RAM ROBOTICS AND MECHATRONICS

EXPLOITING NONLINEAR MODEL PREDICTIVE CONTROL IN CONTACT-BASED AERIAL PHYSICAL INTERACTION

A. (Ayham) Alharbat

MSC ASSIGNMENT

Committee:
A. Franchi, Ph.D HDR
dr. D. Bicego
H. Esmaeeli, MSc
dr. ir. A.Y. Mersha
dr. ir. R.G.K.M. Aarts

October, 2021

069RaM2021
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Abstract

The spatial freedom and the unconstrained work-space of Multi-Rotor Aerial Vehicles (MRAVs) are promising characteristics for many applications that require physical interaction with an environment, such as contact-based inspection, Non-Destructive Testing (NDT), and human assistance in remote areas. However, Aerial Physical Interaction (APhI) control remains a major challenge considering the high non-linearity of MRAVs dynamics, their inherent instability, and limited capabilities of the actuators. Classical interaction control methods, which include hybrid position/force control, impedance, and admittance control, are limited by their reactive nature, where the control action is not optimized for the future horizon, and their ad hoc solutions to try to the system constraints. Model Predictive Control (MPC) exploits the a priori knowledge of the system dynamics to predict its behaviour in the future horizon, and optimize the control action to achieve the control objective, defined with a certain cost function, while the system constraints are included in the optimization problem. In the literature, MPC-based APhI controllers are proposed using hybrid position/force control approach, without any consideration of impedance control, or cascaded control architectures. In this thesis, the use of Nonlinear Model Predictive Control (NMPC) for APhI control is investigated in three configurations, impedance control, admittance control in a cascaded architecture, and hybrid position/force control. Three NMPC-based control approaches, exploring the three configurations, are proposed, implemented, analysed, and validated with real-time simulations of interaction tasks with different environments.

Acknowledgements

First, I would like to thank my daily supervisor Hanieh Esmaeeli for all the support she provided during the last 14 months. She never failed to dive into the details, and find the time for deep discussions. Her detailed feedback and notes were crucial throughout this thesis, on both an academic and a human level, especially in the stressful times of COVID-19. Also, I would like to express my gratitude to my second supervisor, Davide Bicego. I have always enjoyed and appreciated our progress meetings, fruitful discussions, and your detailed feedback.

I am also thankful for Antonio Franchi, for the opportunity to work under his supervision. Antonio's work and contributions were a major inspiration for me to work in this challenging field, and his insights and feedback were invaluable for this work. I have learned a lot from every interaction we had. Finally, I would like to thank Abeje Mersha for the support and feedback that contributed greatly to my research.

On a personal note, I would like to that my father and mother, Nizar and Maysaa, my sister and brother, Raghad and Yaman, for the never-ending love and support, for the trust you always placed in me, and for the freedom you gave me to think and choose. I am forever grateful. I would also like to thank my extended family, especially my grandfather and grandmother, my uncles Mohammad and Saaid and their families, my aunt Jumana, and all my friends around the world who shared this journey with me, especially, Alhassan, Ali, Hamza, Huda, Mhd Shadi, Mustafa, Nawwar, Omar, Rajaa. Thank you all.

Ayham

Contents

List of Figures

List of Tables

List of Acronyms

APhI Aerial Physical Interaction.

AR Aerial Robot.

CoG Center of Gravity.

CoM Center of Mass.

DoF Degree of Freedom.

ESC Electronic Speed Controller.

HFPC Hybrid Force/Position Control.

IMU Inertial Measurement Unit.

MPC Model Predective Control.

MRAV Multi-Rotor Aerial Vehicle.

NDT Non-Destructive Testing.

NLP NonLinear Programming.

NMPC Nonlinear Model Predective Control.

OCP Optimal Control Problem.

QP Quadratic Programming.

ROS Robot Operating System.

RTI Real Time Iteration.

SDF Simulation Description Format.

VTOL Vertical Takeoff and Landing.

1 Introduction

Multi-Rotor Aerial Vehicles (MRAVs) have been reliably deployed in many fields such as aerial photography, precision farming, search and rescue assistance, and civil infrastructure inspection [1]. These applications require almost no physical interaction with the MRAV surroundings, while the trajectory planner and collision avoidance algorithms try to keep the MRAV away from any physical collision with the surroundings to maintain its stability and safety.

