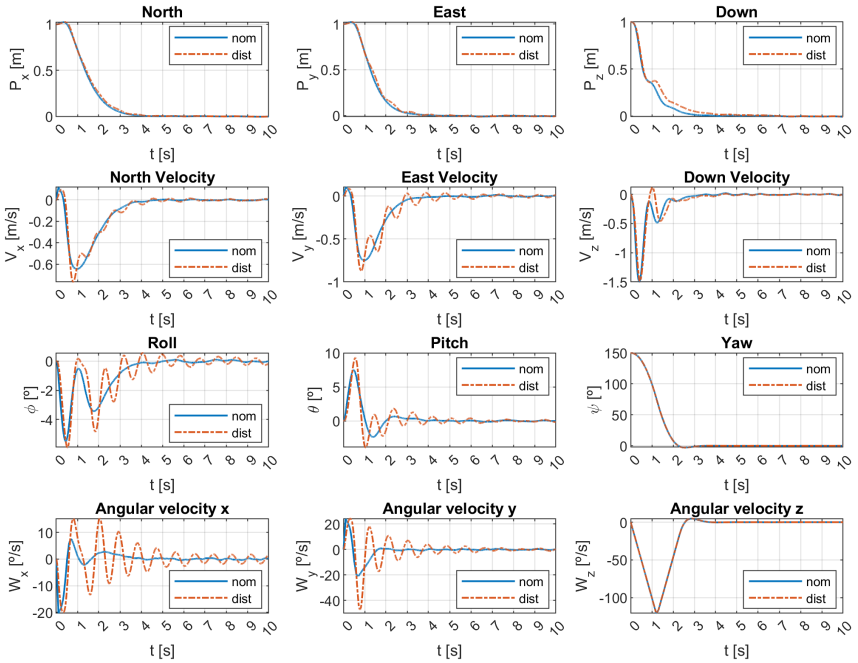


Figure 6.12: State errors of the VIRIATO Test Platform using the NMPCEKF controller. In the nominal *nom* and disturbed *dist* cases, delays of 0.005 and 0.01 seconds, respectively.

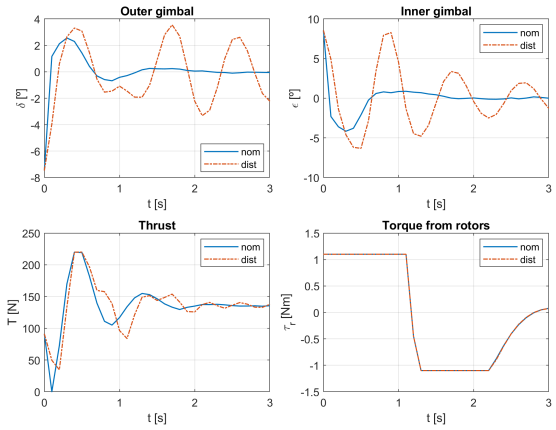


Figure 6.13: Inputs of the VIRIATO Test Platform using the NMPCEKF controller. In the nominal *nom* and disturbed *dist* cases, delays of 0.005 and 0.01 seconds, respectively.

6.3.3 H_∞

The H_∞ results can be visualized in figures 6.14 and 6.15. Contrary to the 2 previous controllers, the H_∞ controller is very robust to actuation delays. In fact, after some tests it was verified that it is able to withstand delays of up to 0.5 seconds, which is 100 times higher than the nominal value of 0.005 seconds, proving that it is quite capable in this domain.

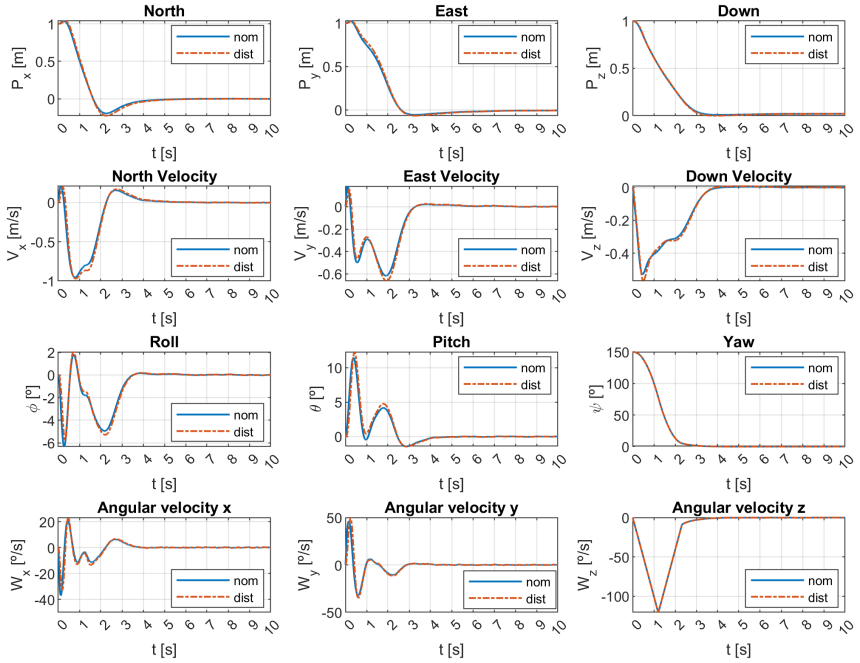


Figure 6.14: State errors of the VIRIATO Test Platform using the H_∞ controller. In the nominal *nom* and disturbed *dist* cases, delays of 0.005 and 0.01 seconds, respectively.

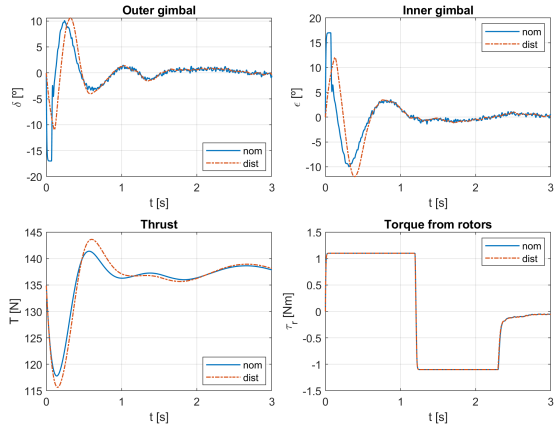


Figure 6.15: Inputs of the VIRIATO Test Platform using the H_∞ controller. In the nominal *nom* and disturbed *dist* cases, delays of 0.005 and 0.01 seconds, respectively.

