UNIVERSIDADE FEDERAL DE MINAS GERAIS
Programa de Pós-Graduação em Engenharia Elétrica
Research group MACRO - Mechatronics, Control and Robotics

Richard Alfonso Andrade Alfaro

TUBE-BASED MODEL PREDICTIVE CONTROL: a fast embedded optimization perspective

Belo Horizonte
2024

Richard Alfonso Andrade Alfaro

TUBE-BASED MODEL PREDICTIVE CONTROL: a fast embedded optimization perspective

Thesis submitted to the Graduate Program in Electrical Engineering of Escola de Engenharia at the Universidade Federal de Minas Gerais, in partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering.

Advisors: Guilherme Vianna Raffo
Co-Advisors: Julio Elias Normey Rico

Belo Horizonte
2024

Alfaro, Richard Alfonso Andrade.
A385t Tube-based model predictive control [recurso eletrônico] : a fast embedded optimization perspective / Richard Alfonso Andrade Alfaro. – 2024.
1 recurso online (121 f. : il., color.) : pdf.

Orientador: Guilherme Vianna Raffo.
Coorientador: Julio Elias Normey-Rico.

Tese (doutorado) – Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f. 113-121.

1. Engenharia elétrica – Teses. 2. Robótica – Teses. 3. Redes de computadores – Administração – Teses. 4. Algoritmos de computador – Teses. 5. Controle preditivo – Teses. 6. Transporte de cargas – Teses. I. Raffo, Guilherme Vianna. II. Normey-Rico, Julio Elias. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pelo Bibliotecário Marcio A. A. Gomes CRB/6 2812
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FOLHA DE APROVAÇÃO

"TUBE-BASED MODEL PREDICTIVE CONTROL: A FAST EMBEDDED OPTIMIZATION PERSPECTIVE"

RICHARD ALFONSO ANDRADE ALFARO

Tese de Doutorado subme da à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica. Aprovada em 04 de abril de 2024. Por:

Prof. Dr. Guilherme Vianna Raffo
DELT (UFMG) - Orientador

Prof. Dr. Julio Elias Normey Rico
DAS (UFSC) - Coorientador

Prof. Dr. Antonio Ferramosca
DIGIP (University of Bergamo)

Prof. Dr. Márcio André Fernandes Mar ns
DEQ (UFBA)

Prof. Dr. Douglas Wildgrube Bertol
DEE (UDESC)

Prof. Dr. Janier Arias García
DELT (UFMG)

Documento assinado eletronicamente por **Guilherme Vianna Raffo**, **Professor do Magistério Superior**, em 04/04/2024, às 17:50, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Antonio Ferramosca**, **Usuário Externo**, em 05/04/2024, às 09:38, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Janier Arias Garcia**, **Professor do Magistério Superior**, em 08/04/2024, às 20:13, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Julio Elias Normey Rico**, **Usuário Externo**, em 09/04/2024, às 09:45, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Douglas Wildgrube Bertol**, **Usuário Externo**, em 11/04/2024, às 11:12, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Marcio André Fernandes Mar ns**, **Usuário Externo**, em 02/05/2024, às 14:11, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A auten cidade deste documento pode ser conferida no site h ps://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3153557** e o código CRC **514A565B**.

To my parents, Fabian and Fanncy,
to my grandmother, Blanquita, and
to my wife, Natalia.

Acknowledgments

I would like to express my deepest gratitude to my parents for their unwavering support throughout my life. They have stood by me in both my triumphs and challenges, always encouraging me to pursue my dreams. Their daily dedication and valuable teachings have shaped me, both personally and professionally.

A special thanks goes to my wife, Natalia, for her constant companionship and support, especially during the difficult moments of this long journey.

I am also profoundly grateful to my advisors, Guilherme Vianna Raffo and Julio Normey Rico, for their continuous guidance and willingness to help me throughout my thesis. Your support has been invaluable, and I truly appreciate all you have done for me. I would also like to extend my thanks to the members of ProVANT and MACRO for their support.

I would like to acknowledge the financial support from the funding agencies CAPES, CNPq, FAPEMIG, and the INCT project, which was essential for the completion of this thesis. My thanks also go to the Graduate Program in Electrical Engineering for providing the necessary infrastructure to carry out this work.

Resumo

Nos próximos anos, a navegação autônoma, impulsionada por técnicas avançadas de controle, se tornará essencial para aplicações como sensoriamento remoto, transporte de cargas e missões de busca e resgate. Para assegurar a execução eficiente dessas tarefas, as técnicas de controle precisam equilibrar desempenho e robustez em sistemas não lineares multivariáveis, considerando as restrições do processo. O controle preditivo (MPC) baseado em tubos surge como uma abordagem promissora para atender a essas exigências, embora sua implementação requeira significativo poder computacional. Esta tese de doutorado propõe esta técnica como uma estratégia de controle para aplicações de navegação autônoma, com foco no desenvolvimento de leis de controle robustas e computacionalmente eficientes, adequadas para sistemas embarcados com recursos limitados. Duas formulações são propostas para implementar essa metodologia em sistemas dinâmicos de alta ordem e dinâmica rápida. A primeira formulação introduz uma metodologia inovadora para o cálculo offline dos conjuntos alcançáveis usando zonotopos, visando reduzir os custos computacionais ao lidar com sistemas de alta ordem baseados em modelos lineares com parâmetros variantes. Propõe-se um novo conjunto de condições de desigualdades lineares matriciais, que considera tanto o valor máximo da ação de controle quanto as incertezas do sistema representadas como um zonotopo. As duas abordagens são utilizadas para calcular os conjuntos nominais de estado e controle, os quais são empregados no problema de otimização nominal do MPC robusto. Este problema de controle nominal é resolvido via otimização multiparamétrica offline, e sua solução explícita é implementada por um novo algoritmo baseado em programação paralela, permitindo o cálculo rápido do sinal de controle. Na segunda formulação, propõe-se um algoritmo ADMM simétrico-escalado para resolver o problema nominal de controle preditivo. Trata-se de um algoritmo de otimização altamente paralelizável, que se diferencia das estruturas tradicionais de programação quadrática. Ele incorpora estratégias de aceleração e normalização, assegurando robustez numérica e rápida convergência. As formulações desenvolvidas são aplicadas a um VANT tiltrotor para tarefas de transporte de carga, modeladas a partir da perspectiva da carga. Resultados experimentais são obtidos usando um ambiente de hardware-in-the-loop durante missões de rastreamento de trajetórias. Os algoritmos propostos são implementados em um computador embarcado, que adquire o comportamento dinâmico do sistema a partir de um simulador 3D de alta fidelidade, desenvolvido no Gazebo e ROS.

Palavras-chave: MPC baseado em tubos; otimização rápida; zonotopos; VANT.

Abstract

In the coming years, autonomous navigation, driven by advanced control techniques, will become essential for applications such as remote sensing, cargo transportation, and search and rescue missions. To ensure the efficient execution of these tasks, control techniques must balance performance and robustness in multivariable nonlinear systems while considering process constraints. Tube-based Model Predictive Control (MPC) emerges as a promising approach to meet these demands, although its implementation requires significant computational resources. This doctoral thesis proposes this technique as a control strategy for autonomous navigation applications, focusing on the development of robust and computationally efficient control laws suitable for embedded systems with limited resources. Two formulations are proposed to implement this methodology in high-order, fast dynamic systems. The first formulation introduces an innovative methodology for the offline computation of reachable sets using zonotopes, aiming to reduce computational costs when dealing with high-order systems based on linear models with varying parameters. A new set of linear matrix inequality conditions is proposed, considering both the maximum control action value and system uncertainties represented as a zonotope. The two approaches are used to calculate nominal state and control sets, which are employed in the nominal optimization problem of robust MPC. This nominal control problem is solved via offline multi-parametric optimization, and its explicit solution is implemented using a new algorithm based on parallel programming, allowing for rapid control signal computation. In the second formulation, a scaled-symmetric ADMM algorithm is proposed to solve the nominal predictive control problem. This is a highly parallelizable optimization algorithm that differs from conventional quadratic programming frameworks. It integrates acceleration and normalization strategies, ensuring numerical robustness and fast convergence. The developed formulations are applied to a tiltrotor UAV for cargo transport tasks, modeled from the load's perspective. Experimental results are obtained using a hardware-in-the-loop environment during trajectory tracking missions. The proposed algorithms are implemented in an embedded computer, which acquires the dynamic behavior of the system from a high-fidelity 3D simulator developed in Gazebo and ROS.

Keywords: tube-based MPC; fast optimization; zonotopes; UAV.

List of Figures

List of Tables

Acronyms

MPC	Model Predictive Control
UAV	Unmanned Aerial Vehicle
TMPC	Tube-based Model Predictive Control
ADMM	Alternating Direction Method of Multipliers
CZ_TMPC	Constrained Zonotopic Tube-based Model Predictive Control
LTI	Linear Time Invariant
LTV	Linear Time Varying
LPV	Linear Parameter-Varying
HIL	Hardware-in-the-loop
FGM	Fast Gradient Method
SBC	Single Board Computer
FPGA	Field Programmable Gate Array
RTOS	Real-time Operating System
GPU	Graphics Processing Unit
SoC	System on a Chip
MSE	Mean Square Error
LQR	Linear Quadratic Regulator
H_∞	H-Infinity (control theory)
TCV	Total Control Variation

Notation

General notation

a	Italic lower case letters denote scalars.
$\boldsymbol{a}$	Boldface italic lower case letters denote vectors.
$\boldsymbol{A}$	Boldface italic upper case letters denote matrices.
t	Continuous time.
k	Discrete time.
$\dot{\boldsymbol{a}}$	Time-derivative of vector a.
$\hat{\boldsymbol{a}}$	Prediction of vector a.
$\bar{\boldsymbol{a}}$	Nominal vector a.
$\tilde{\boldsymbol{a}}$	Error vector of the variable, $\tilde{\boldsymbol{a}} = \boldsymbol{a} - \boldsymbol{a}_r$.
$\boldsymbol{a}'$	Transpose of vector a.
$\boldsymbol{a}_{min}$	Minimum values of variable $\boldsymbol{a}$.
$\boldsymbol{a}_{max}$	Maximum values of variable $\boldsymbol{a}$.
$\boldsymbol{A}^{-1}$	Inverse operator of matrix A.
$\boldsymbol{A}^{+}$	Pseudo inverse operator of matrix A.
$\boldsymbol{A}_n$	Denote the normalized matrix $\boldsymbol{A}$.
$\boldsymbol{I}_n$	Identity matrix of size n.
$\boldsymbol{0}_{nxm}$	Zero matrix of n row and m columns.
$\{\boldsymbol{G}, \boldsymbol{c}\}$	Denote a Zonotope.
$\{\boldsymbol{G}, \boldsymbol{c}, \boldsymbol{A}, \boldsymbol{b}\}$	Denote a Constraint Zonotope.

Symbols and operators

$\mathbb{N}$	Set of natural numbers.
$\mathbb{R}$	Set of real numbers.
$\mathcal{A}$	Denote a Set A.
$\mathbf{0}$	Zero matrix with appropriate dimension.
$\mathbb{I}$	Identity matrix with appropriate dimension.
$\mathbf{0_{n\times m}}$	Zero matrix with n lines and m columns.
$\mathbb{I}_{n\times n}$	Identity matrix with n lines and n columns.

Control Notation

N	Prediction horizon.
M	Control horizon.
$\boldsymbol{P}$	Predicted states matrix.
$\boldsymbol{Q}$	Predicted input matrix.
$\widehat{\boldsymbol{x}}$	Predicted state vector.
$\hat{\boldsymbol{u}}$	Predicted control vector.
$\hat{\boldsymbol{u}}_{\boldsymbol{r}}$	Future control reference vector.
$\boldsymbol{W}_{\boldsymbol{x}}$	Output weighting matrix.
$\boldsymbol{W}_{\boldsymbol{u}}$	Input weighting matrix.
$\boldsymbol{\Sigma}_{\boldsymbol{\rho}}$	Diagonal matrix of states weight.
$\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}$	Diagonal matrix of inputs weight.
$\widehat{\boldsymbol{x}}_{\boldsymbol{r}}$	Future predefined reference trajectory vector.
$\boldsymbol{A}_{\boldsymbol{u}}, \boldsymbol{b}_{\boldsymbol{u}}$	Constraints input matrices.
$\boldsymbol{A}_{\boldsymbol{x}}, \boldsymbol{b}_{\boldsymbol{x}}$	Constraints state matrices.
$\hat{\boldsymbol{u}}_{max}$	Vectors with M copies of maximum values.
$\hat{\boldsymbol{u}}_{min}$	Vectors with M copies of minimum values.

$\hat{\boldsymbol{x}}_{min}$	Vectors with N copies of minimum states values.
$\hat{\boldsymbol{x}}_{max}$	Vectors with N copies of maximum states values.
$\bar{\boldsymbol{x}}_n^+$	Denote the maximum normalization value of the states.
$\bar{\boldsymbol{u}}_n^+$	Denote the maximum normalization value of the input.
$\boldsymbol{C_z}$	Output weight matrix of LMI.
$\boldsymbol{D_z}$	Input weight matrix of LMI.
$\boldsymbol{L}$	Terminal value which is a Lyapunov matrix.

Contents

1 Introduction

1.1 Motivation

In the last years, unmanned aerial vehicles (UAVs) have been widely used in civilian and military applications, chiefly due to the technological innovation in fields like embedded computational systems, availability of highly-accurate on-chip sensors and low cost sensors (Papachristos et al., 2011). These vehicles have been initially developed for military applications due to their ability of providing intelligence, surveillance, and reconnaissance information from hostile areas, and the flexibility to explore dangerous environments. However, in recent years, these vehicles have shown potential for missions like remote sensing, cargo transportation, search and rescue, arousing interest in the academic community (Keane & Carr, 2013).

In the literature, there exist two main groups of UAVs, fixed-wing and rotary-wing aircraft. Rotary-wing aerial vehicles are characterized by their high maneuverability and vertical takeoff and landing capability. On the other side, fixed-wing aircraft have better endurance and higher forward speed. Merging these aircraft configurations, a notable hybrid emerges: the tilt-rotor aircraft. This innovative aerial vehicle combines the benefits of both rotary and fixed wing aircraft, employing tiltable rotors to perform transition between helicopter and airplane flight modes. Consequently, the development of this kind of UAVs involves multiple design challenges, especially in terms of control and localization algorithms, which have strict realtime and fast calculation requirements. The control system design complies with specifications that require the embedded system to compute

the control signal within a specified time interval.

Advanced control systems for these vehicles need to achieve good performance in autonomous flight, deal with the highly nonlinear and time-varying behavior of UAVs, reject aerodynamic disturbances, perform obstacle avoidance and aggressive maneuvers, among other requirements. Therefore, the design of these advanced controllers and its implementation in real time pose a significant challenge.

Nowadays, MPC is one of the advanced controllers most used in aerial vehicle applications with relative success (Hartley et al., 2014; Santos et al., 2018; Andrade et al., 2016; Hu et al., 2018; Pang et al., 2021). The main reason of the MPC usage is the capability to deal with predefined trajectory, to compute smooth control actions using future references, and to consider environment constraints in its formulation (Camacho & Alba, 2013).

The main drawback of the use of MPC in realtime applications for complex systems, implemented in a dedicated computational system, is the high computational demand required to compute the control law at each sampling time (Zometa et al., 2012). MPC formulation needs to solve the optimization problem iteratively in realtime, that becomes a challenge for robotic applications which require fast sampling times, with rates greater than 1 MHz (McInerney et al., 2018).

To address the above issues, two main approaches are found in the literature. The first one is related to the hardware used to compute the control law, that is, using microprocessor units, processor units (SoCs), and FPGAs which is one of the most used devices to implement embedded MPC algorithms. The second approach is making optimization algorithms less computationally and power demanding without sacrificing numerical performance and speed, moving towards the so-called embedded optimization (Suardi et al., 2015). In this context, the optimization algorithms are divided into two principal formulations. The first one is based on online fast computation solutions, such as the active set, interior point, and first-order methods. The second approach is called explicit MPC, where the optimization MPC problem is reformulated in a parametric programming problem, and the calculation of the solution based on regions is performed offline. A family of control laws, each one of them associated to a region given by a polytope, are computed off-line, and a search algorithm chooses one of them at each sampling time.

Thus, from the previous discussion, it is clear that the development of robust embedded MPC algorithms for UAV applications is an interesting topic of research.

1.2 Justification

As previously mentioned, advanced control systems for autonomous navigation applications will have a critical role in the next years in the aerial robotic area. Model predictive control is the most used advanced control strategy in industry and it has a big potential

to be used in UAV applications. MPC is theoretically well established and it can cope with MIMO nonlinear complex systems with constraints. However, because of its optimal control nature, its implementation needs the solution of an optimization problem at each sampling time. For UAV applications, it is necessary to implement the MPC algorithm in low resource computers and to compute the control law in the range of milliseconds. Thus, there is an important research line focused on designing and implementing MPC-based control strategies for fast and complex processes like UAVs in low resources computational systems. These efforts aim to meet closed-loop specifications such as stability, robustness, and disturbance rejection.

1.3 Objective

The main objective of this work is to develop optimization techniques that can be used in the Model Predictive Control framework for a class of linear time-varying systems, particularly considering the robustness of the controller based on tubes. These techniques will be developed focusing on the reduction of computational demand in order to allow the implementation of the controller in the embedded computational system of the UAV.

Specifically, it is intended to investigate the following topics:

1. Development of techniques to implement efficiently the tube-based model predictive formulation in embedded systems for trajectory tracking of an UAV;
2. Development of techniques to solve fast and efficiently the optimization problem of a robust model predictive controller for trajectory tracking of an UAV;
3. Validation of the embedded MPC strategies in embedded systems through hardware-in-the-loop simulation.

1.4 Structure of the Text

The remaining of the text is organized as follows:

- Chapter 2: presents a bibliographic review on the formulations related to the tube-based model predictive control for embedded systems;
- Chapter 3: proposes a methodology for the offline parameters computation of the tube-based model predictive control based on zonotopes set representation;
- Chapter 4: proposes two methodologies for fast optimization of the nominal control problem used in the robust MPC formulation;

- Chapter 5: presents the case study of a tiltrotor UAV with a suspended load, along with the results of the tube-based MPC parameter computation for this specific scenario;
- Chapter 6: presents the simulation framework based on hardware-in-the-loop (HIL) used to verify the performance of the proposed controller, along with the results obtained from the implementation of the proposed formulation;
- Chapter 7: summarizes the conclusions of this doctoral thesis and presents future works.

1.5 List of Publications

The following articles were published, accepted for publication, or submitted during the elaboration of this doctoral thesis:

Journal papers:

1. Andrade, R., E. Normey-Rico, J., & V. Raffo, G. (2024c). Tube-based model predictive control based on constrained zonotopes. *IEEE Access*, in press
2. Andrade, R., E. Normey-Rico, J., & V. Raffo, G. (2024b). Tube-based explicit model predictive control for a tiltrotor uav in cargo transportation tasks. *Journal of Control, Automation and Electrical Systems*, in press
3. Andrade, R., E. Normey-Rico, J., & V. Raffo, G. (2024a). Fast embedded tube-based mpc with scaled-symmetric admm for high-order systems: application to load transportation tasks with uavs. *ISA Transactions*, In revision after first round

Conference papers:

1. Andrade, R., Ferramosca, A., E. Normey-Rico, J., & V. Raffo, G. (2020). Explicit Model Predictive Control for a Tiltrotor UAV in Cargo Transportation Tasks. In *Anais do Congresso Brasileiro de Automatica*

2 Literature Review

This chapter reviews the principal themes explored in this doctoral thesis. It explores the tiltrotor UAVs in cargo transportation tasks and their control challenges. Subsequently, it delves into embedded model predictive control and tube-based model predictive control. Furthermore, it offers an overview of the hardware used for implementing embedded model predictive control techniques.

2.1 Tiltrotor UAV in Cargo Transportation Tasks

Unmanned aerial vehicles (UAVs) have increased interest within the academic community, mainly because of their potential to perform civilian missions such as remote sensing, cargo transportation, and search and rescue operations (Keane & Carr, 2013).

A mission currently quite explored with multi-rotor UAVs is cargo transportation due to their inherent advantages, such as mobility and flexibility in takeoff and landing operations (Sunghun & kim hyun su, 2017). Several works have dealt with cargo transportation using UAVs, in which the load is rigidly attached to the aircraft (Loianno et al., 2018; Bahnemann et al., 2017; Pounds & Dollar, 2014) or suspended using cables (Bulka et al., 2022; Ogunbodede & Singh, 2021). Fixed load is the most common practice due to its coupling to the aircraft. However, it is unsuitable for oversized packages and in the case where the landing task is dangerous. With the load suspended by the aircraft, it is possible to carry large volumes, and the delivery can be carried without needing to land the vehicle, improving the efficiency of the mission (Pizetta et al., 2020).

Regarding the UAV configurations used in cargo transportation tasks, there exist two main groups of UAVs, namely fixed-wing and rotary-wing aircraft. Rotary-wing aerial vehicles are characterized by their high maneuverability and vertical takeoff and landing capability (Pereira et al., 2021). On the other side, fixed-wing aircraft have better endurance and higher forward speed (Ramesh & Muruga Lal Jeyan, 2022). Summing up to these aircraft configurations, a hybrid aircraft, such as the tiltrotor one, can combine advantages from rotary and fixed-wing aerial vehicles, which use tiltable rotors to perform transition between helicopter and airplane flight modes (Ducard & Allenspach, 2021).

Consequently, the tiltrotor aircraft becomes a proper configuration to be used with a suspended load since it has the same mobility as a multi-rotor UAV, it can release a package without landing, and it can improve the delivery time by increasing the flight speed using airplane mode.

Commonly, the control strategies found in the literature for suspended load transportation using UAVs are model-based. Two main approaches exist to deriving the dynamic model of this class of systems: i) from the aircraft perspective (Raffo & Almeida, 2018), resulting in the necessity to estimate the load position, mainly in precise load positioning applications; and ii) from the load perspective (Rego & Raffo, 2019), reducing the load swing and letting the direct trajectory tracking of the load.

Based on these approaches, a variety of controllers have been developed for performing suspended load transportation using UAVs, such as nonlinear control strategies (Lv et al., 2021; Raffo & Almeida, 2018), model reference adaptive control (Altan et al., 2018), model predictive control (MPC) (Altan et al., 2017), mixed techniques between tube-based MPC and input-output feedback linearization (Santos et al., 2018), and linear discrete-time mixed H_2/H_∞ controller (Rego & Raffo, 2019).

2.2 Embedded Model Predictive Control

MPC is one of the most advanced controllers used in aerial vehicle applications (Pang et al., 2021; Hu et al., 2018; Andrade et al., 2016; Hartley et al., 2014), which at each sampling instant finds the future optimal control input sequence and predicts the future process outputs using a discrete predicted model, current process states and disturbances.

MPC minimizes an appropriate cost functional subject to input and output constraints, generally, resulting in an optimization of a quadratic programming problem. At the end, the first control signal from the computed optimal sequence is applied to the process, and the MPC algorithm is repeated at the next sampling time (Hrustic & Prljaca, 2020).

The MPC formulation has proven to be a very successful control strategy due to its capability of returning an optimal control sequence satisfying the physical limitations of the system, dealing explicitly with MIMO systems (Zometa et al., 2012). Also, it can take advantages of a predefined trajectory by computing smooth control actions. Its popularity

has increasingly grown over the past three decades, branching out from the chemical industry into other application areas. However, the main drawback of MPC controlling complex fast dynamics systems is the high computational demand required to compute the control law. For this reason, its applicability is often limited to slow dynamic plants where the sampling time can be on the order of seconds or minutes (Jerez et al., 2011).

Recently, the study of the MPC technique has considerably grown, aiming applications through embedded systems (Teixeira, 2018; Hartley et al., 2014; Shoukry et al., 2013), mainly motivated by the necessity of implementing this multivariable formulation in small processing units to control a variety of fast dynamic systems such as automotive active suspension systems (Shoukry et al., 2013), positioning system inside an atomic force microscope (Jerez et al., 2014), aircrafts (Hartley et al., 2014; Ling et al., 2006), and road vehicles (Morari et al., 2003). The MPC is often used to control tiltrotor aerial vehicles in cargo transportation tasks with relative success (Eskandarpour et al., 2023; Santos, 2018). However, as the main drawback of controlling tiltrotor UAVs with the MPC formulation is the sampling time required in the range of milliseconds, the MPC is often used to control only the transnational position of the aircraft (Bauersfeld et al., 2021).

Conversely, as a general definition, an embedded system is an information processing system that is embedded into a larger product (Marwedel, 2003). However, it could be considered as a system composed by a processor unit, memory, and input-output peripheral devices, in which the main characteristic is to have a dedicated function to control an entire device, including electrical-electronic hardware and mechanical parts. Frequently, embedded systems measure the physical world through sensors and control the environment using actuators.

When an embedded system is used to send commands to an electromechanical device, it must meet realtime constraints. Also, it needs to be reliable and maintainable, and achieve high availability and safety (Marwedel, 2003). However, embedded systems have limited resources, since the platforms used to implement them are systems on-a-chip (SoCs) (Bleris et al., 2006) or single board computers, which could have a reduced memory amount and a slow processing unit. The software running in the processor unit of an embedded system is known as a firmware, which is designed to control the physical process and manage time and concurrency in computational systems.

Considering the aforementioned issues, an embedded MPC is defined as the model predictive control algorithm running on a limited resource computational system. The main drawback of the use of MPC in realtime applications for complex systems in a dedicated computational system is the high computational demand required to compute the control law at each sampling time (Zometa et al., 2012). As stated earlier, the control formulation needs to solve the optimization problem iteratively in realtime, which becomes a challenge for robotic applications requiring fast sampling times, with rates greater than 1 MHz (McInerney et al., 2018).

To address the above issues, two main approaches are found in the literature. The first one is related to the hardware used to compute the control law, that is, using microprocessor units (Gulan et al., 2017; Takács et al., 2016; Zometa et al., 2012), processor units (SoCs) (Hrustic & Prljaca, 2020; Palma et al., 2015; Shoukry et al., 2013), and FPGAs, which is one of the most used devices to implement embedded MPC algorithms (McInerney et al., 2018; Peyrl et al., 2015; Xu et al., 2012). It is noteworthy that, after profiling different MPC algorithms, matrix operations are the main consumer of its computation time (Shoukry et al., 2013).

The second approach is making optimization algorithms less computationally and power demanding without sacrificing numerical performance and speed, moving towards the so-called embedded optimization (Suardi et al., 2015). In this context, the optimization algorithms are divided into two principal formulations, as shows in Figure 2.1. The first one is based on online fast computation solutions, such as active set method (Cimini & Bemporad, 2017; Herceg et al., 2015), interior point method (Wills et al., 2011; Lopes et al., 2009), and first-order methods (Patrinos & Bemporad, 2014; O'Donoghue et al., 2013; Cairano et al., 2013). The second approach, is called explicit MPC (Kouramas et al., 2011; Bemporad et al., 2002), where the optimization MPC problem is reformulated in a parametric programming problem, and the calculation of the solution, based on regions, is performed offline. A family of control laws, each one of them associated to a region given by a polytope, are computed offline, and a search algorithm chooses one of them at each sampling time (Bemporad et al., 2002). The total number of regions (critical regions) depends on the number of constraints and the system complexity, which can increase considerably in some cases (Ahmadi-Moshkenani et al., 2018).

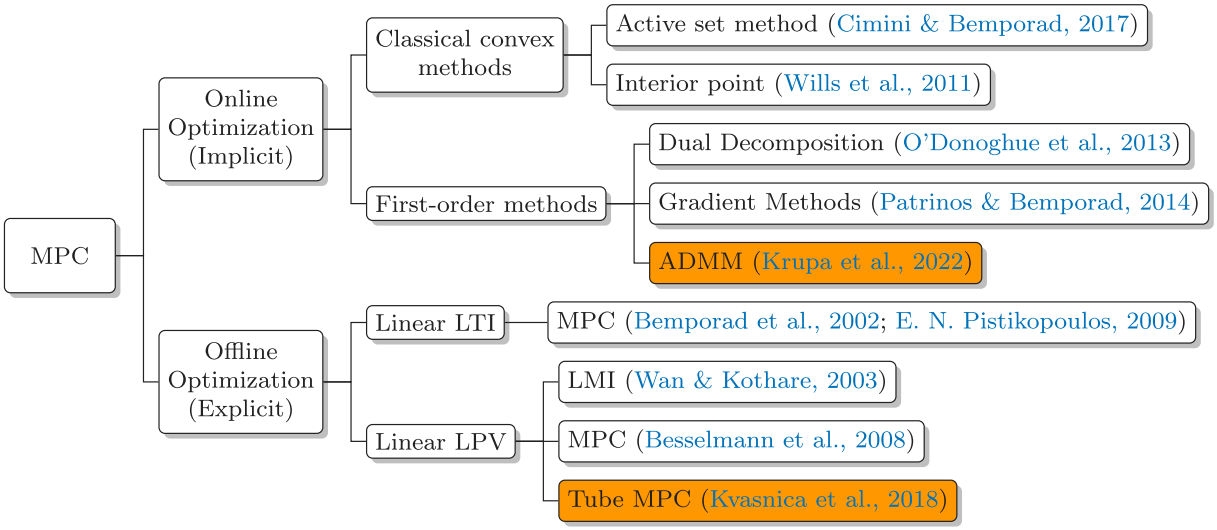

Figure 2.1: Overview of the literature review on Model Predictive Control (MPC) optimization methods, with an emphasis on the key formulations addressed in this thesis.

2.3 Embedded Tube Model Predictive Control

Mechatronic systems are typically represented with time-varying models, which means that some parameters change over time. Additionally, there are always unmodeled dynamics that can interfere with system stability. Hence, robust MPC formulations considering these parameter variations and unmodeled dynamics are required (Hoffmann & Werner, 2015).

One of these robustification methods is the tube-based MPC. This technique considers that the prediction from a nominal model differs from the current evolution of a system due to unmodeled uncertainties (R.Gonzalez et al., 2011). To compensate for this mismatch, in the controller synthesis, a region around the nominal prediction that includes the system's state under any possible uncertainty is calculated using an uncertain model constituted by a restricted collection of unmodeled system behaviors. In the tube-based MPC formulation, as lustrated in Figure 2.2, the control law consists of two terms: i) a state feedback gain that compensates for the differences between the measured and nominal states; and ii) the solution to an optimal control problem based on the nominal model (Mayne et al., 2011).

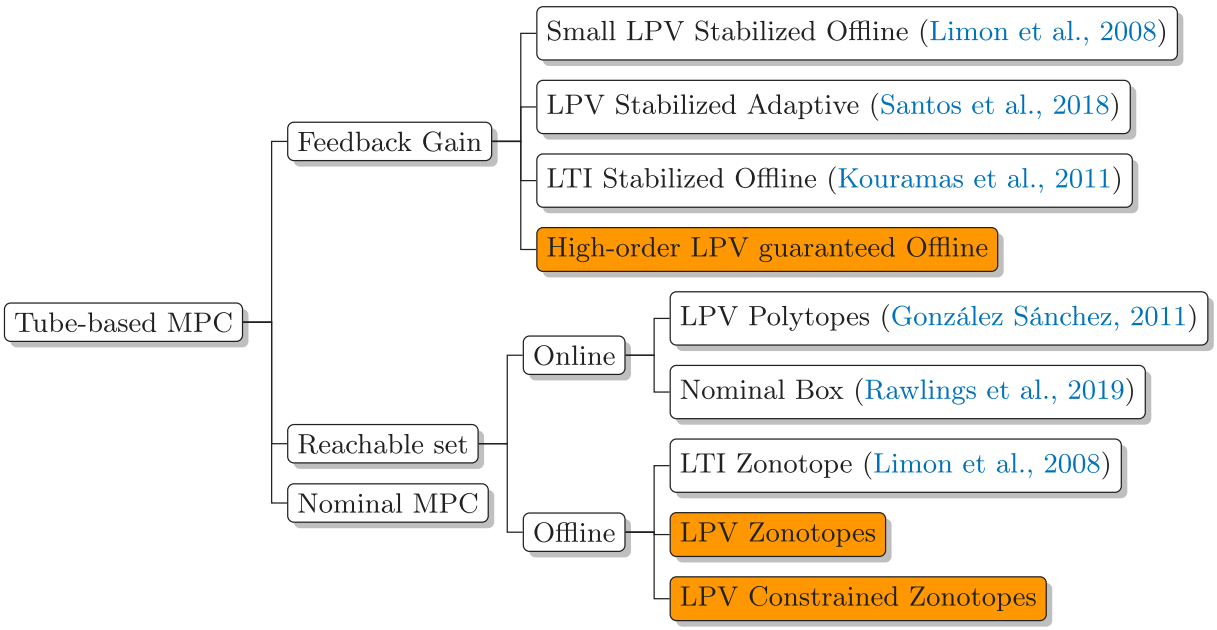

Figure 2.2: Overview of the literature review for each component of the Tube-based Model Predictive Control (TMPC) formulation, emphasizing the key research areas addressed in this thesis that contribute to the implementation of the formulation.

Considering the first term of the tube-based control law, there are two ways to compute the state feedback gain. The first approach involves the offline computation of the state feedback gain that stabilizes the linear time-invariant (LTI) nominal model using well-known methods such as the linear quadratic regulator (LQR) (Jeong & Choi, 2024; Hang et al., 2022) and H_∞ control (Alcalá et al., 2020; Gonzalez et al., 2011). Alternatively, some approaches use an LTI model with additive bounded uncertainty. However, in the

latter case, the algorithm uses uncertainty set vertices during computation, which could be problematic when dealing with high-order systems (Limon et al., 2008). Also, linear time-varying (LTV) models are not considered in the offline approaches, even being an important issue in the robotics area. The other way is the online computation of the feedback gain by minimizing a Lyapunov function based on a LTV model without additive uncertainty (Santos, 2018; R.Gonzalez et al., 2011). Although these formulations consider LTV systems, uncertainties are not considered in the computation. Also, since the feedback gain is computed online, the other parameters, such as the reachable and nominal sets, are computed at each sampling time, increasing the computational cost in a high-order system.

To compute the nominal optimal control sequence in the tube-based MPC, an optimal control problem (OCP) is formulated. This is constrained by the nominal model, in addition to the nominal control and state sets derived from the reachable set of the LTI model with additive uncertainty. Two common approaches exist for computing the reachable set of a system: i) offline computation, which bounds the system variation in an additive term (Han et al., 2016; Limon et al., 2008); and ii) online computation, which compensates for the realizations of the current states (R.Gonzalez et al., 2011).

Although the offline reachable set computation is more suitable from the computational cost perspective, it has some limitations. This approach employs a static local control law to compensate for uncertainties, leading to conservatism in the control design and increasing the set size. Furthermore, there is no consolidated methodology for the offline computation of the reachable set for high-order dynamical systems using time-varying models (Han et al., 2016). Conversely, the online computation of the reachable set has the advantage of obtaining a tighter reachable set by using an adaptive local control law, reducing the conservativeness. Nevertheless, for high-order dynamical systems with more than ten states, the computational feasibility of the reachable set decreases significantly (R.Gonzalez et al., 2011).

The literature describes three types of set representation for reachable sets for both online and offline formulations. The first approach uses the symmetric box approximation of the uncertainty set to simplify the computation of the reachable set at the cost of losing accuracy and increasing conservatism (Rawlings et al., 2019). The second approach uses polytopes to represent the uncertainty set, with the reachable set computation performed through the iterative application of Minkowski sum (Gonzalez et al., 2011). However, this approach is computationally inefficient and could be infeasible for systems with more than ten states. Finally, the third approach defines the reachable set as a positively invariant set using zonotopic representation. This approach is computationally efficient and faster than polytopes, making it suitable for high-order dynamical systems. However, this approach has been designed only to handle time-invariant systems (Han et al., 2016; Limon et al., 2008).

2.3.1 Offline Solution of the Nominal Optimization Problem

The offline solution for the nominal optimization problem has been widely used for embedded applications throughout the years, specifically in the classical model predictive control area (Bemporad et al., 2002). In addition, the works developed with explicit optimization can be applied directly to the Tube MPC formulation, since the optimization problem is similar to the standard MPC.

The explicit MPC has been introduced by Bemporad et al. (2002), which solves the optimization problem via multi-parametric optimization, whith the constraint solution domain being divided into critical regions. Then, for each critical region is computed a control law, which is an affine function of the current state. The total number of critical regions depends on the number of constraints and the system complexity, which can grow considerably (Ahmadi-Moshkenani et al., 2018). Thus, its application for large-scale systems becomes prohibitive due to the large amount of hardware memory consumed by storing the explicit solutions (Gulan et al., 2017).

Along the years, explicit MPC has been deeply studied by the control community to improve the computation of the critical regions. One of the performed studies focused on the reduction of unnecessary partitions of the search space, and consequently, reducing the complexity of the optimization problem of the algorithm (Schulze et al., 2022; Tøndel et al., 2003). This explicit approach has been tested in robotic and automotive applications involving microprocessing and processing systems with success, using linear time invariant (LTI) models to represent the process (E. N. Pistikopoulos, 2009).

However, in many robotic applications, the process dynamics can be better represented by linear time-varying (LTV) models derived from a linearization thoughtout a defined trajectory. Despite this better representation, it makes the computation process of an explicit solution more complex. Some works have tackled this problem in the literature. In Besselmann et al. (2008), the LTV model has been described as a parameter-varying system, in which the schedule parameter lies in a polytope, with its maximum and minimum values representing the polytope vertices. Then, the optimization problem is written as a function of the schedule parameter, and the cost function is transformed into a problem constraint (epigraph formulation). The main drawback of this formulation is the necessity of a more complex optimization solution due to the increased number of constraints, also increasing the number of critical regions.

In Wan & Kothare (2003), an explicit solution of the MPC optimization problem of an LTV system has been computed through a linear matrix inequality (LMI) formulation. In this approach, a set of control laws corresponding to a group of invariant ellipsoids is computed offline. These ellipsoids are contained within one another in the state space and are obtained from a group of arbitrary discrete points of the state vector. However, it requires to perform a second on line step, in which a bisection search between the ellipsoids

is performed to find the smallest ellipsoid containing the current state, and finally the control law is computed. The main advantage of this technique is the ability to deal with large-scale systems. In addition, this kind of formulation can be used to robustify the MPC when dealing with uncertain systems.

An alternative strategy to deal with LTV systems is to split their models into an invariant and a variant part. Then, the explicit MPC solution of the invariant system is robustified with a second control law which takes into account the effect of the variant part compensating the differences between the models. Consequently, the parametric optimization problem is defined using only the LTI part of the model, which makes it more computationally efficient. In addition, the new state and control constraint sets are derived from the second control law (Kouramas et al., 2011). Note that, this approach is close to the tube-based MPC technique, in which two system models are used, a linear nominal model and its extension with an additive uncertainty (Rawlings et al., 2019). Then, the optimum nominal control sequence is obtained based on the nominal model and the nominal control and states sets, which are derived from the reachable set disregarding the uncertainty set. The second part is a feedback gain used to compensate the differences between the real and nominal state vector. This second control law is computed using the linear uncertainty model.

Consequently, based on the similarity of these two last approaches, it is possible to formulate the tube-based MPC for LPV systems, dealing now with both time varying parameters and unknown-but-bounded uncertainties. Therefore, after splitting the LPV model into invariant and variant parts, the latter considers both time varying parameters and uncertainties towards the robustification of the nominal controller (González Sánchez, 2011). Despite the clear advantages provided by this strategy, the reachable set must be computed at each sampling time, which might be computationally prohibitive when dealing with large-scale systems (Limon et al., 2008).

2.3.2 Online Solution of the Nominal Optimization Problem

The online solution for the nominal optimization problem has been widely studied for embedded applications in the last years, seeking for more efficient algorithms to solve the QP problem, by exploiting its particular structures which arise from MPC formulation.

As mentioned before, three main classical convex optimization algorithm classes have being adapted to use in embedded systems, such as the active set method (Cimini & Bemporad, 2017; Herceg et al., 2015), the interior point method (Wills et al., 2011; Lopes et al., 2009), and first-order methods (Patrinos & Bemporad, 2014; O'Donoghue et al., 2013; Cairano et al., 2013). These algorithms have several variants, and many of them have been implemented on embedded systems. In the Interior-Point Method (IPM), the optimal solution is found by solving the nonlinear Karush-Kuhn-Tucker (KKT) system

equations (Ling et al., 2006). This system is solved using an iterative Newton's method, with each iteration consisting of three main steps:

1. Update/linearize the KKT system;
2. Solve the linear KKT system for a step direction;
3. Update the current guess with the step direction.

The main source of computational burden is solving the KKT linear system (Step 1). It has been shown that exploiting the structure of the matrices arising into the KKT problem leads to faster and more scalable solvers.

The Active Set Method (ASM) algorithm finds the optimal point by solving a sequence of equality-constrained subproblems, locating the set of active inequality constraints at optimality. In each iteration, the inequality constraints are converted to equality constraints based upon whether they have been violated in the previous iteration (Yang et al., 2012; Wills et al., 2012).

These two methodologies are oriented to perform traditional optimization algorithms, and some variants are oriented to perform the computation in a parallel way in order to accelerate the computation time of the optimum solution. However, the drawback of these algorithms is related to the sequential manner of solving the optimization, while parallel programming is only used for matrix multiplication.