However, the spatial freedom, unlimited work-space, and Vertical Takeoff and Landing (VTOL) capabilities of MRAVs are very tempting characteristics to utilize in tasks that require physical interaction with the environment, such as maintenance, contact-based inspection, and objects manipulation in remote or limited-access areas [2]. These applications require physical interaction for a sustained period of time, during which the MRAV is exchanging energy with the environment, and its motion is constrained in one or more of its Degrees of Freedom (DoFs) in a near-hovering state.

Due to the challenges associated with Aerial Robots (ARs), such as high nonlinearity; inherent instability; aerodynamic effects; under-actuation; and limited actuators capabilities, Aerial Physical Interaction (APhI) has been one of the main challenges in the field of ARs. Numerous methods and approaches have been proposed to tackle this problem in terms of the platform design, and also control system design.

As far as the platform design is concerned, standard MRAV platforms are usually actuated with fixed collinear propellers, this design choice limits the platform thrust generation to only one direction, which makes these platforms *under-actuated*, because these vehicles cannot follow an arbitrary trajectory in 3D space without changing their orientation. To compensate for the unactuated DoFs in the interaction tasks, the under-actuated MRAV can be equipped with an n-DoFs manipulator which will provide more dexterous manipulation. However, it also increases the UAV inertia, adds more complexity to the system [3], and the Center of Gravity (CoG) of the entire system will keep shifting following the manipulator motion [4].

The standard MRAV design can also be modified, where the rotors can be tilted, such that their thrust generation direction is not collinear anymore, and the tilt angles can be chosen to allow for thrust generation in the 3D space. This means that the vehicle does not have to change its orientation in order to follow an arbitrary 3D trajectory. These platforms are called fully-actuated MRAVs or Multi-Directional Thrust MRAVs, and there has been an increasing interest in these designs to overcome the limitations of the standard under-actuated platforms [5].

A passive tool, such as a fixed end-effector [6], or a partially-passive tool, such as a gripper, can be attached to a fully-actuated MRAV to serve as an end-effector in APhI tasks. These attached tools do not add any significant dynamic behaviour to the system, and they keep the mechanical structure simple and light-weight [7]. This design can be perceived as a flying end-effector [8] that can control the full interaction wrench in 6D and its motion is not constrained by any coupled DOFs. Such MRAVs that are equipped with an end-effector will be referred to as ARs in this thesis.

APhI control has also been a major challenge that has been tackled in many contributions. In the literature, two main classes of controllers can be noticed, hybrid position/force control; and impedance/admittance control [9]. In hybrid position/force control, the controller tracks references for the position and the interaction force separately in two feedback loops, such that the position is controlled in the unconstrained DoFs, and the force is controlled in the constrained DoFs, as in [10; 11; 12].

On the contrary, the impedance/admittance control does not directly control neither the position nor the force, but rather the dynamical behaviour of the system during both free-motion and interaction, as in [13; 14]. Therefore, these methods are usually referred to as indirect interaction control methods.

1.1 Problem definition

The aforementioned control approaches have their limitations and drawbacks. One of the main limitations of these control approaches is their reactive nature, where the control action at any time step, is a direct reaction of the calculated error at that time step, without optimizing the control input for the future time horizon.

Another limitation of these control approaches, is that there is no trivial method to include the physical constraints of the system in the controller without introducing a saturation action, which breaks the continuity of the control loop and can destabilize the system. In other words, the reactive control action is not optimized around the constraints of the system, whether they are physical limitations, safety, or operational constraints. Additionally, these reactive controllers do not exploit any prior knowledge of the system to predict its future evolution and optimize the control action based on those predictions.

Recently, there has been an increasing interest in using optimal/predictive control in MRAVs & APhI control mainly because of its ability to handle equality and inequality constraints. For instance, in [11], a quadratic programming problem is designed to optimally control an AR performing APhI tasks with energy and actuators constraints. One of the most prominent model-based optimization control approaches is Model Predective Control (MPC). MPC superiority over other model-based optimization control techniques is due to its ability to handle modelling uncertainty and correct for them using state-feedback, which also makes it very responsive to disturbances as it benefits from repeated optimization in a receding horizon fashion [15].

MPC has first been deployed for process control in the petrochemical industry in the 1980s where it gained its popularity after successfully replacing classical controllers in that field, but its applications were limited to systems and processes with slow dynamics. Recent advancements in the computational power of embedded computers, together with recent developments of more efficient optimization algorithms [16; 17], which can satisfy soft real-time constraints, has opened the door for exploiting the capabilities of MPC in time-critical applications with fast dynamic systems, such as robotics, and aerial robotics fields.