6.4 Wind Gusts

In this section the controllers are compared against a wind gust of 5 m/s coming from South, but this time they are asked to stand still (references of 0 in all states).

The wind gust dynamics can be approximated by the model in the Military Specification MIL-F-8785C [33] and the algorithm is implemented in the block of *Simulink*, *Discrete Wind Gust Model* [34]. The chosen wind gust time response is present in figure 6.16.

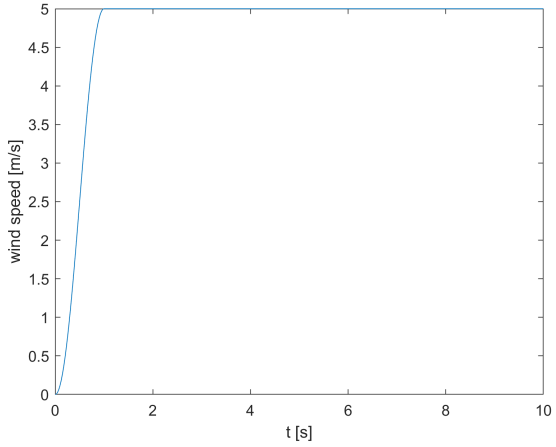


Figure 6.16: Wind gust.

Having characterized the wind gust, it is converted to a resulting force acting on the VIRIATO Test Platform, so the following drag equation is used

$$D = \frac{1}{2}\rho A C_d V^2, \tag{6.1}$$

with D the drag, ρ the air density of 1.225 Kg/m^3, A the lateral area of the VIRIATO Test Platform (when the wind gusts are horizontal), C_d the drag coefficient and V the wind speed. The VIRIATO Test Platform was approximated by a cylinder with height and radius of 1 and 0.3 meters, respectively, leading to a lateral area A of

$$A = 2\pi r h = 2\pi \times 0.3 \times 1 = 0.6\pi\, m^2. \tag{6.2}$$

Using the cylinder approximation, the C_d is [35]

$$C_d \approx 1.2. \tag{6.3}$$

The maximum perturbation, when the wind speed is 5 m/s is

$$D = \frac{1}{2} \times 1.225 \times 0.6\pi \times 1.2 \times 5^2 = 34.636\, N. \tag{6.4}$$

6.4.1 LQG

The LQG results can be visualized in figures 6.17 and 6.18. The controller is able to resist to the perturbation by leaning against the wind gust, but it flies more than 4 meters away from the original

position. If the gains were increased it would have a better resistance to the perturbation and would be able to stay closer to its initial position, but as it has been shown before, the gimbal dynamics are already very fast. This is clearly a limitation of the LQG controller, it is already on its performance limit, the gains can not be increased without risking low stability and robustness.

The inner gimbals (6.18) move in one direction so that the VIRIATO Test Platform faces the wind gust and then in the opposite direction so it stops rotating when the required inclination matches the perturbation.

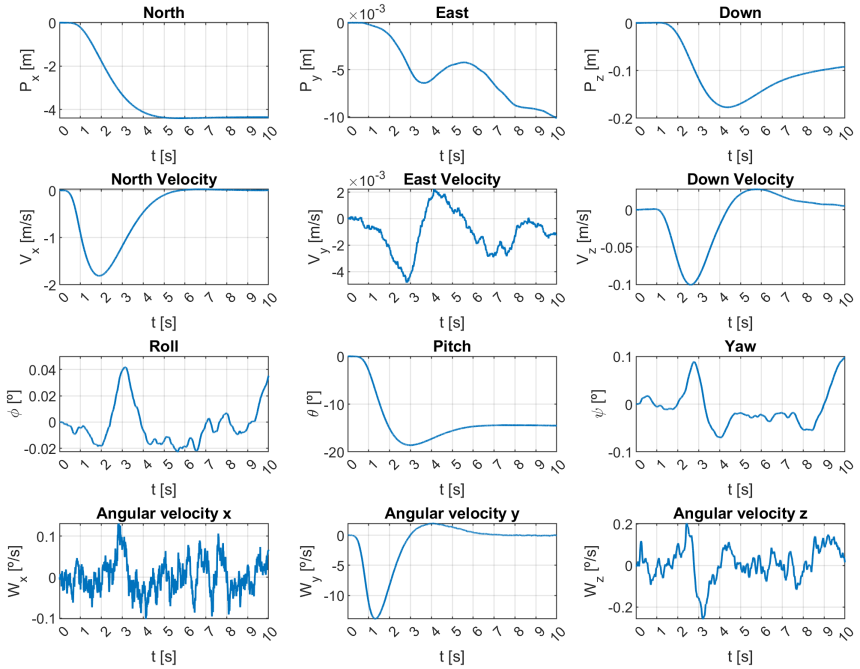


Figure 6.17: States of the VIRIATO Test Platform using the LQG controller.

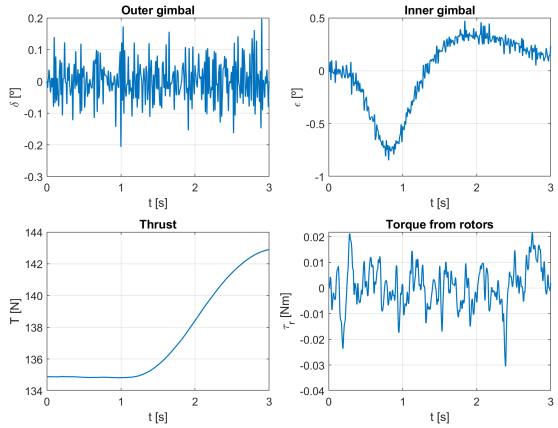


Figure 6.18: Inputs of the VIRIATO Test Platform using the LQG controller.

6.4.2 NMPCEKF

In this domain the behaviour of the NMPCEKF is dominated by the estimates given by the extended Kalman filter. Considering that the filter is using the dynamic equations, depending on the cost of each

variable in the process noise Q matrix, if the costs of certain variables are much higher than the others, the estimation errors can be completely attributed to the ones with the highest costs. As such, by setting the cost in the Q matrix of the angular variables (Euler angles and angular velocity) higher, the EKF gives precise estimates of all variables except the Euler angles, which have a considerable error (figure 6.21). This response, coupled with a mass estimate error (figure 6.22) causes the NMPCEKF to completely reject the wind gust and return to its original position (still with a maximum deviation of 1 meter), acting similarly to an integrator, as shown in figure 6.19.

The inputs throughout this trajectory are shown in figure 6.20 and are similar to the LQG in figure 6.18, with the difference that this time the gimbals move more aggressively.