More recently, methods centered on the idea of performing an iterative computation of the optimum solution using Lagrange multipliers have been studied. These formulations are complemented using fast gradient methods to accelerate the optimization, such as Nesterov Fast Gradient Method (FGM) (Jerez et al., 2014), and the Alternating Direction Method of Multipliers (ADMM) (Zhang et al., 2018; Dang et al., 2015).

In these algorithms, the optimization problem is decomposed into small optimization subproblems, which are solved iteratively until achieving a general solution of the master problem (Goldstein et al., 2014). This approach is widely used to perform decentralized optimization in the model predictive control area, in which a group of agents are used to compute a heavy optimization problem.

These methods utilize only first-order information (e.g., the gradient) in their computations and generally have three main steps (Boyd et al., 2010):

1. Computation of a search direction;
2. Computation of the step size and application of the search direction;
3. Computation of the constraint satisfaction.

Its main drawback is the constraint satisfaction, in Step 3, as it can become computationally complex depending on the problem. However, the first-order methods using a fast

gradient method (specifically, ADMM methods) are particularly attractive for applications of MPC in embedded control systems due to the high parallelization capability of the algorithm (Krupa et al., 2022). Besides, in ADMM methods, the matrix-vector multiplication is the major linear algebra operation, which is less computationally expensive compared with the linear system solver in IPM or ASM approaches. Also, they are relatively easy to implement and have high performance certification guarantees (Jerez et al., 2014).

2.4 Hardware Used for Embedded Model Predictive Control

This section focuses on the embedded systems used in implementing tube-based MPC. However, due to the lack of literature on embedded tube-based MPC formulations, the discussion also considers the embedded hardware platforms commonly employed for standard MPC implementations. This approach is relevant as the robust formulation computes the optimal solution of a nominal MPC.

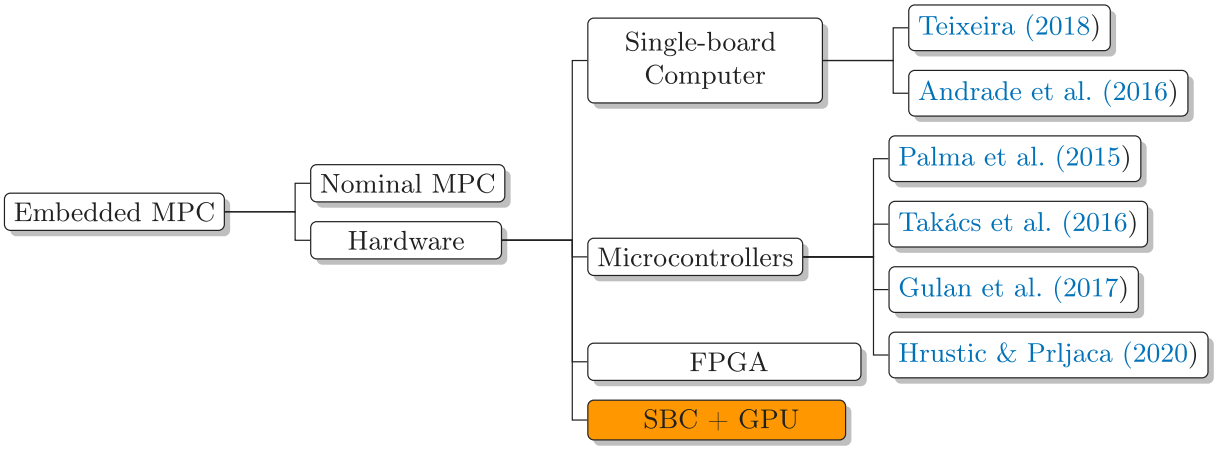

Figure 2.3: Overview of the literature review on devices used for the implementation of Tube-based Model Predictive Control (TMPC), with emphasis on the hardware platforms addressed in this thesis.

A shown in Figure 2.3, it is possible to find three main devices used to implement embedded MPC. The first one uses microcontrollers, which are small processor unites that contain a CPU, small memory, and peripherals, all in only one chip. The CPU generally used is an ARM Cortex M family (Hrustic & Prljaca, 2020; Takács et al., 2016; Palma et al., 2015; Zometa et al., 2012). Also in some implementations, it is used 8 bits processor units, which are microcontrollers with a reduced processing capability compared with an ARM microprocessor (Gulan et al., 2017). The main advantage of these devices is the realtime implementation using a realtime operating system (RTOS) that ensures the execution time of the task and are less susceptible to software failures. However, as the resources are very limited, they are used to control low-order small-scale systems.

The second device class is the FPGAs, which are semiconductor devices that are based on a matrix of configurable logic blocks (CLBs) connected via programmable interconnections. Since the most costly operations in MPC involve matrix and vectors (Bleris et al., 2006), FPGAs devices can be the most suitable hardware to perform these operations due to the ability to perform parallel processing. In this context, some works has been developed, as shown in Table 1, experimenting with a whole variety of optimization algorithms using FPGAs. Nevertheless, the main issue of using FPGAs is that the current programming environments do not have tools as advanced as modern programming languages. Many of the abstractions typically available in general purpose systems, such as high-level languages, abundant libraries, code portability, and automatic resource management, do not exist for FPGAs.

Table 2.1: Model Predictive Control implementation using FPGAs

Source	QP Problem[1]	Algorithm	QP Size[2]	Solver Time
Ling et al. (2006)	D	IP	3/0/60	23.7 ms
Ling et al. (2008)	D	IP	3/0/52	09.1 ms
Vouzis et al. (2009)	D	Newton	2/0/4	688 μs
Basterretxea & Benkrid (2011)	D	IP	3/0/6	120 μs
Wills et al. (2011)	D	IP	12/0/24	< 200 μs
Yang et al. (2012)	D	ASM	3/0/6	20 μs
Wills et al. (2012)	D	ASM	12/0/24	< 30 μs
Peyrl et al. (2015)	D	FGM	15/0/30	0.49 μs
Jerez et al. (2014)	D	FGM	40/0/80	0.53/0.91 μs
Rubagotti et al. (2016)	D	DGP	20/0/208	239 μs
Liu et al. (2014)	S	IP	300/-/600	4 ms
Hartley et al. (2014)	S	IP	377/-/408	12 ms
Jerez et al. (2014)	S	ADMM	216/-/172	4.9/8.52 $m\mu s$
Dang et al. (2015)	S	ADMM	120/80/250	215 ms
Shukla et al. (2017)	S	AMA	384/-/-	900 μs
Zhang et al. (2018)	S	ADMM	204/-/300	30.1 μs

[1] D - Condensed (Dense) formulation, S - Uncondensed (Sparse) formulation.

[2] Decision Variables/Equality Constraints/Inequality Constraints

Therefore, creating a complete application using FPGAs often requires that programmers bring up not only their application, but a significant infrastructure including memory controllers, I/O systems, and drivers for the sensors and actuators. All of these activities require large amounts of time, slowing the FPGA development process (Fleming et al., 2014).

The third class of hardwares is the single-board computer (SBC), which is a complete computer built on a single circuit board, with a microprocessor, memory, I/O system, and other features required of a functional computer. These hardwares currently work with multicore microprocessors that allow parallel processing, and generally, the processors are from the ARM Cortex A family (Teixeira, 2018; Andrade et al., 2016).

The main advantage of these hardwares are to perform realtime computation using an operating system designed for this application class while maintaining high-level languages, abundant libraries, and code portability. Moreover, they have internally the NEON technology, which is a single-instruction multiple data (SIMD) feature, providing an efficient implementation of vector and matrix operations (Bleris et al., 2006).

In the last years, a class of system has been developed to combine the features of FPGA related to the parallel computation with characteristics of an SBC. This kind of hardware is a single-board computer integrated with a graphic processor unit (GPU), which brings the possibility to operate matrix operations in parallel, while a multi-core processor performs other remaining tasks of the application (Smith, 2017). Thus, the graphics unit can execute the optimization problem with a custom algorithm designed to run in a parallel fashion.

2.5 Final Remarks

In this chapter, a specific bibliographic review was presented for embedded model predictive control. Topics related to robustification of MPC based on tubes, predictive control online optimization, explicit model predictive optimization, and hardware used to implement predictive control, were addressed to provide the state of the art on the implementation of tube model predictive control strategies into embedded systems.

Taking into consideration the existing literature and the objective of implementing the tube-based model predictive control in a low computational resource system, the contributions of this doctoral project are focused in the design of custom optimization algorithms to solve the Tube MPC formulation online and offline. Concerning the embedded Tube MPC formulation, the contributions are the extension of the computation of reachable sets for a time-varying and large-scale systems using the zonotope approach.

Thus, for the online optimization of the Tube MPC, it is possible to use ADMM, which has proven to be the fastest algorithm to control small systems, using MPC and implementing it in FPGA (Zhang et al., 2018; Jerez et al., 2014). Also, due to the high parallelizable capability of ADMM, it can be easily implemented using parallel computing. Therefore, some of the optimization algorithms and control formulations proposed in this doctoral project aim to extend the results presented in Krupa et al. (2022), Zhang et al. (2018), and Jerez et al. (2011), to control fast dynamic and large-scale systems using the tube-based robust formulation of the MPC problem.

Additionally, as mentioned previously, the explicit MPC formulation for a time-varying model is still an open issue, since some formulations solve optimization problems with quadratic constraints (Besselmann et al., 2008). Either, the aforementioned works find the optimal solution only for canonical system representation, which might not be feasible for more complex models (Kvasnica et al., 2018). Thus, another objective in this doctoral

research is the generalization of the explicit formulation to large-scale and time-varying model, extending the idea presented in Kvasnica et al. (2018) by the application of the Tube-based MPC formulation.

Additionally, a methodology must be developed to compute the reachable set used into the tube-based robust formulation for large-scale systems, since the presented literature review lacks studies considering high-order systems. Thus, this doctoral project intends to extend the results presented in Limon et al. (2008) to time-varying models using the zonotopic approach.

3 Tube-based Model Predictive Control

In this chapter, a modified Tube-based model predictive control (TMPC) formulation tailored for high-order and fast dynamic systems is presented. The proposed approach enhances computational efficiency by employing zonotope set representation for the reachable set. To tackle the challenges posed by high-order and fast dynamic systems, we propose an offline computation of the reachable set as a Robust Positive Invariant Set (RPI). This innovative method integrates a time-varying linear model with additive uncertainty and its associated constraints in the reachable set computation, ensuring the existence of both nominal control and state sets.

3.1 Problem Formulation

In this thesis, it is assumed that the real process has its dynamics well represented by a linear parameter-varying (LPV) system, and this particular model structure is used to design the control formulation. Consider the following LPV system:

$$\boldsymbol{x_a}(k+1) = \boldsymbol{A_z}(\boldsymbol{\sigma})\boldsymbol{x_a}(k) + \boldsymbol{B_z}\boldsymbol{u}(k) + \boldsymbol{D_z}\boldsymbol{d}(k), \tag{3.1}$$

where $\boldsymbol{x_a} \in \mathcal{X} \subset \mathbb{R}^n$, $\boldsymbol{u} \in \mathcal{U} \subset \mathbb{R}^m$ and $\boldsymbol{d} \in \mathcal{D} \subset \mathbb{R}^l$ are the state, control, and disturbance vectors, respectively. The parameter-varying state matrix $\boldsymbol{A}_z(\sigma) \in \mathbb{R}^{n\times n}$ is defined as $\boldsymbol{A}_z(\sigma) = \bar{\boldsymbol{A}}_z + \Delta\boldsymbol{A}_z$, where $\bar{\boldsymbol{A}}_z \in \mathbb{R}^{n\times n}$ is the linear invariant part and $\Delta\boldsymbol{A}_z \in \mathbb{R}^{n\times n}$ is the varying part of the matrix. Additionally, it is assumed that the state matrix $\boldsymbol{A_z}(\boldsymbol{\sigma})$ is

affine with respect to the parameters $\boldsymbol{\sigma} \in \mathbb{R}^p$, where $\boldsymbol{\sigma}_{min} \leq \boldsymbol{\sigma} \leq \boldsymbol{\sigma}_{max}$. Also, the control matrix is $\boldsymbol{B}_z \in \mathbb{R}^{n \times m}$, and $\boldsymbol{D}_z \in \mathbb{R}^{n \times l}$ is the disturbance matrix.

Let also consider the uncertainty vector $\boldsymbol{w} \in \mathcal{W} \subset \mathbb{R}^n$ defined by the set operation $\mathcal{W} = \mathcal{W}_u \oplus \mathcal{W}_d$, where $\mathcal{W}_u = \{\boldsymbol{w_u} \mid \boldsymbol{w}_{u_{min}} \leq \boldsymbol{w_u} \leq \boldsymbol{w}_{umax}\}$ is the parametric and structural uncertainty set, while the external disturbance set is defined by $\mathcal{W}_d = \boldsymbol{D_z}\mathcal{D}$ with $\mathcal{D} = \{\boldsymbol{d} \mid \boldsymbol{d}_{min} \leq \boldsymbol{d} \leq \boldsymbol{d}_{max}\}$. Thus, the tube-based MPC formulation initiates by defining the discrete LPV system with additive uncertainty as follows

$$\boldsymbol{x_a}(k+1) = \boldsymbol{A_z}(\sigma)\boldsymbol{x_a}(k) + \boldsymbol{B_z}\boldsymbol{u}(k) + \boldsymbol{w}(k). \tag{3.2}$$

Then, by considering the parameter-varying state matrix $\boldsymbol{A}_z(\boldsymbol{\sigma})$ as the sum of a linear invariant component $\bar{\boldsymbol{A}}_z$ at $\boldsymbol{\sigma} = \boldsymbol{0}$ and a time-varying component $\Delta\boldsymbol{A}_z(\boldsymbol{\sigma})$ that varies within a convex combination of the maximum and minimum parameter values, the discrete nominal model is given by

$$\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k+1) = \bar{\boldsymbol{A}}_{\boldsymbol{z}}\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) + \boldsymbol{B_z}\bar{\boldsymbol{u}}(k), \tag{3.3}$$

where $\bar{\boldsymbol{x}}_{\boldsymbol{a}} \in \bar{\mathcal{X}} \subset \mathbb{R}^n$, $\bar{\boldsymbol{u}} \in \bar{\mathcal{U}} \subset \mathbb{R}^m$, with $\bar{\mathcal{X}}$ and $\bar{\mathcal{U}}$ being the nominal state and control sets, respectively.

Thus, the main goal of the proposed approach is: (i) to solve the nominal MPC problem presented in Section 3.5, which considers the nominal model constrained to the nominal state and control sets, aiming to obtain $\bar{\boldsymbol{u}}(k)$; and (ii) to compute an additional state-feedback control law added to the nominal optimal control sequence, as follows

$$\boldsymbol{u}(k) = \bar{\boldsymbol{u}}(k) + \boldsymbol{K}(\boldsymbol{x_a}(k) - \bar{\boldsymbol{x}}_{\boldsymbol{a}}(k)). \tag{3.4}$$

Note that, as the model used in the MPC optimization problem dismisses the varying part, a second control law is used to compensate for the differences between the models, using a feedback gain and the error between the nominal and real states.

In addition, the nominal control and state sets must be computed, which are obtained through the Pontryagin difference as

$$\bar{\mathcal{X}} = \mathcal{X} \ominus \mathcal{Z}, \tag{3.5}$$

$$\bar{\mathcal{U}} = \mathcal{U} \ominus \boldsymbol{K}\mathcal{Z}, \tag{3.6}$$

where $\mathcal{Z}$ is the reachable set computed using the discrete-time linear system with additive uncertainty, and $\boldsymbol{K}$ is a feedback gain that must be designed to stabilize the LPV system.

The approach used to compute the feedback gain is presented in Section 3.2, while the Reachable set $\mathcal{Z}$ considering the LPV model is obtained by the a new formulation proposed in Section 3.3.

3.2 Computation of the Feedback Gain

This section introduces a methodology for computing the feedback gain, $\boldsymbol{K}$, used in the tube-based MPC control law (3.4). This control law aims to compensate for the discrepancies between the nominal model and the uncertain parameter-varying models. Notably, this methodology focuses on the application to high-order systems.

Initially, we define the uncertainty set $\mathcal{W}$ as a zonotope, which is posed as follows.

Definition 3.1. *A set $\mathcal{W}$ is a zonotope if there exists the pair $(\boldsymbol{H_w}, \boldsymbol{w_0}) \in \mathbb{R}^{n\times r} \times \mathbb{R}^n$ such that*

$$\mathcal{W} = \{\boldsymbol{w_0} + \boldsymbol{H_w}\zeta : \zeta \in \mathbb{R}^r, \|\zeta\|_\infty \leq 1\}, \tag{3.7}$$

where $\boldsymbol{w_0}$ is the center and $\boldsymbol{H_w}$ is the generator matrix of the zonotope.

Also, we consider the following lemma about the equivalence between l_2 and l_∞ norms.

Lemma 3.1 (Lemma 2 in Han et al. (2016))**.** *The l_2 and l_∞ norms for a given finite dimensional vector space satisfy the inequality*

$$\|\zeta\|_\infty \leq \|\zeta\|_2 \leq \sqrt{n}\|\zeta\|_\infty. \tag{3.8}$$

Then, the computation of the feedback gain $\boldsymbol{K}$ for the stability of an LPV closed-loop system is provided by the following theorem.

Theorem 3.1. *Consider the linear parameter-varying closed-loop system with additive uncertainty*

$$\boldsymbol{x_a}(k+1) = (\boldsymbol{A_z}(\boldsymbol{\sigma}) + \boldsymbol{B_z}\boldsymbol{K})\boldsymbol{x_a}(k) + \boldsymbol{w}(k), \tag{3.9}$$

for which the pair $(\boldsymbol{A_z}(\boldsymbol{\sigma}), \boldsymbol{B_z})$ is assumed controllable, with $\boldsymbol{A_z}(\boldsymbol{\sigma})$ being a group of matrices representing each vertex of the polytope formed by the maximum and minimum values of the parameter $\boldsymbol{\sigma}$, $\boldsymbol{K}\boldsymbol{x_a} \in \mathcal{U}$, and $\boldsymbol{w} \in \mathcal{W}$, representing $\mathcal{W}$ as a zonotope. Consider also the matrices $\boldsymbol{W} > 0$, $\boldsymbol{Y} > 0$, $\boldsymbol{C_q} > 0$, $\boldsymbol{D_q} > 0$, $\boldsymbol{C_q'}\boldsymbol{D_q} = 0$, the parameter $\kappa > 0$, the convergence rate λ, and the control input bound u_{max}. Then, there exists a feedback gain $\boldsymbol{K}$ which ensures the stability of the uncertain parameter-varying system and is computed by solving the following optimization problem subjected to LMI conditions:

$$
\begin{aligned}
&\min \quad \kappa \qquad (3.10)\\
&\text{subject to :}\\
&\begin{bmatrix} \kappa & 0 \\ 0 & \boldsymbol{W} \end{bmatrix} > 0,\\
&\begin{bmatrix} -\boldsymbol{W} & \boldsymbol{W}\boldsymbol{C}_q' + \boldsymbol{Y}'\boldsymbol{D}_q' & \boldsymbol{W}\boldsymbol{A}_z(\boldsymbol{\sigma})' + \boldsymbol{Y}'\boldsymbol{B}_z' \\ \boldsymbol{C}_q\boldsymbol{W} + \boldsymbol{D}_q\boldsymbol{Y} & -I & 0 \\ \boldsymbol{A}_z(\boldsymbol{\sigma})\boldsymbol{W} + \boldsymbol{B}_z\boldsymbol{Y} & 0 & -\boldsymbol{W} \end{bmatrix} < 0,\\
&\begin{bmatrix} \lambda\boldsymbol{W} & 0 & \boldsymbol{W}\boldsymbol{A}_z(\boldsymbol{\sigma})' + \boldsymbol{Y}'\boldsymbol{B}_z' \\ 0 & 1-\lambda & \zeta'\boldsymbol{H}_w' \\ \boldsymbol{A}_z(\boldsymbol{\sigma})\boldsymbol{W} + \boldsymbol{B}_z\boldsymbol{Y} & \boldsymbol{H}_w\zeta & \boldsymbol{W} \end{bmatrix} \geq 0,\\
&\begin{bmatrix} 1 & \zeta' \\ \zeta & I \end{bmatrix} \geq 0,\\
&\begin{bmatrix} u_{max}^2 & \boldsymbol{Y} \\ \boldsymbol{Y}' & \boldsymbol{W} \end{bmatrix} \geq 0,\\
&\boldsymbol{W} > 0.
\end{aligned}
$$

Proof. First, we consider the ellipsoid $\varepsilon(\boldsymbol{L}, 1) = \{\boldsymbol{x}_a' \in \mathbb{R}^n : \boldsymbol{x}_a'\boldsymbol{L}\boldsymbol{x}_a \leq 1\}$, defined as a robust invariant set, with $\boldsymbol{L}$ being a positive-definite matrix. In order to satisfy the condition $\varepsilon(k+1)(\boldsymbol{L}, 1) \subset \varepsilon(\boldsymbol{L}, 1)$, the following inequality is posed:

$$(1 - \boldsymbol{x}_a(k+1)'\boldsymbol{L}\boldsymbol{x}_a(k+1)) - \lambda(1 - \boldsymbol{x}_a(k)'\boldsymbol{L}\boldsymbol{x}_a(k)) \geq 0, \qquad (3.11)$$

which turns the system λ-stable, meaning that the system is stable with the convergence rate λ. Considering the closed-loop uncertainty system (3.9), inequality (3.11) can be written in a follows

$$\begin{bmatrix} \boldsymbol{x}_a'(k) \\ 1 \end{bmatrix}' \begin{bmatrix} \lambda\boldsymbol{L} - \boldsymbol{A}_k(\boldsymbol{\sigma})'\boldsymbol{L}\boldsymbol{A}_k(\boldsymbol{\sigma}) & -\boldsymbol{A}_k(\boldsymbol{\sigma})'\boldsymbol{L}\boldsymbol{w} \\ -\boldsymbol{w}'\boldsymbol{L}\boldsymbol{A}_k(\boldsymbol{\sigma}) & (1-\lambda) - \boldsymbol{w}'\boldsymbol{L}\boldsymbol{w} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_a(k) \\ 1 \end{bmatrix} \geq 0. \qquad (3.12)$$

By applying Schur's complement to the matrix of (3.12), the following inequality is obtained:

$$\begin{bmatrix} \lambda\boldsymbol{L} & 0 \\ 0 & (1-\lambda) \end{bmatrix} - \begin{bmatrix} \boldsymbol{A}_k(\boldsymbol{\sigma})' \\ \boldsymbol{w}' \end{bmatrix} \boldsymbol{L} \begin{bmatrix} \boldsymbol{A}_k(\boldsymbol{\sigma}) & \boldsymbol{w} \end{bmatrix} \geq 0, \qquad (3.13)$$

Thereafter, it is written as the following LMI:

$$\begin{bmatrix} \lambda \boldsymbol{L} & 0 & \boldsymbol{A_k}(\boldsymbol{\sigma})' \\ 0 & 1-\lambda & \boldsymbol{w}' \\ \boldsymbol{A_k}(\boldsymbol{\sigma}) & \boldsymbol{w} & \boldsymbol{L}^{-1} \end{bmatrix} \geq 0. \tag{3.14}$$

Therefore, if we use definition (3.1), assuming $\boldsymbol{w_0} = 0$ and defining $\boldsymbol{W} = \boldsymbol{L}^{-1} > 0$, and $\boldsymbol{K} = \boldsymbol{Y}\boldsymbol{W}^{-1}$, the following LMI is obtained pre and post multiplying (3.14) the diagonal matrix $\mathbf{diag}([\boldsymbol{W}\ \boldsymbol{I}\ \boldsymbol{I}])$:

$$\begin{bmatrix} \lambda \boldsymbol{W} & 0 & \boldsymbol{W}\boldsymbol{A_z}(\boldsymbol{\sigma})' + \boldsymbol{Y}'\boldsymbol{B_z}' \\ 0 & 1-\lambda & \zeta' \boldsymbol{H_w}' \\ \boldsymbol{A_z}(\boldsymbol{\sigma})\boldsymbol{W} + \boldsymbol{B_z}\boldsymbol{Y} & \boldsymbol{H_w}\zeta & \boldsymbol{W} \end{bmatrix} \geq 0, \tag{3.15}$$

To guarantee the existence of $\bar{\mathcal{U}}$, for all $x \in \varepsilon(\boldsymbol{L}, 1)$, the condition $max|\boldsymbol{K}x| \leq u_{max}$ must be satisfied using the inequality $u_{max}^2 - \boldsymbol{u}(k)'\boldsymbol{L}\boldsymbol{u}(k) \geq 0$, which is written in the following form by considering the control $\boldsymbol{u}(k) = \boldsymbol{K}\boldsymbol{x_a}(k)$:

$$\begin{bmatrix} 1 \\ \boldsymbol{x_a}'(k) \end{bmatrix}' \begin{bmatrix} u_{max}^2 & 0 \\ 0 & -\boldsymbol{K}'\boldsymbol{L}\boldsymbol{K} \end{bmatrix} \begin{bmatrix} 1 \\ \boldsymbol{x_a}(k) \end{bmatrix}' \geq 0. \tag{3.16}$$

Then, inequality (3.16) can be posed in the Schur´s complement form as

$$\begin{bmatrix} u_{max}^2 & 0 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 \\ \boldsymbol{K}' \end{bmatrix} \boldsymbol{L} \begin{bmatrix} 0 & \boldsymbol{K} \end{bmatrix} \geq 0, \tag{3.17}$$

which leads to the following LMI:

$$\begin{bmatrix} u_{max}^2 & 0 & 0 \\ 0 & 0 & \boldsymbol{Y}' \\ 0 & \boldsymbol{Y} & \boldsymbol{W} \end{bmatrix} \geq 0. \tag{3.18}$$

It is worth highlighting that LMI (3.18) can be reduced to the equivalent form presented in (3.10), which is given by:

$$\begin{bmatrix} u_{max}^2 & \boldsymbol{Y}' \\ \boldsymbol{Y} & \boldsymbol{W} \end{bmatrix} \geq 0. \tag{3.19}$$

To ensure stability of the closed-loop system, the following discrete Riccati equation is considered (Andrade et al., 2016):

$$(\boldsymbol{A_z}(\boldsymbol{\sigma}) + \boldsymbol{B_z}\boldsymbol{K})'\boldsymbol{W}(\boldsymbol{A_z}(\boldsymbol{\sigma}) + \boldsymbol{B_z}\boldsymbol{K}) - \boldsymbol{W} + \boldsymbol{C_q}'\boldsymbol{C_q} + \boldsymbol{K}'\boldsymbol{D_q}'\boldsymbol{D_q}\boldsymbol{K} < 0, \tag{3.20}$$

which is rewritten as the following LMI:

$$\begin{bmatrix} -\boldsymbol{W} & \boldsymbol{W}\boldsymbol{C}_q' + \boldsymbol{Y}'\boldsymbol{D}_q' & \boldsymbol{W}\boldsymbol{A}_z(\boldsymbol{\sigma})' + \boldsymbol{Y}'\boldsymbol{B}_z' \\ \boldsymbol{C}_q\boldsymbol{W} + \boldsymbol{D}_q\boldsymbol{Y} & -I & 0 \\ \boldsymbol{A}_z(\boldsymbol{\sigma})\boldsymbol{W} + \boldsymbol{B}_z\boldsymbol{Y} & 0 & -\boldsymbol{W} \end{bmatrix} < 0, \tag{3.21}$$

where $\boldsymbol{C}_q'\boldsymbol{C}_q = \boldsymbol{\Sigma}_\rho$ and $\boldsymbol{D}_q'\boldsymbol{D}_q = \boldsymbol{\Sigma}_\lambda$ must satisfy the condition of $\boldsymbol{C}_q > 0$, $\boldsymbol{D}_q > 0$, and $\boldsymbol{C}_q'\boldsymbol{D}_q = 0$. With the objective to minimize $\boldsymbol{W}$, the following inequality is also considered:

$$\kappa - \boldsymbol{x}_a(\boldsymbol{0})'\boldsymbol{W}\boldsymbol{x}_a(\boldsymbol{0}) < 0, \tag{3.22}$$

from which, by applying the Schur's complement, the following LMI is obtained:

$$\begin{bmatrix} \kappa & 0 \\ 0 & \boldsymbol{W} \end{bmatrix} > 0. \tag{3.23}$$

Then, by minimizing the parameter κ, the matrix $\boldsymbol{W}$ is minimized. Lastly, Definition 3.1 and Lemma 3.1, the unitary box condition $\|\zeta\|_\infty \leq 1$ is transformed into $\|\zeta\|_2 \leq 1$. Then, the inequality $\|\zeta\|_2 \leq 1$ is written as an LMI as follows

$$\begin{bmatrix} 1 & \zeta' \\ \zeta & I \end{bmatrix} \geq 0, \tag{3.24}$$

which completes the LMI conditions used in (3.10), concluding the proof. □

Theorem 3.1 offers a significant advantage in using the zonotope definition to represent the set of uncertainties $\mathcal{W}$. This approach effectively reduces the computational burden associated with obtaining the feedback gain for high-order dynamical systems. In contrast to the methodologies presented in Gonzalez et al. (2011) and Limon et al. (2008), where the vertices of the uncertainty set are considered, thereby increasing the complexity of the optimization problem.

The solution provided by Theorem 3.1 yields the Lyapunov matrix $\boldsymbol{L}$ and the feedback control matrix $\boldsymbol{K}$. This feedback gain ensures system stability, as the theorem employs a Linear Matrix Inequality (LMI) for stabilizing all worst-case scenarios of the LPV model (Eq. 3.21). Additionally, it makes us of an LMI that guarantees γ-stability for the system, considering uncertainties (Eq. 3.15).

It is worth highlighting that the control action generated by the feedback matrix $\boldsymbol{K}$ is constrained within a maximum value (Eq. 3.19), assuming a symmetric control action concerning the origin. Moreover, the selection of a feedback gain $\boldsymbol{K}$ that minimizes the robust positive invariant set (RPI) $\mathcal{Z}$ ensures the existence of the nominal state and control sets $\mathcal{X} \ominus \mathcal{Z}$ and $\mathcal{U} \ominus \boldsymbol{K}\mathcal{Z}$, respectively, thus maximizing the constraint domain of the nominal sets (Limon et al., 2008).

3.3 Reachable Set Computation

In this section, we present an algorithm to compute the reachable set for high-order systems. As stated in González Sánchez (2011), a reachable set contains the real state of a given uncertain system for every possible realization of the uncertainty, which is given by

$$\mathcal{Z}_{k+i+1} = (\boldsymbol{A}_z(k+1) + \boldsymbol{B}_Z\boldsymbol{K})\mathcal{Z}_{k+i} \oplus \mathcal{W}, \tag{3.25}$$

where $i = 0, \ldots, N-1$ and $\mathcal{Z}_0 = 0$. It is worth noting that the computation of the reachable set $\mathcal{Z}$ is typically performed online, employing polytope representation for sets $\mathcal{W}$ and $\mathcal{Z}$. However, the process of successive additions becomes highly complex or even unfeasable for high-order systems, leading to significant increases in the computation time of the reachable set.

To address this challenge, we propose an offline computation approach for the reachable set. This set is regarded as a robust positive invariant set (RPI) and is represented by using zonotopes. Furthermore, the algorithm incorporates a parameter-varying model in the computation process to mitigate the conservatism often associated with the use of linear time invariant (LTI) models with a system depicted by an LPV model.

3.3.1 Preliminaries

Consider the following definition of a RPI set:

Definition 3.2 (Rawlings et al. (2019))**.** *A set $\mathcal{Z} \subseteq \mathbb{R}^n$ is RPI for a system $\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{w}(k))$ with the constraint set $(\mathcal{X}, \mathcal{W})$ if $\mathcal{Z} \subset \mathcal{X}$ and $f(\boldsymbol{x}(k), \boldsymbol{w}(k)) \in \mathcal{Z}$, $\forall \boldsymbol{w} \in \mathcal{W}$, $\forall \boldsymbol{x} \in \mathcal{Z}$.*

Consider the uncertain LTI model $\boldsymbol{x}_a(k+1) = \bar{\boldsymbol{A}}_z\boldsymbol{x}_a(k) + \boldsymbol{B}_z\boldsymbol{u}(k) + \boldsymbol{w}(k)$ and the nominal model (3.3). Let also consider the control law (3.4) and the deviation of the current state $\boldsymbol{x}_a(k)$ from the nominal state $\bar{\boldsymbol{x}}_a(k)$ given by $\boldsymbol{e}(k) = \boldsymbol{x}_a(k) - \bar{\boldsymbol{x}}_a(k)$. Then, $\boldsymbol{x}_a(k)$ satisfies

$$\boldsymbol{x}_a(k+1) = \bar{\boldsymbol{A}}_z\boldsymbol{x}_a(k) + \boldsymbol{B}_z\bar{\boldsymbol{u}}(k) + \boldsymbol{B}_z\boldsymbol{K}\boldsymbol{e}(k) + \boldsymbol{w}(k), \tag{3.26}$$

and the deviation $\boldsymbol{e}(k) = \boldsymbol{x}_a(k) - \bar{\boldsymbol{x}}_a(k)$ satisfies

$$\boldsymbol{e}(k+1) = \bar{\boldsymbol{S}}_z\boldsymbol{e}(k) + \boldsymbol{w}(k), \tag{3.27}$$

where $\bar{\boldsymbol{S}}_z = \bar{\boldsymbol{A}}_z + \boldsymbol{B}_z\bar{\boldsymbol{K}}$ is the LTI closed-loop matrix and $\bar{\boldsymbol{K}}$ stabilizes the system by design. Then, the equation (3.27) can be expanded as

$$\boldsymbol{e}(k+i) = \bar{\boldsymbol{S}}_z^i\boldsymbol{e}(k) + \sum_{j=0}^{i-1} \bar{\boldsymbol{S}}_z^j\boldsymbol{w}(k+j). \tag{3.28}$$

If $\boldsymbol{e}(k) = 0$, then $\boldsymbol{e}(k+i) \in \mathcal{F}(i)$ where the set $\mathcal{F}(i)$ is defined as

$$\mathcal{F}(i) = \sum_{j=0}^{i-1} \bar{\boldsymbol{S}}_{\boldsymbol{z}}^{j} \mathcal{W} = \mathcal{W} \oplus \bar{\boldsymbol{S}}_{\boldsymbol{z}} \mathcal{W} \oplus \bar{\boldsymbol{S}}_{\boldsymbol{z}}^{2} \mathcal{W} \oplus \cdots \oplus \bar{\boldsymbol{S}}_{\boldsymbol{z}}^{i-1} \mathcal{W}, \tag{3.29}$$

in which $\oplus$ denote set addition.

According to Rawlings et al. (2019), if $i \to \infty$ and $\bar{\boldsymbol{S}}_{\boldsymbol{z}}$ is stable, the following set

$$\mathcal{F}_{\infty} = \sum_{j=0}^{\infty} \bar{\boldsymbol{S}}_{\boldsymbol{z}}^{j} \mathcal{W}, \tag{3.30}$$

exists and is the minimum robust positive invariant (mRPI) set of an LTI system. However, the computation of $\mathcal{F}(\infty)$ is impractical when the order of the system is high.

For this reason, instead of computing the mRPI, an approximation of the set $\mathcal{F}_{\infty}$ is obtained, given by the set $\mathcal{Z}$, such that $\mathcal{F}_{\infty} \subseteq \mathcal{Z} \subseteq \mathcal{F}_{\infty} \oplus \varepsilon \mathcal{B}$ for a given error ε. Therefore, the computation of the RPI set $\mathcal{Z}$ is outlined by the following theorem previously stated by Rakovic et al. (2005).

Theorem 3.2. *Given a large enough value of j such that $(1-\alpha(j))^{-1}\alpha(j)\beta(j) \leq \varepsilon$, there exists a RPI set which is an approximation of the mRPI set $\mathcal{Z}_{\infty}$ with an error less than ε, which is given by*

$$\mathcal{Z} = (1-\alpha(j))^{-1} F_j, \tag{3.31}$$

where

$$\alpha(j) = \min \ \alpha : \ (\bar{\boldsymbol{S}}_z)^j \mathcal{W} \subseteq \alpha \mathcal{W}, \tag{3.32}$$

$$\beta(j) = \min \ \beta : \ F_j \subseteq \beta \mathcal{B}^n. \tag{3.33}$$

Proof. Refer to Rakovic et al. (2005). □

It is noteworthy that Theorem 3.2 poses certain challenges when applied to high-order systems. The algorithm requires multiple solutions of linear programming problems, combined with the representation of sets as polytopes, which makes it impractical for high-order systems. Furthermore, the requirement to choose the parameter ε a priori, without prior knowledge of the size of the mRPI set, introduces the risk of a wrong selection of the parameter.

In this context, we propose the use of a modified formulation to compute an approximated RPI set. Originally introduced by Limon et al. (2008) for LTI systems, the method employs a criterion based on the set size, removing the dependency on the parameter ε. Initially, the criterion $(1-\alpha(j))^{-1}\alpha(j)\beta(j) \leq \varepsilon$ used to determine the value of j, is replaced by the inequality equation of $\alpha(j)$ for a given relative error bound $\lambda \in (0,1)$, as follows

$$\alpha(j) \leq \frac{\lambda}{\lambda + 1}. \tag{3.34}$$

Then, the use of zonotope set representation is considered to reduce the computational cost encountered in high-order systems, which can also be defined as follows.

Definition 3.3. *Considering the unitary interval* $\mathcal{B} \triangleq [-1, 1]$ *and the unitary hypercube* $\mathcal{B}^n$*, a set* $\mathcal{Z}$ *is a zonotope if there exists* $(\boldsymbol{H}, \boldsymbol{c}) \in \mathbb{R}^{n\times r} \times \mathbb{R}^n$ *such that*

$$\mathcal{Z} = \boldsymbol{c} \oplus \boldsymbol{H}\mathcal{B}^n \tag{3.35}$$

where $\boldsymbol{c}$ *is the center of the set and* $\boldsymbol{H}$ *is the generator matrix*

Hence, the constraint set $\mathcal{W}$ is represented as a zonotope $\boldsymbol{w_0} + \boldsymbol{H_w}\zeta$, with $\boldsymbol{H_w}$ and $\boldsymbol{w}_0$ being the generator matrix and the center of the uncertainty set, respectively. As mentioned above, the constraint set is give by $\mathcal{W} = \mathcal{W}_u \oplus \mathcal{W}_d$, where $\mathcal{W}_u$ is the unmodeled dynamic set and $\mathcal{W}_d = \boldsymbol{D_z}\mathcal{D}$ is the external disturbance set.

Finally, the RPI set $\mathcal{Z}$ used as a reachable set is computed as

$$\mathcal{Z} = (1 - \alpha(j))^{-1}\boldsymbol{H_z}(j)\mathcal{B}^{sn} \oplus (I_n - \bar{\boldsymbol{S}}_z)^{-1}\boldsymbol{w}_0, \tag{3.36}$$

where the parameters j and $\alpha(j)$ are computed recursively until satisfy the inequality

$$||\boldsymbol{H_w}^{-1}(\bar{\boldsymbol{S}}_z)^j\boldsymbol{H_w}||_\infty \leq \frac{\lambda}{\lambda + 1}, \tag{3.37}$$

using the following lemma presented in Limon et al. (2008).

Lemma 3.2 (Lemma 2 in Limon et al. (2008))**.** *Consider the set* $\mathcal{W} = \boldsymbol{w_0} \oplus \boldsymbol{H_w}\zeta$*, where* $\boldsymbol{H_w}$ *is a nonsingular matrix. Define the matrix* $\boldsymbol{H_z}(j) = [\bar{\boldsymbol{S}}_z^{j-1}\boldsymbol{H_w}, \bar{\boldsymbol{S}}_z^{j-2}\boldsymbol{H_w}, \ldots, \boldsymbol{H_w}]$*. Then,*

$$||\boldsymbol{H_w}^{-1}(\bar{\boldsymbol{S}}_z)^j\boldsymbol{H_w}||_\infty = min\ \alpha :\ (\bar{\boldsymbol{S}}_z)^j\mathcal{W} \subseteq \alpha\mathcal{W},$$
$$||\boldsymbol{H_z}(j)||_\infty = min\ \beta :\ F_j \subseteq \beta\mathcal{B}^n.$$

3.3.2 Reachable Set Computation for Parameter-Varying Systems

In the preceding section, we introduced a methodology for computing the reachable set as an approximation of the minimum RPI using zonotope representation. While this formulation is well-suited for high-order systems, it is limited to LTI systems. Consequently, in this section, we propose an extension of this formulation to incorporate LPV systems (as a contribution to this thesis) commonly employed in robotics formulations.

Therefore, we present the following theorem for an LPV closed-loop system.