MPC is suitable for ARs applications because the system constraints are included in the controller design, and MPC optimizes the control action to satisfy these constraints. These constraints can be on the system states, control inputs, or other general constraints, and they can represent physical constraints, e.g. actuators limits, or safety/operational constraints, e.g. obstacle avoidance, and limited workspace. Recent publications by [18] and [19] have explored the use of MPC in MRAVs trajectory tracking control. However, instead of linearizing the nonlinear dynamic prediction model of the MRAV so that it can be used in the MPC, the full nonlinear model is used in the MPC, which is known as Nonlinear Model Predective Control (NMPC). The interested reader is referred to [20] for a survey of the design and applications of MPC for MRAVs.

1.1.1 Related work

In terms of physical interaction control using **(N)**MPC, three categories can be found in the literature, direct pose/force control, where the Optimal Control Problem (OCP) deals with both pose and force tracking; indirect interaction control, such that the OCP includes certain impedance/admittance requirements; and finally cascaded control, where an outer-loop is re-

Table 1.1: Overview of the contributions of this thesis compared to the state of the art.

The key of **(N)**MPC role: **P**=Pose tracking, **P/F**=Pose & Force tracking, **I**=Impedance control, **A**=Admittance control.

Paper	(N)MPC role	AR	Cascaded control	Force tracking	Compliant behaviour	Experimental validation
Peric et al. 2021 [21]	**P/F**	✓	✗	✓	✗	✓
Darivianakis et al. 2014 [22]	**P/F**	✓	✗	✓	✗	✓
Tzoumanikas et al. 2020 [23]	**P/F**	✓	✗	✓	✗	✓
Matschek et al. 2020 [24]	**P/F**	✗	✗	✓	✗	✓
Bednarczyk et al. [25]	**I**	✗	✗	✗	✓	✓
Kazim et al. 2018 [26]	**P**	✗	✗	✓	✓	✗
Pankert and Hutte 2020 [27]	**P**	✗	✓	✓	✓	✓
Wahrburg and Listmann 2016 [28]	**A**	✗	✓	✗	✓	✗
Contributions of this thesis						
NMPC cascaded control	**P**	✓	✓	✗	✓	✗
NMPC impedance control	**I**	✓	✗	✗	✓	✗
NMPC hybrid control	**P/F**	✓	✗	✓	✗	✗

sponsible for interaction control, and the inner-loop is a trajectory tracking controller, one of those controllers can be an **(N)**MPC-based controller.

Direct pose/force control with (N)MPC

An explicit MPC is proposed in [22] for hybrid pose/force control, where the capabilities of the controller are illustrated in an aerial writing task on a smooth flat surface with a quadrotor. In [23] accurate aerial writing was demonstrated using a hexarotor that is equipped with a parallel arm, and controlled by a hybrid force and position NMPC. A recent publication [21] proposed an NMPC to directly control force and pose of an AR during push-and-slide tasks. The control approach uses a smooth switching strategy between 3 different modes of control, Free-flight, static, and dynamic interaction.

Indirect interaction control with (N)MPC

Another approach that can be noticed in the literature is the use of MPC to impose desired impedance behaviour. In [25] an MPC controller is designed to behave as static state feedback impedance control in the absence of constraints, but when defining constraints, the MPC's objective function is defined to drive the system with the specified impedance properties while respecting the defined constraints. The controller design was verified with interaction tasks using a manipulator. On the contrary, [26] proposed a different approach where admittance control is combined with a nonlinear model predictive path following control in one OCP using the admittance dynamics outputs as virtual states. These virtual states will modify the reference trajectory to achieve the desired interaction behaviour. The proposed approach was verified with simulations of a writing task using a lightweight robotic arm.

Cascaded control with (N)MPC

For a mobile robot that is equipped with a manipulator, [27] proposed a cascaded structure with an admittance controller in the outer-loop and a whole-body MPC controller for pose tracking in the inner-loop. The defined MPC tracks task-space trajectories of the manipulator's end-effector, and at the same time complying with physical and obstacle avoidance constraints. On the other hand, [28] designed an MPC-based admittance controller as the outer-loop interaction controller, while the inner-loop controller is a classic velocity control at the joints level of the manipulator. The MPC-based admittance controller mimics the classical admittance control scheme in terms of its inputs and outputs, but thanks to the imposed constraints, the interaction forces can be constrained to certain bounds and continuous contact with the environment is guaranteed.