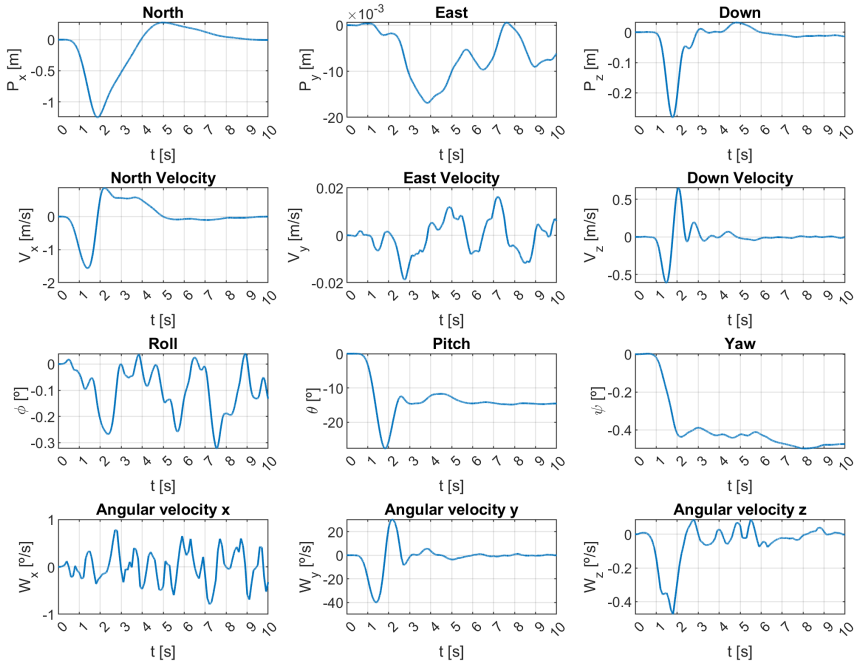


Figure 6.19: States of the VIRIATO Test Platform using the NMPCEKF controller with Q cost higher for the Euler angles and angular velocities.

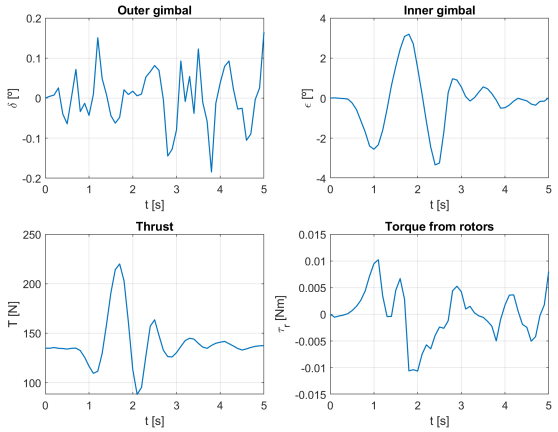


Figure 6.20: Inputs of the VIRIATO Test Platform using the NMPCEKF controller.

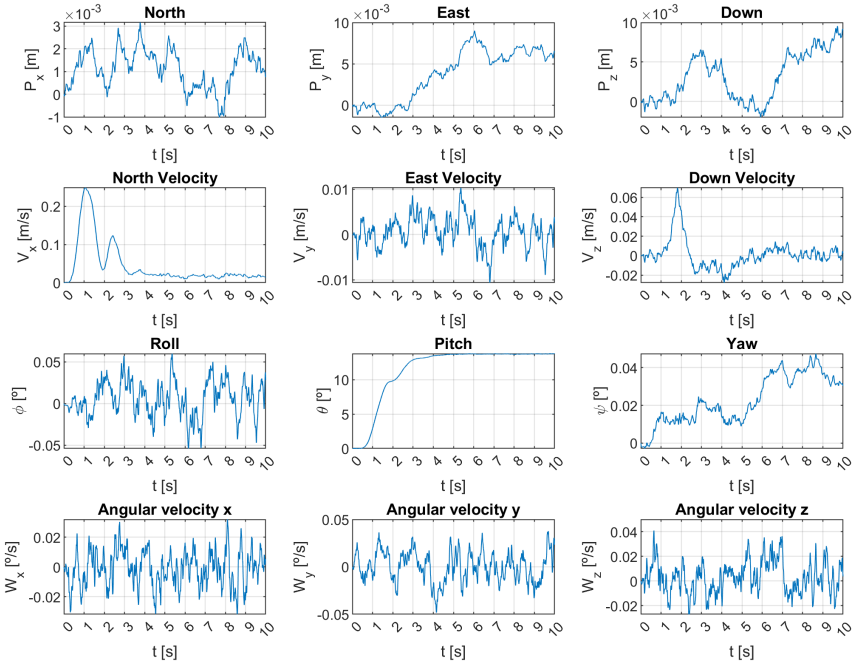


Figure 6.21: Estimation errors of the VIRIATO Test Platform using the NMPCEKF controller with Q cost higher for the Euler angles and angular velocities.

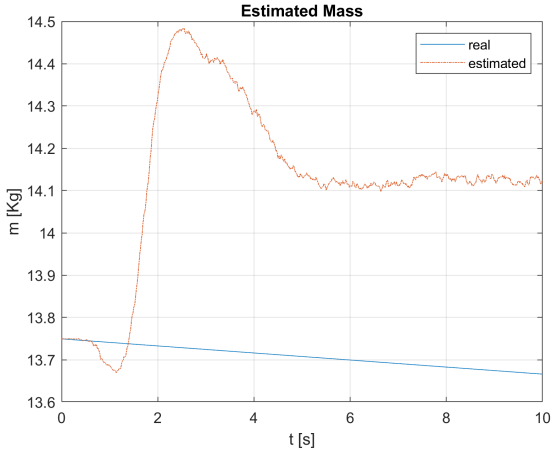


Figure 6.22: Mass estimate of the VIRIATO Test Platform using the NMPCEKF controller with Q cost higher for the Euler angles and angular velocities.

6.4.3 H_∞

The H_∞ results can be visualized in figures 6.23 and 6.24. Although the H_∞ controller is not able to completely reject the wind gust perturbation (it has no integrator), it is able to attenuate it effectively, being moved from the original position 0.6 meters, which is a better result than the LQG and is less than the max deviation of 1 meter of the NMPCEKF.

As expected, the movement of the inner gimbal is the most aggressive among the 3 controllers, which is required to attenuate considerably the wind gust and stop the VIRIATO Test Platform from being dragged.