Theorem 3.3. *Consider the linear parameter-varying closed-loop system with additive uncertainty*

$$\boldsymbol{x_a}(k+1) = (\boldsymbol{A_z}(\boldsymbol{\sigma}) + \boldsymbol{B_z}\boldsymbol{K})\boldsymbol{x_a}(k) + \boldsymbol{w}, \tag{3.38}$$

for which the pair $(\boldsymbol{A_z}(\boldsymbol{\sigma}), \boldsymbol{B_z})$ *is assumed controllable, with* $\boldsymbol{A_z}(\boldsymbol{\sigma})$ *being a group of matrices representing each vertex of the polytope formed by the maximum and minimum values of the parameter* $\boldsymbol{\sigma}$*. Consider also the feedback matrix* $\boldsymbol{K}$ *derived from Theorem 3.1, which ensures the stability of the uncertain parameter-varying system. Then, the closed-loop dynamic matrix for an LPV system can be written as*

$$\boldsymbol{S}_z = \bar{\boldsymbol{A}}_{\boldsymbol{k}} + \sum_{i=1}^{n} \Delta \boldsymbol{A}_{\boldsymbol{k}}(\boldsymbol{\sigma}_i), \tag{3.39}$$

where $\bar{\boldsymbol{A}}_{\boldsymbol{k}} = \bar{\boldsymbol{A}}_{\boldsymbol{z}} + \boldsymbol{B_z}\bar{\boldsymbol{K}}$ *is the nominal part of the system, with* $\bar{\boldsymbol{K}}$ *being the nominal feedback gain computed with (3.10) when assuming* $\boldsymbol{\sigma} = \boldsymbol{0}$*.*

Proof. First, we write the parameter-varying state matrix as a convex combination of a LPV system as follows

$$\boldsymbol{A_z}(\boldsymbol{\sigma}) = \bar{\boldsymbol{A}}_{\boldsymbol{z}} + \sum_{i=1}^{n} \Delta \boldsymbol{A_z}(\boldsymbol{\sigma_i}), \tag{3.40}$$

where $\bar{\boldsymbol{A}}_{\boldsymbol{z}}$ is the nominal state matrix assuming $\boldsymbol{\sigma} = \boldsymbol{0}$, and $\Delta \boldsymbol{A_z}(\boldsymbol{\sigma_i})$ is the parameter-varying part of the state matrix.

Second, according to Gonzalez et al. (2011), the feedback gain that ensures the stability of the parameter-varying system can be written as $\boldsymbol{K} = \bar{\boldsymbol{K}} + \Delta \boldsymbol{K}$, where $\bar{\boldsymbol{K}}$ is the nominal feedback gain that stabilizes the LTI system part and $\Delta \boldsymbol{K}$ is a group of feedback gains that stabilizes the LPV part of the model. However, as the first step, it is possible to write the feedback gain as a convex combination given by

$$\boldsymbol{K} = \bar{\boldsymbol{K}} + \sum_{i=1}^{n} \Delta \boldsymbol{K_i}, \tag{3.41}$$

where $\Delta \boldsymbol{K}_i$ is the feedback gain that stabilizes the LPV part of the i-th model.

Third, by replacing equations (3.40) and (3.41) into $\boldsymbol{S_z} = \boldsymbol{A_z}(\boldsymbol{\sigma}) + \boldsymbol{B_z}\boldsymbol{K}$, the closed-loop dynamic equation matrix for an LPV system is written as

$$\boldsymbol{S}_z = \bar{\boldsymbol{A}}_{\boldsymbol{k}} + \sum_{i=1}^{n} \Delta \boldsymbol{A}_{\boldsymbol{k}}(\boldsymbol{\sigma}_i). \tag{3.42}$$

which conclude the proof. □

It is worth mentioning that it is not necessary to compute $\Delta \boldsymbol{K_i}$, since the parameter-varying part of the closed-loop system can be obtained as

$$\Delta \boldsymbol{A}_{\boldsymbol{k}}(\boldsymbol{\sigma}_i) = (\boldsymbol{A_z}(\boldsymbol{\sigma}_i) + \boldsymbol{B_z}\boldsymbol{K}_i) - \bar{\boldsymbol{A}}_{\boldsymbol{k}},$$

where the feedback gain $\boldsymbol{K}_i$ is computed from (3.10) by defining $\boldsymbol{A_z}(\boldsymbol{\sigma}) \triangleq \boldsymbol{A_z}(\boldsymbol{\sigma}_i)$, with $\boldsymbol{A_z}(\boldsymbol{\sigma}_i)$ representing the state matrix of the system in each vertex of the polytope composed by the maximum and minimum values of the varying parameter $\boldsymbol{\sigma}$.

Thus, the reachable set $\mathcal{Z}$ is computed using the LPV closed-loop system as follows

$$\mathcal{Z} = (1 - \alpha(j))^{-1} \boldsymbol{H_z}(j) \mathcal{B}^{sn} \oplus (I_n - \boldsymbol{S}_z)^{-1} \boldsymbol{w}_0, \tag{3.43}$$

where $\boldsymbol{H_z}(j) = [\boldsymbol{S}_z^{j-1}\boldsymbol{H_w}, \boldsymbol{S}_z^{j-2}\boldsymbol{H_w}, \ldots, \boldsymbol{H_w}]$ and $\alpha(j) = ||\boldsymbol{H}_{\boldsymbol{w}}^{-1}(\boldsymbol{S}_z)^j \boldsymbol{H_w}||_\infty$. Furthermore, the parameter j is computed recursively until satisfying the inequality

$$||\boldsymbol{H}_{\boldsymbol{w}}^{-1}(\boldsymbol{S}_z)^j \boldsymbol{H_w}||_\infty \leq \frac{\lambda}{\lambda + 1}.$$

3.4 Nominal Control and State Sets Computation

In this section, two methodologies are presented to compute the nominal control and state sets used to formulate the nominal optimization problem. The optimal solution derived from this problem constitutes the second control term within the control law (3.4). As mentioned above, these sets are obtained through the following set difference:

$$\bar{\mathcal{X}} = \mathcal{X} \ominus \mathcal{Z}, \tag{3.44}$$

$$\bar{\mathcal{U}} = \mathcal{U} \ominus \boldsymbol{K}\mathcal{Z}, \tag{3.45}$$

In the first methodology, the reachable set is represented by a zonotope while the state and control sets are considered as polytopes in the set difference (3.44) and (3.45). Thus the set difference between a polytope and a zonotope can be performed using the following proposition.

Proposition 3.1 (Raimondo et al. (2013), Proposition 3)**.** *Let* $\mathcal{Z} = \{G,\ c\}$ *and* $\mathcal{P} \equiv \{z \in \mathbb{R}^n : h_i^T z \leq k_i,\ i = 1, ..., m\}$*, where* $h_i \in \mathbb{R}^n, h_i \neq 0$*, and* $k_i \in \mathbb{R}$*. Then* $P \ominus Z = \{z \in \mathbb{R}^n : h_i^T z \leq k_i',\ i = 1, ..., m\}$*, where* $k_i' \equiv k_i - h_i^T c - \|h_i^T G\|_1$*.*

Consequently, by applying Proposition 3.1, the sets resulting from equations (3.44) and (3.45) are represented as polytopes.

In the second methodology, a novel formulation is proposed in which the reachable set is represented as a zonotope, while the state and control sets are considered as constrained zonotopes. The latter set representation offers clear advantages, mainly enabling operations with asymmetric convex polytopes. In addition, they exhibit computational efficiency, particularly in relation to addition, multiplication, and subtraction set operations.

Thus, consider the following definition of constrained zonotopes.

Definition 3.4 (Scott et al. (2016))**.** *A set* $\mathcal{Z}$ *is a constrained zonotope if there exists* $(\boldsymbol{H_z}, \boldsymbol{z_0}, \boldsymbol{A}, \boldsymbol{b}) \in \mathbb{R}^{n \times r} \times \mathbb{R}^n \times \mathbb{R}^{m \times r} \times \mathbb{R}^m$ *such that*

$$\mathcal{Z} = \{\boldsymbol{z_0} + \boldsymbol{H_z}\zeta : \zeta \in \mathbb{R}^r, \|\zeta\|_\infty \leq 1, \boldsymbol{A}\zeta == \boldsymbol{b}\}, \tag{3.46}$$

where $\boldsymbol{z_0}$ *is the center,* $\boldsymbol{H_z}$ *is the generator matrix,* $\boldsymbol{A}$ *and* $\boldsymbol{b}$ *are the equality constraint matrix and vector.*

Then, the set difference between a constrained zonotope and a zonotope can be performed using the following lemma.

Lemma 3.3 (Lemma 5 in Raghuraman & Koeln (2022))**.** *If* $\mathcal{P} = \{\boldsymbol{H_p}, \boldsymbol{p_0}, \boldsymbol{A_p}, \boldsymbol{b_p}\}$ *and* $\mathcal{Z} = \{\boldsymbol{H_z},\ \boldsymbol{z_0}\}$*, then the Pontryagin difference* $\mathcal{Z}_d = \mathcal{P} \ominus \mathcal{Z}$ *is computed using the* $\boldsymbol{n}_g$ *generators* $\boldsymbol{h}_i$ *of* $\mathcal{Z}$ *by applying the following recursion:*

$$\mathcal{Z}_{int}^{(0)} = \mathcal{Z} - \boldsymbol{p_0} \tag{3.47}$$

$$\mathcal{Z}_{int}^{(i)} = (\mathcal{Z}_{int}^{(i-1)} + \boldsymbol{h}_i) \cap (\mathcal{Z}_{int}^{(i-1)} - \boldsymbol{h}_i) \tag{3.48}$$

$$\mathcal{Z}_d = \mathcal{Z}_{int}^{(\boldsymbol{n}_g)} \tag{3.49}$$

Consequently, by applying Lemma 3.3, the sets resulting from equations (3.45) and (3.45) are represented as constrained zonotopes.

It is worth mentioning that in the second approach, constrained zonotopes offer computational efficiency in set subtraction operations, unlike polytopic approaches, which often require complex and computationally expensive algorithms. Additionally, in zonotope approaches, the operation result is not exact and may not be a closed zonotope (Raghuraman & Koeln, 2022).

3.5 Nominal Model Predictive Optimal Control Problem

As mentioned before, the nominal control term of the control law (3.4) is computed via a nominal model predictive control. The optimization problem makes use of the nominal model (3.3) constrained by the nominal state and control set (3.5) and (3.6), respectively. The cost functional used in the optimization problem is casted as follows

$$V_{N,M}(\bar{\boldsymbol{x}}_{\boldsymbol{a}}, \bar{\boldsymbol{u}}) = \sum_{i=0}^{M-1} \ell(\bar{\boldsymbol{x}}_{\boldsymbol{a}}(i), \bar{\boldsymbol{u}}(i)) + \sum_{i=M}^{N-1} \ell(\bar{\boldsymbol{x}}_{\boldsymbol{a}}(i), \bar{\boldsymbol{u}}(M-1)) + V_f(\bar{\boldsymbol{x}}_{\boldsymbol{a}}(N)), \tag{3.50}$$

where $\ell(.)$ and $V_f(.)$ denote the stage cost and the terminal cost functionals, respectively. The parameter N is the prediction horizon, and M is the control horizon, with $M \leq N$.

The stage cost functional is defined as $\ell(\bar{\boldsymbol{x}}_{\boldsymbol{a}}, \bar{\boldsymbol{u}}) = \bar{\boldsymbol{x}}_{\boldsymbol{a}}' \boldsymbol{\Sigma_\rho} \bar{\boldsymbol{x}}_{\boldsymbol{a}} + \bar{\boldsymbol{u}}' \boldsymbol{\Sigma_\lambda} \bar{\boldsymbol{u}}$ where the matrix $\boldsymbol{\Sigma_\rho} = \text{diag}(\rho_1, \rho_2, \rho_3, ..., \rho_{\text{n}})$ and $\boldsymbol{\Sigma_\lambda} = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_{\text{m}})$, with ρ_i and λ_i being the state and control weights, respectively. The terminal cost functional is defined as $V_f(\bar{\boldsymbol{x}}_{\boldsymbol{a}}, \bar{\boldsymbol{u}}) = \bar{\boldsymbol{x}}_{\boldsymbol{a}}' \boldsymbol{L} \bar{\boldsymbol{x}}_{\boldsymbol{a}}$, where $\boldsymbol{L}$ is a Lyapunov matrix, obtained from Theorem 3.1 which guarantees stability in a finite horizon.

The nominal control sequence is obtained through the following model predictive control optimization problem, where the variable $\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k)$ is considered as a parameter:

$$\begin{aligned}
\min_{\bar{\boldsymbol{x}}_{\boldsymbol{a}},\bar{\boldsymbol{u}}} \quad & V_{N,M}(\bar{\boldsymbol{x}}_{\boldsymbol{a}},\bar{\boldsymbol{u}}) \\
\text{s. t.:} \quad & \bar{\boldsymbol{x}}_{\boldsymbol{a}}(0) = \bar{\boldsymbol{x}}_{\boldsymbol{a}}(k), \\
& \bar{\boldsymbol{x}}_{\boldsymbol{a}}(i+1) = f(\bar{\boldsymbol{x}}_{\boldsymbol{a}}(i),\bar{\boldsymbol{u}}(i)), i \in \mathbb{I}_{0:M-1}, \\
& \bar{\boldsymbol{x}}_{\boldsymbol{a}}(i+1) = f(\bar{\boldsymbol{x}}_{\boldsymbol{a}},\bar{\boldsymbol{u}}(M-1)), i \in \mathbb{I}_{M:N-1}, \\
& \bar{\boldsymbol{u}}(i) \in \bar{\mathcal{U}}, i \in \mathbb{I}_{0:M-1}, \\
& \bar{\boldsymbol{x}}_{\boldsymbol{a}}(i) \in \bar{\mathcal{X}}, i \in \mathbb{I}_{0:N-1},
\end{aligned} \tag{3.51}$$

where $f(\bar{\boldsymbol{x}}_{\boldsymbol{a}},\bar{\boldsymbol{u}})$ is the discrete-time nominal system (3.3).

Then, the optimization problem can be written in a matrix form by considering the predicted state vector, $\hat{\boldsymbol{x}}$, and the predicted control vector, $\hat{\boldsymbol{u}}$, given by

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \bar{\boldsymbol{x}}_{\boldsymbol{a}}(1) \\ \bar{\boldsymbol{x}}_{\boldsymbol{a}}(2) \\ \bar{\boldsymbol{x}}_{\boldsymbol{a}}(3) \\ \vdots \\ \bar{\boldsymbol{x}}_{\boldsymbol{a}}(N) \end{bmatrix}, \quad \hat{\boldsymbol{u}} = \begin{bmatrix} \bar{\boldsymbol{u}}(0) \\ \bar{\boldsymbol{u}}(1) \\ \bar{\boldsymbol{u}}(2) \\ \vdots \\ \bar{\boldsymbol{u}}(M-1) \end{bmatrix}. \tag{3.52}$$

Besides, by expanding the nominal system (3.3) throughout the prediction horizon N and the control horizon M, the predicted output model is computed as

$$\hat{\boldsymbol{x}} = \boldsymbol{P}\hat{\boldsymbol{u}} + \boldsymbol{Q}\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k), \tag{3.53}$$

where the prediction matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ are given by:

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{B}_{\boldsymbol{z}} & 0 & 0 & \cdots & 0 \\ \bar{\boldsymbol{A}}_{\boldsymbol{z}}\boldsymbol{B}_{\boldsymbol{z}} & \boldsymbol{B}_{\boldsymbol{z}} & 0 & \cdots & 0 \\ \bar{\boldsymbol{A}}_{\boldsymbol{z}}^2\boldsymbol{B}_{\boldsymbol{z}} & \bar{\boldsymbol{A}}_{\boldsymbol{z}}\boldsymbol{B}_{\boldsymbol{z}} & \boldsymbol{B}_{\boldsymbol{z}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \bar{\boldsymbol{A}}_{\boldsymbol{z}}^M\boldsymbol{B}_{\boldsymbol{z}} & \bar{\boldsymbol{A}}_{\boldsymbol{z}}^{M-1}\boldsymbol{B}_{\boldsymbol{z}} & \bar{\boldsymbol{A}}_{\boldsymbol{z}}^{M-2}\boldsymbol{B}_{\boldsymbol{z}} & \cdots & \boldsymbol{B}_{\boldsymbol{z}} + \boldsymbol{\varepsilon}(k,j) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \bar{\boldsymbol{A}}_{\boldsymbol{z}}^{N-1}\boldsymbol{B}_{\boldsymbol{z}} & \bar{\boldsymbol{A}}_{\boldsymbol{z}}^{N-2}\boldsymbol{B}_{\boldsymbol{z}} & \bar{\boldsymbol{A}}_{\boldsymbol{z}}^{N-3}\boldsymbol{B}_{\boldsymbol{z}} & \cdots & \bar{\boldsymbol{A}}_{\boldsymbol{z}}^{N-M}\boldsymbol{B}_{\boldsymbol{z}} + \boldsymbol{\varepsilon}(k,j) \end{bmatrix}, \text{ and } \boldsymbol{Q} = \begin{bmatrix} \bar{\boldsymbol{A}}_{\boldsymbol{z}} \\ \bar{\boldsymbol{A}}_{\boldsymbol{z}}^2 \\ \vdots \\ \bar{\boldsymbol{A}}_{\boldsymbol{z}}^N \end{bmatrix},$$

with

$$\boldsymbol{\varepsilon}(k,j) \begin{cases} 0 & \text{if} \quad M = N \\ \sum_{i=1}^{N-M} \bar{\boldsymbol{A}}_{\boldsymbol{z}}^{N-M-i}\boldsymbol{B}_{\boldsymbol{z}} & \text{if} \quad M < N. \end{cases}$$

By considering the predicted model (3.53), the cost functional (3.50) is written in the matrix form as

$$V(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}) = \hat{\boldsymbol{x}}' \, \boldsymbol{W}_x \, \hat{\boldsymbol{x}} + \hat{\boldsymbol{u}}' \, \boldsymbol{W}_u \, \hat{\boldsymbol{u}}, \tag{3.54}$$

with the weighting matrices $\boldsymbol{W}_x$ and $\boldsymbol{W}_u$ corresponding to the state and control, respectively, and being defined as diagonal block matrices, where their diagonals are given by $\boldsymbol{W}_x = (\boldsymbol{\Sigma}_\rho, \boldsymbol{\Sigma}_\rho, \ldots, \boldsymbol{\Sigma}_\rho, \boldsymbol{L})$ and $\boldsymbol{W}_u = (\boldsymbol{\Sigma}_\lambda, \boldsymbol{\Sigma}_\lambda, \ldots, \boldsymbol{\Sigma}_\lambda)$.

Finally, by considering the nominal state and control sets as the polytopes $\boldsymbol{A}_{px}\bar{\boldsymbol{x}}(k) \leq \boldsymbol{b}_{px}$ and $\boldsymbol{A}_{pu}\bar{\boldsymbol{u}}(k) \leq \boldsymbol{b}_{pu}$, respectively. The nominal control signal is computed at each sampling time by minimizing the following optimal control problem:

$$\begin{aligned} \min_{\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}} \quad & V(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}) \\ \text{s. t.:} \quad & \hat{\boldsymbol{x}} = \boldsymbol{P}\hat{\boldsymbol{u}} + \boldsymbol{Q}\bar{\boldsymbol{x}}_a(k), \\ & \boldsymbol{A}_u \hat{\boldsymbol{u}} \leq \boldsymbol{b}_u, \\ & \boldsymbol{A}_x \hat{\boldsymbol{x}} \leq \boldsymbol{b}_x, \end{aligned} \tag{3.55}$$

where the diagonals of the block matrices $\boldsymbol{A}_u$ and $\boldsymbol{A}_x$ are given by $(\boldsymbol{A}_{pu}, \boldsymbol{A}_{pu}, \ldots, \boldsymbol{A}_{pu})$ and $(\boldsymbol{A}_{px}, \boldsymbol{A}_{px}, \ldots, \boldsymbol{A}_{px})$, respectively. Furthermore, the vectors $\boldsymbol{b}_u$ and $\boldsymbol{b}_x$ are defined as $\boldsymbol{b}_u = [\boldsymbol{b}_{pu}, \boldsymbol{b}_{pu}, \ldots, \boldsymbol{b}_{pu}]$ and $\boldsymbol{b}_x = [\boldsymbol{b}_{px}, \boldsymbol{b}_{px}, \ldots, \boldsymbol{b}_{px}]$, respectively.

By solving (3.55), the optimum solution of $\hat{\boldsymbol{u}}$ is obtained, where the first term $\bar{\boldsymbol{u}}(0)$ is the nominal control action $\bar{\boldsymbol{u}}(k)$ used in equation (3.4).

3.6 Final Remarks

This chapter developed a methodology to reduce the computation drawback of the tube-based MPC formulation when controlling high-order systems. The proposed algorithm for computing the reachable set as an RPI set allows to deal with high-order dynamical systems, building upon the methodology proposed in Limon et al. (2008) and extending it to parameter-varying systems. Theorem 3.1 offers a significant advantage by using the zonotope definition to represent the set of uncertainties $\mathcal{W}$. This approach effectively reduces the computational burden by the off-line computation of the feedback gain $\boldsymbol{K}$, conversely to the methodologies presented in Gonzalez et al. (2011) and Limon et al. (2008) that consider the vertices of the uncertainty set, thereby increasing the complexity of the optimization problem.

Moreover, as the feedback gain is also employed in the subsequent computation of the reachable set, selecting a feedback gain $\boldsymbol{K}$ that minimizes the size of the RPI set $\mathcal{Z}$ is essential. Hence, the criteria of Theorem 3.1 ensures the existence of the nominal state and control sets $\mathcal{X} \ominus \mathcal{Z}$ and $\mathcal{U} \ominus \boldsymbol{K}\mathcal{Z}$, respectively, thereby maximizing the constraint domain of nominal sets (Limon et al., 2008). Furthermore, it is presented a new formulation based

on the nominal states and control set represented as constrained zonotopes, aiming to reduce the computation time of the nominal optimization problem.

From now on, we will discuss the methodologies for fast optimization of the MPC problem to achieve a tube-based MPC formulation capable of being applied to high-order and fast dynamic systems.

4

Tube-based MPC Using Fast Optimization

This chapter focuses on developing methodologies to achieve fast optimization of the problem presented in Chapter 3, considering high-order and fast dynamic systems. To accomplish this, the thesis proposes the use of two formulations. The first one relies on the explicit solution of the optimization problem (3.55), computed offline. The second formulation involves online optimization employing the alternate direction method of multipliers (ADMM) alongside several techniques aimed at diminishing computational costs and expediting the optimization process.

4.1 Explicit Multi-parametric Optimization

The first methodology proposes to solve the optimization problem (3.55) offline for all $\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k)$ within the nominal state set $\bar{\mathcal{X}}$, using multi-parametric optimization. This results in a group of piecewise affine functions where the controller is represented in a totally equivalent way, in which the explicit dependency of $\bar{\boldsymbol{u}}(k)$ on $\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k)$ is obtained.

Consider the nominal optimization problem (3.55) given by

$$\min_{\hat{\boldsymbol{x}},\hat{\boldsymbol{u}}} \quad \hat{\boldsymbol{x}}'\boldsymbol{W}_{\boldsymbol{x}}\hat{\boldsymbol{x}} + \hat{\boldsymbol{u}}'\boldsymbol{W}_{\boldsymbol{u}}\hat{\boldsymbol{u}} \tag{4.1}$$

$$\text{s. t.:} \quad \hat{\boldsymbol{x}} = \boldsymbol{P}\hat{\boldsymbol{u}} + \boldsymbol{Q}\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k), \tag{4.2}$$

$$\boldsymbol{A}_{\boldsymbol{u}}\hat{\boldsymbol{u}} \leq \boldsymbol{b}_{\boldsymbol{u}}, \tag{4.3}$$

$$\boldsymbol{A}_{\boldsymbol{x}}\hat{\boldsymbol{x}} \leq \boldsymbol{b}_{\boldsymbol{x}}. \tag{4.4}$$

First, we transform the optimization problem (3.55) into a parametric quadratic programming (pQP) problem by replacing the predictive model (4.2) into the cost functional (4.1) as follows

$$\frac{1}{2}\hat{\boldsymbol{u}}'\boldsymbol{H}_q\hat{\boldsymbol{u}} + \bar{\boldsymbol{x}}_a(k)'\boldsymbol{F}_q\hat{\boldsymbol{u}} + \frac{1}{2}\bar{\boldsymbol{x}}_a(k)'\boldsymbol{Y}_q\bar{\boldsymbol{x}}_a(k), \tag{4.5}$$

where $\boldsymbol{H}_q = \boldsymbol{P}'\boldsymbol{W}_y\boldsymbol{P} + \boldsymbol{W}_u$, $\boldsymbol{F}_q = \boldsymbol{Q}'\boldsymbol{W}_y\boldsymbol{P}$, $\boldsymbol{Y}_q = \boldsymbol{Q}'\boldsymbol{W}_y\boldsymbol{Q}$, $\boldsymbol{H} \geq 0$, and $\bar{\boldsymbol{x}}_a(k)$ is considered as a parametric variable. Moreover, to place constraints in the pQP form, the nominal state constraint $\boldsymbol{A}_x\hat{\boldsymbol{x}} \leq \boldsymbol{b}_x$ must be written with respect to the predicted control vector. Thus, by replacing the predictive model (4.2) in (4.4), the state constraint is rewritten as $\boldsymbol{A}_x(\boldsymbol{P}\hat{\boldsymbol{u}} + \boldsymbol{Q}\bar{\boldsymbol{x}}_a(k)) \leq \boldsymbol{b}_x$. Consequently, the inequality constraint of the optimization problem is given by

$$\underbrace{\begin{bmatrix}\boldsymbol{A}_u\\ \boldsymbol{A}_x\boldsymbol{P}\end{bmatrix}}_{G_q}\hat{\boldsymbol{u}} \leq \underbrace{\begin{bmatrix}\boldsymbol{b}_u\\ \boldsymbol{b}_x\end{bmatrix}}_{W_q} + \underbrace{\begin{bmatrix}\boldsymbol{0}\\ -\boldsymbol{A}_x\boldsymbol{Q}\end{bmatrix}}_{E_q}\bar{\boldsymbol{x}}_a(k). \tag{4.6}$$

Therefore, from the cost functional (4.5) and the constraints (4.6), the optimization problem (3.55) is written in the pQP form as follows

$$\begin{aligned}\min_{\hat{\boldsymbol{u}}} \quad & \frac{1}{2}\hat{\boldsymbol{u}}'\boldsymbol{H}_q\hat{\boldsymbol{u}} + \bar{\boldsymbol{x}}_a(k)'\boldsymbol{F}_q\hat{\boldsymbol{u}} + \frac{1}{2}\bar{\boldsymbol{x}}_a(k)'\boldsymbol{Y}_q\bar{\boldsymbol{x}}_a(k)\\ \text{s. t.:} \quad & \boldsymbol{G}_q\hat{\boldsymbol{u}} \leq \boldsymbol{W}_q + \boldsymbol{E}_q\bar{\boldsymbol{x}}_a(k).\end{aligned} \tag{4.7}$$

According with Jones & Morari (2006), in order to apply the multi-parametric optimization, the problem (4.7) is solved using the parametric linear complementary programming (LCP), where the pQP problem is placed into the standard form using the Lagrange dual problem

$$\begin{aligned}\min_{\boldsymbol{\mu}} \quad & \frac{1}{2}\,\boldsymbol{\mu}'\,\boldsymbol{H}_d\,\boldsymbol{\mu} + (\boldsymbol{F}_d\boldsymbol{x}(k) + \boldsymbol{c}_d)'\boldsymbol{\mu}\\ \text{s. t.:} \quad & \boldsymbol{G}_d\,\boldsymbol{\mu} \;\geq\; \boldsymbol{W}_d + \boldsymbol{E}_d\,\boldsymbol{x}(k),\\ & \boldsymbol{\mu} \geq 0,\end{aligned} \tag{4.8}$$

where $\boldsymbol{\mu}$ is the dual variable, $\boldsymbol{H}_d$ is a positive semi-definite matrix, and matrices $\boldsymbol{F}_d, \boldsymbol{G}_d, \boldsymbol{E}_d$, and vectors $\boldsymbol{W}_d, \boldsymbol{c}_d$ are derived from the Lagrange dual method applied to the problem (4.7). Then, the Karush-Kuhn-Tucker(KKT) optimality conditions, derived from the optimization problem (4.8), are guiven by

$$\begin{aligned}\boldsymbol{H}_d\boldsymbol{\mu} + \boldsymbol{E}_d\boldsymbol{x}(k) + \boldsymbol{c}_d - \boldsymbol{G}_d'\boldsymbol{\lambda} - \boldsymbol{\upsilon} &= 0,\\ \boldsymbol{\lambda}'(\boldsymbol{G}_d\boldsymbol{\mu} - \boldsymbol{E}_d)\boldsymbol{x}(k) - \boldsymbol{W}_d) &= 0,\\ \boldsymbol{\upsilon}'\boldsymbol{\mu} &= 0,\\ \boldsymbol{G}_d\boldsymbol{\mu} \geq \boldsymbol{W}_d + \boldsymbol{E}_d\boldsymbol{x}(k),\ \boldsymbol{\mu} &\geq 0,\end{aligned} \tag{4.9}$$

Thereby, the pLCP problem is written considering the slack variable $\boldsymbol{\sigma} = \boldsymbol{G_d} - \boldsymbol{E_d}\boldsymbol{x}(k)$ into the KKT conditions (4.9), as follows

$$\begin{bmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\sigma} \end{bmatrix} - \begin{bmatrix} \boldsymbol{H_d} & -\boldsymbol{G_d'} \\ \boldsymbol{G_d} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F_d} \\ -\boldsymbol{E_d} \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} \boldsymbol{c_d} \\ -\boldsymbol{W_d} \end{bmatrix}, \tag{4.10}$$

$$\begin{bmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\sigma} \end{bmatrix}' \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\lambda} \end{bmatrix} = 0, \quad \boldsymbol{\upsilon}, \boldsymbol{\sigma}, \boldsymbol{\mu}, \boldsymbol{\lambda} \geq 0, \tag{4.11}$$

where $\boldsymbol{\lambda}$, $\boldsymbol{\upsilon}$ are Lagrange multiplier vectors. The control inputs are given by the solution of the following multi-parametric LCP problem:

$$\begin{aligned} \text{find} \quad & \boldsymbol{w}, \boldsymbol{z} \\ s.t. \quad & \boldsymbol{w} - \boldsymbol{M}\boldsymbol{z} = q + \mathcal{Q}\boldsymbol{x}, \\ & \boldsymbol{w}'\boldsymbol{z} = 0, \\ & \boldsymbol{w}, \boldsymbol{z} \geq 0, \\ & \boldsymbol{x}(k) \in \bar{\mathcal{X}}, \end{aligned} \tag{4.12}$$

where $\boldsymbol{w} = [\boldsymbol{\upsilon}\ \boldsymbol{\sigma}]'$, $\boldsymbol{z} = [\boldsymbol{\mu}\ \boldsymbol{\lambda}]'$, $\boldsymbol{q} = [\boldsymbol{c_d}\ -\boldsymbol{W_d}]$, $\mathcal{Q} = [\boldsymbol{F_d}\ -\boldsymbol{E_d}]'$, and $\boldsymbol{M} = \begin{bmatrix} \boldsymbol{H_d} & -\boldsymbol{G_d'} \\ \boldsymbol{G_d} & 0 \end{bmatrix}$.

In a general way, the multi-parametric optimization splits the set $\bar{\mathcal{X}}$ into subsets, named critical regions, defined as

$$CR^j = \{\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) \in \mathbb{R}^n : \boldsymbol{A_{Rj}}\ \bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) \leq \boldsymbol{b_{Rj}}\},$$

and computes an affine function for each critical region in the following form:

$$\bar{\boldsymbol{u}}(k) = \begin{cases} \boldsymbol{K}_1\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) + \boldsymbol{c}_1 & if \quad \bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) \in CR^1, \\ \boldsymbol{K}_2\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) + \boldsymbol{c}_2 & if \quad \bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) \in CR^2, \\ \quad \vdots & \quad \vdots \\ \boldsymbol{K}_s\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) + \boldsymbol{c}_s & if \quad \bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) \in CR^s, \end{cases} \tag{4.13}$$

for all $j = 1, 2, .., s$, where s is the number of critical regions derived from the optimization problem, $\boldsymbol{K}_j$ is a matrix of appropriate dimension, and $\boldsymbol{c}_j$ is a constant vector.

The critical region subsets CR^j, computed by the MPT toolbox, are characterized by Polyhedrons in the form of $\boldsymbol{A_R}\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) \leq \boldsymbol{b_R}$. As presented in Figure 4.1, these critical regions are stored in a multi-dimensional matrix $\boldsymbol{CR} \in \mathbb{R}^3$, where each slide of the matrix contains the region information represented by $\boldsymbol{CR}_j = [\boldsymbol{A_{Rj}}\ \boldsymbol{b_{Rj}}]$, with $\boldsymbol{A_{Rj}} \in \mathbb{R}^2$, $\boldsymbol{b_{Rj}} \in \mathbb{R}^2$ and $j \in \mathbb{Z}$ for all $j = 1, ..., s$. Moreover, the control laws are stored in a multi-dimensional matrix $\boldsymbol{K_R} \in \mathbb{R}^3$ composed by $\boldsymbol{K_{Rj}} = [\boldsymbol{K}_j\ \boldsymbol{c_{Rj}}]$, where $\boldsymbol{K}_j \in \mathbb{R}^2$ and $\boldsymbol{c_{Rj}} \in \mathbb{R}^2$.

In order to compute the control law at each sample time, it is proposed Algorithm 4.1,

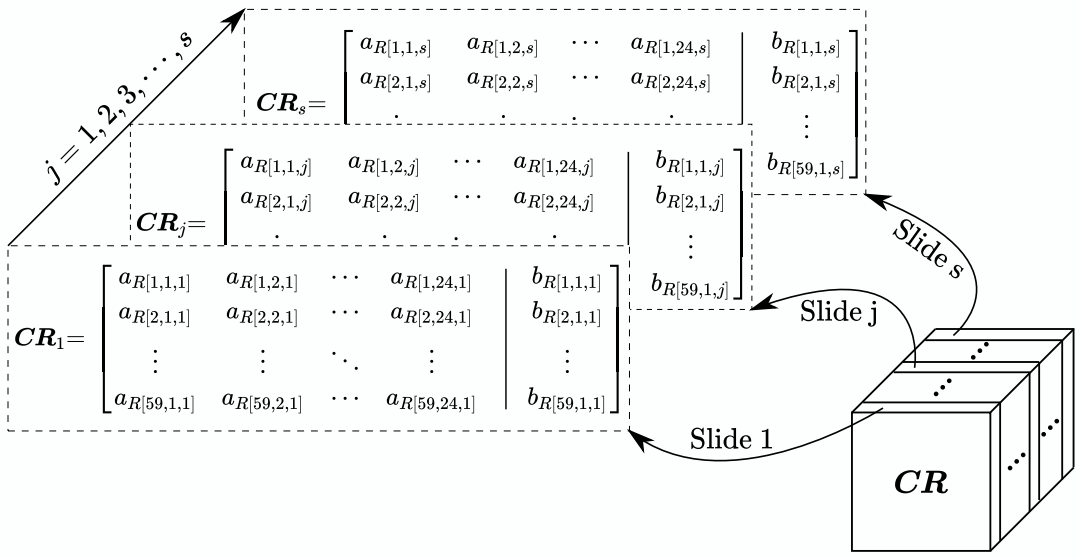

Figure 4.1: Structure of the Critical Regions Multi-dimensional Matrix.

for which it is necessary to find the critical region where the current state belongs. The algorithm is capable of accessing multiple slides of the matrix $\boldsymbol{CR}$ at the same time by the use of parallel programming. By defining the vector $\boldsymbol{x_c}(k) = [\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) \ -1]$, the algorithm verifies if the current state belongs to any of the accessed critical regions, then the search loop will stop when a critical region is founded. With the number of the found region, the algorithm computes the control signal using the slide of the matrix $\boldsymbol{K_R}$ belonging to that region and redefining the vector $\boldsymbol{x_c}(k) = [\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) \ 1]$. It is worth noting that the algorithm performance depends on the number of execution threads, which is advantageous depending on the device where the program is executed.

Algorithm 4.1 Control Law Calculation

Input: $\boldsymbol{x}(k)$
Output: $\bar{\boldsymbol{u}}(k)$
`/* Performing Parallel Execution */`
# pragma omp parallel for
for $j \leftarrow 0$ **to** $s - 1$ **do**
  $\boldsymbol{x_c} = [\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) \ -1] \quad aux = \boldsymbol{CR}_j \cdot \boldsymbol{x_c}(k)$
  if $aux \leq \varepsilon$ **then**
    $\boldsymbol{x_c} = [\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) \ 1] \quad \bar{\boldsymbol{u}}(k) = \boldsymbol{K}_{\boldsymbol{Rj}} \cdot \boldsymbol{x_c}(k)$
    break

To implement the explicit formulation of the tube-based MPC, it was used the multi-parametric toolbox (MPT3) (Herceg et al., 2013). Besides, for parallel programming, the *armadillo* c/c++ library was considered to perform multithread operations via OpenMP.

4.2 Fast MPC Optimization Based on ADMM

The second methodology focuses on developing an optimization algorithm capable to achieve a fast solution to the optimization problem outlined in (3.55). It is proposed the use of sparse optimization employing the alternating direction multiplier method (ADMM) enhanced with several techniques to reduce computational cost and accelerate the optimization process.

The methodology adopts a different approach by considering the optimization problem (3.55), contrary to other works, such as Jerez et al. (2014); Dang et al. (2015); Zhang et al. (2017), which replace the equality constraint into the cost functional and convert it in a quadratic programming (QP) function. Hence, the process starts by defining the augmented Lagrangian function (AL) as follows

$$\boldsymbol{\mathcal{L}}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}, \boldsymbol{\lambda}) = \hat{\boldsymbol{x}}' \boldsymbol{W}_y \hat{\boldsymbol{x}} + \hat{\boldsymbol{u}}' \boldsymbol{W}_u \hat{\boldsymbol{u}} + \boldsymbol{\lambda}_p' (\hat{\boldsymbol{x}} - \boldsymbol{P}\hat{\boldsymbol{u}} - \boldsymbol{Q}\bar{\boldsymbol{x}}_a(k)) + \frac{1}{2}\|\hat{\boldsymbol{x}} - \boldsymbol{P}\hat{\boldsymbol{u}} - \boldsymbol{Q}\bar{\boldsymbol{x}}_a(k)\|_{\boldsymbol{\rho}}^2, \tag{4.14}$$

where $\boldsymbol{\lambda}_p \in \mathbb{R}^n$ is the Lagrangian multipliers and $\boldsymbol{\rho} > 0$ is a predefined quadratic penalty matrix.

Then, we use the Gauss-Seidel decomposition (He et al., 2016), also known as alternating minimization, to minimize the AL function (4.14). In this process, the decision variables are updated one by one, resulting in the following standard ADMM algorithm:

$$\textbf{$\hat{\boldsymbol{x}}$-primal update: } \hat{\boldsymbol{x}}^k = \min_{\hat{\boldsymbol{x}} \in \bar{\mathcal{X}}} \boldsymbol{\mathcal{L}}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}^{k-1}, \boldsymbol{\lambda}^{k-1}), \tag{4.15}$$

$$\textbf{$\hat{\boldsymbol{u}}$-primal update: } \hat{\boldsymbol{u}}^k = \min_{\hat{\boldsymbol{u}} \in \bar{\mathcal{U}}} \boldsymbol{\mathcal{L}}(\hat{\boldsymbol{x}}^k, \hat{\boldsymbol{u}}, \boldsymbol{\lambda}^{k-1}), \tag{4.16}$$

$$\textbf{dual update: } \boldsymbol{\lambda}_p^k = \boldsymbol{\lambda}_p^{k-1} + \boldsymbol{\rho}(\hat{\boldsymbol{x}}^k - \boldsymbol{P}\hat{\boldsymbol{u}}^k - \boldsymbol{Q}\bar{\boldsymbol{x}}_a(k)), \tag{4.17}$$

where the upper index k refers to the iteration step of the algorithm. In the ADMM algorithm, the subproblems (4.15) and (4.16) optimize each decision variables $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{u}}$, respectively. In addition, equation (4.17) updates the Lagrange multipliers at each iteration.