1.1.2 Thesis contributions

It can be noticed that the aerial robotics contributions in the literature are limited to the direct control approach, while other control architectures remain unexplored. In this thesis, three different approaches for APhI control using NMPC are proposed, implemented, and validated through real-time simulations, an overview of the contributions of this thesis compared to the state-of-the-art is presented in Table 1.1. The first proposed controller is a cascaded controller that uses NMPC for motion tracking in the inner-loop, while the interaction is controlled by the outer-loop admittance controller. The second controller is an NMPC-based impedance controller that is designated to track a desired impedance behaviour and indirectly control the APhI. Finally, the third controller is an NMPC hybrid controller that controls both the position and the interaction force during physical interaction.

1.1.3 Research questions

The research questions that are addressed in this thesis are:

RQ1 How to exploit NMPC capabilities for APhI control?

RQ2 How to choose an NMPC-based control approach for a specific task?

RQ3 Can NMPC be combined with Admittance control to create a "Constrained Admittance Control"?

1.2 Report layout

The rest of the thesis is organized as follows:

Chapter 2 will present an introduction to the theoretical background of this work. Where the modelling of the AR is discussed together with the actuation principles and limitations. Also, an introduction to direct and indirect interaction control approaches is presented.

Chapter 3 will build on the literature review of Chapter 1, and the theoretical background of Chapter 2 to propose NMPC-based APhI control approaches. The control architecture of the three controller is designed, and the respective OCP is proposed.

Chapter 4 discusses the practical implementation of the proposed controllers in MATLAB/Simulink using MATMPC tool, and the NMPC parameters that affect the feasibility of the proposed controllers.

Chapter 5 presents the validation of the proposed controllers through a simulations campaign in Gazebo simulation environment. The simulation results are presented and discussed.

Chapter 6 will conclude this thesis by summarizing the conclusions, reviewing the research questions, and proposing future extensions of this work.

2 Theoretical background

In this chapter, a brief review of the theoretical background of this thesis is presented. First, the mathematical modeling of the AR is presented in Section 2.1, then Section 2.2 introduces direct force control methods, and in Section 2.3 the basic concepts of indirect force control are discussed. Then, Section 2.4 will present the contact forces model, and finally in Section 2.5 the basic building blocks of NMPC controllers are introduced.

Notation. A consistent notation will be used throughout this thesis, where:

- Vectors and matrices will be denoted in bold font, with lower and upper cases, respectively.
- Upper case subscripts of vectors refer to the reference frame that the vector is expressed in, while Lower case subscripts are descriptive subscripts of the vector. In $\mathbf{p}_{W,r}$ for example, W refers to the world frame, and r means that this is *reference* vector.
- The rotation matrix $\mathbf{R}^{\star}_{\bullet}$ describes the orientation of frame $\bullet$ w.r.t $\star$. When $\star$ is not specified, the world frame is implied.
- The operator $\bullet^{\top}$ will denote the transpose operation of the matrix $\bullet$.
- $\mathbf{0}_{m\times n}$ and $\mathbf{1}_{m\times n}$ denote the matrices with all its elements equal to 0 and 1, respectively, and a size of m rows and n columns.
- The matrix $\mathbf{I}_n$ denotes the identity matrix of size $(n \times n)$
- $\mathbb{R}^{n\times n}_{>0}$ denotes the set of positive-definite real square matrices.
- The vectors $\mathbf{e}_1 = [1,0,0]^{\top}, \mathbf{e}_2 = [0,1,0]^{\top}, \mathbf{e}_3 = [0,0,1]^{\top} \in \mathbb{R}^3$
- SO(3) refers to the special orthogonal group, which represents 3D rotations. While $\mathfrak{so}(3)$ refers to the Lie-algebra of SO(3)
- The operator $\bullet_{\times} \in \mathfrak{so}(3)$ denotes the skew symmetric matrix of the vector $\bullet \in \mathbb{R}^3$
- The operator $\bullet^{\vee}$ is the inverse of $\bullet_{\times}$ and it denotes the $\mathbb{R}^3$ vector of any skew symmetric matrix $\in \mathfrak{so}(3)$.