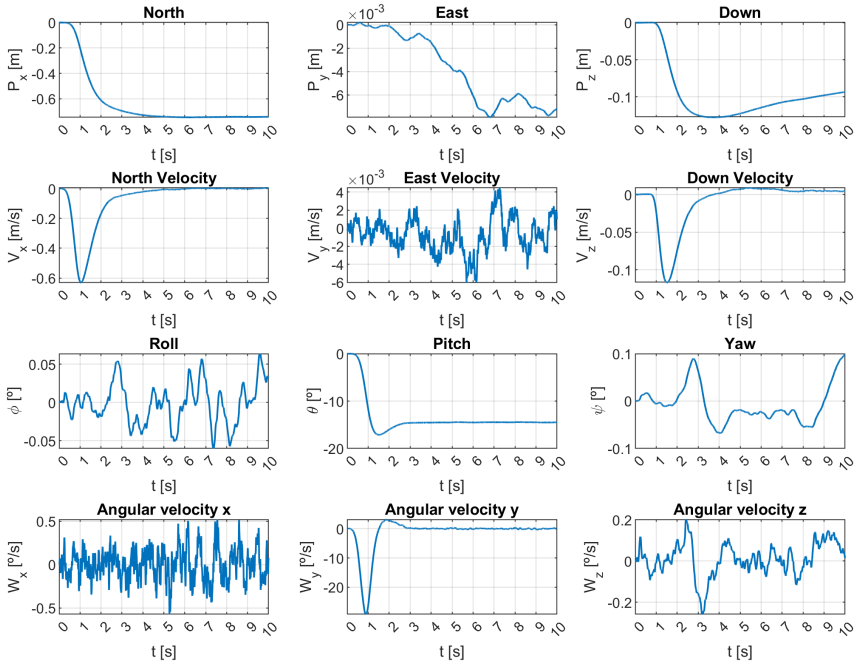


Figure 6.23: States of the VIRIATO Test Platform using the H_∞ controller.

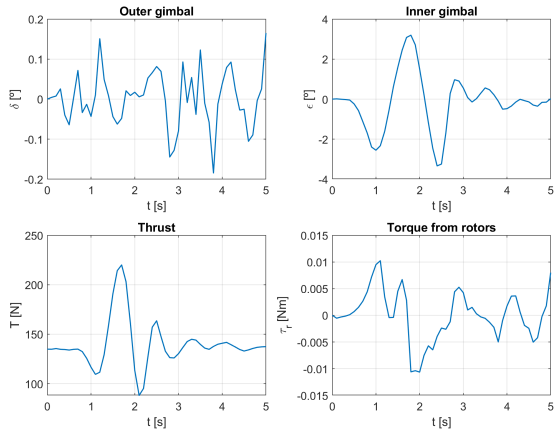


Figure 6.24: Inputs of the VIRIATO Test Platform using the NMPCEKF controller.

6.5 Overall results

After the thorough tests, some conclusions can be taken regarding the 3 controllers. Firstly, the height and yaw control are successfully controlled using PID and PD controllers only, so more complex techniques are not necessary to ensure good tracking performance, disturbance rejection and robustness. The yaw controller was able to track step references with a 2.5 s settling time and was unaffected by the different introduced perturbations. The height controller showed a settling time of 3 to 4 seconds and was also little affected by the Pitch and Roll variations, caused by North and East step references, of the VIRIATO Test Platform.

Contrarily, the position control could benefit from more sophisticated controllers such as the NMPCEKF or the H_∞ controllers. Both of them offer different benefits, namely the NMPCEKF is better suited for complex trajectories which have simultaneous varying references, whereas the H_∞ controller excels in simple trajectories, ensuring the highest level of robustness and disturbance rejection.

A summary of the results is present in table 6.1. All controllers are capable of dealing with the expected mass variation of -2.5 Kg or 20 $\%$, the NMPCEKF and H_∞ controllers are the fastest and the latter has the best robustness and disturbance rejection properties, as mentioned in the previous paragraph.

Lastly, the LQG controller has worse performance in almost all parameters, which shows that it could serve as a baseline controller that could be safely implemented first, as its tuning is just a matter of changing the gains and is known to most control engineers.

Table 6.1: Simulation results of the 3 controllers.

	LQG	NMPCEKF	Hinf
Approximate settling time [s]	5	4	4
Robustness to mass variation	yes		
Maximum tolerable actuation delay [s]	0.1	0.1	0.5
Wind gusts maximum position error [m]	4.2	1.2*	0.6

*In the previous simulations the settling times of the LQG and H_∞ are affected by the varying yaw. If the yaw was kept constant, the settling times would be as indicated in the table.
**There was also an error in the Down position and after 10 seconds the NMPCEKF was able to return to its initial position.

Chapter 7

Guidance

In this chapter, using the H_∞ controller derived and tested previously, 3 trajectories will be fed to the controller, namely a square through waypoints, a circular trajectory while pointing to the center and a more complex one that resembles an 8 or the infinity, again pointing at the center of each circunference. Additionally, a constant 2 m/s wind perturbation coming from South (pushing the VIRIATO Test Platform towards North) is added.

Given that the H_∞ controller is able to control position directly, the guidance algorithm only has 2 concerns, limit the position reference to the control loop and generate trajectories for the VIRIATO Test Platform to follow.

To limit the references, a simple solution was found,

$$\begin{aligned} e &= r - x \\ e_l &= min(2, max(-2, e)) \\ r_l &= x + e_l \end{aligned} \tag{7.1}$$

with e, r, x, e_l, r_l the error, reference, current state, limited error and reference, respectively. This way, the error is bounded to -2 and 2, which ensures that the inputs stay within a reasonable value.

7.1 Square Waypoint Trajectory

In this section, the simplest trajectory is fed to the VIRIATO Test Platform. The goal is to follow the path defined in the orange, dotted line in figure 7.2, while keeping a constant yaw of 0. The states during the movement, in figure 7.1 show that the VIRIATO Test Platform is able to follow the North and East position references with a delay of 1 second, which is expected given the performance shown in the previous chapter. It can also be noted that when the VIRIATO Test Platform changes direction, it tilts, which causes a disturbance in the Down position, originating maximum errors of 0.4 meters. This Down error could be reduced by increasing the gains of the height controller, but this would increase noise, reduce stability margins and increase fuel consumption. In fact, after the VIRIATO Test Platform tilts, it

goes back to its original straight position after a few seconds, recovering some of the lost height. Note that the North and East references are not steps and have a maximum error of 2 meters, as intended with the limiting technique defined earlier.