To improve the computational efficiency of the algorithm for higher-order systems, it is employed two modified versions of the standard ADMM methodologies. One such variant is the scaled ADMM formulation, which reduces the number of mathematical operations compared to the standard ADMM formulation. By defining the residual $\boldsymbol{r} = \hat{\boldsymbol{x}}^k - \boldsymbol{P}\hat{\boldsymbol{u}}^k - \boldsymbol{Q}\bar{\boldsymbol{x}}_a$ and considering the AL function (4.14), the following expression is given

$$\begin{aligned} \boldsymbol{\lambda}_p' \boldsymbol{r} + \frac{1}{2}\|\boldsymbol{r}\|_{\boldsymbol{\rho}}^2 &= \frac{1}{2}\|\boldsymbol{r} + \boldsymbol{\rho}^{-1}\boldsymbol{\lambda}_p\|_{\boldsymbol{\rho}}^2 - \frac{1}{2}\|\boldsymbol{\rho}^{-1}\boldsymbol{\lambda}_p\|_{\boldsymbol{\rho}}^2 \\ &= \frac{1}{2}\|\boldsymbol{r} + \boldsymbol{\lambda}\|_{\boldsymbol{\rho}}^2 - \frac{1}{2}\|\boldsymbol{\lambda}\|_{\boldsymbol{\rho}}^2 \end{aligned} \tag{4.18}$$

Thus, the augmented Lagrangian function for the scaled formulation is given as follows

(Boyd et al., 2010)

$$\boldsymbol{\mathcal{L}}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}, \boldsymbol{\lambda}) = \hat{\boldsymbol{x}}' \boldsymbol{W}_y \hat{\boldsymbol{x}} + \hat{\boldsymbol{u}}' \boldsymbol{W}_u \hat{\boldsymbol{u}} - \frac{1}{2}\|\boldsymbol{\lambda}\|_{\boldsymbol{\rho}}^2 + \frac{1}{2}\|\hat{\boldsymbol{x}} - \boldsymbol{P}\hat{\boldsymbol{u}} - \boldsymbol{Q}\bar{\boldsymbol{x}}_a(k) + \boldsymbol{\lambda}\|_{\boldsymbol{\rho}}^2, \tag{4.19}$$

where the scaled dual variable $\boldsymbol{\lambda} = \boldsymbol{\rho}^{-1}\boldsymbol{\lambda}_p$. The new lagrangian multiplier update is given by $\boldsymbol{\lambda}^k = \boldsymbol{\lambda}^{k-1} + (\hat{\boldsymbol{x}}^k - \boldsymbol{P}\hat{\boldsymbol{u}}^k - \boldsymbol{Q}\boldsymbol{x}(k))$ and the optimization sub-problems are posed as shown in equations (4.15) and (4.16). Note that the new lagrangian multiplier update does not depend on the penalty parameter step $\boldsymbol{\rho}$, reducing the influence of this parameter in the lagrangian computation.

The second variant is the symmetric ADMM formulation, where the primal and dual updates follow a symmetric framework. The benefit of symmetric ADMM over the standard algorithm is that it yields a faster convergence rate, as a dual update step follows each primal update block in the algorithm.

Therefore, it is proposed the following scaled-symmetric ADMM optimization algorithm given by

$$\textbf{u-primal update: } \hat{\boldsymbol{u}}^k = \min_{\hat{\boldsymbol{u}} \in \bar{\mathcal{U}}} \boldsymbol{\mathcal{L}}(\hat{\boldsymbol{x}}^{k-1}, \hat{\boldsymbol{u}}, \boldsymbol{\lambda}^{k-1}), \tag{4.20}$$

$$\textbf{dual update: } \boldsymbol{\lambda}^{\frac{1}{2}k} = \boldsymbol{\lambda}^{k-1} + r(\hat{\boldsymbol{x}}^{k-1} - \boldsymbol{P}\hat{\boldsymbol{u}}^k - \boldsymbol{Q}\bar{\boldsymbol{x}}_a(k)), \tag{4.21}$$

$$\textbf{x-primal update: } \hat{\boldsymbol{x}}^k = \min_{\hat{\boldsymbol{x}} \in \bar{\mathcal{X}}} \boldsymbol{\mathcal{L}}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}^k, \boldsymbol{\lambda}^{\frac{1}{2}k}), \tag{4.22}$$

$$\textbf{dual update: } \boldsymbol{\lambda}^k = \boldsymbol{\lambda}^{\frac{1}{2}k} + s(\hat{\boldsymbol{x}}^k - \boldsymbol{P}\hat{\boldsymbol{u}}^k - \boldsymbol{Q}\bar{\boldsymbol{x}}_a(k)), \tag{4.23}$$

where parameters r and s are the dual step size. Note that the standard ADMM formulation is a special case of the symmetric formulation when $r = 0$ and $s = 1$. Also, the dual step size parameters are generally chosen as $r > 1$ and $s > 1$ to achieve faster convergence (Yang et al., 2022).

Taking into account the augmented Lagrangian function (4.19), it is proposed to split the function as shown in equations (4.20) and (4.22), where the main advantage is the ability to pose minimum-maximum optimization subproblems to find a closed-form solution for each subproblem. Thus, the equation (4.20) will be restricted only to the nominal state set $\bar{\mathcal{X}}$, and the equation (4.22) to the nominal control set $\bar{\mathcal{U}}$.

Thus, considering the function (4.19) and the subproblem (4.20), the optimization problem is posed as

$$\min_{\hat{\boldsymbol{u}} \in \bar{\mathcal{U}}} \hat{\boldsymbol{u}}' \boldsymbol{W}_u \hat{\boldsymbol{u}} + \frac{1}{2}\|\hat{\boldsymbol{x}} - \boldsymbol{P}\hat{\boldsymbol{u}} - \boldsymbol{Q}\bar{\boldsymbol{x}}_a(k) + \boldsymbol{\lambda}\|_{\boldsymbol{\rho}}^2. \tag{4.24}$$

We can write it as a quadratic function

$$\min_{\hat{\boldsymbol{u}} \in \bar{\mathcal{U}}} \frac{1}{2}\hat{\boldsymbol{u}}' \boldsymbol{H}_1 \hat{\boldsymbol{u}} + \boldsymbol{q}_1' \hat{\boldsymbol{u}}, \tag{4.25}$$

where $\boldsymbol{H_1} = \boldsymbol{P}'\boldsymbol{\rho}\boldsymbol{P} + 2\boldsymbol{W_u}$ and $\boldsymbol{q_1} = \boldsymbol{P}'\boldsymbol{\rho}(\boldsymbol{\lambda}^{k-1} + \hat{\boldsymbol{x}}^{k-1} - \boldsymbol{Q}\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k))$. Then, the closed-form solution for the optimization subproblem (4.25) is given by the derivative of this function with respect to the decision variable $\hat{\boldsymbol{u}}$, as follows

$$\boldsymbol{H_1}\hat{\boldsymbol{u}}^* + \boldsymbol{q_1} = 0,$$

which leads to

$$\hat{\boldsymbol{u}}^* = (\boldsymbol{H_1})^{-1}\boldsymbol{q_1}. \tag{4.26}$$

Remark 4.1. *The positive definiteness of the matrix $\boldsymbol{\rho}$ implies that the term $\boldsymbol{P}'\boldsymbol{\rho}\boldsymbol{P}$ is also a positive definite matrix. Consequently, the inverse of the matrix $\boldsymbol{H_1}$ exists since $\boldsymbol{W_u}$ and $\boldsymbol{P}'\boldsymbol{\rho}\boldsymbol{P}$ are positive definite matrices.*

Finally, the optimal solution to the subproblem (4.24) is given by (4.26), where each $i-$th element of the vector $\hat{\boldsymbol{u}}^*$ is restricted to the set $\bar{\mathcal{U}}$. Thus, the constraint is applied to all $i = 1, 2, ..., M$ elements of vector $\hat{\boldsymbol{u}}^*$ as follows

$$\hat{\boldsymbol{u}}^k[i] = \begin{cases} \hat{\boldsymbol{u}}_{\max}[i], & if \quad \hat{\boldsymbol{u}}^*[i] \geq \hat{\boldsymbol{u}}_{\max}[i], \\ \hat{\boldsymbol{u}}_{\min}[i], & if \quad \hat{\boldsymbol{u}}^*[i] \leq \hat{\boldsymbol{u}}_{\min}[i], \\ \hat{\boldsymbol{u}}^*[i], & otherwise. \end{cases}$$

The terms $\hat{\boldsymbol{u}}_{\max}$ and $\hat{\boldsymbol{u}}_{\min}$ are the maximum and minimum control signal values, respectively, obtained from the nominal control set $\bar{\mathcal{U}}$.

The same procedure is applied to the second optimization subproblem by substituting equation (4.19) into equation (4.22), resulting in the following optimization problem

$$\underset{\hat{\boldsymbol{x}} \in \bar{\mathcal{X}}}{min}\ \hat{\boldsymbol{x}}'\boldsymbol{W_y}\hat{\boldsymbol{x}} + \frac{1}{2}\|\hat{\boldsymbol{x}} - \boldsymbol{P}\hat{\boldsymbol{u}} - \boldsymbol{Q}\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) + \boldsymbol{\lambda}\|_{\boldsymbol{\rho}}^2. \tag{4.27}$$

Therefore, the closed-form solution for the optimization problem (4.27) is given by

$$\hat{\boldsymbol{x}}^* = (\boldsymbol{H_2})^{-1}\boldsymbol{q_2}, \tag{4.28}$$

where $\boldsymbol{H_2} = \boldsymbol{\rho} + 2\boldsymbol{W_x}$ and $\boldsymbol{q_2} = \boldsymbol{\rho}(\boldsymbol{P}\hat{\boldsymbol{u}}^k + \boldsymbol{Q}\bar{\boldsymbol{x}}_{\boldsymbol{a}} - \boldsymbol{\lambda}^{k-1})$. Then, each $i = 1, 2, ..., N$ element of the vector $\hat{\boldsymbol{x}}^*$ is restricted to the set $\bar{\mathcal{X}}$ as follows

$$\hat{\boldsymbol{x}}^k[i] = \begin{cases} \hat{\boldsymbol{x}}_{\max}[i], & if \quad \hat{\boldsymbol{x}}^*[i] \geq \hat{\boldsymbol{x}}_{\max}[i], \\ \hat{\boldsymbol{x}}_{\min}[i], & if \quad \hat{\boldsymbol{x}}^*[i] \leq \hat{\boldsymbol{x}}_{\min}[i], \\ \hat{\boldsymbol{x}}^*[i], & otherwise \end{cases}$$

where $\hat{\boldsymbol{x}}_{\max}$ and $\hat{\boldsymbol{x}}_{\min}$ are the maximum and minimum state values, respectively, obtained from the nominal state set $\bar{\mathcal{X}}$.

Remark 4.2. *By following the same procedure as mentioned in Remark 4.1, it can be*

observed that the inverse of the matrix $\boldsymbol{H_2}$ *exists since both* $\boldsymbol{\rho}$ *and* $\boldsymbol{W_x}$ *are positive symmetric definite matrices.*

4.2.1 Convergence Acceleration Strategies on the ADMM-based Optimization

In this section, two techniques are proposed to enhance the convergence rate of gradient-based optimization algorithms, specifically to tackle fast and high-order optimization problems. The first technique involves applying Nesterov's accelerated gradient descent to the scaled-symmetric ADMM optimization algorithm. Additionally, it is introduced a modified reset criterion that relies on the primal-dual residuals of the proposed ADMM algorithm. This criterion effectively addresses the issue of the algorithm becoming stuck during optimization.

The Nesterov technique introduces an interpolation scheme to compute the prediction of the Lagrangian multipliers and the decision variables that will be used in the next iteration. The interpolation is performed at each iteration of the algorithm optimization as follows

$$\hat{\boldsymbol{x}}_{\boldsymbol{e}}^{k} = \hat{\boldsymbol{x}}^{k} + \frac{\alpha^{k-1}-1}{\alpha^{k}}(\hat{\boldsymbol{x}}^{k} - \hat{\boldsymbol{x}}^{k-1}), \tag{4.29}$$

$$\boldsymbol{\lambda}_{\boldsymbol{e}}^{k} = \boldsymbol{\lambda}^{k} + \frac{\alpha^{k-1}-1}{\alpha^{k}}(\boldsymbol{\lambda}^{k} - \boldsymbol{\lambda}^{k-1}), \tag{4.30}$$

based on the step size given by

$$\alpha^{k} = \frac{1+\sqrt{1+4(\alpha^{k-1})^{2}}}{2}. \tag{4.31}$$

The scaled-symmetric ADMM algorithm is modified by applying equations (4.29) and (4.30) into equations (4.20) and (4.21) respectively, resulting in the following update equations:

$$\hat{\boldsymbol{u}}^{k} = \min_{\hat{\boldsymbol{u}}\in\mathcal{U}} \boldsymbol{\mathcal{L}}(\hat{\boldsymbol{x}}_{\boldsymbol{e}}^{k-1}, \hat{\boldsymbol{u}}, \boldsymbol{\lambda}_{e}^{k-1}), \tag{4.32}$$

$$\boldsymbol{\lambda}^{\frac{1}{2}k} = \boldsymbol{\lambda}_{e}^{k-1} + r(\hat{\boldsymbol{x}}_{\boldsymbol{e}}^{k-1} - \boldsymbol{P}\hat{\boldsymbol{u}}^{k} - \boldsymbol{Q}\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k)). \tag{4.33}$$

This technique is particularly attractive, as it has the potential to improve the convergence rate of first-order gradient methods from $\mathcal{O}(1/k)$ to $\mathcal{O}(1/k^2)$ (Yang et al., 2022). This represents a significant improvement in computational efficiency when using first-order information and can be considered as the best achievable rate (Nesterov, 2014).

The Reset Criteria based on the primal-dual residuals for the ADMM optimization is used as a rule to reset the Nesterov algorithm when the optimization is stagnant. According to Goldstein et al. (2014), this criterion is used when the ADMM formulation is used to

optimize a weakly convex optimization problem. However, it is proposed here the use of this algorithm to accelerate the optimization of a strongly convex function applied to a symmetric-scaled ADMM formulation.

Considering the primal feasibility condition for the optimization problem (3.55) is $\hat{\boldsymbol{x}}^* - \boldsymbol{P}\hat{\boldsymbol{u}}^* - \boldsymbol{Q}\bar{\boldsymbol{x}}_{\boldsymbol{a}}$, we can write equation (4.23) as follows

$$\hat{\boldsymbol{x}}^k - \boldsymbol{P}\hat{\boldsymbol{u}}^{\boldsymbol{k}} - \boldsymbol{Q}\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) = \boldsymbol{\rho}^{-1}(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^{k-1}).$$

Hence, the primal residual for the ADMM optimization algorithm is given by

$$\boldsymbol{c}_1^k = \boldsymbol{\rho}^{-1}(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^{k-1}).$$

Conversely, considering the dual feasibility condition for the optimization problem (3.55) is $\partial f_1(\hat{\boldsymbol{u}}^*) - \boldsymbol{P}\boldsymbol{\lambda}^*$ (Boyd et al., 2010), the partial derivative of (4.24) with respect to $\hat{\boldsymbol{u}}$ can be written as

$$\partial f_1(\hat{\boldsymbol{u}}) + \frac{1}{2}\partial(\|\hat{\boldsymbol{x}}^{k-1} - \boldsymbol{P}\hat{\boldsymbol{u}} - \boldsymbol{Q}\bar{\boldsymbol{x}}_{\boldsymbol{a}}(k) + \boldsymbol{\lambda}^{k-1}\|_{\boldsymbol{\rho}}^2) \in 0,$$

or equivalently,

$$\partial f_1(\hat{\boldsymbol{u}}^k) - \boldsymbol{P}\boldsymbol{\lambda}^k \in \boldsymbol{\rho}(\hat{\boldsymbol{x}}^k - \hat{\boldsymbol{x}}^{k-1}).$$

Therefore, the dual residual of the ADMM algorithm is given by

$$\boldsymbol{c}_2 = \boldsymbol{\rho}(\hat{\boldsymbol{x}}^k - \hat{\boldsymbol{x}}^{k-1}).$$

Hence, according to Goldstein et al. (2014), the size of the primal and dual residuals can be used as a measure to know how far the iterations are from the optimal solution, because these two residuals converge to zero as the ADMM algorithm proceeds (Boyd & Vandenberghe, 2004).

To use these residuals as reset criteria, we compute a global measure of the residuals applying the Nesterov acceleration algorithm, as follows

$$c^k = \|\boldsymbol{\rho}^{-1}(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}_{\boldsymbol{e}}^{k-1})\|_2 + \| - \boldsymbol{P}'\boldsymbol{\rho}(\hat{\boldsymbol{x}}^k - \hat{\boldsymbol{x}}_{\boldsymbol{e}}^{k-1})\|_2, \tag{4.34}$$

where, if the current value c^k is not η times less than the previous one, Algorithm 4.2 is performed.

The parameter $\eta \in (0, 1)$ is a factor used to test if the residuals are decreasing in the range that it is desired. Otherwise, the most recent iteration is thrown out and the algorithm is reset.

Finally, using equations (4.22), (4.23), (4.32), (4.33), the Nesterov's acceleration algorithm, and the reset criteria, the proposed symmetric-scaled ADMM optimization for

Algorithm 4.2 Reset Criteria based on ADMM residuals

if $c^k < \eta\ c^{k-1}$ **then**

$\alpha^k = \frac{1+\sqrt{1+4(\alpha^{k-1})^2}}{2}$

$\hat{\boldsymbol{u}}_e^k = \hat{\boldsymbol{u}}^k + \frac{\alpha^{k-1}-1}{\alpha^k}(\hat{\boldsymbol{u}}^k - \hat{\boldsymbol{u}}^{k-1})$

$\boldsymbol{\lambda}_e^k = \boldsymbol{\lambda}^k + \frac{\alpha^{k-1}-1}{\alpha^k}(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^{k-1})$

else

$\alpha^k = 1$

$c^k = \eta^{-1} c^{k-1}$

$\hat{\boldsymbol{u}}_e^k = \hat{\boldsymbol{u}}^{k-1}$

$\boldsymbol{\lambda}_e^k = \boldsymbol{\lambda}^{k-1}$

high-order and fast dynamic systems is summarized in Algorithm 4.3.

Algorithm 4.3 ADMM Optimization Algorithm

Result: Optimun $\hat{\boldsymbol{u}}$

k=1, $\boldsymbol{\lambda}_e^0 = 0, \boldsymbol{\lambda}^0 = 0, \hat{\boldsymbol{u}}^0 = 0, \hat{\boldsymbol{u}}_e^0 = 0, \alpha^0 = 1$

while $\|\Gamma\|_\infty \leq \epsilon$ **do**

$\hat{\boldsymbol{x}}^k = \underset{\hat{\boldsymbol{x}} \in \bar{\mathcal{X}}}{min}\ \boldsymbol{\mathcal{L}}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}_e^{k-1}, \boldsymbol{\lambda}_e^{k-1})$

$\hat{\boldsymbol{u}}^k = \underset{\hat{\boldsymbol{u}} \in \bar{\mathcal{U}}}{min}\ \boldsymbol{\mathcal{L}}(\hat{\boldsymbol{x}}^k, \hat{\boldsymbol{u}}, \boldsymbol{\lambda}^{k-1})$

$\boldsymbol{\Gamma} = \hat{\boldsymbol{x}}^k - \boldsymbol{P}\hat{\boldsymbol{u}}^k - \boldsymbol{Q}\boldsymbol{x}(k)$

$\boldsymbol{\lambda}^k = \boldsymbol{\lambda}_e^{k-1} + \boldsymbol{\rho}(\boldsymbol{\Gamma})$

$c_1 = \|\boldsymbol{\rho}^{-1}(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}_e^{k-1})\|_2$

$c_2 = \| - \boldsymbol{P}'\boldsymbol{\rho}(\hat{\boldsymbol{x}}^k - \hat{\boldsymbol{x}}_e^{k-1})\|_2$

$c^k = c_1 + c_2$

if $c^k < \eta\ c^{k-1}$ **then**

$\alpha^k = \frac{1+\sqrt{1+4(\alpha^{k-1})^2}}{2}$

$\hat{\boldsymbol{u}}_e^k = \hat{\boldsymbol{u}}^k + \frac{\alpha^{k-1}-1}{\alpha^k}(\hat{\boldsymbol{u}}^k - \hat{\boldsymbol{u}}^{k-1})$

$\boldsymbol{\lambda}_e^k = \boldsymbol{\lambda}^k + \frac{\alpha^{k-1}-1}{\alpha^k}(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^{k-1})$

else

$\alpha^k = 1$

$c^k = \eta^{-1} c^{k-1}$

$\hat{\boldsymbol{u}}_e^k = \hat{\boldsymbol{u}}^{k-1}$

$\boldsymbol{\lambda}_e^k = \boldsymbol{\lambda}^{k-1}$

end

$k = k + 1$

end

Algorithm 4.3 has the particularity to be well suited for concurrent computation. This is due to the nature of the operations involved in the algorithm, which mainly consist of comparison operations and matrix-matrix or matrix-vector multiplication. The main advantage of these types of operations is that they can be efficiently accelerated using parallel computation techniques, improving the efficiency and speed of the algorithm.

4.3 Nominal MPC Optimization Using Constrained Zonotopes

In this section, a novel MPC formulation is developed based on the constrained zonotope representation. In the proposed formulation, the sets $\bar{\mathcal{U}}$ and $\bar{\mathcal{X}}$ are exactly mapped from the polytopic set representation to the constrained zonotope set representation, as shown in Figure 4.2. Therefore, it is expected that the proposed MPC problem achieve an optimal solution similar to the standard MPC. Conversely, an improvement in computation time is expected due to the computational efficiency afforded by the set representation employed.

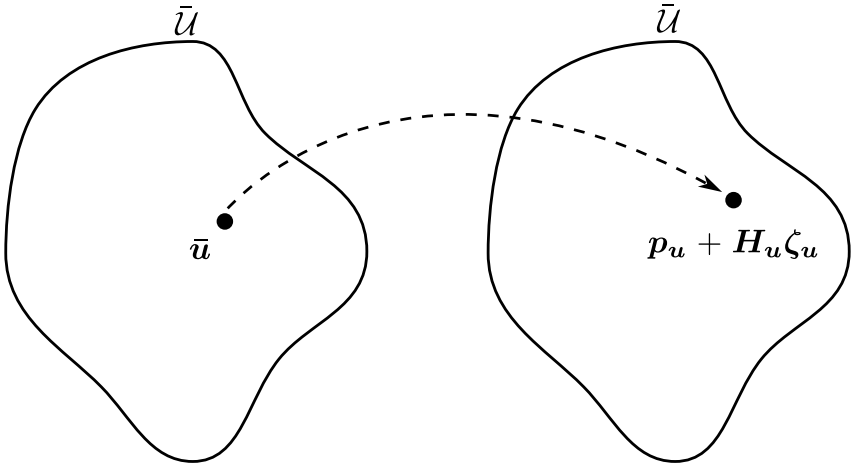

Figure 4.2: Mapping an specific nominal control vector from the polytopic set representation to the constrained zonotope set representation.

The process starts by mapping the nominal set from the polytopic to the constrained zonotope set representation. Therefore, the sets $\bar{\mathcal{U}}$ and $\bar{\mathcal{X}}$ are considered as constrained zonotope defined as follows

$$\bar{\boldsymbol{u}}(i) = \boldsymbol{p_u} + \boldsymbol{H_u}\boldsymbol{\zeta_u}(i), \quad \boldsymbol{A_u}\boldsymbol{\zeta_u}(i) == \boldsymbol{b_u}, \tag{4.35}$$

$$\bar{\boldsymbol{x}}_{\boldsymbol{a}}(i) = \boldsymbol{p_x} + \boldsymbol{H_x}\boldsymbol{\zeta_x}(i), \quad \boldsymbol{A_x}\boldsymbol{\zeta_x}(i) == \boldsymbol{b_x}. \tag{4.36}$$

Then, the state prediction vector, in equation (3.52), is depicted in the constrained zonotope set representation by the following two equations:

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \bar{\boldsymbol{x}}_{\boldsymbol{a}}(1) \\ \bar{\boldsymbol{x}}_{\boldsymbol{a}}(2) \\ \bar{\boldsymbol{x}}_{\boldsymbol{a}}(3) \\ \vdots \\ \bar{\boldsymbol{x}}_{\boldsymbol{a}}(N) \end{bmatrix} = \begin{bmatrix} \boldsymbol{p_x} + \boldsymbol{H_x}\boldsymbol{\zeta_x}(1) \\ \boldsymbol{p_x} + \boldsymbol{H_x}\boldsymbol{\zeta_x}(2) \\ \boldsymbol{p_x} + \boldsymbol{H_x}\boldsymbol{\zeta_x}(3) \\ \vdots \\ \boldsymbol{p_x} + \boldsymbol{H_x}\boldsymbol{\zeta_x}(N) \end{bmatrix}, \quad \begin{bmatrix} \boldsymbol{A_x}\boldsymbol{\zeta_x}(1) \\ \boldsymbol{A_x}\boldsymbol{\zeta_x}(2) \\ \boldsymbol{A_x}\boldsymbol{\zeta_x}(3) \\ \vdots \\ \boldsymbol{A_x}\boldsymbol{\zeta_x}(N) \end{bmatrix} = \begin{bmatrix} \boldsymbol{b_x} \\ \boldsymbol{b_x} \\ \boldsymbol{b_x} \\ \vdots \\ \boldsymbol{b_x} \end{bmatrix}, \tag{4.37}$$

However, by defining the prediction vector $\hat{\boldsymbol{\zeta}}_{\boldsymbol{x}} = [\boldsymbol{\zeta}_{\boldsymbol{x}}(1), \boldsymbol{\zeta}_{\boldsymbol{x}}(2), \ldots, \boldsymbol{\zeta}_{\boldsymbol{x}}(N)]$, the vectors in equation (4.37) are written in a matrix form as following:

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{p}_{\boldsymbol{x}} \\ \boldsymbol{p}_{\boldsymbol{x}} \\ \boldsymbol{p}_{\boldsymbol{x}} \\ \vdots \\ \boldsymbol{p}_{\boldsymbol{x}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{H}_{\boldsymbol{x}} & 0 & 0 & \ldots & 0 \\ 0 & \boldsymbol{H}_{\boldsymbol{x}} & 0 & \ldots & 0 \\ 0 & 0 & \boldsymbol{H}_{\boldsymbol{x}} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \boldsymbol{H}_{\boldsymbol{x}} \end{bmatrix} \hat{\boldsymbol{\zeta}}_{\boldsymbol{x}} \quad \rightarrow \boldsymbol{c}_{\boldsymbol{x}} + \boldsymbol{G}_{\boldsymbol{x}} \hat{\boldsymbol{\zeta}}_{\boldsymbol{x}} \tag{4.38}$$

$$\begin{bmatrix} \boldsymbol{A}_{\boldsymbol{x}} & 0 & 0 & \ldots & 0 \\ 0 & \boldsymbol{A}_{\boldsymbol{x}} & 0 & \ldots & 0 \\ 0 & 0 & \boldsymbol{A}_{\boldsymbol{x}} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \boldsymbol{A}_{\boldsymbol{x}} \end{bmatrix} \hat{\boldsymbol{\zeta}}_{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{b}_{\boldsymbol{x}} \\ \boldsymbol{b}_{\boldsymbol{x}} \\ \boldsymbol{b}_{\boldsymbol{x}} \\ \vdots \\ \boldsymbol{b}_{\boldsymbol{x}} \end{bmatrix} \quad \rightarrow \boldsymbol{F}_{\boldsymbol{x}} \hat{\boldsymbol{\zeta}}_{\boldsymbol{x}} == \boldsymbol{f}_{\boldsymbol{x}}. \tag{4.39}$$

Similarly, the control prediction vector of equation (3.52) is rewritten, using the constrained zonotope set representation (4.36), as follows:

$$\hat{\boldsymbol{u}} = \begin{bmatrix} \bar{\boldsymbol{u}}(0) \\ \bar{\boldsymbol{u}}(1) \\ \bar{\boldsymbol{u}}(2) \\ \vdots \\ \bar{\boldsymbol{u}}(M-1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_{\boldsymbol{u}} + \boldsymbol{H}_{\boldsymbol{u}} \boldsymbol{\zeta}_{\boldsymbol{u}}(0) \\ \boldsymbol{p}_{\boldsymbol{u}} + \boldsymbol{H}_{\boldsymbol{u}} \boldsymbol{\zeta}_{\boldsymbol{u}}(1) \\ \boldsymbol{p}_{\boldsymbol{u}} + \boldsymbol{H}_{\boldsymbol{u}} \boldsymbol{\zeta}_{\boldsymbol{u}}(2) \\ \vdots \\ \boldsymbol{p}_{\boldsymbol{u}} + \boldsymbol{H}_{\boldsymbol{u}} \boldsymbol{\zeta}_{\boldsymbol{u}}(M-1) \end{bmatrix}, \quad \begin{bmatrix} \boldsymbol{A}_{\boldsymbol{u}} \boldsymbol{\zeta}_{\boldsymbol{u}}(0) \\ \boldsymbol{A}_{\boldsymbol{u}} \boldsymbol{\zeta}_{\boldsymbol{u}}(1) \\ \boldsymbol{A}_{\boldsymbol{u}} \boldsymbol{\zeta}_{\boldsymbol{u}}(2) \\ \vdots \\ \boldsymbol{A}_{\boldsymbol{u}} \boldsymbol{\zeta}_{\boldsymbol{u}}(M-1) \end{bmatrix} == \begin{bmatrix} \boldsymbol{b}_{\boldsymbol{u}} \\ \boldsymbol{b}_{\boldsymbol{u}} \\ \boldsymbol{b}_{\boldsymbol{u}} \\ \vdots \\ \boldsymbol{b}_{\boldsymbol{u}} \end{bmatrix} \tag{4.40}$$

Then, by defining the prediction control vector $\hat{\boldsymbol{\zeta}}_{\boldsymbol{u}} = [\boldsymbol{\zeta}_{\boldsymbol{u}}(0), \boldsymbol{\zeta}_{\boldsymbol{u}}(1), \ldots, \boldsymbol{\zeta}_{\boldsymbol{u}}(M-1)]$, the equalities in (4.40) are written in a matrix form as follows

$$\hat{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{p}_{\boldsymbol{u}} \\ \boldsymbol{p}_{\boldsymbol{u}} \\ \boldsymbol{p}_{\boldsymbol{u}} \\ \vdots \\ \boldsymbol{p}_{\boldsymbol{u}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{H}_{\boldsymbol{u}} & 0 & 0 & \ldots & 0 \\ 0 & \boldsymbol{H}_{\boldsymbol{u}} & 0 & \ldots & 0 \\ 0 & 0 & \boldsymbol{H}_{\boldsymbol{u}} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \boldsymbol{H}_{\boldsymbol{u}} \end{bmatrix} \hat{\boldsymbol{\zeta}}_{\boldsymbol{u}} \quad \rightarrow \boldsymbol{c}_{\boldsymbol{u}} + \boldsymbol{G}_{\boldsymbol{u}} \hat{\boldsymbol{\zeta}}_{\boldsymbol{u}}, \tag{4.41}$$

$$\begin{bmatrix} \boldsymbol{A}_{\boldsymbol{u}} & 0 & 0 & \ldots & 0 \\ 0 & \boldsymbol{A}_{\boldsymbol{u}} & 0 & \ldots & 0 \\ 0 & 0 & \boldsymbol{A}_{\boldsymbol{u}} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \boldsymbol{A}_{\boldsymbol{u}} \end{bmatrix} \hat{\boldsymbol{\zeta}}_{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{b}_{\boldsymbol{u}} \\ \boldsymbol{b}_{\boldsymbol{u}} \\ \boldsymbol{b}_{\boldsymbol{u}} \\ \vdots \\ \boldsymbol{b}_{\boldsymbol{u}} \end{bmatrix} \quad \rightarrow \boldsymbol{F}_{\boldsymbol{u}} \hat{\boldsymbol{\zeta}}_{\boldsymbol{u}} = \boldsymbol{f}_{\boldsymbol{u}}. \tag{4.42}$$

The next step is to express the cost functional (3.54) in terms of the constrained zonotope parameters $\hat{\boldsymbol{\zeta}}_{\boldsymbol{x}}$ and $\hat{\boldsymbol{\zeta}}_{\boldsymbol{u}}$. By replacing equations (4.38) and (4.41) into (3.54), the

cost functional can be written as follows

$$V(\hat{\boldsymbol{\zeta}}_{\boldsymbol{x}}, \hat{\boldsymbol{\zeta}}_{\boldsymbol{u}}) = \boldsymbol{f_1}(\hat{\boldsymbol{\zeta}}_{\boldsymbol{u}}) + \boldsymbol{f_2}(\hat{\boldsymbol{\zeta}}_{\boldsymbol{x}}), \tag{4.43}$$

where the costs $\boldsymbol{f_1}(\hat{\boldsymbol{\zeta}}_{\boldsymbol{u}})$ and $\boldsymbol{f_2}(\hat{\boldsymbol{\zeta}}_{\boldsymbol{x}})$ are given by

$$\begin{aligned}
\boldsymbol{f_1}(\hat{\boldsymbol{\zeta}}_{\boldsymbol{u}}) &= \hat{\boldsymbol{\zeta}}_{\boldsymbol{u}}' \left[\boldsymbol{G}_{\boldsymbol{u}}' \boldsymbol{W}_{\boldsymbol{u}} \boldsymbol{G}_{\boldsymbol{u}}\right] \hat{\boldsymbol{\zeta}}_{\boldsymbol{u}} + 2 \left[\boldsymbol{c}_{\boldsymbol{u}}' \boldsymbol{W}_{\boldsymbol{u}} \boldsymbol{G}_{\boldsymbol{u}}\right] \hat{\boldsymbol{\zeta}}_{\boldsymbol{u}} + \boldsymbol{c}_{\boldsymbol{u}}' \boldsymbol{W}_{\boldsymbol{u}} \boldsymbol{c}_{\boldsymbol{u}}, \\
\boldsymbol{f_2}(\hat{\boldsymbol{\zeta}}_{\boldsymbol{x}}) &= \hat{\boldsymbol{\zeta}}_{\boldsymbol{x}}' \left[\boldsymbol{G}_{\boldsymbol{x}}' \boldsymbol{W}_{\boldsymbol{x}} \boldsymbol{G}_{\boldsymbol{x}}\right] \hat{\boldsymbol{\zeta}}_{\boldsymbol{x}} + 2 \left[\boldsymbol{c}_{\boldsymbol{x}}' \boldsymbol{W}_{\boldsymbol{x}} \boldsymbol{G}_{\boldsymbol{x}}\right] \hat{\boldsymbol{\zeta}}_{\boldsymbol{x}} + \boldsymbol{c}_{\boldsymbol{x}}' \boldsymbol{W}_{\boldsymbol{x}} \boldsymbol{c}_{\boldsymbol{x}}.
\end{aligned}$$

Furthermore, the predicted output model (3.53) can be rewritten in terms of $\hat{\boldsymbol{\zeta}}_{\boldsymbol{x}}$ and $\hat{\boldsymbol{\zeta}}_{\boldsymbol{u}}$. Thus, the predicted output model is given by

$$\boldsymbol{G}_{\boldsymbol{x}} \hat{\boldsymbol{\zeta}}_{\boldsymbol{x}} - \boldsymbol{P} \boldsymbol{G}_{\boldsymbol{u}} \hat{\boldsymbol{\zeta}}_{\boldsymbol{u}} = \boldsymbol{P} \boldsymbol{c}_{\boldsymbol{u}} - \boldsymbol{c}_{\boldsymbol{x}} + \boldsymbol{Q} \bar{\boldsymbol{x}}_{\boldsymbol{a}}(\boldsymbol{k}). \tag{4.44}$$

Considering the constrained zonotope Definition 3.4, the condition $\|\boldsymbol{\zeta}\|_\infty \leq 1$ must be written in a form that can be used in the nominal optimization problem in order to satisfy the constrained zonotope condition. Therefore, the infinity norm $\|\boldsymbol{\zeta}\|_\infty$ can be written as the maximum condition $\underset{1 \leq i \leq m}{max} |\zeta(i)|$.

Then, $\underset{1 \leq i \leq m}{max} |\zeta(i)|$ can be transformed into a unitary box, as follows

$$\left.\begin{array}{l} |\zeta(1)| \leq 1 \\ |\zeta(2)| \leq 1 \\ \quad \vdots \\ |\zeta(m)| \leq 1 \end{array}\right\} = \left\{\begin{array}{c} -1 \leq \zeta(1) \leq 1 \\ -1 \leq \zeta(2) \leq 1 \\ \vdots \\ -1 \leq \zeta(m) \leq 1 \end{array}\right\} = -1 \leq \boldsymbol{\zeta} \leq 1. \tag{4.45}$$

Finally, by considering the cost functional (4.43), the predicted output model (4.44) and the constraints (4.38), (4.39), (4.41), (4.42), and (4.45), the nominal MPC optimization problem based on constrained zonotopes can be posed as

$$\begin{aligned}
\underset{\hat{\boldsymbol{\zeta}}_{\boldsymbol{u}}, \hat{\boldsymbol{\zeta}}_{\boldsymbol{x}}}{min} \quad & \left\{\boldsymbol{f_1}(\hat{\boldsymbol{\zeta}}_{\boldsymbol{u}}) + \boldsymbol{f_2}(\hat{\boldsymbol{\zeta}}_{\boldsymbol{x}})\right\} \\
\text{s. t.:} \quad & \boldsymbol{G}_{\boldsymbol{x}} \hat{\boldsymbol{\zeta}}_{\boldsymbol{x}} - \boldsymbol{P} \boldsymbol{G}_{\boldsymbol{u}} \hat{\boldsymbol{\zeta}}_{\boldsymbol{u}} = \boldsymbol{P} \boldsymbol{c}_{\boldsymbol{u}} - \boldsymbol{c}_{\boldsymbol{x}} + \boldsymbol{Q} \bar{\boldsymbol{x}}_{\boldsymbol{a}}(\boldsymbol{k}) \\
& \boldsymbol{F}_{\boldsymbol{u}} \hat{\boldsymbol{\zeta}}_{\boldsymbol{u}} = \boldsymbol{f}_{\boldsymbol{u}} \\
& \boldsymbol{F}_{\boldsymbol{x}} \hat{\boldsymbol{\zeta}}_{\boldsymbol{x}} = \boldsymbol{f}_{\boldsymbol{x}}, \\
& -1 \leq \hat{\boldsymbol{\zeta}}_{\boldsymbol{u}} \leq 1, \\
& -1 \leq \hat{\boldsymbol{\zeta}}_{\boldsymbol{x}} \leq 1.
\end{aligned} \tag{4.46}$$

It is worth highlighting that the solution of the optimization problem is the optimal control sequence of the variable $\hat{\boldsymbol{\zeta}}_{\boldsymbol{u}}$. Therefore, the optimal control signal $\bar{\boldsymbol{u}}(0)$ is obtained

as

$$\bar{\boldsymbol{u}}(0) = \boldsymbol{p_u} + \boldsymbol{H_u}\boldsymbol{\zeta_u}(0). \tag{4.47}$$

4.4 Final Remarks

In this chapter, two methodologies were developed to solve the optimization problem (3.55). The first approach used a parametric optimization method to find offline the optimal solution of the nominal MPC problem. It resulted in a group of matrices used by a search algorithm proposed in this chapter. It takes advantage of concurrent computing to perform a parallel search of the nominal control law belonging to the current system state. The second approach involved an online solution of the nominal optimization problem via ADMM. A novel splitting of the quadratic cost function was proposed to perform the exact optimization of the subproblems, which leads to a fast computation of the optimal solution of each subproblem. Moreover, fast gradient computation was employed with a new reset algorithm to accelerate the optimization of the master problem. This reset algorithm was modified to include a penalty matrix, improving the performance with MIMO systems. Furthermore, a new MPC optimization problem based on constrained zonotopes was developed to accelerate the search for an optimal solution due to the efficient computation properties of this set representation. In Chapter 6, the proposed formulation is applied to a tilt-rotor UAV carrying a suspended load to verify the algorithm's performance.

5 Case Study

This chapter introduces the case study considered for the experiments performed in this doctoral thesis. The system is a tiltrotor UAV with suspended load, characterized by presenting fast and high-order dynamics, making it suitable to verify the applicability of the proposed control formulations. In addition, this chapter presents the obtained offline computation results for the tube-based MPC developed in Chapter 3 when applied to the case study.

5.1 Tiltrotor UAV Load Perspective Non-linear Model

This section presents the multi-body dynamic model of a tiltrotor UAV with suspended load, for which the equations of motion of the system are obtained from the load perspective. The system is composed by four rigid bodies (see Figure 5.1), using eight reference frames defined as: frame $\mathcal{B}$ attached to the rotation axis of the UAV; the inertial reference frame $\mathcal{I}$; frames $\mathcal{C}_i$ which are attached to the center of mass of each rigid body; auxiliary frames $\mathcal{A}_i$ attached to the connection point of each rigid body with the main body; and finally, frame $\mathcal{L}$ attached to the center of mass of the load.

The translational position vector of the load frame $\mathcal{L}$ with respect to $\mathcal{I}$ is described by $\boldsymbol{\xi} = \begin{bmatrix} x & y & z \end{bmatrix}'$[1], while the rotational position of frame $\mathcal{L}$ with respect to $\mathcal{I}$ is parameterized by the Euler angles, $\boldsymbol{\eta} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}'$, using the ZYX convention around the local axes.

[1]The prime $'$ notation denotes the transpose operator.

The angular positions of the right and left tilting mechanisms are represented by $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_r & \alpha_l \end{bmatrix}'$, while the angular position of the suspended load around axis x and y of frame $\mathcal{B}$ is given by $\boldsymbol{\gamma} = \begin{bmatrix} \gamma_1 & \gamma_2 \end{bmatrix}'$. Consequently, the generalized coordinates vector is composed by $\boldsymbol{q} = \begin{bmatrix} \boldsymbol{\xi}' & \boldsymbol{\eta}' & \boldsymbol{\gamma}' & \boldsymbol{\alpha}' \end{bmatrix}'$.