2.1 Modeling of the Aerial Robot

Different modeling formalisms can be used to model the mechanical system of the MRAVs such as the Newton–Euler, and the Lagrangian formalisms. In this thesis, the Newton–Euler formalism will be used, mainly because it is susceptible to the development of efficient recursive models [29], which will be used to solve the OCP in the MPC.

2.1.1 Aerial Robot model

As shown in Fig. 2.1 the inertial world frame is defined as $\mathcal{F}_W = O_W\{\mathbf{x}_W, \mathbf{y}_W, \mathbf{z}_W\}$, while the frame $\mathcal{F}_B = O_B\{\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B\}$ is the body frame, which is attached to the Center of Mass (CoM) of the AR and aligned with the rigid body's principal inertia axes, i.e. the inertia matrix $\mathbf{J} \in \mathbb{R}^{3\times 3}_{>0}$ is a diagonal matrix. The frame $\mathcal{F}_E = O_E\{\mathbf{x}_E, \mathbf{y}_E, \mathbf{z}_E\}$ is rigidly attached to the end-effector, while the frame $\mathcal{F}_S = O_S\{\mathbf{x}_S, \mathbf{y}_S, \mathbf{z}_S\}$ is the surface frame which is attached to the contact point of the contact surface. Finally, the frame $\mathcal{F}_{A_i} = O_{A_i}\{\mathbf{x}_{A_i}, \mathbf{y}_{A_i}, \mathbf{z}_{A_i}\}$ is associated with the i-th actuator, where O_{A_i} coincides with the thrust generation point, and $\mathbf{z}_{A_i}$ is aligned with the thrust generation direction of the i-th actuator, which is assumed to be constant in this thesis. An overview of the main symbols and notations is presented in Table 2.1.

Table 2.1: Overview of the main symbols of this thesis.

Definition	Symbol	Set
Inertial world frame with origin O_W and orthogonal axes $\{\mathbf{x}_W, \mathbf{y}_W, \mathbf{z}_W\}$	$\mathcal{F}_W$	
Aerial robot body frame with origin O_B and orthogonal axes $\{\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B\}$	$\mathcal{F}_B$	
End-effector frame with origin O_E and orthogonal axes $\{\mathbf{x}_E, \mathbf{y}_E, \mathbf{z}_E\}$	$\mathcal{F}_E$	
Surface frame with origin O_S and orthogonal axes $\{\mathbf{x}_S, \mathbf{y}_S, \mathbf{z}_S\}$	$\mathcal{F}_S$	
i-th actuator frame with origin O_{A_i} and orthogonal axes $\{\mathbf{x}_{A_i}, \mathbf{y}_{A_i}, \mathbf{z}_{A_i}\}$	$\mathcal{F}_{A_i}$	
Position, velocity, acceleration of $\mathcal{F}_E$ w.r.t $\mathcal{F}_W$	$\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}}$	$\mathbb{R}^3$
Euler angles representing the orientation of $\mathcal{F}_B$ w.r.t $\mathcal{F}_W$	$\boldsymbol{\eta}$	$\mathbb{R}^3$
Rotation matrix of $\mathcal{F}_B$ w.r.t $\mathcal{F}_W$	$\mathbf{R}_B$	$\mathrm{SO}(3), \mathbb{R}^{3\times3}$
Rotation matrix of $\mathcal{F}_S$ w.r.t $\mathcal{F}_W$	$\mathbf{R}_S$	$\mathrm{SO}(3), \mathbb{R}^{3\times3}$
Angular velocity of $\mathcal{F}_B$ w.r.t $\mathcal{F}_W$ expressed in $\mathcal{F}_B$	$\boldsymbol{\omega}$	$\mathbb{R}^3$
Angular acceleration of $\mathcal{F}_B$ w.r.t $\mathcal{F}_W$ expressed in $\mathcal{F}_B$	$\dot{\boldsymbol{\omega}}$	$\mathbb{R}^3$
Actuators thrusts expressed in $\mathcal{F}_{A_i}$	$\boldsymbol{\gamma}$	$\mathbb{R}^6$
Actuators generated forces expressed in $\mathcal{F}_B$	$\mathbf{f}_\mathrm{a}$	$\mathbb{R}^3$
Actuators generated torques expressed in $\mathcal{F}_B$	$\boldsymbol{\tau}_\mathrm{a}$	$\mathbb{R}^3$
Contact force applied to the aerial robot expressed in $\mathcal{F}_S$	$\mathbf{f}_\mathrm{c}$	$\mathbb{R}^3$
Mass of the aerial robot	m	$\mathbb{R}_{>0}$
Inertia matrix of the aerial robot w.r.t. to $\mathcal{F}_B$	$\mathbf{J}$	$\mathbb{R}^{3\times3}_{>0}$
Wrench map which maps $\boldsymbol{\gamma}$ to $\mathbf{f}_\mathrm{a}$ and $\boldsymbol{\tau}_\mathrm{a}$	$\mathbf{G}$	$\mathbb{R}^{6\times6}$
Position of the end-effector frame w.r.t the body frame	$\mathbf{o}_{\mathrm{B,E}}$	$\mathbb{R}^3$
Position of the i-th actuator frame w.r.t the body frame	$\mathbf{o}_{\mathrm{B,A_i}}$	$\mathbb{R}^3$