In figure 7.2 the square trajectory is closely matched, with a constant error in the North position due to the wind perturbation of 2 m/s, shifting the square trajectory to the right (the wind comes from South). In addition to this, in the corners there is a slight increase in error due to the fact that a simple mechanism was adopted to change the reference to the next waypoint, namely when the error is less than a threshold, which in this case was 0.5 m. The inputs of the VIRIATO Test Platform behave smoothly, moving the gimbals accordingly, tilting the VIRIATO Test Platform in the right direction to accelerate and then roll or pitch back to the original position. The thrust also increases in the beginning of the direction change to increase speed and then compensates to keep the height at zero.

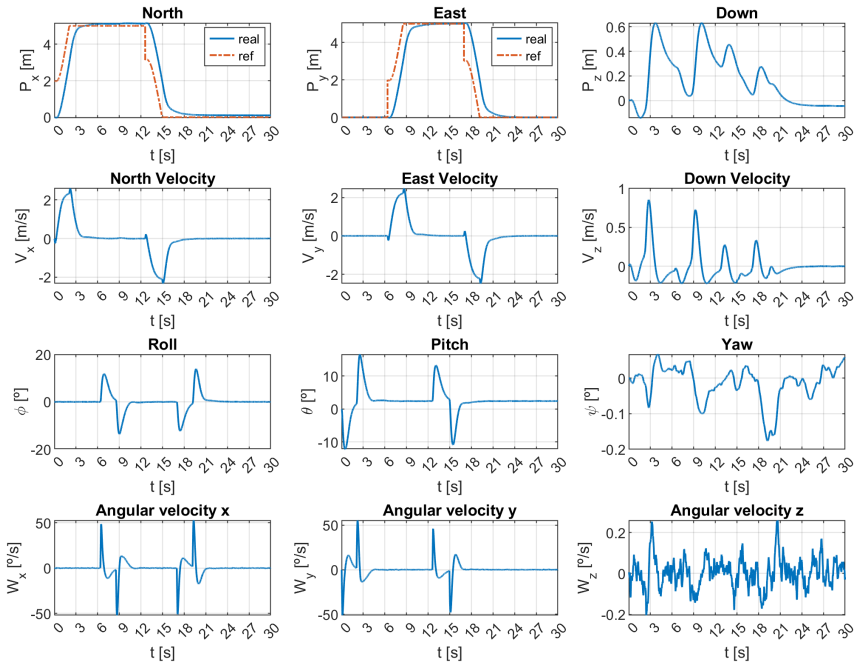
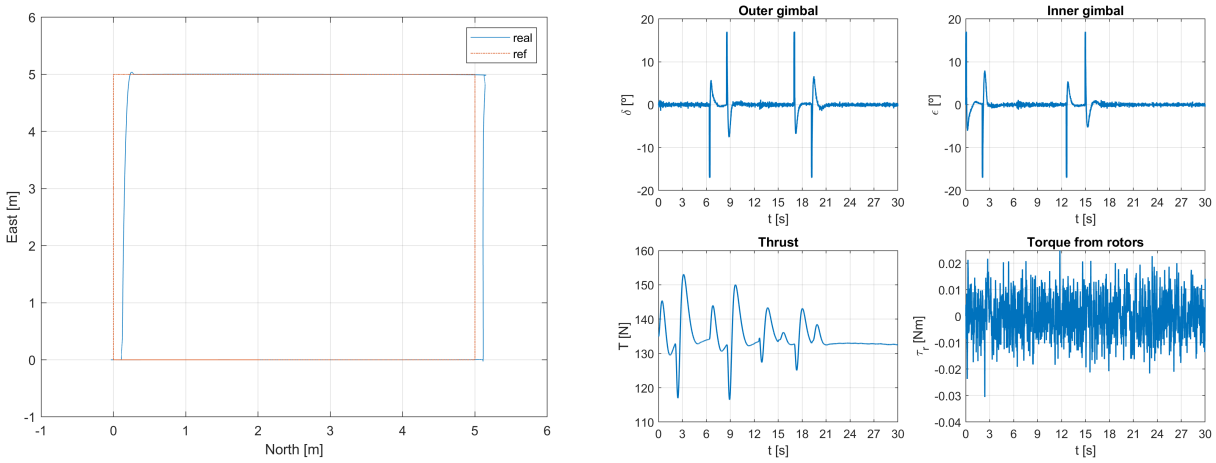


Figure 7.1: States of the VIRIATO Test Platform and horizontal position references for the square trajectory



(a) Path.

(b) Inputs.

Figure 7.2: Square path in the North-East plane and inputs of the VIRIATO Test Platform.

7.2 Loop Trajectory

Here, a slightly more complex trajectory is given to the VIRIATO Test Platform. It has to describe a circumference while pointing at its center, as depicted in the red line in figure 7.4.

In figure 7.3, the controller is able to follow the North, East positions and velocities, with 1 second delay, while the yaw has a smaller delay of 0.2 seconds. Given that the states are very similar to the references, only having a delay, the path followed is almost the same as the reference (figure 7.4), but the trajectory is delayed. Due to the wind perturbation, the loop is also shifted to the right by around 30 meters.

Notice, in figure 7.4, that the yaw is almost pointing to the center of the circumference, with an error of 20º, which could be corrected by using an integrator at the cost of robustness and overshoot. The input gimbals are smoother than in the square trajectory, having an initial abrupt movement, then making small adjustments to keep in track and in the end they roll and pitch the VIRIATO Test Platform so it stops moving. The thrust has a maximum error of 0.4 m, due to the tilting and requested horizontal velocity and the rotors provide an initial torque so the VIRIATO Test Platform starts spinning at the correct rate for the loop and then an opposite torque so it stops.

.