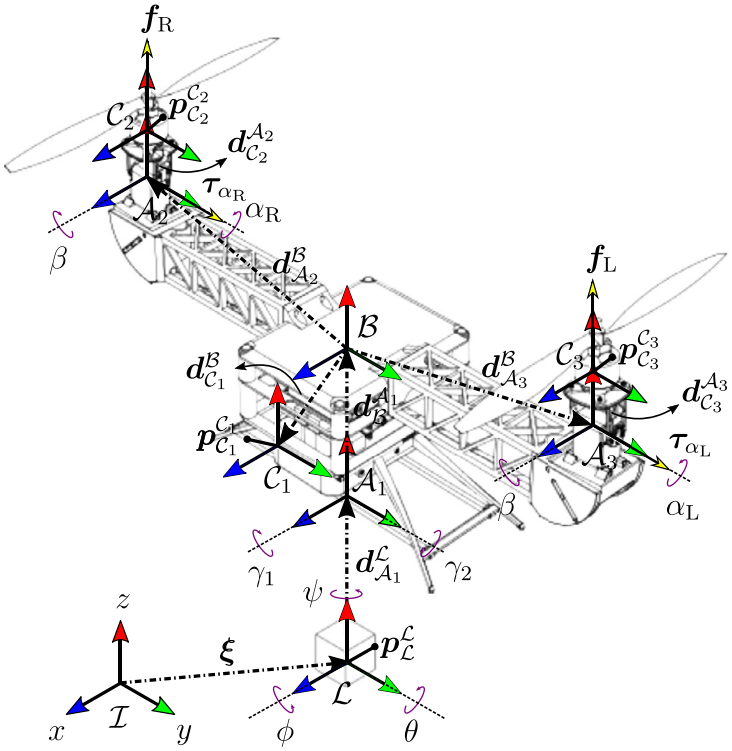

Figure 5.1: The tiltrotor UAV with suspended load description, in which the reference frames, kinematic parameters, control inputs and generalized coordinates are illustrated.

The Euler-Lagrange formulation is applied to obtain the dynamic model of the tiltrotor UAV with suspended load, from which the canonical form is written as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\vartheta}_{in} + \boldsymbol{\vartheta}_{fr} + \boldsymbol{\vartheta}_{db}, \tag{5.1}$$

where $\boldsymbol{M}(\boldsymbol{q})$ is the inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is the Coriolis and centripetal forces matrix, and $\boldsymbol{G}(\boldsymbol{q})$ is the gravitational force vector. $\boldsymbol{M}(\boldsymbol{q})$ can be obtained from the sum of the kinetic energies of all bodies, expressing it in the form $\boldsymbol{K} = \frac{1}{2}\dot{\boldsymbol{q}}'\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}$. The matrix $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is derived from $\boldsymbol{M}(\boldsymbol{q})$ using the Christoffel symbols of the first kind, while the gravitational force vector is computed as $\boldsymbol{G}(\boldsymbol{q}) = \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{q}}$, with $\boldsymbol{P}$ being the sum of the potential energies of all bodies constituting the aircraft.

In the right side of equation (5.1), $\boldsymbol{\vartheta}_{in}$ is the vector of generalized forces/torques, $\boldsymbol{\vartheta}_{fr}$ is known as the drag force vector, and $\boldsymbol{\vartheta}_{db}$ is the disturbance force vector applied to the system. The applied generalized force/torque vector, $\boldsymbol{\vartheta}_{in}$, can be rewritten as a product of a matrix $\boldsymbol{L}_{in}(\boldsymbol{q})$ and the control input vector $\boldsymbol{u}(t) = \begin{bmatrix} f_r & f_l & \tau_r & \tau_l \end{bmatrix}'$, where f_r and f_l are, respectively, the thrusts generated by the right and left propellers, and τ_r and τ_l are,

respectively, the right and left torques applied to the servomotors. Besides, $\boldsymbol{\vartheta_{fr}}$ can be assumed proportional to the velocities of the system using $\boldsymbol{\vartheta_{fr}} = -\boldsymbol{L}_{fr}\dot{\boldsymbol{q}}$. Finally, $\boldsymbol{\vartheta_{db}}$ can be expressed as $\boldsymbol{\vartheta_{db}} = \boldsymbol{L}_{db}\boldsymbol{d}$, where $\boldsymbol{L}_{db} = [I_{3\times3}\ 0_{3\times3}\ 0_{3\times2}\ 0_{3\times1}\ 0_{3\times1}]'$ and $\boldsymbol{d}$ is the external disturbance force vector applied to the suspended the load.

Therefore, from equation (5.1), the generalized coordinate acceleration vector is defined as

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}(\boldsymbol{q})^{-1}(\boldsymbol{L}_{in}(\boldsymbol{q})\boldsymbol{u} - [\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{L}_{fr}]\dot{\boldsymbol{q}} - \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{L}_{db}\boldsymbol{d}). \tag{5.2}$$

Defining the state space vector $\boldsymbol{x_s}(t) \triangleq \begin{bmatrix}\boldsymbol{q} & \dot{\boldsymbol{q}}\end{bmatrix}'$, the nonlinear state-space model of the system is expressed as

$$\dot{\boldsymbol{x}}_{\boldsymbol{s}}(t) = f(\boldsymbol{x_s}(t), \boldsymbol{u}(t), \boldsymbol{d}(t)) = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}(\boldsymbol{q})^{-1}(\boldsymbol{L}_{in}(\boldsymbol{q})\boldsymbol{u} - [\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{L}_{fr}]\dot{\boldsymbol{q}} - \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{L}_{db}\boldsymbol{d}) \end{bmatrix}. \tag{5.3}$$

Assuming $\dot{\boldsymbol{x}}_{\boldsymbol{s}}(t) = 0$, the equilibrium point $(\boldsymbol{u}^*, \boldsymbol{q}^*, \dot{\boldsymbol{q}}^*, \boldsymbol{d}^*)$, with $\boldsymbol{d}^* = 0$, can be calculated by solving the system from equation (5.3). For more details about the modeling of the tiltrotor UAV with suspended load, see Rego & Raffo (2019).

5.2 Linearization Through a Known Trajectory

In this section, the discrete-time linear error model of the tiltrotor UAV with suspended load is obtained by linearizing the nonlinear model around a generic desired trajectory, to be used in Chapter 6 in the tube-based MPC approaches.

Reference Control Computation

As the system has eight DOF and only four control signals, it is considered an underactuated system. Thus, the reference trajectory vector is defined to require trajectory tracking of four DOF, while the remaining states must be stabilized around an equilibrium point. Thus, the reference vector is given by $\boldsymbol{x_{sr}}(t) = [x_r(t)\ y_r(t)\ z_r(t)\ \phi^*\ \theta^*\ \psi_r(t)\ \alpha_r^*\ \alpha_l^*\ \gamma_1^*\ \gamma_2^*\ \dot{x}_r(t)\ \dot{y}_r(t)\ \dot{z}_r(t)\ 0\ 0\ 0\ 0\ 0\ 0\ 0]'$, where prefix r denotes reference trajectory, and $*$ means equilibrium point.

Thereafter, the reference control vector is computed using the nonlinear state model (5.3) with

$$\boldsymbol{u_r} = \boldsymbol{L}_{in}(\boldsymbol{q_r})^{\#}(\boldsymbol{M}(\boldsymbol{q_r})\ddot{\boldsymbol{q}}_{\boldsymbol{r}} + [\boldsymbol{C}(\boldsymbol{q_r},\dot{\boldsymbol{q}}_{\boldsymbol{r}}) + \boldsymbol{L}_{fr}]\dot{\boldsymbol{q}}_{\boldsymbol{r}} + \boldsymbol{G}(\boldsymbol{q_r})), \tag{5.4}$$

where $\boldsymbol{q_r}$ is assumed feasible and $\boldsymbol{L}_{in}(\boldsymbol{q_r})^{\#}$ exists[2].

[2] # means the pseudo inverse operator of a matrix.

Discrete-Time Linear Error Model

The following continuous-time linear error model is obtained from (5.3) using the first order terms of the Taylor's series expansion around the desired trajectory $(\boldsymbol{x}_{sr}(t), \boldsymbol{u}_r(t), 0)$ and defining the error state, the error control and the disturbance vectors as $\tilde{\boldsymbol{x}}_s(t) = \boldsymbol{x}_s(t) - \boldsymbol{x}_{sr}(t)$, $\tilde{\boldsymbol{u}}(t) = \boldsymbol{u}(t) - \boldsymbol{u}_r(t)$ and $\tilde{\boldsymbol{d}} = \boldsymbol{d}$, respectively:

$$\begin{aligned} \dot{\tilde{\boldsymbol{x}}}_s(t) &= \boldsymbol{A}(t)\tilde{\boldsymbol{x}}_s(t) + \boldsymbol{B}\tilde{\boldsymbol{u}}(t) + \boldsymbol{D}\boldsymbol{d}(t), \\ \tilde{\boldsymbol{y}}_s(t) &= \boldsymbol{C}\tilde{\boldsymbol{x}}_s(t), \end{aligned} \tag{5.5}$$

where $\boldsymbol{A}(t) = \frac{\partial f(\boldsymbol{x_s},\boldsymbol{u},\boldsymbol{d})}{\partial \boldsymbol{x_s}}\Big|_{\substack{\boldsymbol{x_s}=\boldsymbol{x_{sr}} \\ \boldsymbol{u}=\boldsymbol{u_r}}}$, $\boldsymbol{B} = \frac{\partial f(\boldsymbol{x_s},\boldsymbol{u},\boldsymbol{d})}{\partial \boldsymbol{u}}\Big|_{\substack{\boldsymbol{x_s}=\boldsymbol{x_{sr}} \\ \boldsymbol{u}=\boldsymbol{u_r}}}$, $\boldsymbol{D} = \frac{\partial f(\boldsymbol{x_s},\boldsymbol{u},\boldsymbol{d})}{\partial \boldsymbol{d}}\Big|_{\substack{\boldsymbol{x_s}=\boldsymbol{x_{sr}} \\ \boldsymbol{u}=\boldsymbol{u_r}}}$, and

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}.$$

Since matrix $\boldsymbol{A}(t)$ is time varying, the state-space system (5.5) is considered a linear time-varying model (LTV). According to Rego & Raffo (2019), the reference control vector (5.4) is used to calculate the state matrix $\boldsymbol{A}(t)$ of the system (5.5). As $\boldsymbol{u}_r$ is an affine function of the acceleration of the desired trajectory, it implies that the state matrix is also an affine function of the acceleration of the desired trajectory, and varies with time. Assuming that the acceleration of the desired trajectory is known, has no negligible values and defining the vector $\boldsymbol{\sigma} = \ddot{\boldsymbol{\xi}}_r$, we have that $\boldsymbol{u}_r$ is an affine function of $\boldsymbol{\sigma}$. Then, we can rewrite the time-varying matrix $\boldsymbol{A}(t)$ as a parameter-varying matrix $\boldsymbol{A}(\boldsymbol{\sigma})$, which is also affine in the parameters $\boldsymbol{\sigma}$.

With the objective of performing path tracking and rejecting constant disturbances, the state space vector is extended with integral action of the state errors of x, y, z, and the yaw angle. Thus, the new augmented system is given by

$$\dot{\tilde{\boldsymbol{x}}}_a(t) = \boldsymbol{A}_a(\boldsymbol{\sigma})\tilde{\boldsymbol{x}}_a(t) + \boldsymbol{B}_a\tilde{\boldsymbol{u}}(t) + \boldsymbol{D}_a\boldsymbol{d}(t), \tag{5.6}$$

with $\boldsymbol{A}_a(\boldsymbol{\sigma}) = \begin{bmatrix} \boldsymbol{A}(\boldsymbol{\sigma}) & 0 \\ \boldsymbol{C} & 0 \end{bmatrix}$, $\boldsymbol{B}_a = \begin{bmatrix} \boldsymbol{B} \\ 0 \end{bmatrix}$, $\boldsymbol{D}_a = \begin{bmatrix} \boldsymbol{D} \\ 0 \end{bmatrix}$, and the augmented error state vector is written as

$$\tilde{\boldsymbol{x}}_a(t) = \begin{bmatrix} \boldsymbol{x}_s(t) - \boldsymbol{x}_{sr}(t) \\ \int (x(t) - x_r(t))dt \\ \int (y(t) - y_r(t))dt \\ \int (z(t) - z_r(t))dt \\ \int (\psi(t) - \psi_r(t))dt \end{bmatrix}. \tag{5.7}$$

Thus, from the linear parameter-varying (LPV) error model (5.6), the following discrete-

time system can be obtained to be used in the explicit predictive control formulation:

$$\tilde{\boldsymbol{x}}_{\boldsymbol{a}}(k+1) = \boldsymbol{A}_{\boldsymbol{z}}(\boldsymbol{\sigma})\tilde{\boldsymbol{x}}_{\boldsymbol{a}}(k) + \boldsymbol{B}_{\boldsymbol{z}}\tilde{\boldsymbol{u}}(k) + \boldsymbol{D}_{\boldsymbol{z}}\boldsymbol{d}(k), \tag{5.8}$$

where $\boldsymbol{A}_{\boldsymbol{z}}(\boldsymbol{\sigma}) = \boldsymbol{I} - \boldsymbol{A}_{\boldsymbol{a}}(\boldsymbol{\sigma})T$, $\boldsymbol{B}_{\boldsymbol{z}} = \boldsymbol{B}_{\boldsymbol{a}}T$, and $\boldsymbol{D}_{\boldsymbol{z}} = \boldsymbol{D}_{\boldsymbol{a}}T$, in which T is the sampling time chosen small enough to capture the dynamics of the aircraft[3]. It is noteworthy that, in this work, we assume that all states of the discrete-time system are available to the controller by measurement and/or state estimation.

Remark 5.1. *The discrete-time LPV model (5.8) of the tiltrotor UAV with suspended load has the same structure as the system presented in (3.1). Thus, we can apply the entire methodology presented in Chapters 3 and 4.*

Additionally, considering that the system is represented by a linear-error parameter varying model (5.8), the tube-based control law (3.4) is extended as follows

$$\boldsymbol{u}(k) = \boldsymbol{u}_{\boldsymbol{r}}(k) + \bar{\boldsymbol{u}}(k) + \boldsymbol{K}(\tilde{\boldsymbol{x}}_{\boldsymbol{a}}(k) - \bar{\boldsymbol{x}}_{\boldsymbol{a}}(k)), \tag{5.9}$$

where the reference control vector $\boldsymbol{u}_{\boldsymbol{r}}(k)$ is taken from (5.4), $\bar{\boldsymbol{u}}(k)$ is the optimal nominal control sequence obtained from (3.55). Also, in the mismatch term $\boldsymbol{K}(\boldsymbol{x}_{\boldsymbol{a}}(k) - \bar{\boldsymbol{x}}_{\boldsymbol{a}}(k))$, $\boldsymbol{K}$ is the feedback gain computed by (3.10) and $\tilde{\boldsymbol{x}}_{\boldsymbol{a}}(k)$ is the error measured state from (5.7).

An important aspect of our methodology is the use of a normalized nominal linear model in the optimization process. By normalizing the nominal system, we address the numerical challenges arising from the magnitude disparity of the variables. When considering our case study, the translational position states (x, y, and z) have magnitudes in meters, whereas the angular states are in radians. The significant range difference can lead to numerical instability and imprecision in the optimization algorithm. However, by appropriately scaling the variables, we ensure that the algorithm operates with variables in a similar range, enhancing the reliability and numerical efficiency of the optimization algorithm.

The normalization process is applied to the nominal model of the tiltrotor UAV with suspended load, which is derived from (5.8) setting $\boldsymbol{d} = 0$ and $\boldsymbol{\sigma} = 0$. The process begins by defining the normalized nominal state vector as $\bar{\boldsymbol{x}}_{\boldsymbol{n}} = \boldsymbol{N}_{\boldsymbol{x}}\bar{\boldsymbol{x}}_{\boldsymbol{a}}$ and the normalized nominal control vector as $\bar{\boldsymbol{u}}_{\boldsymbol{n}} = \boldsymbol{N}_{\boldsymbol{u}}\bar{\boldsymbol{u}}$, where $\boldsymbol{N}_{\boldsymbol{x}}$ and $\boldsymbol{N}_{\boldsymbol{u}}$ are diagonal matrices with each diagonal elements given by

$$\begin{aligned} \boldsymbol{N}_{\boldsymbol{x}}[i] &= \frac{\bar{\boldsymbol{x}}_{\boldsymbol{n}}^{+}[i]}{\bar{\boldsymbol{x}}_{\boldsymbol{a}}^{+}[i]}, \forall i = 1, ..., n, \\ \boldsymbol{N}_{\boldsymbol{u}}[i] &= \frac{\bar{\boldsymbol{u}}_{\boldsymbol{n}}^{+}[i]}{\bar{\boldsymbol{u}}^{+}[i]}, \forall i = 1, ..., m, \end{aligned} \tag{5.10}$$

where the symbol $\bar{\boldsymbol{x}}^{+}$ denote the maximum value of the variable.

[3]The discretization is performed using Euler method.

Therefore, the normalization of the nominal model (3.3) of the tiltrotor UAV is defined as

$$\bar{\boldsymbol{x}}_{\boldsymbol{n}}(k+1) = \boldsymbol{A}_{\boldsymbol{n}}\bar{\boldsymbol{x}}_{\boldsymbol{n}}(k) + \boldsymbol{B}_{\boldsymbol{n}}\bar{\boldsymbol{u}}_{\boldsymbol{n}}(k), \tag{5.11}$$

where $\boldsymbol{A}_{\boldsymbol{n}} = \boldsymbol{N}_{\boldsymbol{x}}\bar{\boldsymbol{A}}_{\boldsymbol{z}}\boldsymbol{N}_{\boldsymbol{x}}^{-1}$ and $\boldsymbol{B}_{\boldsymbol{n}} = \boldsymbol{N}_{\boldsymbol{x}}\boldsymbol{B}_{\boldsymbol{z}}\boldsymbol{N}_{\boldsymbol{u}}^{-1}$.

Moreover, if the parameters of the optimization problem (3.51) are initially tuned for the nominal model, we can normalize the weighting matrices and terminal cost as

$$\boldsymbol{\Sigma}_{\boldsymbol{\rho n}} = \boldsymbol{N}_{\boldsymbol{x}}'\boldsymbol{\Sigma}_{\boldsymbol{\rho}}\boldsymbol{N}_{\boldsymbol{x}},\ \boldsymbol{\Sigma}_{\boldsymbol{\lambda n}} = \boldsymbol{N}_{\boldsymbol{u}}'\boldsymbol{\Sigma}_{\boldsymbol{\lambda}}\boldsymbol{N}_{\boldsymbol{u}},\ \text{and}$$
$$\boldsymbol{L}_{\boldsymbol{n}} = \boldsymbol{N}_{\boldsymbol{x}}'\boldsymbol{L}\boldsymbol{N}_{\boldsymbol{x}}.$$

Similarly, the constraints sets can be normalized as

$$\bar{\mathcal{X}} : \boldsymbol{A}_{\boldsymbol{xn}}\boldsymbol{b}_{\boldsymbol{xn}} \leq \boldsymbol{b}_{\boldsymbol{xn}},\ \boldsymbol{b}_{\boldsymbol{xn}} = \boldsymbol{N}_{\boldsymbol{x}}\boldsymbol{b}_{\boldsymbol{x}},$$
$$\bar{\mathcal{U}} : \boldsymbol{A}_{\boldsymbol{un}}\boldsymbol{b}_{\boldsymbol{un}} \leq \boldsymbol{b}_{\boldsymbol{un}},\ \boldsymbol{b}_{\boldsymbol{un}} = \boldsymbol{N}_{\boldsymbol{u}}\boldsymbol{b}_{\boldsymbol{u}},$$

where $\boldsymbol{A}_{\boldsymbol{xn}}$ and $\boldsymbol{A}_{\boldsymbol{un}}$ are identity matrices.

5.3 Reachable Set Computation Analysis

In this section, the reachable set $\mathcal{Z}$ computation methodology, presented in Chapter 3, is applied to the tiltrotor UAV with suspended load. It uses the constrained control signals $[f_r, f_l] \in [0, 30]\ N$ and $[\tau_r, \tau_l] \in [-2, 2]\ N.m$, which correspond to the actuator specifications for the aircraft presented in Rego & Raffo (2019) and are used to build the control set $\mathcal{U}$. Also, the state constraints to build the state set $\mathcal{X}$ are settled as: $[\tilde{x}, \tilde{y}] \in [-1, 1]\ m$, $\tilde{z} \in [-1, 1]\ m$, $[\tilde{\phi}, \tilde{\theta}, \tilde{\psi}] \in [-0.5, 0.5]\ rad$, $[\tilde{\alpha}_r, \tilde{\alpha}_l] \in [-0.5, 0.5]\ rad$, $[\tilde{\gamma}_1, \tilde{\gamma}_2] \in [-0.5, 0.5]\ rad$. In addition, the intervals of translational (m/s) and rotational (rad/s) velocities are $[\pm 0.6, \pm 1, \pm 0.2, \pm 1, \pm 1, \pm 0.5, \pm 2, \pm 1, \pm 0.2, \pm 0.2]$, and the constraint of the integral action of the error states are $[\pm 1.5, \pm 1.5, \pm 1.5, \pm 0.5]$ that were obtained via simulation of the aircraft model described in Section 5.1.

The feedback gain $\boldsymbol{K}$ in (3.42) is calculated using the following weighting parameters for the control signals and states, respectively:

$$\begin{aligned}
\lambda &= [1.642\ 1.646\ 38.730\ 38.730], \\
\rho &= [1.265\ 1.265\ 1.265\ 0.450\ 0.450\ 0.712\ 0.637\ 0.637\ 1.273 \\
&\quad 1.273\ 0.250\ 0.250\ 0.250\ 0.955\ 0.955\ 1.273\ 1.061\ 1.061 \\
&\quad 0.011\ 0.011\ 1.789\ 1.789\ 1.789\ 0.158].
\end{aligned}$$

As presented in (3.34), the positive invariant set is calculated recursively, until achieving the error bound λ set as 0.05.

With the objective to exemplify the difference between the reachable sets generated by the proposed approach and the one presented in Limon et al. (2008), some projections of these sets are shown below. It is worth mentioning that the projection is used due to the difficulty of plotting a highly order set. Also, note that projections of a zonotope are not necessarily a zonotope. Figure 5.2 illustrates the reachable set projection of the x, y and z axis. It is worth highlighting that the proposed method reaches a smaller invariant set size when compared to the set generated by the method presented in Limon et al. (2008). This behavior occurs because the LPV formulation is less conservative due to the use of the time varying parts of the system to compute the reachable set. Therefore, the less conservatism of the algorithm results in a smaller reachable set.

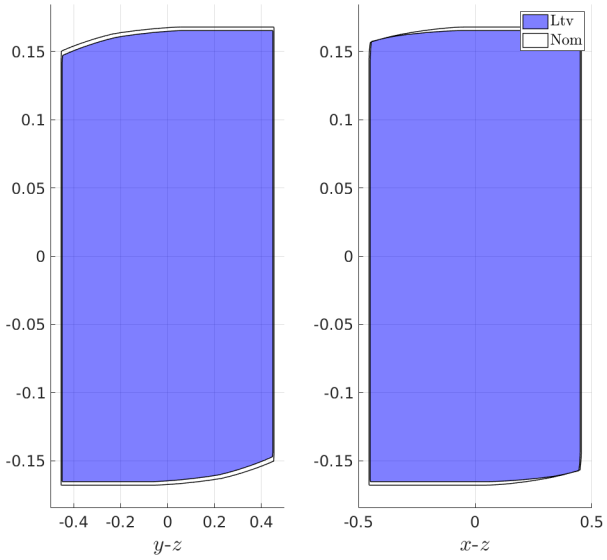

Figure 5.2: Graphic showing the projection of the Reachable Set for the x, y, and z states

The same behavior is shown in Figures 5.3 for the case of the projection of the tilting angles' states. Besides, the size of the set does not exceed the limits settled for these states, and the limits of the reachable set are 60% lower. Also, it is noteworthy that the presence of the disturbance set in the formulation of the uncertainty set does not allows a further reduction of the set size.

We used the the 1-norm of the radius and the volume of the set as measures to compare the size of the sets generated by the presented methods. Both cases use the approximation of the set as a box, where the 1-norm of the radius is given by the summation of the diameters of the set intervals, and the volume is computed by multiplying all intervals diameters.

Thus, from Table 5.1, note that the size of the reachable set computed by the LPV formulation is smaller than the nominal approach, in which its volume is around nine percent smaller than the nominal formulation presented in Limon et al. (2008). In order to demonstrate the importance of using the time varying part of the system in the formulation, it is presented the set projection of the nominal state and control sets, where the size

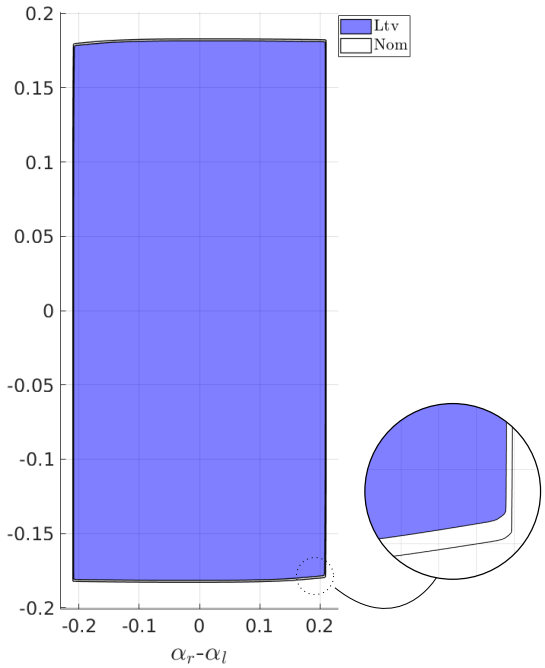

Figure 5.3: Graphic showing the projection of the Reachable Set for the α_r and α_l states

Table 5.1: Indexes values comparing the sizes of the reachable sets generated by each formulation.

	LPV	Nominal	Performance
1-Norm Radius	11.386	11.433	0.409%
Volume	1.48×10^{-9}	1.63×10^{-9}	9.135%

increase of the domain of attraction for the optimization problem is observed.

In Figures 5.4, the nominal control set projection for x, y, and z nominal states are presented. This result shows that both sets are almost similar. However, note that, in the zoomed images, the limits of the $x - y$ and $x - z$ states are enlarged using the LPV formulation, which is mainly caused by the size reduction of the reachable set. This shows that the method using the LPV model gives an smaller RPI set than the nominal system. Consequently, the nominal state set obtained from equation (3.5) results bigger.

Also, it is worth noting the enlargement of the domain of attraction of the nominal state set in Figures 5.5 through the projections of $\phi - \theta$ states. However, in the $\phi - \psi$ projection, the limits of the set obtained by the LPV computation method are bigger than the nominal approach. Since the projection of the states does not necessarily show the real size of the set, we used the 1-norm of the radius and the volume set as comparative indexes between both formulations.

As the reachable set sizes of both formulations are similar and the feedback gain of the LPV formulation takes into account the nominal part of the system, the nominal control sets generated by the formulations are similar, as it is shown in Figures 5.6.

Considering the 1-Norm radius and the volume for the nominal state and control set,

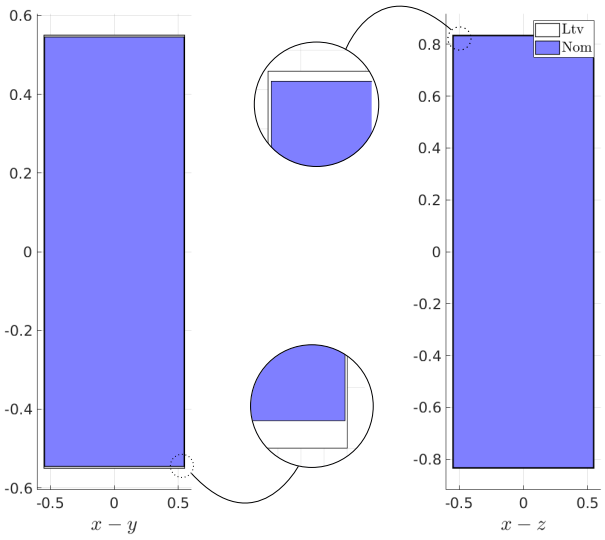

Figure 5.4: Graphic showing the projection of the Nominal State Set for the x, y and z states.

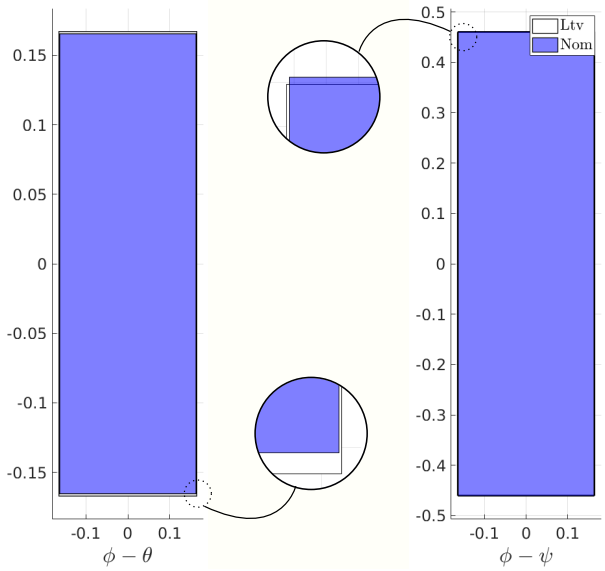

Figure 5.5: Graphic showing the projection of the Nominal State Set for the ϕ, θ and ψ states.

Table 5.2 supports the above assertions about the size of the sets are bigger for the LPV formulation, which means that using this sets in the MPC optimization problem, the domain attraction is increased by almost five percent.

5.4 Final Remarks

This chapter introduced the system that will be used in the experiments to verify the performance of the proposed control algorithms. The nonlinear model of the tiltrotor UAV with a suspended load was linearized through a desired trajectory, resulting in an LPV model characterized by fast and high-order dynamics. It is important to note that

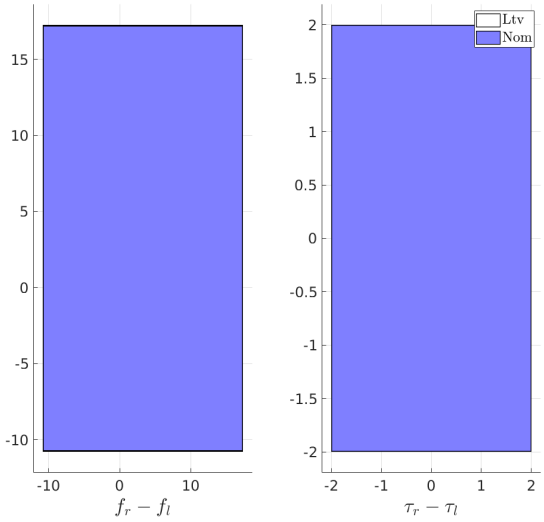

Figure 5.6: Graphic showing the projection of the Nominal Control Set.

Table 5.2: Indexes values comparing the size of the nominal sets generated by each formulation.

	1-Norm Radius	Volume
Nominal State	174.093	7.269×10^{17}
LPV State	174.046	6.913×10^{17}
Peformance	0.0268%	4.891%
Nominal Control	31.9485	1.243×10^{4}
LPV Control	32.0205	1.248×10^{4}
Peformance	0.0102%	0.0306%

the reachable set computation algorithm presented in Section 5.3 could be successfully applied to the high-order LPV model of the tiltrotor aircraft. The proposed methodology considering the parameter-varying part of the model gives a slightly smaller robust positive invariant set compared with the LTI approach, improving the tube-based MPC formulation since it needs a reachable set as close as possible to the minimum robust positive invariant set. Also, note that none of the nominal states and control sets exceed the imposed constraints due to the use of the LMIs designed in Theorem 3.1. Besides, the proposed approach increases the domain of attraction of the MPC optimization problem, a clear advantage to be used with the optimization of the MPC formulations presented in Chapter 4.

6 Experimental Results

This chapter presents the experimental environment and the results of controlling the tiltrotor UAV with suspended load using the proposed tube-based MPC formulation. The functionality of the controllers is verified through numerical experiments using realtime implementation in a Hardware-In-the-Loop (HIL) framework.

6.1 Hardware-In-The-Loop Simulation Environment

The experimental environment setup is presented in Figure 6.1, which consists of an embedded computer Nvidia Jetson and the ProVANT simulator, which interchange the state and control information using serial communication. This system emulates the final application, testing the hardware performance used in the real aircraft.

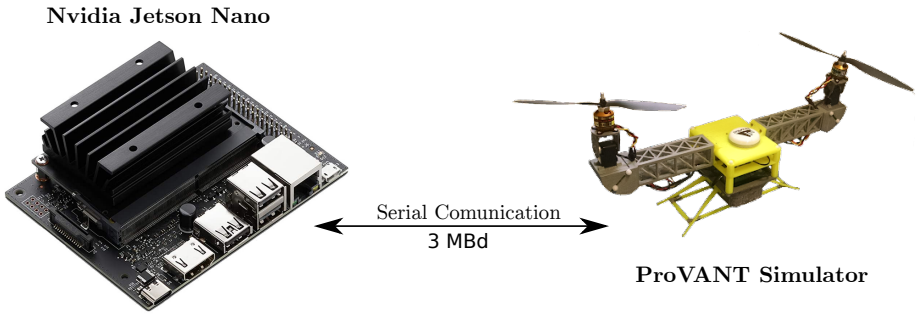

Figure 6.1: Hardware composition scheme used to perform HIL simulations.

6.1.1 ProVANT Simulator

In the HIL framework, the ProVANT Simulator is responsible for emulating the dynamic behavior of the tiltrotor UAV with a suspended load using a physics engine to accurately compute the dynamics in a non-deterministic manner. Figure 6.2 depicts the three main modules employed by the ProVANT simulator during the simulation process. Module 1 illustrates the Gazebo physics engine, which consists of the Physics Engine and Update Model State plugins. This module is responsible for computing the system's dynamic behavior in a non-deterministic manner. It uses the physics engine provided by Gazebo, in conjunction with a mechanical model based on a Computer-Aided Design (CAD) developed using a 3D modeling program. Module 2, known as Gazebo Plugins, is responsible for updating the state vector and control inputs in the virtual model. In order to measure the states, it utilizes a plugin that acts as a virtual sensor inside the simulation, reading the values generated by the physics engine and facilitating the generation of the state vector. Additionally, another plugin is employed to update the control inputs in the virtual model, enabling control over the virtual model within the physics engine. Module 3, known as Communication Plugins, manages the transmission of state and control vectors between the Embedded System and the Gazebo Plugins module via the serial communication interface. It is worth noting that the serial port is configured as *ASYNC_LOW_LATENCY* to minimize latency.

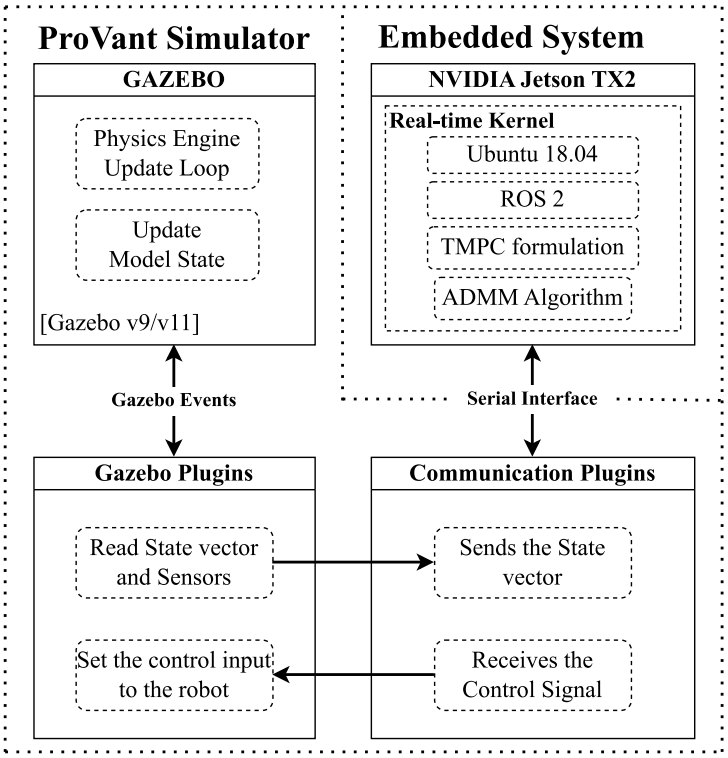

Figure 6.2: Hardware in the Loop (HIL) structure used as an experimental testbench, illustrating the main modules of the ProVant Simulator software and the realtime framework characteristics of the embedded computer.

6.1.2 Embedded System

The embedded system is the second component of the HIL testbench, as illustrated in Figure 6.2. It is used the NVIDIA Jetson TX2 single-board computer to implement the proposed algorithms. This computer features a dual-core 64-bit NVIDIA Denver 2 CPU at 2.2 GHz, a quad-core ARM A57 processor running at 2 GHz, a 256-core NVIDIA Pascal GPU, and 8GB DDR4 RAM. The operating system employed is Ubuntu 18.04, with a customized kernel using the PREEMPT_RT patch to transform it into a fully preemptible kernel, allowing the system to better respond to urgent tasks and improve overall multitasking.

The embedded system is responsible for computing the control signal based on the state vector obtained from the ProVANT Simulator. One of the main advantages of this computer is the presence of a GPU, which allows for efficient execution of dense parallel operations. Taking advantage of this characteristic, it has developed a highly parallel and scalable optimization methodologies to exploit the parallel processing capabilities of the embedded system, enabling efficient and rapid computation for high-order and fast dynamic systems.

The structure of the software implemented in the embedded system is presented in Figure 6.3. Despite having developed a control algorithm that satisfies realtime requirements, it is implemented a backup control law to address situations where the computation time of the TMPC formulation exceeds the sampling time. The algorithm presented in Figure 6.3 executes a fast computation control law in parallel when the TMPC formulation performs the optimization. By having an auxiliary control law, it is ensured that the control system remains responsive and stable, even when the TMPC formulation cannot meet the sampling time constraint.

Figure 6.3 illustrates the execution sequence of the algorithm, outlining the main steps involved. The process begins with the memory allocation of the system model variables and TMPC parameters in CPU and GPU memory. This memory allocation reduces the data sharing between the two devices in the online phase, minimizing data transfer overhead.

Subsequently, the embedded computer communicates with the ProVant Simulator to request the system's state vector. Upon acquiring the state vector, the computation of the TMPC begins. When the optimization is required, the state vector is transmitted to the thread executing the optimization algorithm in parallel. For the case of the ADMM optimization, the algorithm is executed in the GPU, while the explicit and the constrained zonotope formulations are executed in the CPU.

Concurrently, the auxiliary control law is computed as a backup control strategy. The system remains idle, awaiting to achieve the desired sampling time. Once the time threshold is reached, the algorithm evaluates the response of the ADMM optimization step. If the optimization process does not respond within the prescribed time constraint, the

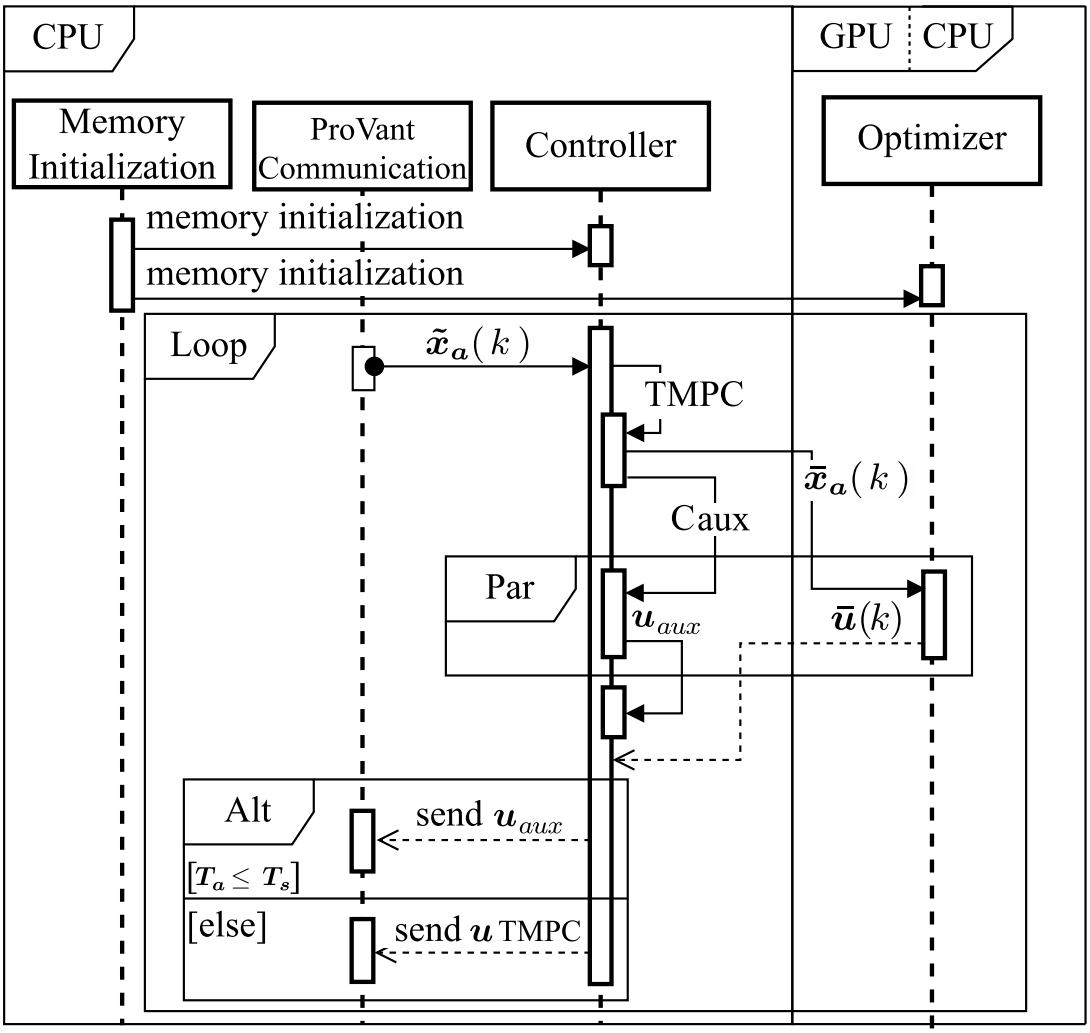

Figure 6.3: General structure of the software implemented in the embedded computer, showcasing the main modules executed in the CPU and GPU devices. $TMPC$ is the proposed algorithm, C_{aux} is the auxiliary control law, and the interfaces 'Par' and 'Alt' represent a parallel execution and a decision blocks, respectively.

control values computed by the auxiliary control law are sent to the ProVant Simulator. Conversely, if the optimization algorithm provides a timely response, the control value computed by the TMPC formulation is transmitted.