In APhI, it is convenient to consider the linear dynamics of the end-effector instead of the AR body, and hence the linear dynamics will be derived for the end-effector frame w.r.t the world frame. Therefore, $\mathbf{p}, \dot{\mathbf{p}}, \ddot{\mathbf{p}} \in \mathbb{R}^3$ will represent the position, velocity, and acceleration of $\mathcal{F}_E$ w.r.t $\mathcal{F}_W$. On the other hand, the orientation of the AR can be expressed by the rotation matrix $\mathbf{R}_B^W \in \mathrm{SO}(3)$, or $\mathbf{R}_B$, which denotes the orientation of $\mathcal{F}_B$ w.r.t $\mathcal{F}_W$, while $\boldsymbol{\omega} \in \mathfrak{so}(3)$ represents the angular velocity of $\mathcal{F}_B$ w.r.t $\mathcal{F}_W$ expressed in $\mathcal{F}_B$, and finally the angular acceleration is $\dot{\boldsymbol{\omega}} \in \mathbb{R}^3$.

The evolution of the rotation matrix, that represents the AR orientation, w.r.t the angular velocity is expressed by:

$$\dot{\mathbf{R}}_B = \mathbf{R}_B [\boldsymbol{\omega}]_\times \tag{2.1}$$

The translational dynamics will be expressed in the inertial world frame, while the rotational dynamics are expressed in the non-inertial body frame. This choice of reference frame is a common practice when modelling an aerial vehicle and it has several advantages such as: the inertia matrix $\mathbf{J}$ being constant and not dependent on the vehicle orientation; and relatively simplified equations.

Finally, for an AR with a mass of $m \in \mathbb{R}_{>0}$ and an inertia matrix of $\mathbf{J}$, the equations of motion are:

$$\begin{aligned}
\underbrace{m\mathbf{I}_3\ddot{\mathbf{p}}}_{Inertia} &= \underbrace{-mg\mathbf{e}_3}_{Gravity} + \underbrace{\mathbf{R}_B\mathbf{f}_\mathrm{a}}_{Actuators} + \underbrace{\mathbf{R}_S\mathbf{f}_\mathrm{c}}_{Contact} + \underbrace{\mathbf{f}_{W,\mathrm{ext}}}_{External} + \underbrace{m\mathbf{I}_3\mathbf{R}_B\left(\dot{\boldsymbol{\omega}} \times \mathbf{o}_{B,E} + [\boldsymbol{\omega}]_\times^2 \mathbf{o}_{B,E}\right)}_{Fictitious} \\
\underbrace{\mathbf{J}\dot{\boldsymbol{\omega}}}_{Inertia} &= \underbrace{-\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}}_{Coriolis} + \underbrace{\boldsymbol{\tau}_\mathrm{a}}_{Actuators} + \underbrace{\mathbf{o}_{B,E} \times \left(\mathbf{R}_B^\top \mathbf{R}_S \mathbf{f}_\mathrm{c}\right)}_{Contact} + \underbrace{\boldsymbol{\tau}_{B,\mathrm{ext}}}_{External}
\end{aligned} \tag{2.2}$$

where g is the gravitational acceleration, $\mathbf{f}_\mathrm{a}, \boldsymbol{\tau}_\mathrm{a}$ are the forces and torques generated by the actuators (which will be discussed extensively in Section 2.1.3) expressed in the body frame, and