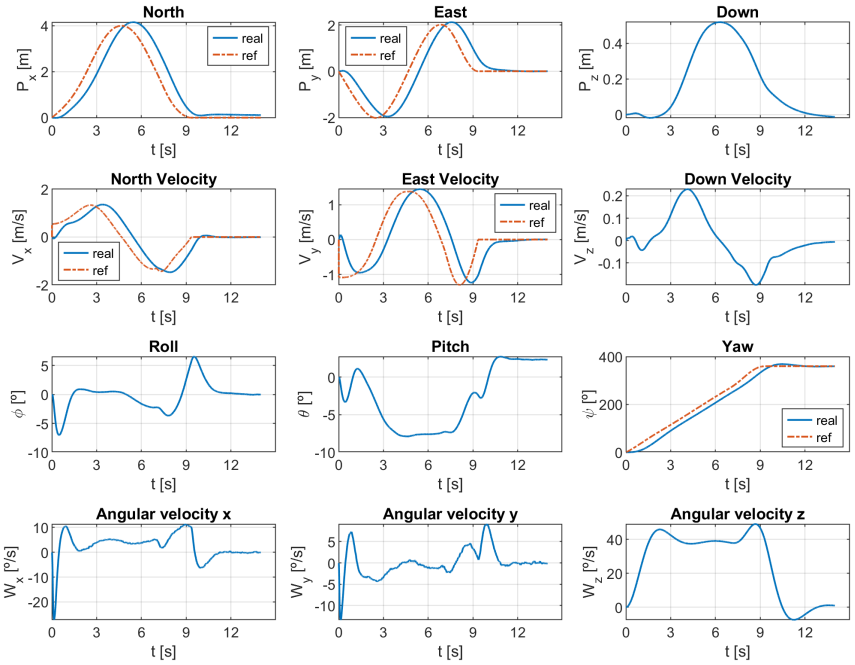
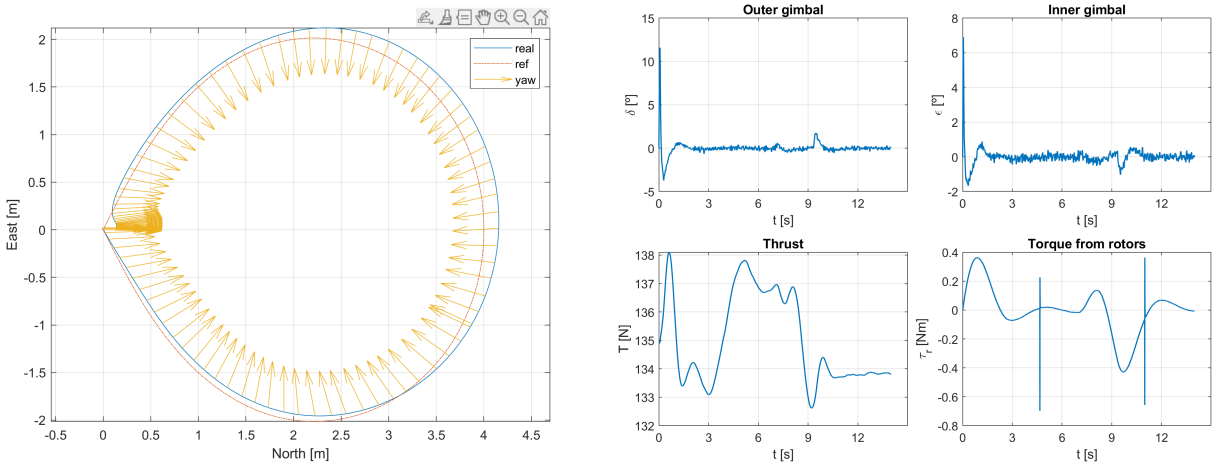


Figure 7.3: States of the VIRIATO Test Platform and horizontal position references for the loop trajectory



(a) Path. (b) Inputs.

Figure 7.4: Loop path in the North-East plane and inputs of the VIRIATO Test Platform.

7.3 Infinity Trajectory

Lastly, the most complex tested trajectory is presented. Similarly to the previous 2 sections, it is shown in the red line in figure 7.6. The VIRIATO Test Platform has to follow a trajectory that resembles the infinity symbol ∞ or an 8, while pointing at the center of the first circumference at North, East = (2, 0) when its North position is less than 4 m and to the second center at North, East = (6, 0) when the North position is bigger than 4 m.

The tracking results of the position and velocity states in figure 7.5 are similar to the loop trajectory in figure 7.3, exhibiting a smooth trajectory with a delay of 1 second. Contrary to the previous loop trajectory, when the VIRIATO Test Platform crosses the 4 m North mark, the yaw reference changes 180º, behaving as a step reference, to whom the VIRIATO Test Platform responds in approximately 3 seconds with little overshoot.

In figure 7.6, the path is closely followed, except in the segment where yaw has to change 180º, consequently affecting the position of the VIRIATO Test Platform. This behaviour exists because the position controller is not able to react as quickly as the yaw controller, thus suffering position errors. This also happened in the previous chapter to the H_∞ controller, and its the most significant drawback when compared to the NMPCEKF. Furthermore, similarly to the previous 2 cases, the wind perturbation shifts the trajectory to the right by 0.3 meters.

Similarly to the previous loop trajectory section, the gimbals move abruptly in the beginning to tilt the VIRIATO Test Platform in the right direction and then do small changes to keep the VIRIATO Test Platform in the desired trajectory. Notice that in seconds 9 and 21 the inner gimbal shows again an abrupt response, which corresponds to the crossing of the 4 m North (figure 7.5 and a change of 180º of yaw, so the position controller has to compensate the yaw changes. The thrust also varies significantly at the 9 and 21 seconds mark, for the same reason as described previously and it is enough to keep the Down error below 0.4 m (figure 7.5.