6.2 Experimental Parameters

In this thesis, all experimental scenarios use the same task in which it is required to solve the trajectory tracking problem of the predefined trajectory depicted by the following equations:

$$x_r(t) = 5\cos(\frac{\pi}{40}t)\ [m],\ y_r(t) = 5\sin(\frac{\pi}{40}t)\ [m],$$
$$z_r(t) = 4 - 3\cos(\frac{\pi}{40}t)\ [m].$$

Also, it is assumed that the take-off position of the aircraft is $\boldsymbol{q}(0) = \begin{bmatrix} 5 & 0 & 3 & 0_{1\times 7} \end{bmatrix}$, coincident with the desired trajectory's first point.

With the objective of testing the disturbance rejection capability, constant external forces are applied to the the first three degrees of freedom (x, y, and z) of the load motions at different instants, as defined in Table 6.1.

Table 6.1: Disturbance Parameters

Disturbance	Value	Time
p_x	$1\ N$	$20\ s$
p_y	$1\ N$	$40\ s$
p_z	$1\ N$	$60\ s$

The physical parameters of the UAV are the same as used in Rego & Raffo (2019), which are presented in Table 6.2.

Table 6.2: Mass and inertia matrix values of the the tiltrotor UAV bodies used to compute the linear error model and controller parameters.

Parameter	Value
$(m_{\mathcal{L}}, m_1, m_2, m_3)$	$(0.5, 1.7068, 0.08978, 0.08978)$ Kg
$\boldsymbol{I}_{\mathbb{L}}$	$8.333 \cdot 10^{-6} \cdot \mathbb{I}_{3\times 3}$ Kg·m²
$\boldsymbol{I}_1$	$\begin{bmatrix} 4047.04 & 0.860582 & 9.65766 \\ 0.860582 & 881.618 & -0.873079 \\ 9.65766 & -0.873079 & 4173.18 \end{bmatrix} \cdot 10^{-6}$ Kg · m²
$\boldsymbol{I}_1$	$\begin{bmatrix} 335.737 & -1.33011 \cdot 10^{-15} & -2.85046 \cdot 10^{-15} \\ -1.33011 \cdot 10^{-15} & 335.737 & -6.81597 \cdot 10^{-16} \\ -2.85046 \cdot 10^{-15} & -6.81597 \cdot 10^{-16} & 641.59 \end{bmatrix} \cdot 10^{-6}$ Kg · m²
$\boldsymbol{I}_1$	$\begin{bmatrix} 335.737 & -5.26914 \cdot 10^{-16} & 2.9351 \cdot 10^{-15} \\ -5.26914 \cdot 10^{-16} & 335.737 & -1.24077 \cdot 10^{-16} \\ 2.9351 \cdot 10^{-15} & -1.24077 \cdot 10^{-16} & 641.59 \end{bmatrix} \cdot 10^{-6}$ Kg · m²

The equilibrium point of the aircraft is defined as $\bar{\phi}(t) = 0$ rad, $\bar{\theta}(t) = 0$ rad, $\bar{\gamma}_1 = 1.3170 \times 10^{-4}$ rad, $\bar{\gamma}_2 = 0.01396$ rad, $\bar{\alpha}_r = 0.0140$ rad, and $\bar{\alpha}_l = 0.0138$ rad, calculated at $\psi_r(t) = 0$. Furthermore, it is considered a sampling time of 12 ms, which is the parameter T used in the discretization of the LPV error model.

In order to evaluate the performance of the proposed formulations, the trajectories performed by the aircraft are evaluated using the Root Mean Square Error (RMSE) index, and the control signals generated by the actuators are compared using the total control

variation index. The comparative analysis between the indices was conducted using the following equation:

$$\%I = 100 - \frac{100 \cdot I_p}{I_b}. \tag{6.1}$$

where I_p and I_b are the proposed and base formulation indices, respectively.

Also, it is measured the time expended by the embedded computer to calculate the control signals of the formulations. The methodology to measure this time is performed as follows: first, it is executed ten simulations, from which it is measured the time spend by the embedded computer at each sampling time; then, this information data is used to compute the maximum, minimum and average values of each simulations for both formulations. Finally, in order to determine the time performance of the proposed formulation, an average of the time values calculated before are used.

6.3 Simulation Results for Explicit Tube-based MPC

This section analyzes the explicit tube-based model predictive control formulation derived from mixing the TMPC algorithm designed in Chapter 3 with the offline optimization methodology presented in Section 4.1. A comparative analysis is conducted from two experimental cases, where the first case considers the LTV model (eTMPC LTV) presented in Theorem (3.3), while the second case considers the nominal one (eTMPC NOM), presented in Limon et al. (2008), to verify the performance of the proposed formulation.

The solution of the multi-parametric optimization problem (4.7) generates $s = 420$ and $s = 437$ critical regions for the nominal and LTV approaches, respectively. The final size of the multi-dimensional matrices $\boldsymbol{CR}$ and $\boldsymbol{K_R}$ derived form the MPT toolbox are presented in Table 6.3.

Table 6.3: Size Matrices obtained from the multi-parametric solution

Matrices	$eTMPCLTV$	$eTMPCNOM$
$\boldsymbol{CR}$	$64 \times 25 \times 437$	$64 \times 25 \times 420$
$\boldsymbol{K_R}$	$4 \times 25 \times 437$	$4 \times 25 \times 420$

The error of the tracked reference by the suspended load during the simulation is illustrated in Figure 6.4. Both tube-based explicit MPC formulations presented good results in the path tracking scenario. Notice that, as the controllers consider the integral actions in the translational position error states, the controllers are able to perform path tracking with almost null error. Also, the rejection of the disturbances applied to the system can be observed at the instants 20, 40 and 60 seconds. Besides, it is worth highlighting that the behavior of both formulations is very similar. However, the eTMPC LTV formulation has smaller overshoots, for example, the transient response at the 20 seconds in the $\tilde{x}$ state, which means that the eTMPC LTV formulation is smoother than

the nominal one (eTMPC NOM).

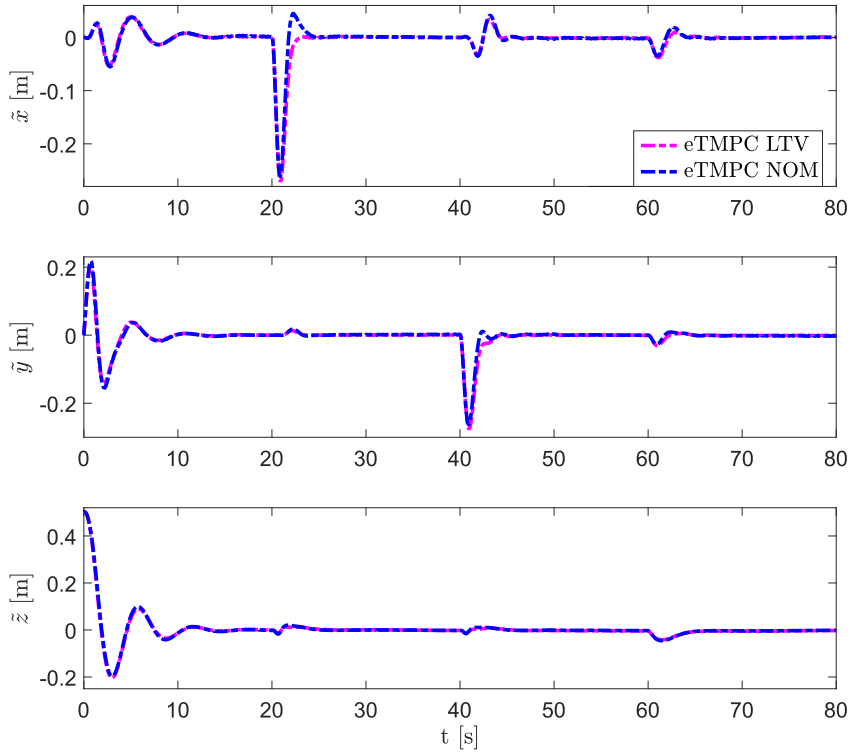

Figure 6.4: Trajectory made by the tiltrotor with the suspended load.

Figure 6.5 shows the time response of Euler angles of the load, where both formulations present a similar behavior. In order to track the desired reference, it is possible to see that the angles ϕ and θ change the equilibrium points when disturbances affect the system. For example, the equilibrium point of θ changes when a force is applied to the x-motion at instant 20s. In the same way, the equilibrium point of ϕ changes when a force is applied to the y-motion at 40 and 60 seconds. Conversely, as the angle ψ has an integral action, it has the ability to reject step disturbances and track a constant reference trajectory, which is zero in this case.

Similar performance is observed on the servomotors' angle responses presented in Figure 6.6, where the eTMPC LTV formulation presents a more aggressive behavior than the one obtained with the eTMPC NOM. As it can be observed, the time response of eMPC LTV approach presents higher overshoot than the nominal formulation through the simulation. It is worth mentioning that both controllers stabilize the system in different equilibrium points depending on the disturbance applied to the system.

The aggressiveness of the eTMPC LTV is more evident in Figure 6.7, which presents the load's angles performance. The main difference between both formulations is the oscillatory behavior of the tube-based eTMPC LTV formulation at the 60 seconds time simulation. It is worth highlighting that the nominal formulation is less oscillatory and the stabilization is faster than the LTV approach, for both states. This behavior is given because both controllers were tuned with the same parameters in order to avoid stability

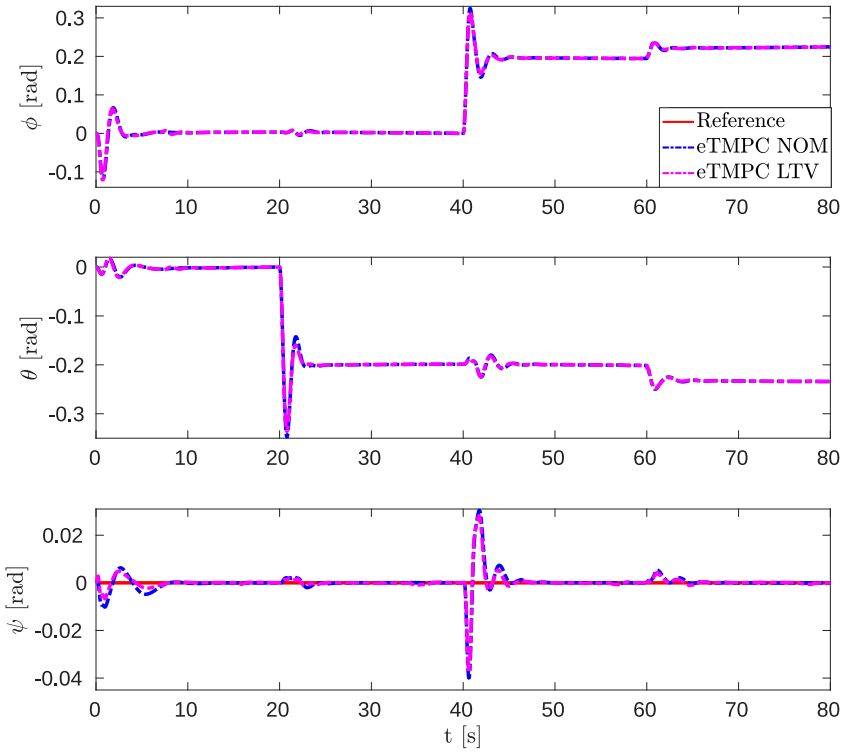

Figure 6.5: Trajectory of the Euler angles performed by the suspended load during the simulations.

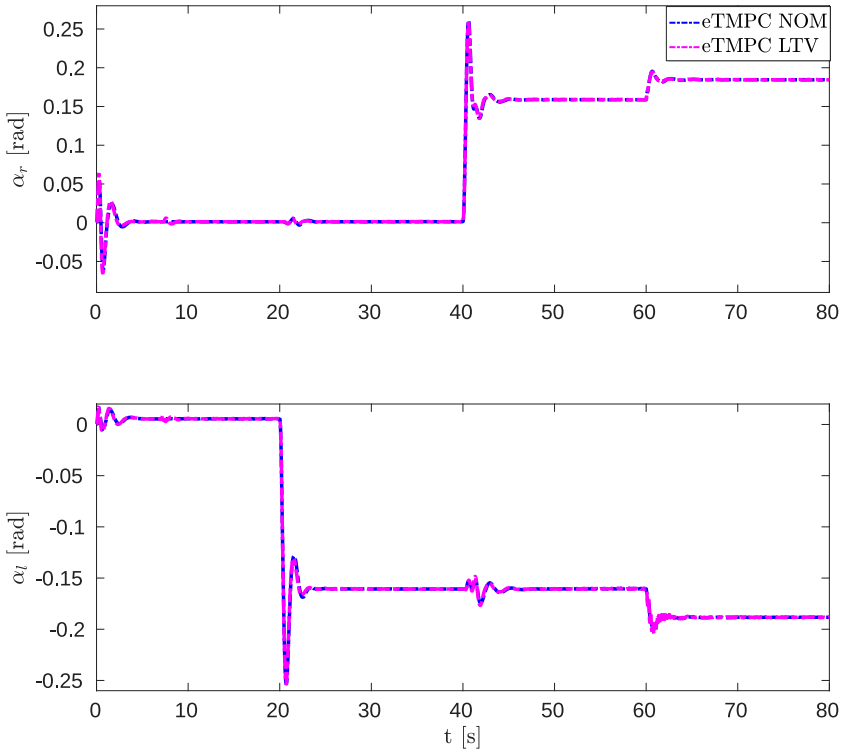

Figure 6.6: Time evolution of the Servo-motor angles.

issues of the nominal formulation. However, improvement on the tuning process can reduce the aggressive behavior of the eTMPC LTV approach.

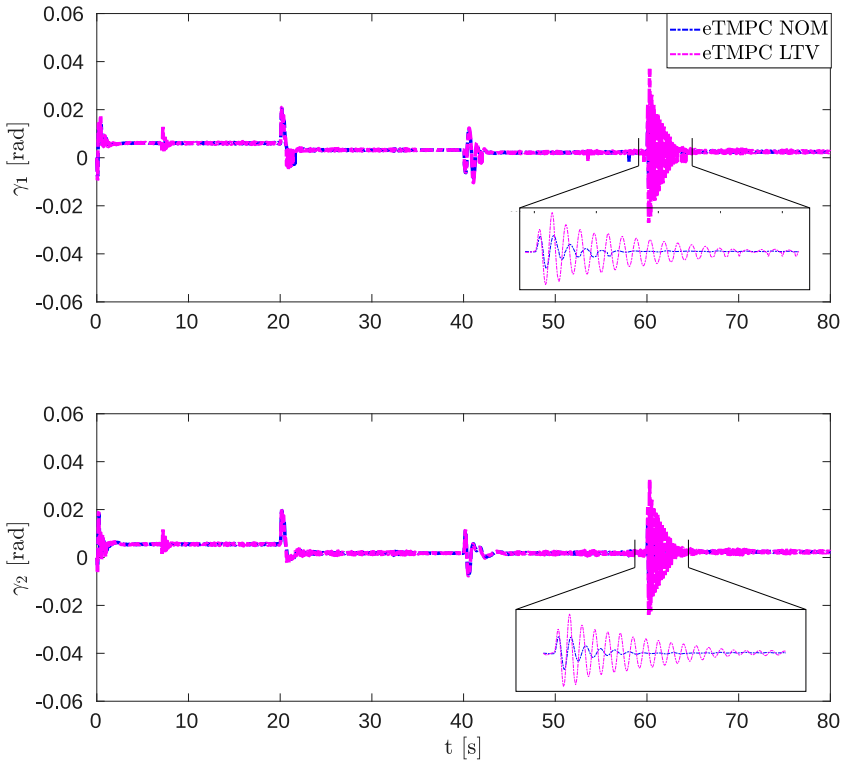

Figure 6.7: Time evolution of the Load angles showing how eTMPC reduced the oscillation behavior of the states.

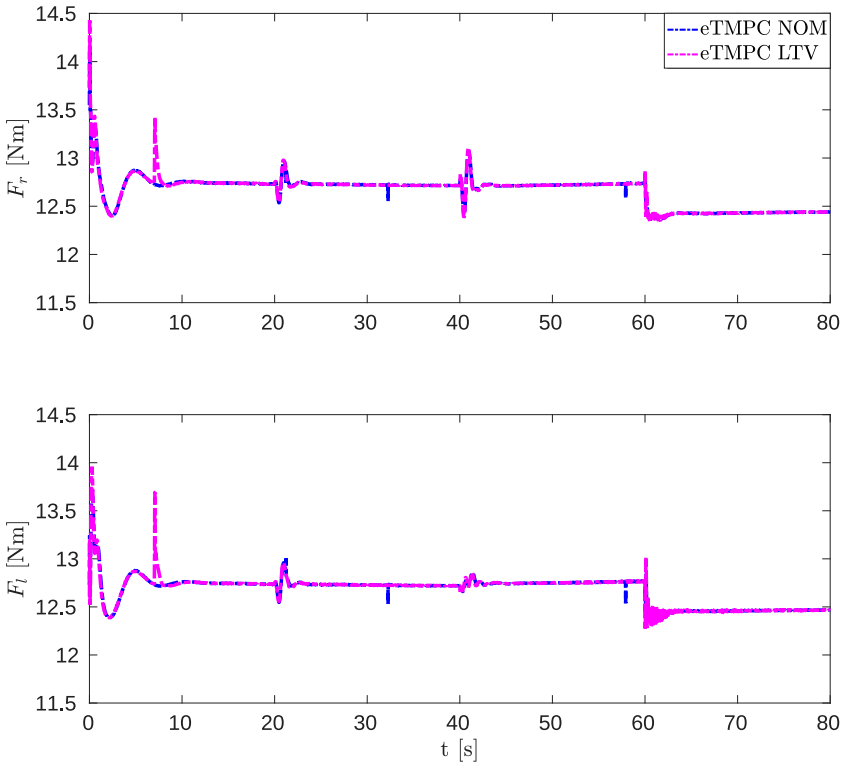

Figure 6.8: Thrust forces applied to the Tiltrotor aircraft.

Figs. 6.8 and 6.9 show the thrust forces and torques applied to the aircraft. From Figure 6.9, observe how the controller manages the oscillations presented in the load angles. Due to the aggressiveness of the eTMPC LTV, oscillatory torques are required to stabilize the load.

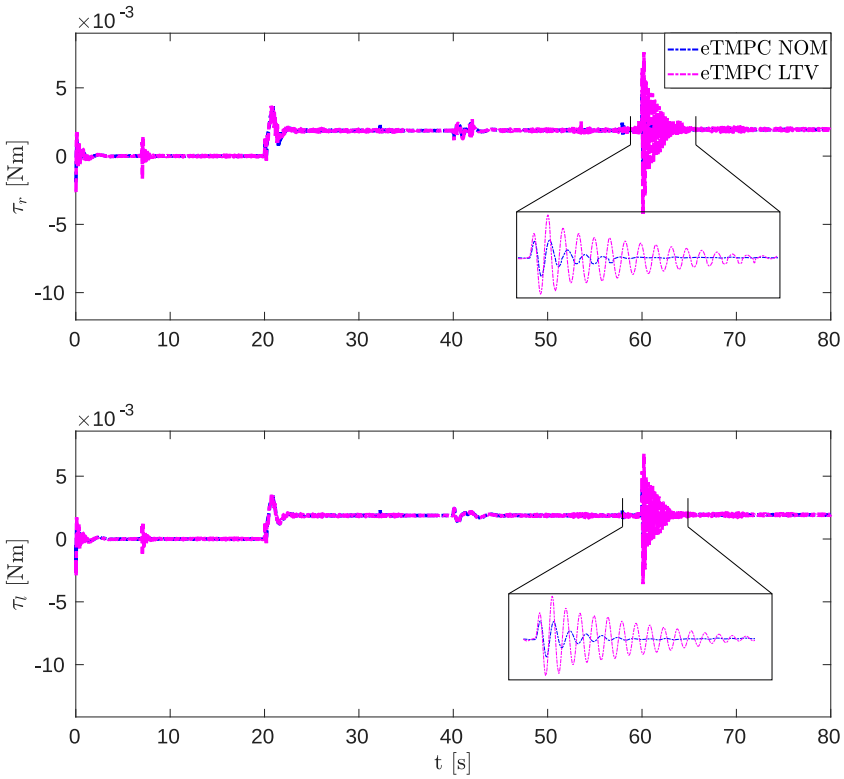

Figure 6.9: Torques applied to the tilting mechanism of the Tiltrotor aircraft.

As aforementioned, the MSE index is used with the objective to measure the performance of both formulations. Table 6.4 presents the values obtained from simulation results and the relative error between them. The $*$ mark represents that the tube-based eTMPC nominal formulation achieved better performance than the LTV one.

In general, the performance of both formulations is very similar. However, it is possible to contrast the information with the graphics presented before, in which the state ψ shows smaller amplitude oscillation with the eTMPC LTV approach, reflected by the MSE index which is 27.34% better than the one obtained with the eTMPC NOM. Also, some states showed a great improvement using the proposed formulation, e.g. x, γ_1 and ψ, γ_2 states.

In addition, the total control variation index was calculated to evaluate the control effort of both formulations. Table 6.5 shows that the tube-based eTMPC nominal approach expends less control effort than the LTV one. This result corroborates the aggressiveness of the proposed tube-based eTMPC LTV approach, which on the other hand achieved better performance during the trajectory tracking.

Additionally, the computational time spent to compute the control signals of both techniques was also evaluated. As mentioned before, ten simulations were performed with

Table 6.4: Mean Square Error indices of the trajectory performed by the aircraft.

States	eTMPC LTV	eTMPC Nominal	Relative Error(%)
x	0.000846	0.000749	-13.214
y	14.5354	14.5406	0.0364 (*)
z	40.3971	40.3915	-0.0144
ϕ	0.02224	0.02226	0.0605 (*)
θ	0.03371	0.03375	0.1120 (*)
ψ	0.000016	0.000022	27.34
γ_1	0.000022	0.000017	-23.1977
γ_2	0.000131	0.000126	-3.2095
α_r	0.01270	0.01270	0.0000
α_l	0.02551	0.02552	-0.0298 (*)

Table 6.5: Values of the Total Control Variation index generated by the actuators of the tiltrotor UAV with suspended load.

Control	eTMPC LTV	eTMPC Nominal	Relative Error
f_l	6178.28	6020.42	-0.1994
f_r	6019.54	6178.62	0.1985
τ_l	9.70	9.58	-1.3400
τ_l	9.57	9.47	-1.1600

each proposed technique. Then, the maximum, minimum, and average values obtained from each simulation are depicted in Figure 6.10. It is worth mentioning that any of the worst cases do not exceed the sampling time of the system, which is twelve milliseconds, because the algorithm send the second control law when the sampling time is exceeded. This ensures the ability of the proposed scheme to work in a realtime framework.

The violation of the execution time is not related with the formulations, but to the implementation in the embedded computer. The main reason that support this statement is because the moments when the sample time is exceeded are aleatory, as shown in Table 6.6.

Table 6.6: Instant of times in which the sampling period of the system was exceeded.

Experiments	Ex1	Ex2	Ex5	Ex6	Ex9	Ex10
LTV $[s]$	-	-	-	-	-	17.21
Nominal $[s]$	53.93	37.66	63.61	5.45	12.20	-

In cases where the sampling time is not violated, the biggest computation time is lower than 12 ms. This is due to the parallel algorithm proposed in Section 6.1.2, which can compute several matrices at the same time. In addition, the proposed algorithm has the characteristic to be scalable to be used with a GPU in order to increase the computational capability of the system.

Table 6.7 summarizes Figure 6.10, in which it is presented the average of the maximum, minimum and mean time values of both formulations. It is important to note that the

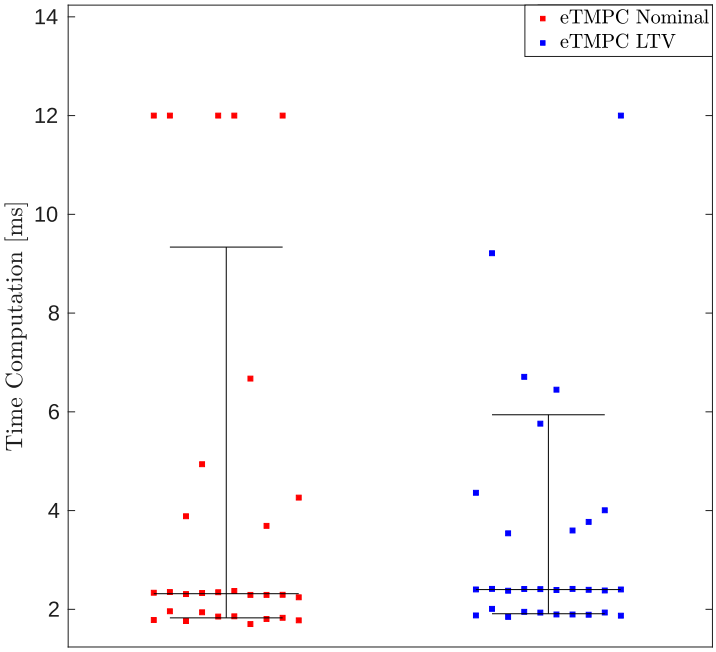

Figure 6.10: Resume of the maximum, mean and minimum time values spent by the embedded system computing the control law in the experiments.

maximum time value of the eTMPC LTV formulation was $5.9ms$, which is 50.8% of the sampling time for the worst case, given a good performance considering the number of critical regions.

Table 6.7: Computational Time Spent on the eTMPC algorithm.

time (ms)	$eTMPC\ LTV$	$eTMPC\ NOM$
max	5.9	9.3
$average$	2.3	2.4
min	1.9	1.8

It is important to note that the proposed formulation, combining tube-based MPC with zonotopes and offline multi-parametric optimization, can be implemented in a realtime framework using an embedded computer. This formulation successfully executed the path-tracking problem of a tiltrotor UAV with suspended load characterized by fast and high-order dynamics. While the performance differences between the LTV and Nominal formulations are not very noticeable, the LTV formulation is less conservative, as demonstrated by the number of critical regions generated by the multi-parametric optimization. The larger number of critical regions in the LTV formulation indicates a smaller size of the reachable set compared to the nominal one, which allows for larger sizes of the nominal state and control sets. However, the explicit solution of the nominal optimization problem has limitations concerning the number of restrictions that can be added to the problem since the number of critical regions depends directly on the number of critical regions and grows exponentially. Consequently, the implementation of optimization

problems considering many constraints could be infeasible.

6.4 Simulation Results for Tube-based MPC using ADMM

This section analyses the tube-based model predictive control formulation presented in Chapter 3 using the ADMM optimization algorithm developed in Section 4.2. To evaluate the performance of the proposed formulation, it was conducted three experimental cases, each of which was executed ten times.

In experimental case one, it is used Gurobi (Gurobi Optimization, LLC, 2023) to solve the nominal optimization problem instead of the proposed ADMM. Gurobi is a well-known solver widely used in optimization fields, based on a barrier method. This experiment aims to establish a baseline for comparison with the proposed optimization algorithm in terms of flight performance.

In experimental case two, it is employed the proposed ADMM strategy in the optimization process, with parameters set as $\epsilon = 1e^{-3}$ and $\eta = 0.999$. In the first two experimental cases, constant aerodynamic forces are applied to the aircraft to test the capability of the system to reject disturbances.

In the third experimental case, disturbances are not applied to the aircraft. Then the number of iterations and computation time in the third experimental case are compared with those in experimental case two, where disturbances are present, with the objective to analyze the effect of the disturbances in the optimization algorithm.

The performance of the formulation is evaluated using the indices mentioned in Section 6.2. However, it is also conducted a T-test using the RMSE to determine if there is a significant difference in trajectory tracking performance between experimental cases 1 and 2. Similarly, besides of analyzing the maximum, minimum, and mean computation time values, it is performed a statistical analysis by examining the distribution and probability density of the computation time values.

Figure 6.11 presents the trajectories executed by the load tethered to the aircraft for both formulations. As previously mentioned, TMPC using Gurobi serves as a baseline for performance comparison with TMPC using ADMM optimization. The load's path was successfully tracked using the proposed Tube-MPC method with ADMM optimization, which also effectively rejects constant disturbances due to the utilization of an extended error model with integral actions. In Figure 6.11, it is evident that both formulations exhibit similar trajectories, indicating that the two optimization algorithms generated comparable optimal solutions.

This similarity persists even in the face of disturbance rejection. Analyzing Figure 6.12, which illustrates the translation error state behavior of the aircraft, the transient

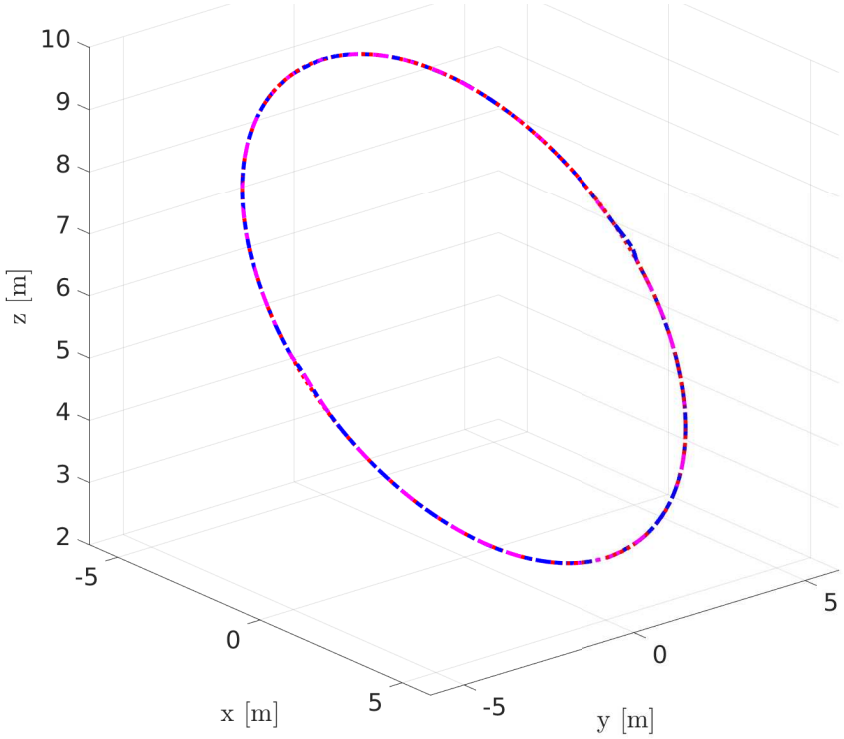

Figure 6.11: Path Tracking performed by the tiltrotor UAV with suspended load, in which the tracking reference and the disturbances behavior are illustrated.

response to disturbances appears similar in both formulations.

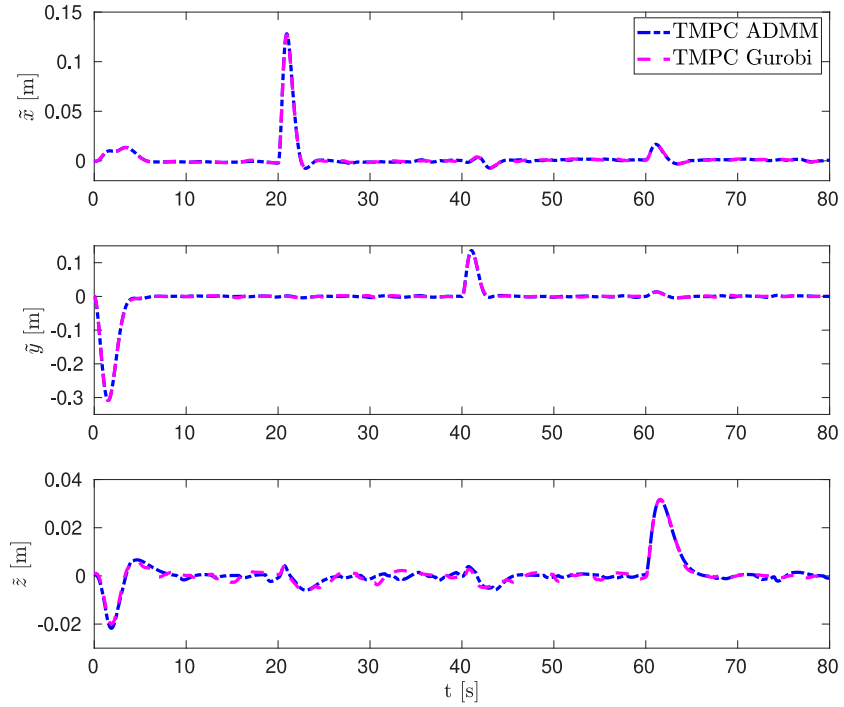

Figure 6.12: Time response for the error state of the translational position performed by the tiltrotor UAV with suspended load, in which the behavior of the system when the disturbances at times 20, 40 and 60 affect it.

In Figures 6.13 and 6.14, the stable behavior of the remaining degrees of freedom of the system is observable as the load follows its trajectory. The stable behavior observed in the load and tilt states also implies in a stable behavior of the UAV, as these variables implicitly describe the dynamic of the aircraft with respect to the inertial frame.

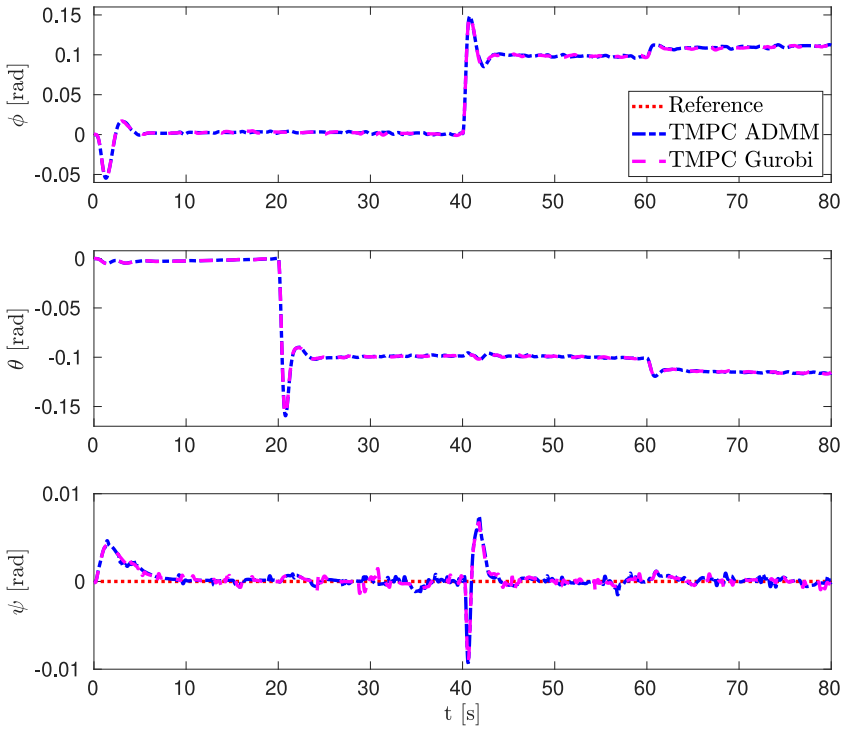

Figure 6.13: Time response for the rotational position performed by the tiltrotor UAV with suspended load, showing the behavior of the system when disturbances at times 20, 40 and 60 affects the system.

Observing the states γ_1 and γ_2 in Figure 6.14, noticeable numerical noise becomes apparent. This behavior is intricately linked with the optimization solutions obtained by both algorithms. As solvers seek optimal solutions through numerical formulations, they adhere to specific criteria to stop the computation algorithm. Consequently, the identified solution is often suboptimal, leading to abrupt changes, as illustrated in Figure 6.15 and Figure 6.16. In the case of the proposed ADMM algorithm, the stop criteria is set by the parameter $\epsilon = 1e-3$. This means that the algorithm does not precisely reach the ideal stop criteria of $\|\Gamma\|_\infty \leq 0$, but instead stops the computation when it is close enough to zero.

In Table 6.8, the computation of the Root Mean Square Error (RMSE) of the trajectories performed by the aircraft using both the Gurobi and the proposed ADMM formulation is presented. Additionally, the table shows the result of the student's t-test with a confidence interval of 95%.

Analyzing Table 6.8, it became evident that both algorithms computed similar optimal solutions, as the p-Values are bigger than 0.005, with the exception of state α_l, which presented a significant difference being the ADMM formulation better. The similarity of

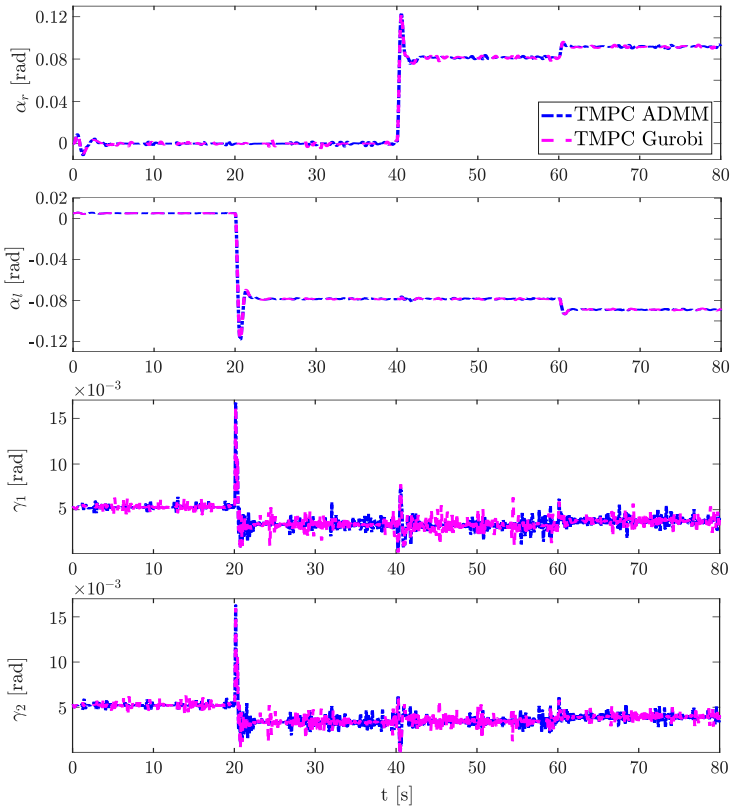

Figure 6.14: Time response for the rotational position of the tilt mechanisms and load performed by the aircraft, showing the behavior of the system when disturbances at times 20, 40 and 60 affects the system.

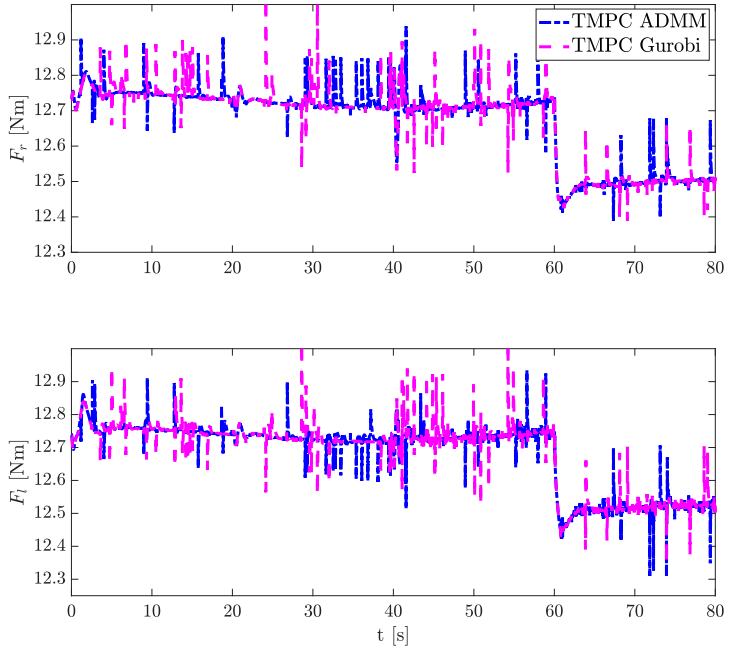

Figure 6.15: Time response for the rotational position of the tilt mechanisms and load performed by the aircraft, showing the behavior of the system when disturbances at times 20, 40 and 60 affects the system.

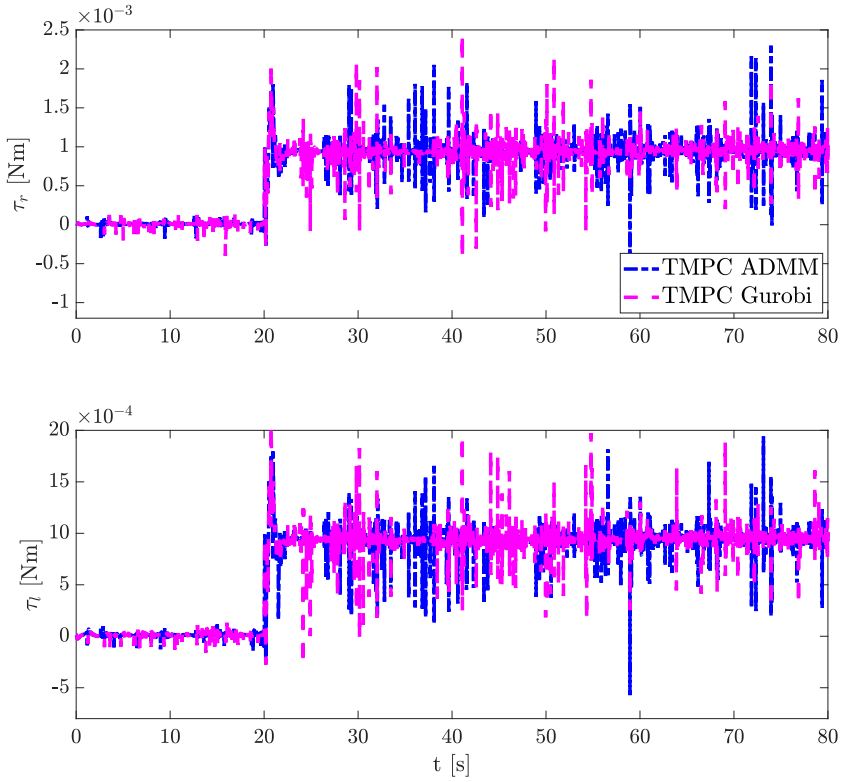

Figure 6.16: Time response for the rotational position of the tilt mechanisms and load performed by the aircraft, showing the behavior of the system when disturbances at times 20, 40 and 60 affects the system.

Table 6.8: Table showing the mean values of the MSE and the student's t test results for the ten first states performed by the aircraft in the first two experiment cases

	TMPC ADMM	**TMPC Gurobi**	**T-test**	
State	Mean	Mean	p-Value	Std
x	0.0040	0.0040	0.9760	1.276e-05
y	0.0122	0.0122	0.9903	2.436e-05
z	0.0028	0.0028	0.9517	3.073e-05
ϕ	1.4151	1.4151	0.5874	1.585e-04
θ	2.0053	2.0053	0.8317	2.509e-04
ψ	1.0306	1.0306	0.9177	5.993e-04
α_r	3.7275	3.6724	0.0628	6.215e-02
α_l	10.1135	9.9474	0.0287	15.622e-02
γ_1	1.2042	1.2043	0.1353	2.3000e-04
γ_2	2.1736	2.1734	0.2233	2.5924e-04

the optimal solution is also corroborated by the graphics presented in this section, where the aircraft behavior is similar in both cases.

6.4.1 Solver Performance

In this section, it is analyze the time performance of the proposed ADMM formulation. The analysis uses the numerical results taken from the experimental cases 2 and 3 in order to determine if the algorithm is robust when disturbances affect the system. However, it is not compared with the time computation spend by the Gurobi optimization solver since the proposed formulation is executed in the GPU of the embedded system, and Gurobi can only be executed in the CPU.

Taking into account Figure 6.17, the maximum average computation time of the proposed algorithm is close to 10 milliseconds. Since the sampling time of the system is 12 milliseconds, the developed methodology can be implemented in a realtime framework. It is important to note that in cases where the computation time is longer than the sample time, the controller sends the second control law to sustain the realtime framework. This behavior can be observed in simulations 2 and 9 of the Non-Disturbed and Disturbed scenarios, as shown in Figure 6.17.

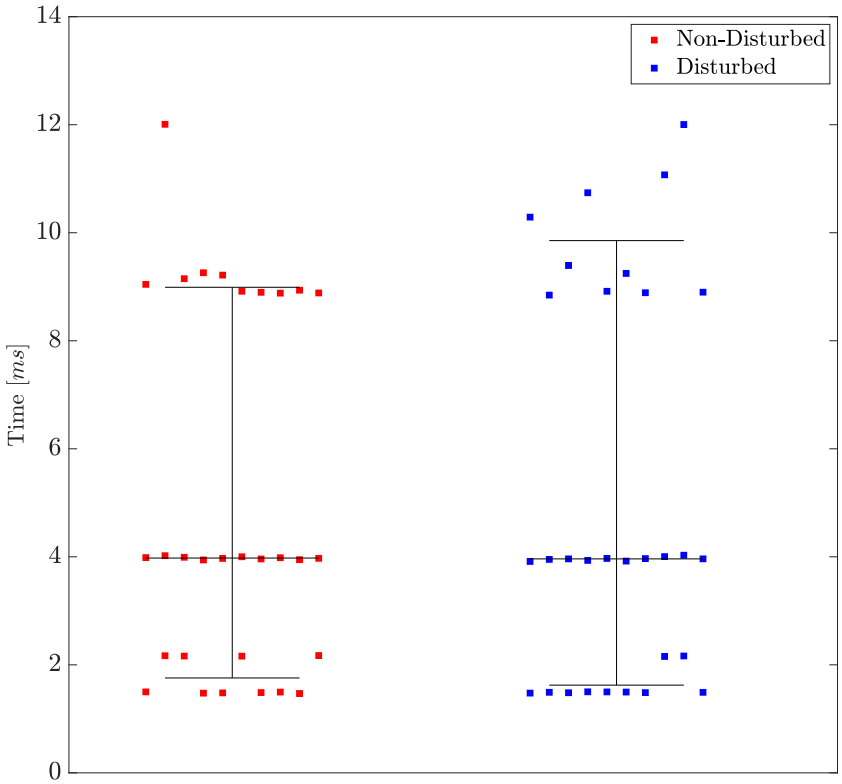

Figure 6.17: Graphic showing the maximum, minimum and mean values of the time computation spent by the embedded computer for the Non-Disturbed and Disturbed experimental scenarios.

Figure 6.18 shows that the execution time of the proposed algorithm is consistent

over time since the maximum, minimum, and mean average time values spent by the algorithm at each iteration are similar in both cases. Therefore, comparing Figure 6.18 with Figure 6.17, it can be concluded that in some cases, the proposed algorithm increments the number of iterations performed by the algorithm to compute the optimal solution when disturbances are presented. However, these disturbances never produce larger time computations than the system's sample time.

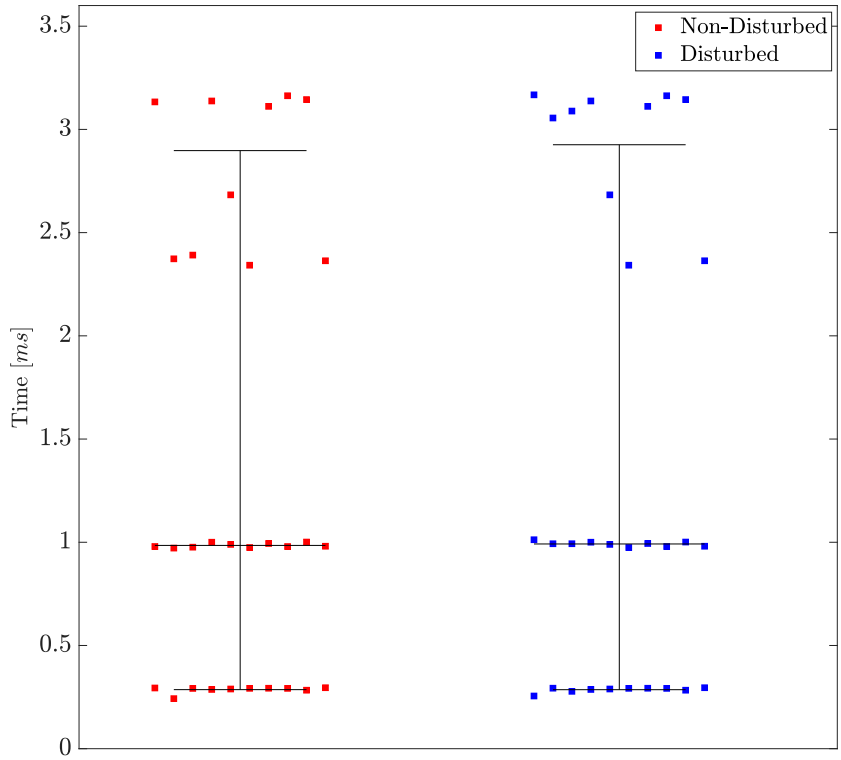

Figure 6.18: Graphic showing the time computation spent by the embedded computer for each iteration. The results are analyzed by the maximum, minimum and mean values of each experiment.

Figure 6.19 illustrates the numerical distribution and data density of the computation time spent by the controller during the third experimental case. The density curve indicates that most of the time the algorithm required 2.3 milliseconds to compute the optimal solution. Importantly, the algorithm consistently met the system's sampling time without any violations.

Similarly, the density curve in Figure 6.20 indicates that most of the time, the algorithm required the same computation time as in the third experimental case to find the optimal solution, demonstrating a slight influence of the disturbances on the algorithm's computation time. However, there are instances in this experimental case where the computation time exceeds the sampling time, where the second control law is activated.

The consistency of the computation time for the ADMM optimization algorithm is more evident in Figure 6.21. In this figure, the results from the ten executions of each experimental case are combined for a comprehensive analysis of the algorithm's

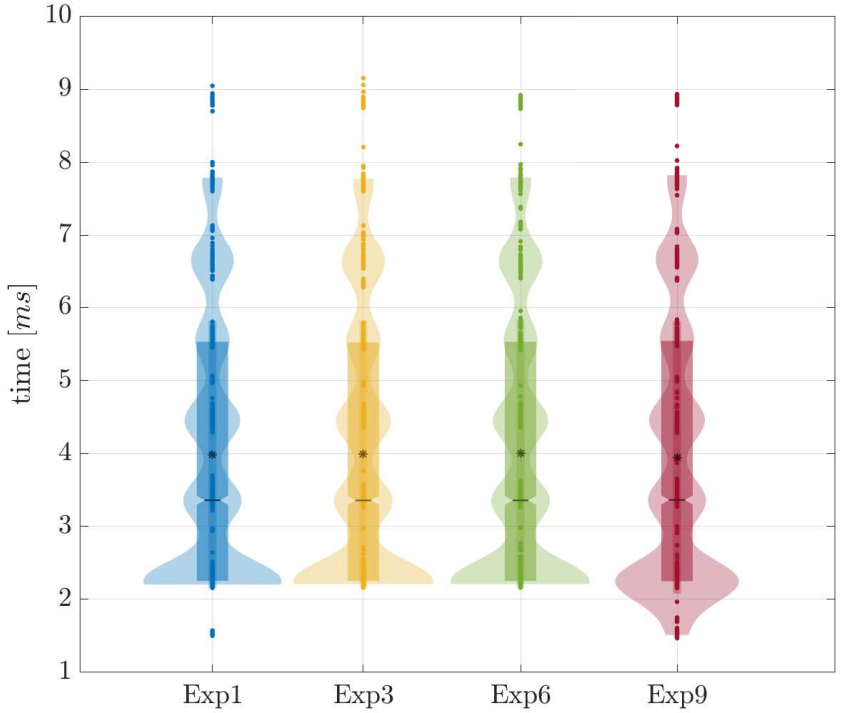

Figure 6.19: Violin plot showing the computational time spent by the controller while the tilt-rotor UAV performs the tracking trajectory of the suspended load without applying disturbances.

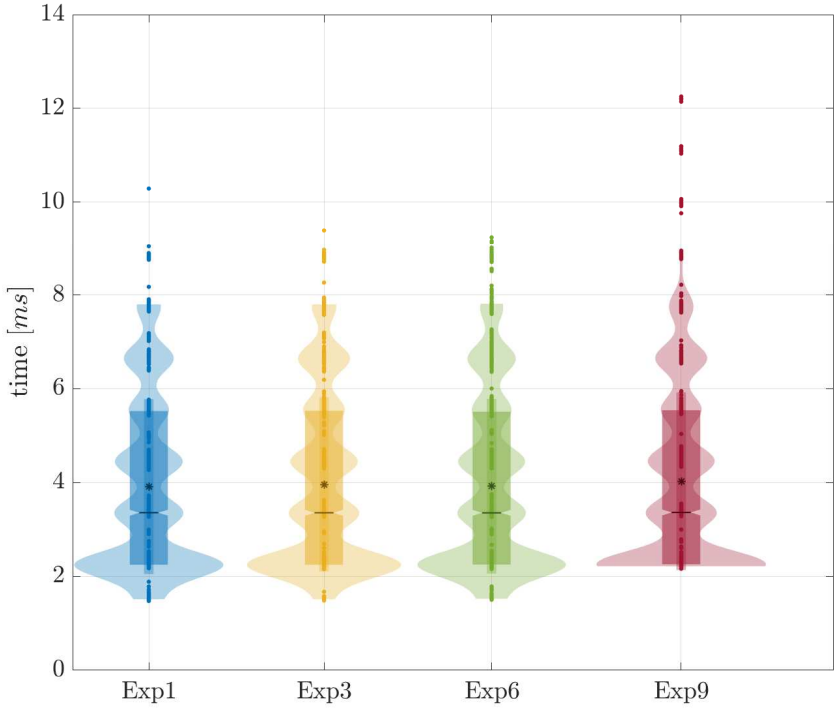

Figure 6.20: Violin plot showing the computational time spent by the controller while the tilt-rotor UAV performs the tracking trajectory of the suspended load when disturbances are not applied to the system.

performance. It can see that both cases show the same density curve shapes, where the highest density concentration is around 2.3 milliseconds. This implies that the algorithm exhibits consistent behavior both in the presence and in the absence of disturbances.

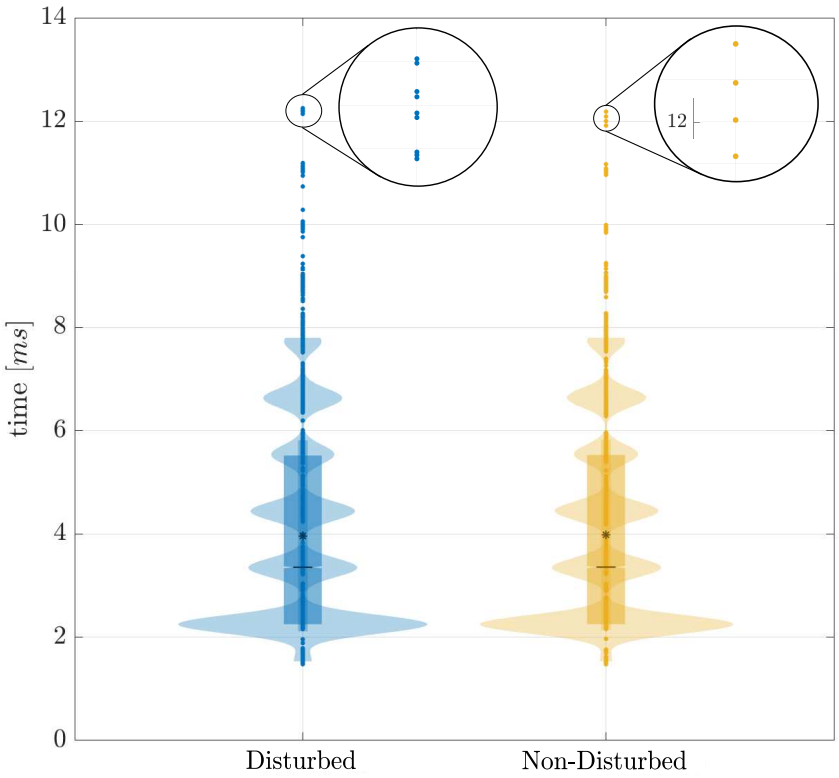

Figure 6.21: Time response for the error state of the translational position performed by the tiltrotor UAV with suspended load. Behavior of the system when the disturbances at times 20, 40 and 60 affect it.

Accordingly, the proposed scaled-symmetric ADMM algorithm demonstrates a notable capability for execution within a real-time framework. Table 6.9 presents the system sampling time violation rate, with a maximum rate of less than 0.02%. This underscores the algorithm's efficiency in meeting the required time constraints. It is worth mentioning that, on occasions when the algorithm's computation time exceeds the system's sampling time, the second control law is activated to maintain real-time performance. Nonetheless, the proposed ADMM formulation is employed 99.98% of the time to compute the control signal.

Table 6.9: Number of sampling times where the algorithm calculation time was greater than the system sampling time

	Exp. Case 2	**Exp. Case 3**
Size Data	66680	66680
Violation Number	9	4
Violation Percentage	0.0135%	0.006%

6.5 Simulation Results for TMPC using CZ

The performance of the proposed tube-based MPC formulation is evaluated by performing three experimental scenarios, each are executed ten times. In the first experimental scenario, we use the standard nominal MPC problem using polytopes within the tube-based (TMPC), while the reachable set computation algorithm is based on zonotopes. This experiment aims to establish a baseline for comparison between an MPC formulation using polytopes with the proposed optimization algorithm based on constrained zonotopes in terms of flight performance. In contrast, the second experimental scenario employs the proposed MPC problem based on constrained zonotopes (4.46) in the tube-based formulation (CZ_TMPC).

The third experimental scenario involves a Monte Carlo simulation using the CZ_TMPC formulation. In this scenario, ten simulations are performed, applying random variations to the system's parameters between ±10% according to a normal distribution in each simulation. The objective of this experiment is to verify the robustness of the formulation. In all experimental scenarios, the Gurobi solver is used to find the optimal solution.

The performance of the formulation is evaluated using the indices mentioned in Section 6.2. In addition, a T-test is conducted to determine if there is a significant difference in the trajectory tracking performance between experimental scenarios. Regarding the execution time performance, it is performed a statistical analysis by examining the distribution and probability density of the computation time values.

The trajectories executed by the load tethered to the aircraft for both formulations are presented in Figure 6.22. As mentioned previously, the first experimental scenario provides a basis for a performance comparison with the CZ_TMPC formulation. The load's path was successfully tracked using our proposed CZ_TMPC, which also effectively rejects disturbances. In Figure 6.22, it is evident that both formulations exhibit similar trajectories, indicating that the two optimization algorithms generated equivalent optimal solutions.

Analyzing Figure 6.23, where the translational position error behavior of the suspended load is presented, the similarity persists even in the face of disturbance rejection, and the transient response to disturbances appears similar in both formulations.

While the load tracks its trajectories, the remaining degrees of freedom of the system have a stable behavior, as shown in Figures 6.24 and 6.25. In fact, the stability observed in the load and tilt states inherently implies the stability of the UAV. These variables implicitly describe the behavior of the aircraft concerning the inertial frame, making their stability a direct indicator of the UAV's stability.

Numerical noise becomes apparent when states γ_1 and γ_2 are analyzed in Figure 6.25. This behavior is closely tied to the stop criterion of the numerical optimization algorithms. As the solvers seek optimal solutions, this criterion stops the computation. Consequently,

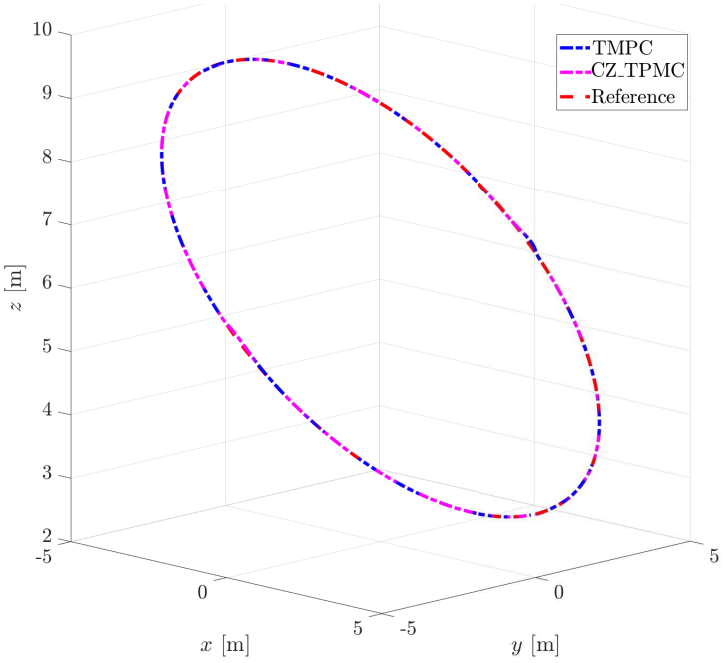

Figure 6.22: TMPC and CZ_TMPC comparison of the path tracking performed by the suspended load carried by the tiltrotor UAV.

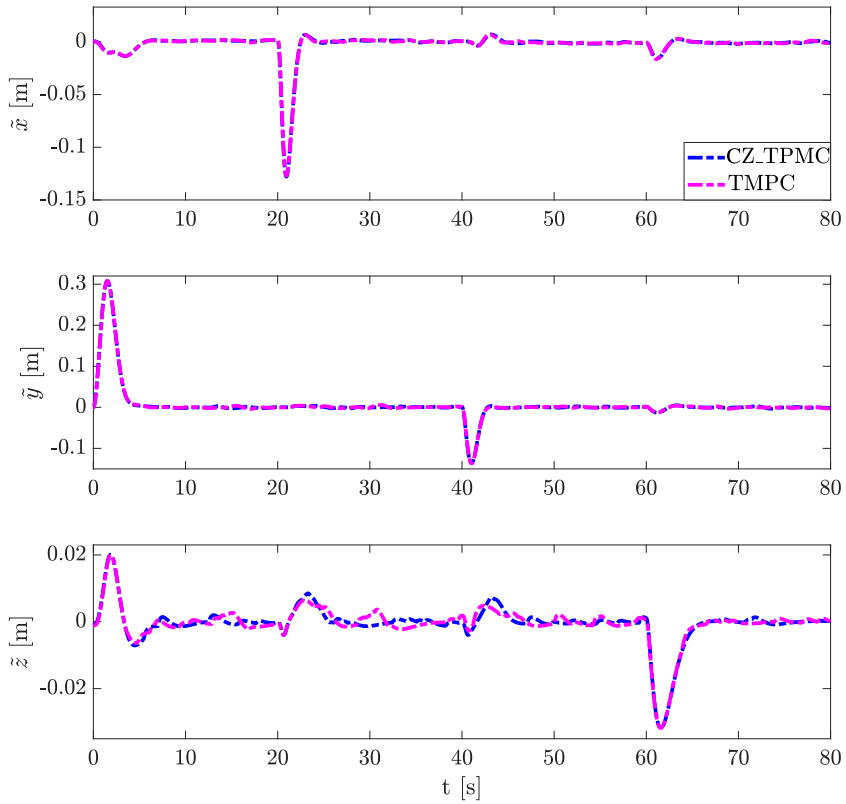

Figure 6.23: TMPC and CZ_TMPC comparison of the time response for the error state of the translational position performed by the suspended load carried by the tiltrotor UAV, highlighting the behavior of the system when the disturbances affect it at instants 20, 40 and 60 seconds.

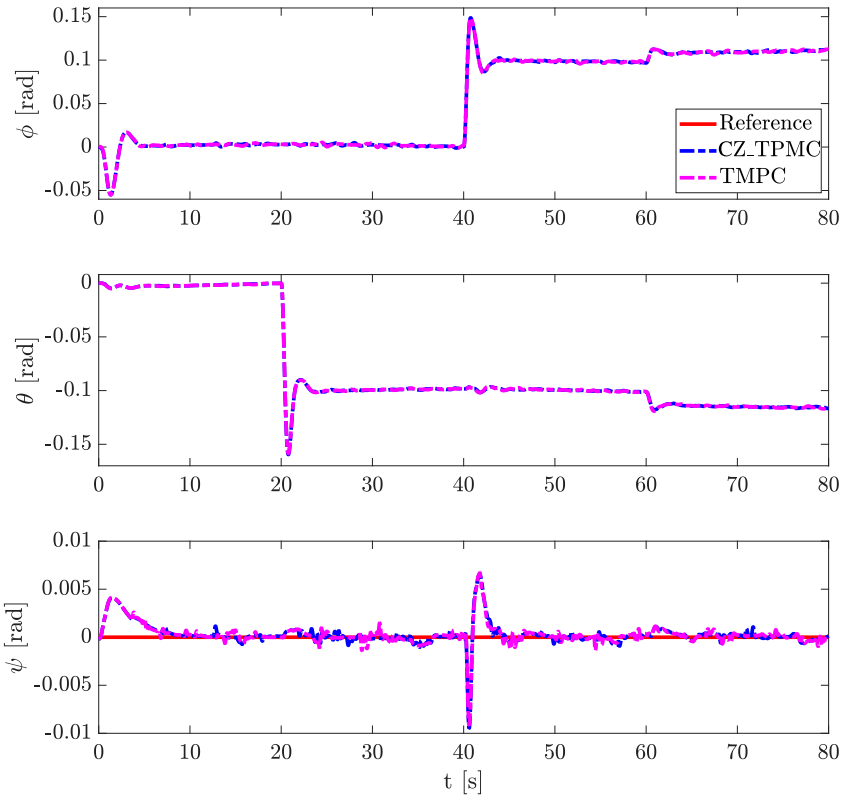

Figure 6.24: TMPC and CZ_TMPC comparison of the time response for the rotational position states performed by suspended load carried by the tiltrotor UAV, showing the behavior of the system when disturbances affect it at times 20, 40 and 60 seconds.

the founded solution is often suboptimal, producing abrupt changes, as illustrated in Figure 6.26 and Figure 6.27.

It is noteworthy that the proposed formulation is not intended to enhance the performance of the system's dynamic behavior. Nevertheless, its main objective is to reduce the computational cost of the optimization algorithm without modifying the system's optimal behavior, as shown in Figures 6.24 and 6.25. In fact, in the proposed formulation, the sets $\bar{\mathcal{U}}$ and $\bar{\mathcal{X}}$ are exactly mapped from the polytopic set representation to the constrained zonotope set representation. Therefore, it is expected that the proposed MPC problem achieve an optimal solution similar to the standard MPC. Conversely, an improvement in computation time is expected due to the computational efficiency afforded by the set representation employed.

The trajectories performed by the aircraft using the TMPC and the CZ_TMPC are evaluated using the Root Mean Square Error (RMSE). A comparative analysis was conducted using the T-test at a confidence interval of 95%, revealing similar optimal solutions generated by both algorithms, as shown in Table 6.10. In particular, states α_l and γ_1 exhibit a significant difference due to their p-values lower than 0.005, indicating a significant difference of time responses between these two states. In addition, all states exhibit a percentage error lower than 3%.

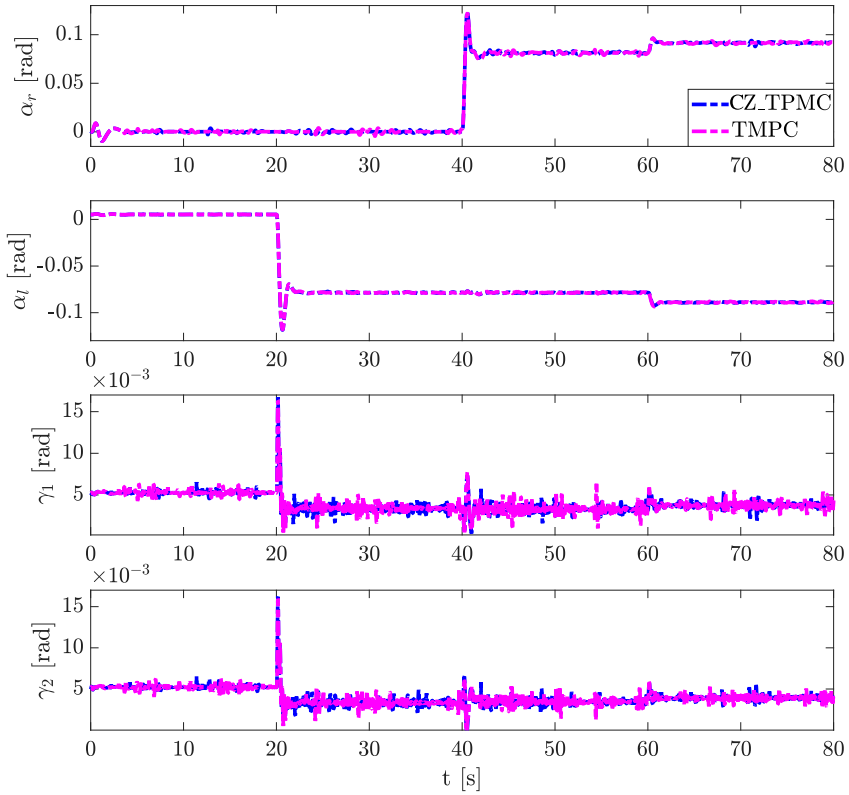

Figure 6.25: TMPC and CZ_TMPC comparison of the time response for the rotational position of the tilt mechanisms and load angles performed by the tiltrotor UAV, showing the behavior of the system when disturbances affect it at instants 20, 40 and 60 seconds.

Table 6.10: The mean values of the MSE and the t test results for the ten first states performed by the aircraft in the experiments one and two

	TMPC	**CZ TMPC**	**Diff**	**T-test**	
State	Mean	Mean	%	p-Value	Std
x	0.0040	0.0039	2.5	0.0791	1.1837e-05
y	0.0122	0.0123	−0.8	0.2305	2.5698e-05
z	0.0028	0.0027	3.0	0.3147	3.4697e-05
ϕ	1.4151	1.4150	0.0	0.1283	1.7149e-04
θ	2.0053	2.0053	0.0	0.2919	1.6316e-04
ψ	1.0306	1.0307	0.0	0.7913	6.4541e-04
α_r	3.6724	3.7243	−1.4	0.0708	6.215e-02
α_l	9.9474	10.0834	−1.4	0.0436	0.0605e-00
γ_1	1.2043	1.2041	0.0	0.0054	0.1402e-00
γ_2	2.1734	2.1736	0.0	0.0170	1.5580e-04

In Table 6.11, the mean values of the TVC indices are presented for the four control actions applied to the aircraft in both experimental scenarios. This table shows that the control signal torques have significantly different behavior between the two experimental scenarios. It can also be corroborated in Figure 6.27, where both signals exhibit different behaviors, but the equilibrium points are the same.

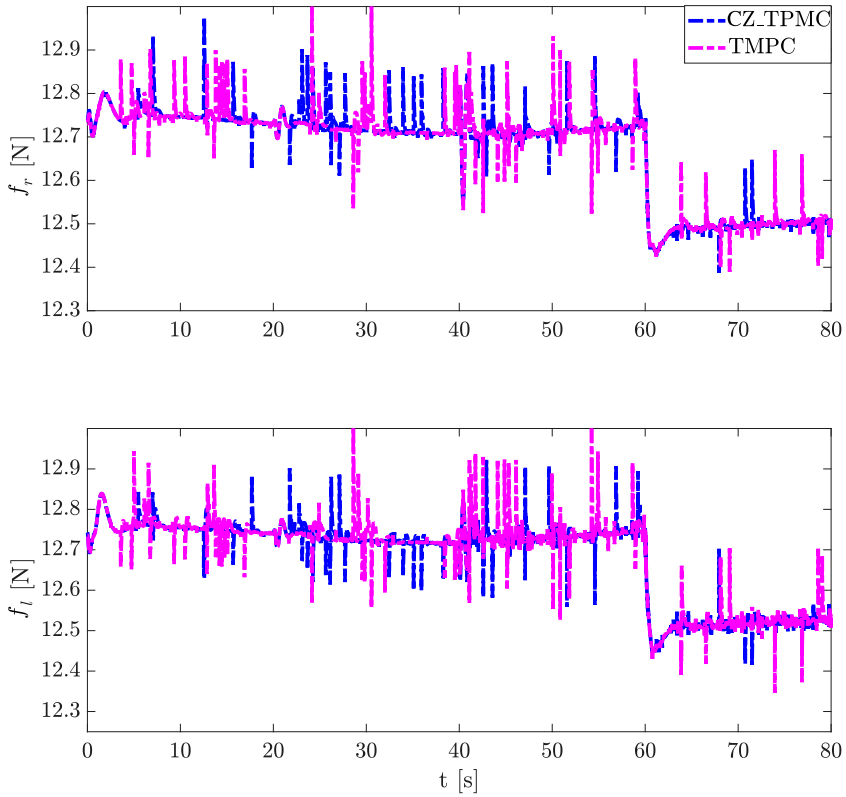

Figure 6.26: TMPC and CZ_TMPC comparison of the time response of the thrust forces applied to the propulsion system, showing the behavior the control signals when disturbances affects the system at instants 20, 40 and 60 seconds.

Table 6.11: Table showing the mean values of the Total Control Value (TCV) index, along with the T-test results for the control signals applied to the aircraft's actuators in both experimental scenarios.

	TMPC	**CZ-TMPC**	**T-test**	
State	Mean	Mean	p-Value	Std
f_r	6.2866e+03	6.2788e+03	0.0865	9.5518
f_l	6.1057e+03	6.0988e+03	0.1160	9.2898
τ_r	4.7758e+00	4.7662e+00	0.0116	0.0077
τ_l	4.6930e+00	4.6853e+00	0.0110	0.0061

Moreover, we conducted a Montecarlo simulation to verify the robustness of the proposed formulation. In these experiments, the inertia matrix elements and masses of the UAV bodies vary between ±10% using a random uniform distribution. Then, ten simulations are performed, applying random variations to the system's parameters in each simulation. As illustrated in Figure 6.28, the general behavior of the controlled error states is similar to the case in which the nominal parameters are used, corroborating the

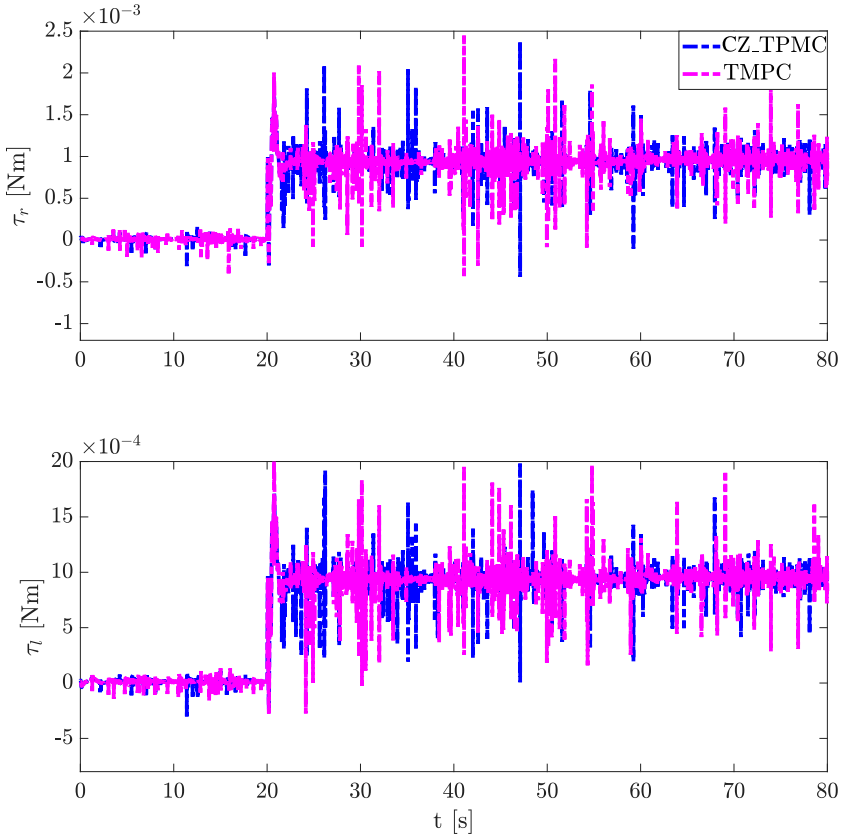

Figure 6.27: TMPC and CZ_TMPC comparison of the time response of the torque forces applied to the tilting mechanisms, showing the behavior the control signals when disturbances affect the system at instants 20, 40 and 60 seconds.

robustness of the proposed tube-based MPC formulation. It is noteworthy that the time responses of the system's states across all ten simulations exhibit similarities. This is primarily due to the second term of the tube-based control law, which compensates for the differences between the nominal model and the UAV system using a feedback gain $\boldsymbol{K}$. This gain is designed within a set defined by the maximum and minimum bounds of the uncertainties.

All regulated states were stabilized in different equilibrium points depending on the UAV parameter values. This behavior is more evident when the system is affected by a disturbance, as illustrated in Figure 6.29 by the time reposnses of the ϕ and θ states. As we are varying the inertia matrix elements of the UAV bodies, the control formulation modifies the angles' equilibrium points to compensate for this difference. However, the controlled states are always near the reference due to the integrator used in the extended state vector, as depicted by the ψ state's time response in Figure 6.29.

Figure 6.29 demonstrates that the error state z consistently converges to 0, indicating that the controller adjusts the thrust forces to track the trajectory. During the Monte Carlo simulation, we varied the masses of the UAV bodies, implying a change in the equilibrium point of the thrust forces to compensate for these variations. This change in

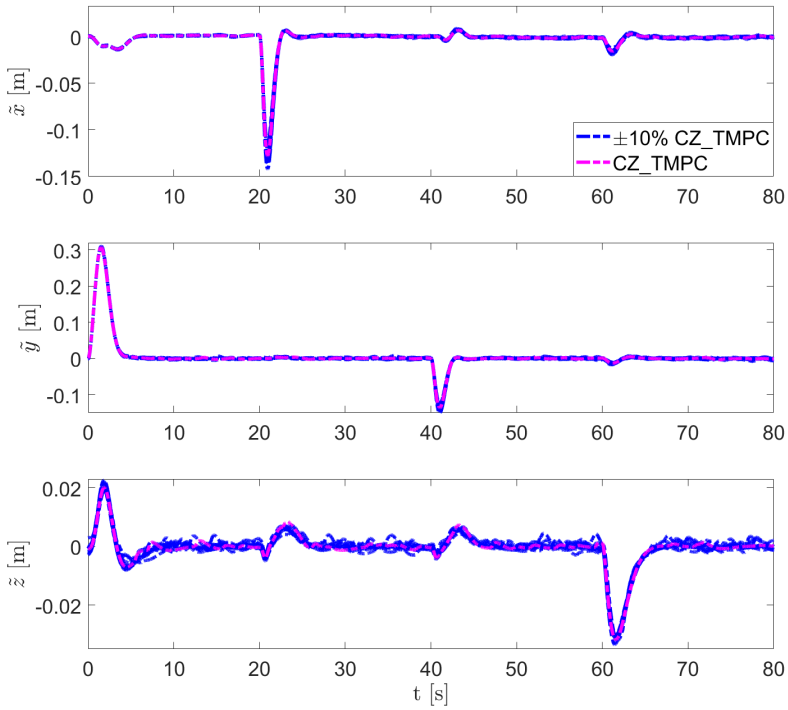

Figure 6.28: Comparison of the time responses of the translational position error states of the suspended load using nominal and modified inertia and mass parameters values, showing the system's behavior when disturbances affect the system at instants 20, 40, and 60 seconds.

equilibrium points is illustrated in Figure 6.30, showing different equilibrium points from 0 to 60 seconds. Before the 60-second mark, a positive disturbance force is applied to the load, causing it to be pushed vertically upward. The controller interprets this disturbance as a reduction in the load mass, leading to a decrease in the thrust forces.

In addition, we conducted an analysis of the time performance associated with the proposed formulation. We examine the computational time spent by the solver to find the optimal solution in both experimental scenarios. The main objective is to determine whether employing a set representation based on a constrained zonotope contributes to a reduction in the computational time of the solver.

In Figure 6.31, it is evident that the maximum average computation time for the optimization problem based on the constrained zonotope is lower than the minimum average value observed in the standard MPC formulation using polytope representation. This observation strongly suggests that employing a constrained zonotope representation could significantly assist the solver in reducing the computational time required to obtain the optimal solution.

The numerical distribution and data density presented in Figure 6.32 illustrate the computation time spent by the controller during the first experimental scenario. The density curve indicates that most of the time the algorithm required 29 milliseconds to compute the optimal solution.