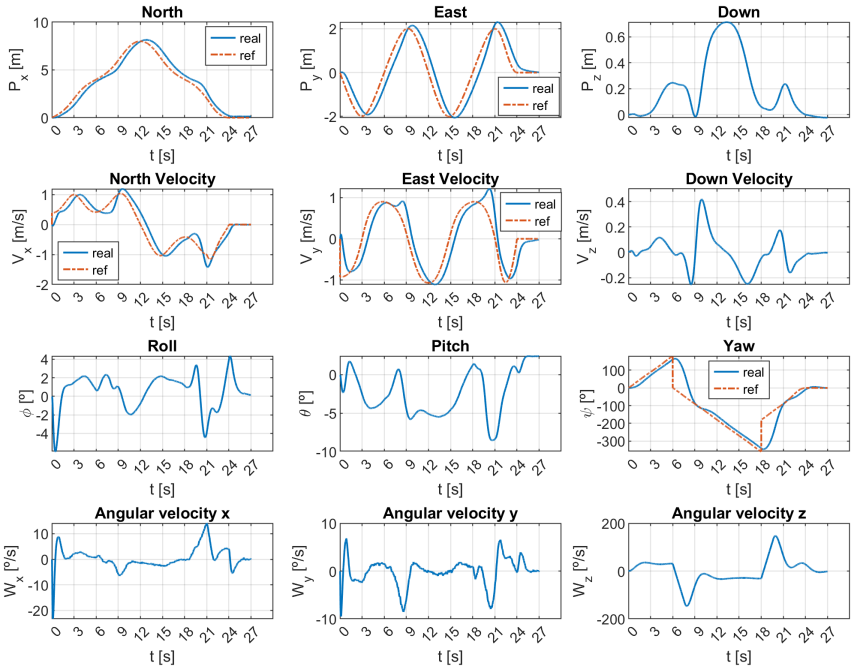
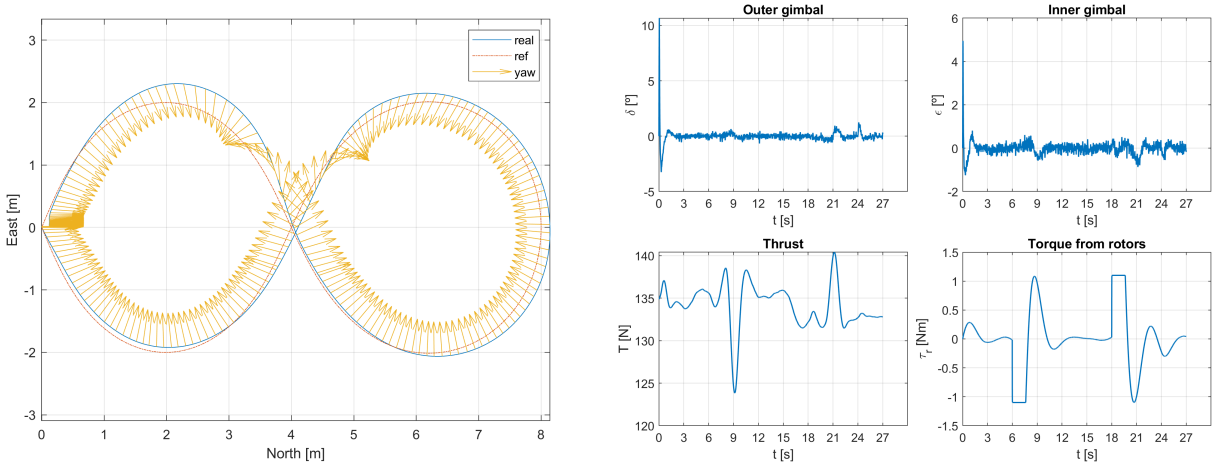


Figure 7.5: States of the VIRIATO Test Platform and horizontal position references for the infinity trajectory



(a) Path. (b) Inputs.

Figure 7.6: Infinity path in the North-East plane and inputs of the VIRIATO Test Platform.

Chapter 8

Conclusions

In this chapter the main conclusions of this work are presented.

8.1 Achievements

Firstly, the nonlinear and linear dynamics, both in transfer function and state space form were derived. Additionally, the noise specifications of the sensors were presented and the position and acceleration measurements were combined to produce cleaner position and velocity estimates.

Then, classical PD controllers (both using filtered derivatives and cascaded), were applied to control height, Euler angles and position states. In addition to this, MIMO LQG and MPC controllers were formulated to control the horizontal position of the VIRIATO Test Platform. The main design results of these controllers were shown, achieving the desired performance for the control of the Euler angles and height, but the bandwidth of the horizontal position PD and LQG controllers was 0.12 and 0.14, respectively, far from the desired 1 Hz.

Having concluded that the previously developed position controllers did not achieve the desired performance, 3 robust controllers H_2, H_∞ and μ synthesis were designed. Their tuning using weighting functions $\mathbf{W}_1$, $\mathbf{W}_2$ and $\mathbf{W}_3$ proved to be challenging, but responses with a bandwidth of 0.3 Hz were found for the H_2 and H_∞ controllers, which is significantly better than the PD and LQG controllers. Moreover, the stability margins for these controllers are similar to the slower linear controllers developed in 3, so this design proved to be an improvement.

To test another control position approach, a nonlinear MPC controller (NMPC) and an extended Kalman filter (EKF) with mass error states were developed. The EKF receives position, Euler angles and angular velocity measurements and the known actuator inputs given to VIRIATO (gimbals, thrust and torque of the rotors) and provides estimates of the real position, velocity, Euler angles and angular velocity, as well as the mass error. The NMPC, receives these estimates and using the CasADi interior point solver calculates the optimal trajectory in the prediction horizon, whose curve is shaped by its Q and R cost matrices. Together these 2 systems form the NMPCEKF (nonlinear model predictive control with an extended Kalman filter) and originate the fastest response, with the littlest overshoot in nominal

conditions (no perturbations or delay) among all the controllers.

Having designed linear, robust and nonlinear controllers, the best performing controller of each category was chosen: LQG, H_∞ and NMPCEKF, respectively. These are compared against the maximum expected mass variation of -2.5 Kg, actuation delays of 0.1 s in the gimbals and wind gusts of 5 m/s equalling 35 N. All 3 controllers were able to deal with the expected mass variation with little effort, but the H_∞ controller showed the best rejection to perturbations, moving a maximum of 0.6 m to a wind gust of 5 m/s, while the LQG moved 4 m and the NMPCEKF 1 m. Note that if the cost matrix Q of the EKF was tuned so that little trust was given to the Euler angles and angle velocity measurements, the NMPCEKF would completely reject the wind gust disturbance, having a max deviation of 1 m and then returning to its original position. This is an interesting result but could be detrimental in other situations due the fact that this low trust in the Euler angles and angular velocity measurements lead to an estimation error of the Euler angles of 15° which is a dangerous result. Lastly, the H_∞ controller had again the best robustness against actuation delays, easily withstanding the presented delay of 0.1 s, while the other controllers struggled and were barely stable.

Having tested extensively the most promising controllers, the H_∞ controller proved to be the most reliable given its fast response, disturbance rejection and robustness properties. Externally to this controller, an outer guidance loop was developed that generates the trajectories for the position controller to follow and limits the references at each time step not to exceed a certain threshold, chosen as 2 m. This set up was able to follow the desired square and loop paths with a maximum error of 0.3 m, mostly due to the wind perturbation of 2 m/s and only deviated a maximum of 0.5 m in the infinity path, in the segment with a drastic variation of the yaw reference of 180° .

All things considered, a guidance and control system was developed that enables the VIRIATO Test Platform to follow trajectories ranging from square waypoints to rotating loops, having little path and yaw errors except when asked abrupt yaw variations. The proposed final controller is faster than the usual PD or LQG algorithms and exhibits robustness to mass variations of -2.5 Kg or 20% of the nominal value, 0.1 s of actuation delay and wind gusts of 5 m/s or 35 N.

8.2 Future Work

To continue developing on this work, the algorithms could be implemented in real time in the VIRIATO Test Platform. The tests should start with the simplest classical PD controller, increasing in complexity until the H_∞ controller. Perhaps it would be possible to test the NMPCEKF controller, but it would require a faster microcontroller.