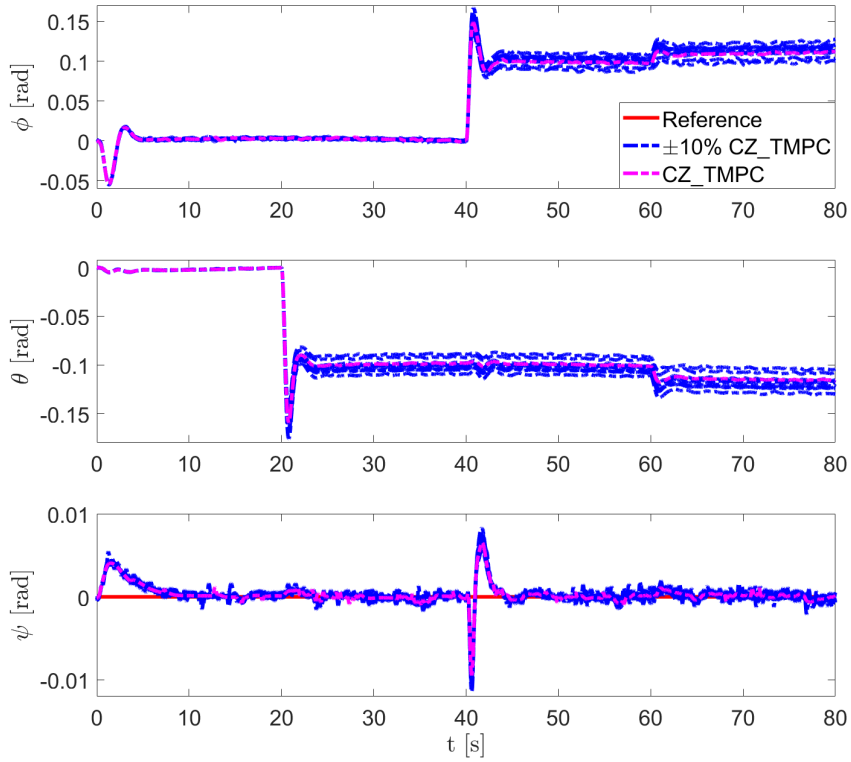

Figure 6.29: Comparison of the time responses of the rotational position states of the suspended load using nominal and modified inertia and mass parameter values, showing the system's behavior when disturbances affect the system at instants 20, 40, and 60 seconds.

In contrast, the density curve for the computational time of experimental case two, in Figure 6.33, indicates that most of the time, the solver required 20.9 milliseconds, which is 30% better than the formulation using polytope representation.

In Figure 6.34, the results of the ten executions of each experimental scenario are combined for a comprehensive analysis. We can see a notable reduction in the median value and data distribution in the formulation employing constrained zonotopes compared to the standard one. This performance improvement is related to the unitary box condition of the zonotopic constraints. In the standard formulation, the constraints of the OCP are represented by polytopes, while the proposed optimization problem considers unitary boxes to represent the same sets. Therefore, from the optimization perspective, solving a problem subjected to box constraints is generally faster than solving the same problem subject to polytopic constraints.

It is important to note that the algorithm's computation time exceeded the system's sampling time, as it was not executed on the embedded computer's GPU. However, the formulation based on constrained zonotopes significantly improved the computation time of the optimal solution.

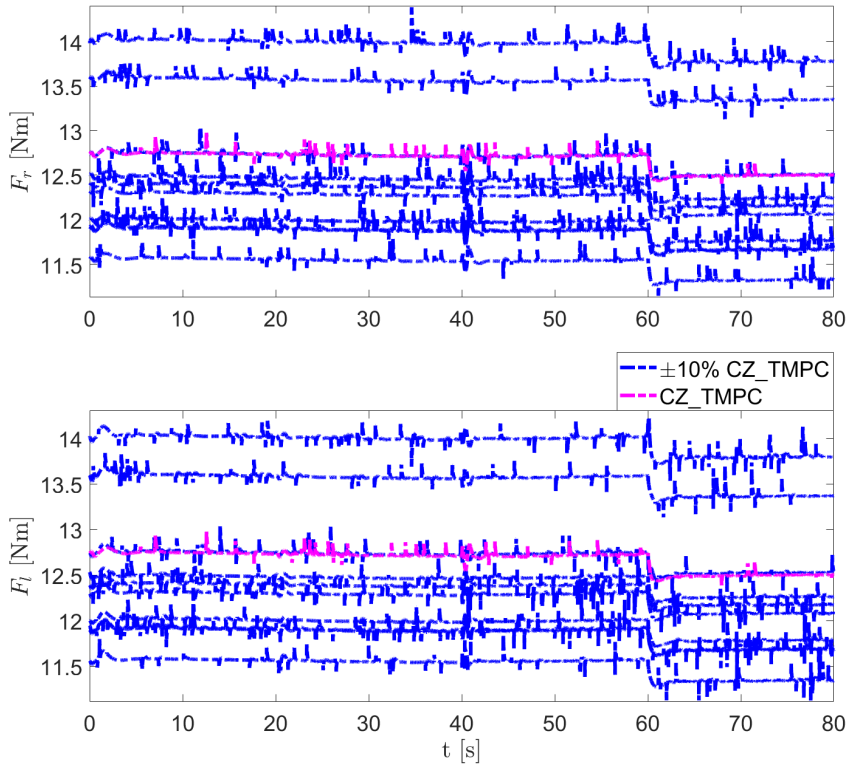

Figure 6.30: Thrust forces applied to the propulsion system using nominal and modified inertia and mass parameter values, showing the system's behavior when disturbances affect the system at instants 20, 40, and 60 seconds.

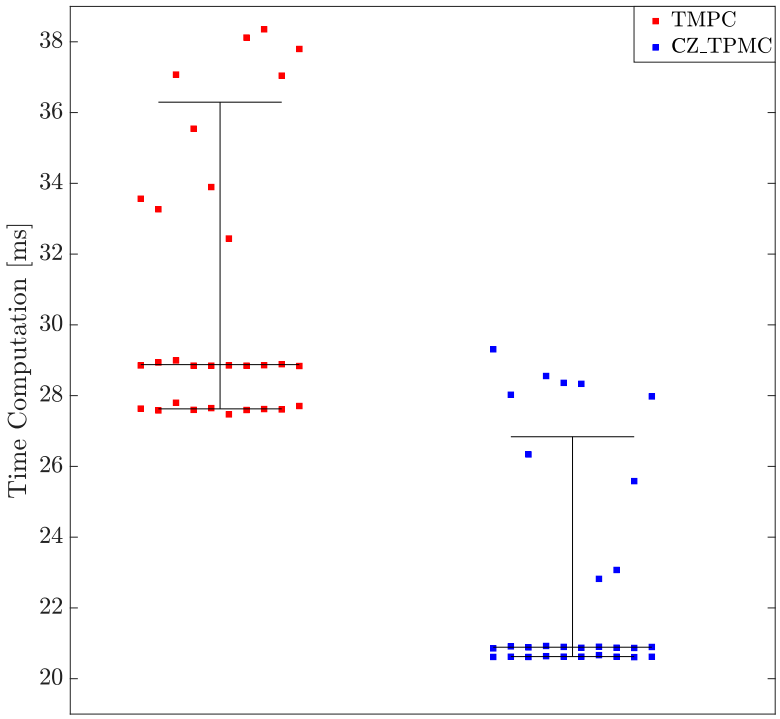

Figure 6.31: TMPC and CZ_TMPC comparison of the maximum, minimum and mean values of the time computation spent by the embedded computer.

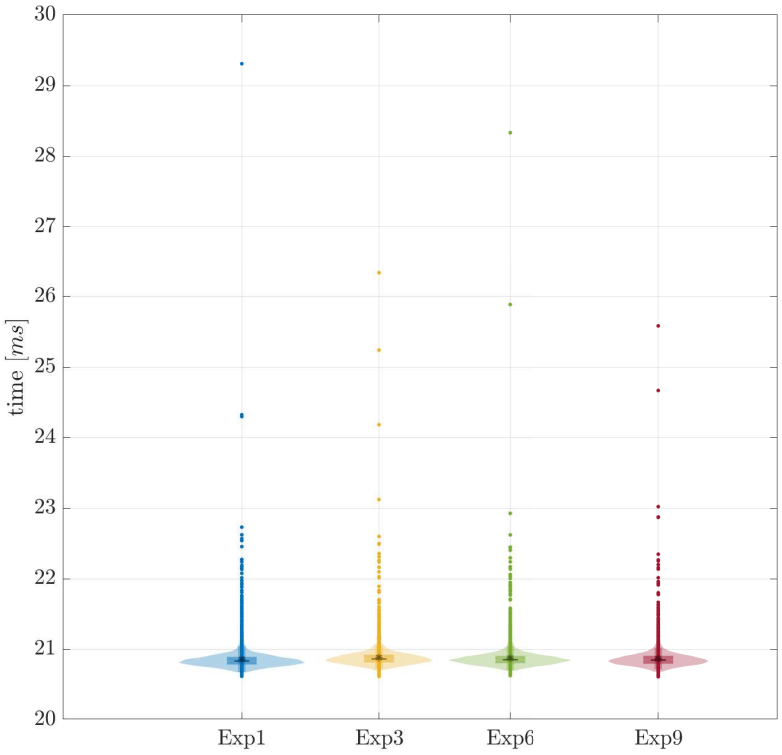

Figure 6.32: Violin plot regarding the computational time spent by the controller using the standard MPC optimization while the tilt-rotor UAV performs the trajectory tracking of the suspended load.

6.6 Final Remarks

This chapter analyzed the proposed formulations through HIL experiments, seeking realtime implementation in an embedded computer to control fast and high-order dynamic systems. The first methodology used the new tube-based MPC formulation, computing the nominal control law term through the explicit solution of the nominal MPC problem (3.55). The solution is implemented in a Jetson Nano embedded computer using the new searching algorithm developed in Section 4.1 running in the CPU. The second methodology used the new tube-based MPC formulation using the novel ADMM optimization algorithm (Section 4.2) to solve the nominal MPC problem (3.55). The control algorithm is implemented in a Jetson TX2 embedded computer and runs in the GPU. The third formulation proposed the use of the tube-based MPC with constrained zonotopes in the nominal MPC optimization presented in Section 4.3. The algorithm was also implemented in a Jetson TX2 embedded computer running in the CPU.

All formulations were successfully implemented in embedded systems, improving the standard tube-based MPC and making the implementation of the robust approach in fast and high-order dynamic systems feasible.

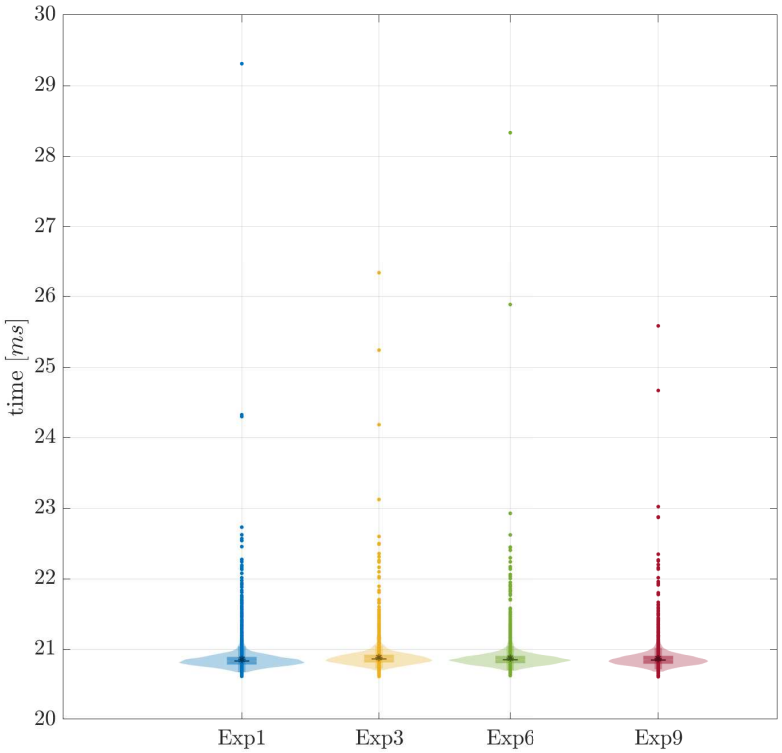

Figure 6.33: Violin plot regarding the computational time spent by the controller using the constrained zonotope-based MPC optimization while the tilt-rotor UAV performs the trajectory tracking of the suspended load.

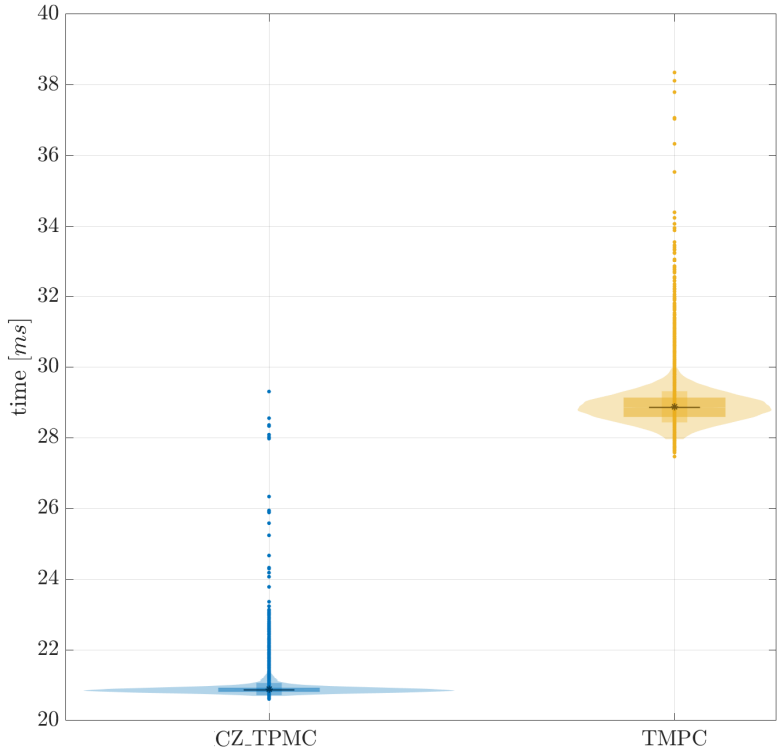

Figure 6.34: Violin plots combining the ten executions results for TMPC and CZ_TMPC formulations regarding the computational time expended by controller while tilt-rotor UAV performs the trajectory tracking of the suspended load.

7 Conclusion

This chapter provides a summary of the main contributions of this doctoral thesis and concludes the text. Additionally, proposals for future work are presented and detailed at the end of this chapter.

7.1 Overview and Contributions

Motivated by the importance to run robust model predictive control algorithms on low resource systems (embedded systems) for performing autonomous navigation in cargo transportation tasks, this thesis proposed to investigate the problem of fast and efficient optimization, to compute the control law of tube-based model predictive control formulation for fast dynamic systems. The problem was addressed in the control design level, either with fast optimization algorithms, the use of new hardware platforms, and parallel computing, in such a way we were able to provide the development of novel formulations gathering distinct objectives together.

Three approaches were proposed. In the first one, the optimization problem was performed offline using multi-parametric optimization, while in the second one the first-order optimization method, specifically ADMM optimization, was used to accelerate the online solution of the optimization problem. Both methodologies exploited the use of parallel computing and hardware designed for concurrent computing to accelerate vector and matrix operations. Besides, this work focused on the application of the tube-based model predictive control to fast dynamic and large-scale systems, providing a robust

guarantee to the application and giving important theoretical findings. This manuscript is structured into seven chapters, each contributing to the overall research as detailed below.

Chapter 1 served as the foundation for this doctoral thesis, motivating the development of a tube-based MPC formulation suitable for fast and high-order dynamic systems. It highlighted the limitations of the classic formulation when computing the reachable set for high-order systems. Also, it exposed the computation time drawbacks of the MPC optimization when applied to fast dynamic systems. Additionally, Chapter 1 outlined this doctoral thesis's justification and objectives.

Chapter 2 provided a comprehensive literature review on the implementation of tube-based MPC formulations in high-order systems. It also discussed methodologies for fast optimization of MPC techniques and the standard hardware used to implement fast optimization algorithms. Additionally, all the works that served as basis for this doctoral thesis were presented.

In Chapter 3, a comprehensive methodology was developed, emphasizing the offline computation of the tube-based MPC parameters for high-order systems. A new algorithm to compute the reachable set as a Robust Positive Invariant (RPI) set for an LPV system using zonotopes was proposed. For the computation of the feedback gain, a group of LMIs was proposed to consider the maximum value of the control action and the system uncertainty represented as a zonotope set, granting the existence of the nominal control and state sets. The contributions of this chapter are summarized as follows:

- The proposal of a novel tube-based MPC for fast and high-order dynamical systems considering uncertain LPV models.
- The extension of the formulation to compute the reachable sets as a robust invariant set using zonotope proposed in Limon et al. (2008) to consider LPV model.
- The development of novel LMI conditions to obtain the feedback gains considering the maximum value of the control action and the system uncertainty represented as a zonotope set.
- The development of novel tube-based MPC formulation considering the control and states constraint sets, represented as constrained zonotopes aiming to reduce the computational time

Chapter 4 developed three methodologies to solve the nominal optimization problem. The first approach proposed the multi-parametric optimization of the nominal MPC. The solution is a group of critical regions and control law sets that solve the nominal part of the formulation. Also, with the explicit solution, a new parallel search algorithm was developed to find the critical region where the current state belongs, which uses the advantages of the concurrent computer to accelerate the computation of the nominal control signal. The

second approach proposed the online optimization based on ADMM to solve the MPC optimization. It proposed the division of the cost function into two optimum subproblems for the control signal and states. The subproblems were solved by exact computation of the optimum value, accelerating the computation time using parallel computing. In addition, it was proposed the use of the Nesterov fast gradient approach. Besides, a new reset criteria for the algorithm was developed to reduce the convergence rate of the algorithm. The third methodology proposed a new MPC formulation based on constrained zonotope. The optimization control problem were transformed to the constrained zonotope set representation using the nominal state and control set previously computed as a constrained zonotope sets. The contributions of this chapter are summarized as follows:

- The proposal of a new parallel search algorithm to find the critical region where the current state belongs, which takes advantages of the concurrent computation to accelerate the problem solution.
- The proposal of a novel fast and highly parallelizable optimization algorithm based on a scaled-symmetric ADMM methodology, enhancing computational efficiency for high-order systems.
- The development of a reset criteria based on the primal and dual feasibility conditions of the optimization problem.
- The development of a novel MPC optimization problem refined as a function of the constrained zonotope structure

Chapter 5 introduced the case study used to verify the performance of the control formulations. It developed the LPV model of a Tilt-rotor UAV carrying a suspended load. This system is modeled from the load perspective and based on the system presented by Rego & Raffo (2019). Therefore, the resulting linear model is normalized in order to reduce numerical issues in the developed algorithms. Also, this chapter presented the results for the computation of the reachable set for the mentioned system, characterized by 24 states. It demonstrated the successful application of the reachable set computation algorithm presented in Section 3.3.2 to high-order dynamical systems. The contributions of this chapter are summarized as follows:

- The proposal of a procedure to normalize an LPV model to reduce numerical issues in the optimization algorithms.
- The successful off-line computation of the reachable set for an LPV and high-order system.

Chapter 6 introduced the simulation environment used to verify the performance of the control formulation. It presented the HIL structure, the embedded computer

characteristics used in the test, and the implemented realtime algorithm. Also, the results of the experiment using the Tube-based MPC formulation, optimized by the explicit and ADMM formulation applied to the Tilt-rotor UAV via HIL framework, are presented. Moreover, it is presented a new tube-based MPC formulation using constrained zonotopes. In all experiment scenarios, it was analyzed the performance related to the trajectory performed by the aircraft and the computation time spent by the algorithm for computing the control law. The contributions of this chapter are summarized as follows:

- The proposal and verification of the explicit tube-based MPC formulation to control a tiltrotor UAV in a realtime framework.
- The proposal and verification of the online tube-based MPC formulation using ADMM to control a tiltrotor UAV in a realtime framework.
- The proposal and verification of the online tube-based MPC formulation using constrained zonotopes to control a tiltrotor UAV in a realtime framework.
- The application of the proposed controllers to solve the trajectory tracking of a suspended load when it is carried by a tiltrotor UAV.
- Numerical experiments obtained via a HIL framework together with a high-fidelity 3D simulator based on CAD model.

7.2 Future Works

This section describes some possible directions for future work derived from the contributions presented in this doctoral thesis:

- The implementation of the control strategies in a real tiltrotor UAV.
- The extension of the tube-based MPC formulation to the use of constrained zonotopes, with the objective of seeking for the efficient online computation of the nominal and state sets. Thorough the constraint zonotope representation, the Pontryagin difference could be computationally efficient and fast.
- If the tube-based MPC formulation is performed using constrained zonotopes, a new multi-parametric optimization algorithm could be implemented using constrained zonotopes.
- The development of a methodology for the exact computation of the quadratic penalty matrix $\boldsymbol{\rho}$ in the ADMM optimization. The convergence of the algorithm is directly related to this matrix, and finding an exact computation method could improve the efficiency and reliability of the optimization process.

- The extension of the tube-based MPC formulation to consider only constrained zonotopes representation. It has been demonstrated that these set representations can improve the computation time of the algorithms, and integrating them into the online formulation could further enhance its efficiency.

Bibliography

Ahmadi-Moshkenani, P., Johansen, T. A., & Olaru, S. (2018). Combinatorial approach toward multiparametric quadratic programming based on characterizing adjacent critical regions. *IEEE Transactions on Automatic Control*, 63(10), 3221–3231.

Alcalá, E., Puig, V., Quevedo, J., & Sename, O. (2020). Fast zonotope-tube-based lpv-mpc for autonomous vehicles. *IET Control Theory and Applications*, 14(20), 3676–3685.

Altan, A., Aslan, Ö., & Hacıoğlu, R. (2017). Model predictive control of load transporting system on unmanned aerial vehicle (uav). In *Fifth international conference on advances in mechanical and robotics engineering* (pp. 1–4).: Institute of Research Engineers and Doctors Rome, Italy.

Altan, A., Aslan, O., & Hacıoğlu, R. (2018). Model reference adaptive control of load transporting system on unmanned aerial vehicle. In *2018 6th International Conference on Control Engineering & Information Technology (CEIT)* (pp. 1–5).

Andrade, R., E. Normey-Rico, J., & V. Raffo, G. (2024a). Fast embedded tube-based mpc with scaled-symmetric admm for high-order systems: application to load transportation tasks with uavs. *ISA Transactions*, In revision after first round.

Andrade, R., E. Normey-Rico, J., & V. Raffo, G. (2024b). Tube-based explicit model predictive control for a tiltrotor uav in cargo transportation tasks. *Journal of Control, Automation and Electrical Systems*, in press.

Andrade, R., E. Normey-Rico, J., & V. Raffo, G. (2024c). Tube-based model predictive control based on constrained zonotopes. *IEEE Access*, in press.

Andrade, R., Ferramosca, A., E. Normey-Rico, J., & V. Raffo, G. (2020). Explicit Model Predictive Control for a Tiltrotor UAV in Cargo Transportation Tasks. In *Anais do Congresso Brasileiro de Automatica*.

Andrade, R., Raffo, G., & Normey-Rico, J. (2016). Model predictive control of a tilt-rotor UAV for load transportation. In *2016 European Control Conference, ECC 2016*.

Andrade, R., Raffo, G. V., & Normey-Rico, J. E. (2016). Model predictive control of a tilt-rotor uav for load transportation. In *2016 European Control Conference (ECC)* (pp. 2165–2170).

Bahnemann, R., Schindler, D., Kamel, M., Siegwart, R., & Nieto, J. (2017). A decentralized multi-agent unmanned aerial system to search, pick up, and relocate objects. In *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)* (pp. 123–128). Shanghai: IEEE.

Basterretxea, K. & Benkrid, K. (2011). Embedded high-speed model predictive controller on a fpga. In *2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)* (pp. 327–335).

Bauersfeld, L., Spannagl, L., Ducard, G. J. J., & Onder, C. H. (2021). Mpc flight control for a tilt-rotor vtol aircraft. *IEEE Transactions on Aerospace and Electronic Systems*, 57(4), 2395–2409.

Bemporad, A., Morari, M., Dua, V., & Pistikopoulos, E. N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1), 3–20.

Besselmann, T., Lofberg, J., & Morari, M. (2008). Explicit model predictive control for linear parameter-varying systems. In *2008 47th IEEE Conference on Decision and Control* (pp. 3848–3853).

Bleris, L. G., Garcia, J., Kothare, M. V., & Arnold, M. G. (2006). Towards embedded model predictive control for system-on-a-chip applications. *Journal of Process Control*, 16(3), 255–264.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1), 1–122.

Boyd, S. & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Bulka, E., He, C., Wehbeh, J., & Sharf, I. (2022). Experiments on collaborative transport of cable-suspended payload with quadrotor uavs. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 1465–1473).

Cairano, S. D., Brand, M., & Bortoff, S. A. (2013). Projection-free parallel quadratic programming for linear model predictive control. *International Journal of Control*, 86(8), 1367–1385.

Camacho, E. & Alba, C. (2013). *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer London.

Cimini, G. & Bemporad, A. (2017). Exact Complexity Certification of Active-Set Methods for Quadratic Programming. *IEEE Transactions on Automatic Control*, 62(12), 6094–6109.

Dang, T. V., Ling, K., & Maciejowski, J. (2015). Embedded admm-based qp solver for mpc with polytopic constraints. In *2015 European Control Conference (ECC)* (pp. 3446–3451).

Ducard, G. J. & Allenspach, M. (2021). Review of designs and flight control techniques of hybrid and convertible vtol uavs. *Aerospace Science and Technology*, 118.

E. N. Pistikopoulos (2009). Perspectives in Multiparametric Programming and Explicit Model Predictive Control. *AIChE Journal*, 59(8), 215–228.

Eskandarpour, A., Mehrandezh, M., Gupta, K., Ramirez-Serrano, A., & Soltanshah, M. (2023). A constrained robust switching MPC structure for tilt-rotor UAV trajectory tracking problem. *Nonlinear Dynamics*, 111(18), 17247–17275.

Fleming, K., Yang, H.-J., Adler, M., & Emer, J. (2014). The leap fpga operating system. In *2014 24th International Conference on Field Programmable Logic and Applications (FPL)* (pp. 1–8).

Goldstein, T., O'Donoghue, B., Setzep, S., & Baraniuk, R. (2014). Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3), 1588–1623.

Gonzalez, R., Fiacchini, M., Guzmán, J. L., Alamo, T., & Rodríguez, F. (2011). Robust tube-based predictive control for mobile robots in off-road conditions. *Robotics and Autonomous Systems*, 59(10), 711–726.

González Sánchez, R. (2011). *Contributions to Modelling and Control of Mobile Robots in Off-Road Conditions*. PhD thesis, Universidad de Almería.

Gulan, M., Takács, G., Anh Nguyen, N., Olaru, S., Rodriguez-Ayerbe, P., & Rohal-Ilkiv, B. (2017). Embedded linear model predictive control for 8-bit microcontrollers via convex lifting. *IFAC-PapersOnLine*, 50(1), 10697–10704.

Gurobi Optimization, LLC (2023). Gurobi Optimizer Reference Manual.

Han, D., Rizaldi, A., El-Guindy, A., & Althoff, M. (2016). On enlarging backward reachable sets via Zonotopic set membership. *IEEE International Symposium on Intelligent Control - Proceedings*, 2016-Septe, 1–8.

Hang, P., Xia, X., Chen, G., & Chen, X. (2022). Active safety control of automated electric vehicles at driving limits: A tube-based mpc approach. *IEEE Transactions on Transportation Electrification*, 8(1), 1338–1349.

Hartley, E. N., Jerez, J. L., Suardi, A., MacIejowski, J. M., Kerrigan, E. C., & Constantinides, G. A. (2014). Predictive control using an FPGA with application to aircraft control. *IEEE Transactions on Control Systems Technology*, 22(3), 1006–1017.

He, B., Xu, H. K., & Yuan, X. (2016). On the Proximal Jacobian Decomposition of ALM for Multiple-Block Separable Convex Minimization Problems and Its Relationship to ADMM. *Journal of Scientific Computing*, 66(3), 1204–1217.

Herceg, M., Jones, C. N., Kvasnica, M., & Morari, M. (2015). Enumeration-based approach to solving parametric linear complementarity problems. *Automatica*, 62, 243–248.

Herceg, M., Kvasnica, M., Jones, C., & Morari, M. (2013). Multi-Parametric Toolbox 3.0. In *2013 European Control Conference* (pp. 502–510). Zürich, Switzerland.

Hoffmann, C. & Werner, H. (2015). A survey of linear parameter-varying control applications validated by experiments or high-fidelity simulations. *IEEE Transactions on Control Systems Technology*, 23(2), 416–433.

Hrustic, N. & Prljaca, N. (2020). An implementation and evaluation of fast embedded model predictive control. In *2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH)* (pp. 1–5).

Hu, H., Feng, X., Quirynen, R., Villanueva, M. E., & Houska, B. (2018). Real-time tube mpc applied to a 10-state quadrotor model. *Proceedings of the American Control Conference*, 2018-June, 3135–3140.

Jeong, D. & Choi, S. B. (2024). Tube-based robust model predictive control for tracking control of autonomous articulated vehicles. *IEEE Transactions on Intelligent Vehicles*, 9(1), 2184–2196.

Jerez, J. L., Constantinides, G. A., Kerrigan, E. C., & Ling, K. V. (2011). Parallel MPC for real-Time FPGA-based implementation. In *IFAC Proceedings Volumes (IFAC-PapersOnline)*.

Jerez, J. L., Goulart, P. J., Richter, S., Constantinides, G. A., Kerrigan, E. C., & Morari, M. (2014). Embedded online optimization for model predictive control at megahertz rates. *IEEE Transactions on Automatic Control*, 59(12), 3238–3251.

Jones, C. N. & Morari, M. (2006). Multiparametric linear complementarity problems. In *Proceedings of the 45th IEEE Conference on Decision and Control* (pp. 5687–5692).

Keane, J. F. & Carr, S. S. (2013). A brief history of early unmanned aircraft. *The Johns Hopkins APL Technical Digest*, 32(3), 558–571.

Kouramas, K. I., Faísca, N. P., Panos, C., & Pistikopoulos, E. N. (2011). Explicit/multi-parametric model predictive control (MPC) of linear discrete-time systems by dynamic and multi-parametric programming. *Automatica*, 47(8), 1638–1645.

Krupa, P., Alvarado, I., Limon, D., & Alamo, T. (2022). Implementation of model predictive control for tracking in embedded systems using a sparse extended admm algorithm. *IEEE Transactions on Control Systems Technology*, 30(4), 1798–1805.

Kvasnica, M., Szucs, A., Fikar, M., & Drgona, J. (2018). Explicit MPC of LPV systems in the controllable canonical form. *2013 European Control Conference (ECC)*, (pp. 1035–1040).

Limon, D., Alvarado, I., Alamo, T., & Camacho, E. (2008). On the design of robust tube-based MPC for tracking. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 17(1 PART 1), 15333–15338.

Ling, K., Wu, B., & Maciejowski, J. (2008). Embedded model predictive control (mpc) using a fpga. *IFAC Proceedings Volumes*, 41(2), 15250–15255.

Ling, K. V., Yue, S. P., & Maciejowski, J. M. (2006). A fpga implementation of model predictive control. In *2006 American Control Conference* (pp. 6 pp.–).

Liu, J., Peyrl, H., Burg, A., & Constantinides, G. A. (2014). Fpga implementation of an interior point method for high-speed model predictive control. In *2014 24th International Conference on Field Programmable Logic and Applications (FPL)* (pp. 1–8).

Loianno, G., Spurny, V., Thomas, J., Baca, T., Thakur, D., Hert, D., Penicka, R., Krajnik, T., Zhou, A., Cho, A., Saska, M., & Kumar, V. (2018). Localization, Grasping, and Transportation of Magnetic Objects by a Team of MAVs in Challenging Desert-Like Environments. *IEEE Robotics and Automation Letters*, 3(3), 1576–1583.

Lopes, A. R., Shahzad, A., Constantinides, G. A., & Kerrigan, E. C. (2009). More flops or more precision? Accuracy parameterizable linear equation solvers for model predictive control. *Proceedings - IEEE Symposium on Field Programmable Custom Computing Machines, FCCM 2009*, (pp. 209–216).

Lv, Z., Wu, Y., & Li, S. (2021). Nonlinear control for a coaxial tilt-rotor uav transporting a slung load. In *2021 China Automation Congress (CAC)* (pp. 2996–3001).

Marwedel, P. (2003). *Embedded System Design*. Springer.

Mayne, D. Q., E.C.Kerrigan, Wyk, E., & Falugi1, P. (2011). Tube-based robust nonlinear model predictive control. *INTERNATIONAL JOURNAL OF ROBUST AND NONLINEAR CONTROL*, 18, 1343–1353.

McInerney, I., Constantinides, G. A., & Kerrigan, E. C. (2018). A Survey of the Implementation of Linear Model Predictive Control on FPGAs. *IFAC-PapersOnLine*, 51(20), 381–387.

Morari, M., Baotić, M., & Borrelli, F. (2003). Hybrid systems modeling and control. *European Journal of Control*, 9(2-3), 177–189.

Nesterov, Y. (2014). *Introductory Lectures on Convex Optimization: A Basic Course*. Louvain-la-Neuve, Belgium: Springer New York, NY, 1 edition.

O'Donoghue, B., Stathopoulos, G., & Boyd, S. (2013). A splitting method for optimal control. *IEEE Transactions on Control Systems Technology*, 21(6), 2432–2442.

Ogunbodede, O. & Singh, T. (2021). Load Vibration Mitigation in Unmanned Aerial Vehicles With Cable Suspended Load. *Journal of Autonomous Vehicles and Systems*, 1(3).

Palma, V. G., Suardi, A., & Kerrigan, E. C. (2015). Sensitivity-based multistep MPC for embedded systems. *IFAC-PapersOnLine*, 48(23), 360–365.

Pang, S., Li, Q., & Ni, B. (2021). Improved nonlinear mpc for aircraft gas turbine engine based on semi-alternative optimization strategy. *Aerospace Science and Technology*, 118, 106983.

Papachristos, C., Alexis, K., Nikolakopoulos, G., & Tzes, A. (2011). Model predictive attitude control of an unmanned tilt-rotor aircraft. *Industrial Electronics (ISIE), 2011 IEEE International Symposium on*, (pp. 922–927).

Patrinos, P. & Bemporad, A. (2014). An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *IEEE Transactions on Automatic Control*, 59(1), 18–33.

Pereira, J. C., Leite, V. J., & Raffo, G. V. (2021). Nonlinear model predictive control on se(3) for quadrotor aggressive maneuvers. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 101.

Peyrl, H., Ferreau, H. J., & Kouzoupis, D. (2015). A Hybrid Hardware Implementation for Nonlinear Model Predictive Control. In *IFAC-PapersOnLine*.

Pizetta, I. H. B., Brandao, A. S., & Sarcinelli-Filho, M. (2020). Load Transportation by Quadrotors in Crowded Workspaces. *IEEE Access*, 8, 223941–223951.

Pounds, P. E. & Dollar, A. M. (2014). Aerial Grasping from a Helicopter UAV Platform. In O. Khatib, V. Kumar, & G. Sukhatme (Eds.), *Experimental Robotics*, volume 79 (pp. 269–283). Berlin, Heidelberg: Springer Berlin Heidelberg.

Raffo, G. V. & Almeida, M. M. (2018). A load transportation nonlinear control strategy using a tilt-rotor uav. *Journal of Advanced Transportation*, 2018.

Raghuraman, V. & Koeln, J. P. (2022). Set operations and order reductions for constrained zonotopes. *Automatica*, 139.

Raimondo, D. M., Marseglia, G. R., Braatz, R. D., & Scott, J. K. (2013). Fault-tolerant model predictive control with active fault isolation. *Conference on Control and Fault-Tolerant Systems, SysTol*, (pp. 444–449).

Rakovic, S., Kerrigan, E., Kouramas, K., & Mayne, D. (2005). Invariant approximations of the minimal robust positively invariant set. *IEEE Transactions on Automatic Control*, 50(3), 406–410.

Ramesh, P. S. & Muruga Lal Jeyan, J. V. (2022). Comparative Analysis of Fixed-Wing, Rotary-Wing and Hybrid Mini Unmanned Aircraft Systems (UAS) from the Applications Perspective. *INCAS Bulletin*, 14(1), 137–151.

Rawlings, J., Mayne, D., & Diehl, M. (2019). *Model Predictive Control: Theory, Computation, and Design 2nd Edition*, volume 197. Nob Hill Publishing.

Rego, B. S. & Raffo, G. V. (2019). Suspended load path tracking control using a tilt-rotor uav based on zonotopic state estimation. *Journal of the Franklin Institute*, 356(4), 1695–1729.

R.Gonzalez, Fiacchini, M., Alamo, T., Guzman, J., & Rodriguez, F. (2011). Online robust tube-based mpc for time-varying systems: a practical approach. *International Journal of Control*, 84(6), 1157–1170.

Rubagotti, M., Patrinos, P., Guiggiani, A., & Bemporad, A. (2016). Real-time model predictive control based on dual gradient projection: Theory and fixed-point fpga implementation. *International Journal of Robust and Nonlinear Control*, 26(15), 3292–3310.

Santos, M. A. (2018). *TUBE-BASED MPC WITH ECONOMICAL CRITERIA FOR LOAD TRANSPORTATION TASKS USING TILT-ROTOR UAVS*. PhD thesis, Universidade Federal de Minas Gerais.

Santos, M. A., Ferramosca, A., & Raffo, G. V. (2018). Tube-based MPC with Nonlinear Control for Load Transportation using a UAV. *IFAC-PapersOnLine*, 51(25), 459–465.

Schulze, L., Bertol, D. W., & Raffo, G. V. (2022). Fast computation of binary search tree for pwa functions representation using intersection classification. *Automatica*, (pp. 110217).

Scott, J. K., Raimondo, D. M., Marseglia, G. R., & Braatz, R. D. (2016). Constrained zonotopes: A new tool for set-based estimation and fault detection. *Automatica*, 69, 126–136.

Shoukry, Y., Watheq El-Kharashi, M., & Hammad, S. (2013). An embedded implementation of the Generalized Predictive Control algorithm applied to automotive active suspension systems. *Computers and Electrical Engineering*.

Shukla, H. A., Khusainov, B., Kerrigan, E. C., & Jones, C. N. (2017). Software and hardware code generation for predictive control using splitting methods. *IFAC-PapersOnLine*, 50(1), 14386–14391.

Smith, R. (2017). Nvidia announces jetson tx2: Parker comes to nvidia's embedded system kit.

Suardi, A., Kerrigan, E. C., & Constantinides, G. A. (2015). Fast FPGA prototyping toolbox for embedded optimization. In *2015 European Control Conference, ECC 2015* (pp. 2589–2594).

Sunghun, J. & kim hyun su (2017). Analysis of amazon prime air uav delivery service. *Journal of Knowledge Information Technology and Systems*, 12(2), 253–266.

Takács, G., Batista, G., Gulan, M., & Rohaľ-Ilkiv, B. (2016). Embedded explicit model predictive vibration control. *Mechatronics*, 36, 54–62.

Teixeira, G. (2018). *MULTI-CORE MODEL PREDICTIVE CONTROL STRATEGY FOR A TILT-ROTOR UAV IN SYSTEM-IN-THE-LOOP SIMULATION*. PhD thesis, Graduate Program in Electrical Engineering.

Tøndel, P., Johansen, T. A., & Bemporad, A. (2003). An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica*, 39(3), 489–497.

Vouzis, P. D., Bleris, L. G., Arnold, M. G., & Kothare, M. V. (2009). A system-on-a-chip implementation for embedded real-time model predictive control. *IEEE Transactions on Control Systems Technology*, 17(5), 1006–1017.

Wan, Z. & Kothare, M. V. (2003). An efficient off-line formulation of robust model predictive control using linear matrix inequalities. *Automatica*, 39(5), 837–846.

Wills, A., Mills, A., & Ninness, B. (2011). Fpga implementation of an interior-point solution for linear model predictive control*. *IFAC Proceedings Volumes*, 44(1), 14527–14532.

Wills, A. G., Knagge, G., & Ninness, B. (2012). Fast linear model predictive control via custom integrated circuit architecture. *IEEE Transactions on Control Systems Technology*, 20(1), 59–71.

Xu, F., Xi, Y., & Chen, H. (2012). Field programmable gate array/system on a programmable chip-based implementation of model predictive controller. *IET Control Theory & Applications*, 6(8), 1055–1063.

Yang, N., Li, D., Zhang, J., & Xi, Y. (2012). Model predictive controller design and implementation on fpga with application to motor servo system. *Control Engineering Practice*, 20(11), 1229–1235.

Yang, Y., Guan, X., Jia, Q.-S., Yu, L., Xu, B., & Spanos, C. J. (2022). A survey of admm variants for distributed optimization: Problems, algorithms and features.

Zhang, P., Zambreno, J., & Jones, P. H. (2017). An embedded scalable linear model predictive hardware-based controller using ADMM. *Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors*, (pp. 176–183).

Zhang, X., Ferranti, L., & Keviczky, T. (2018). An improved primal-dual interior point solver for model predictive control. *2017 IEEE 56th Annual Conference on Decision and Control, CDC 2017*, 2018-Janua(Cdc), 1126–1131.

Zometa, P., Kogel, M., Faulwasser, T., & Findeisen, R. (2012). Implementation aspects of model predictive control for embedded systems. In *2012 American Control Conference (ACC)* (pp. 1205–1210). Fairmont Queen Elizabeth: 2012 American Control Conference.