In the computational simulations domain, there are several ideas that could be explored to enhance the control and guidance of the VIRIATO Test Platform. The dynamical model presented in this master thesis is very simplified, capturing only the main dynamics. A more complex model that captures more complex movements such as sloshing could be developed to run the simulations in a more reliable environment.

The Weighting functions of the Robust controllers were obtained via trial and error, which proved to

be a challenging exercise. Similarly to the LTR technique used to tune the Kalman filter of the LQG, one to tune the weighting functions of the robust controllers could be developed.

The NMPCEKF could be extended with disturbance or delay states to include perturbations such as wind gusts or actuation delays and deal with them optimally.

Lastly, the proposed H_∞ controller performance was impacted when simultaneous position and abrupt yaw references were fed. A solution to this problem would be to include some nonlinear mapping before feeding the position references to the controller, using for example feedback linearization.

Bibliography

[1] SpaceX. Launches. https://www.spacex.com/launches/. Accessed: 2022-10-30.

[2] VentureRadar. Top launcher start-ups. https://www.ventureradar.com/startup/launcher. Accessed: 2022-10-30.

[3] Omnidea. Viriato. https://www.omnidea.net/viriato.html. Accessed: 2022-10-30.

[4] Spin.Works. Viriato – gnc subsystem. https://www.spinworks.pt/portfolio/viriato-gnc-subsystem/. Accessed: 2022-10-30.

[5] DLR. Eagle. https://www.dlr.de/irs/en/desktopdefault.aspx/tabid-11346/. Accessed: 2022-07-02.

[6] M. Dumke, M. Sagliano, P. Saranrittichai, G. F. Trigo, and S. Theil. Eagle - environment for autonomous gnc landing experiments. In *10th International ESA Conference on Guidance, Navigation and Control Systems*, June 2017. URL https://elib.dlr.de/116426/.

[7] M. Sagliano, M. Dumke, and S. Theil. Simulations and flight tests of a new nonlinear controller for the eagle lander. *Journal of Spacecraft and Rockets*, 56(1):259–272, 2019. doi: 10.2514/1.A34161. URL https://doi.org/10.2514/1.A34161.

[8] M. Dumke, G. F. Trigo, M. Sagliano, P. Saranrittichai, and S. Theil. Design, development, and flight testing of the vertical take-off and landing GNC testbed EAGLE. *CEAS Space Journal*, 12(1): 97–113, Jan. 2020. doi: 10.1007/s12567-019-00269-5.

[9] Z. Benić, P. Piljek, and D. Kotarski. Mathematical modelling of unmanned aerial vehicles with four rotors. *Interdisciplinary Description of Complex Systems*, 14:88–100, 01 2016. doi: 10.7906/indecs.14.1.9.

[10] Xsens. Xsens mti 600-series. https://www.xsens.com/products/mti-600-series, . Accessed: 2022-09-29.

[11] Xsens. Mti 600-series performance specifications. https://mtidocs.xsens.com/sensor-specifications-2, . Accessed: 2022-09-29.

[12] M. I. L. de Faria Ribeiro. Kalman and extended kalman filters: Concept, derivation and properties. *Robotics WEBook, http://www.euron.org*, 2004.

[13] MATLAB. kalman. https://www.mathworks.com/help/control/ref/ss.kalman.html, . Accessed: 2022-06-22.

[14] C. Ma. Application of lqr and h2-optimal control for a quadrotor system. Master's thesis, Huazhong University of Science and Technology, 2020.

[15] MATLAB. allmargin. https://www.mathworks.com/help/control/ref/lti.allmargin.html, . Accessed: 2022-06-5.

[16] P. Seiler, A. Packard, and P. Gahinet. An introduction to disk margins [lecture notes]. *IEEE Control Systems Magazine*, 40(5):78–95, 2020. doi: 10.1109/MCS.2020.3005277.

[17] MATLAB. diskmargin. https://www.mathworks.com/help/robust/ref/diskmargin.html, . Accessed: 2022-05-25.

[18] A. Lim, J. Moore, and L. Faybusovich. Separation theorem for linearly constrained lqg optimal control - a continuous time case. In *Proceedings of 35th IEEE Conference on Decision and Control*, volume 4, pages 4152–4157 vol.4, 1996. doi: 10.1109/CDC.1996.577430.

[19] J. P. Hespanha. *Linear Systems Theory*. 2018.

[20] MATLAB. mpc. https://www.mathworks.com/help/mpc/ref/mpc.html, . Accessed: 2022-05-25.

[21] MATLAB. c2d. https://www.mathworks.com/help/control/ref/lti.c2d.html, . Accessed: 2022-05-25.

[22] S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. Wiley, 2^{nd} edition, 2005. ISBN: 978-0-470-01167-6.

[23] K. Zhou, J. C. Doyle, and K. Glover. *ROBUST AND OPTIMAL CONTROL*. Prentice Hall, 1^{st} edition, 1996.

[24] MATLAB. augw. https://www.mathworks.com/help/robust/ref/lti.augw.html, . Accessed: 2022-05-29.

[25] MATLAB. hinfsyn. https://www.mathworks.com/help/robust/ref/lti.hinfsyn.html, . Accessed: 2022-05-29.

[26] MATLAB. musyn. https://www.mathworks.com/help/robust/ref/uss.musyn.html, . Accessed: 2022-05-29.

[27] MATLAB. wcdiskmargin. https://www.mathworks.com/help/robust/ref/lti.wcdiskmargin.html, . Accessed: 2022-05-30.

[28] K. R. H., F. A. R., and J. C. Geromel. H-infinity control design for time-delay linear systems: a rational transfer function based approach. *European Journal of Control*, 18(5):425–436, 2012. doi: 10.3166/EJC.18.425-436.

[29] M. Kelly. Transcription methods for trajectory optimization: a beginners tutorial. 07 2017.

[30] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1): 1–36, 2019. doi: 10.1007/s12532-018-0139-4.

[31] M. Fazlyab, S. Paternain, V. M. Preciado, and A. Ribeiro. Interior point method for dynamic constrained optimization in continuous time, 2015. URL `https://arxiv.org/abs/1510.01396`.

[32] FDNunes. V - gps. DEEC, IST, 2020.

[33] MIL-F-8785C. Military specification: Flying qualities of piloted airplanes. November 1980.

[34] MATLAB. Discrete wind gust model. `https://www.mathworks.com/help/aeroblks/discretewindgustmodel.html`, . Accessed: 2022-10-23.

[35] G. Bruschi, T. Nishioka, K. Tsang, and R. Wang. A comparison of analytical methods drag coefficient of a cylinder. March 2003.