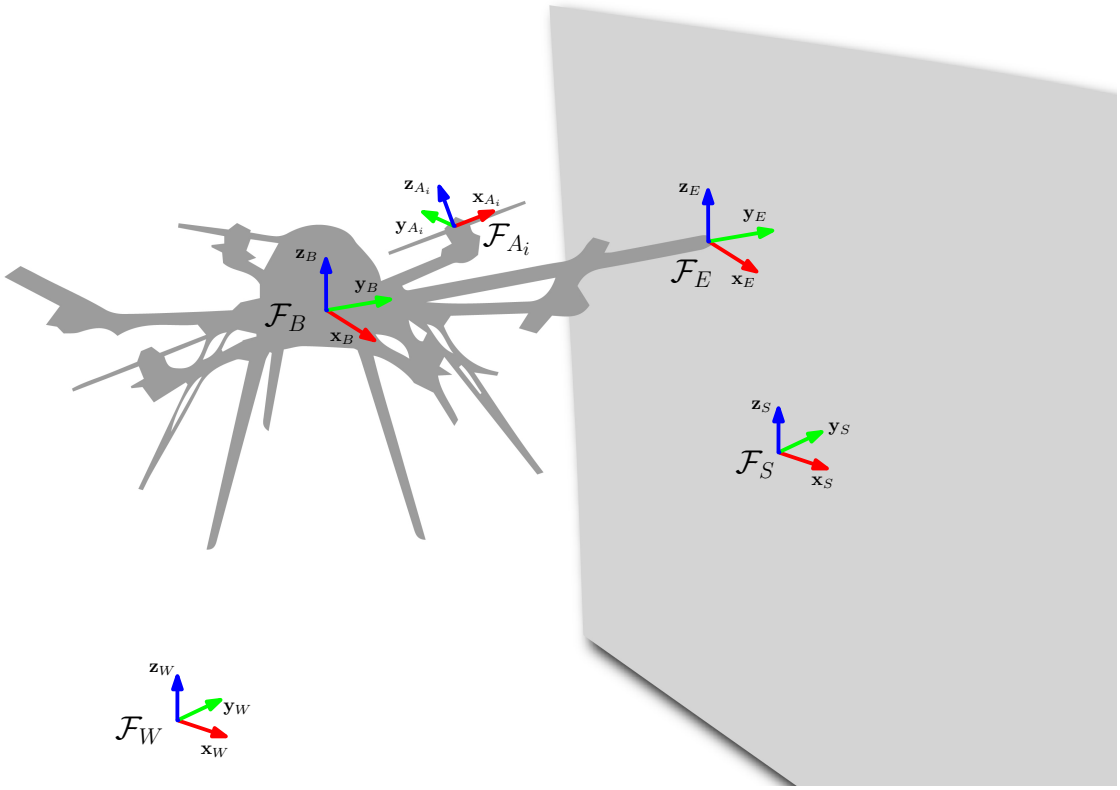


Figure 2.1: Visual representation of an aerial robot with the reference frames and the contact surface.

$\mathbf{f}_c$ is the contact force applied to the end-effector and expressed in the surface frame. Also, $\mathbf{f}_{W,\text{ext}}, \boldsymbol{\tau}_{B,\text{ext}}$ are the external force and torque applied to the AR and expressed in the world and body frames, respectively. This external wrench can be due to external disturbances such as wind, or internal disturbances such as unmodeled dynamics and the aerodynamic effects of the actuators.

2.1.2 Orientation representation

Orientations and rotations of a rigid body in 3D space can be represented in different forms, such as: Euler angles, rotation matrix, and unit quaternions. In this section, the relationship between these three representations and the angular velocity $\boldsymbol{\omega}$ is presented together with the error definition for each representation, the section is concluded with a brief comparison and selection criteria.

Rotation matrix

The evolution of the rotation matrix w.r.t the angular velocity has been presented in Eq. (2.1). The error between two rotation matrices $\mathbf{R}_1$ and $\mathbf{R}_2$, i.e. $\mathbf{e}_R \in \mathbb{R}^3$, is defined as:

$$\mathbf{e}_R = \frac{1}{2}(\mathbf{R}_1^\top \mathbf{R}_2 - \mathbf{R}_2^\top \mathbf{R}_1)^\vee \tag{2.3}$$

Euler angles

For a set of Euler angles $\boldsymbol{\eta} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^\top$, the angular velocity can be expressed as a function of Euler angles rate of change $\dot{\boldsymbol{\eta}}$, or *Euler rates*, such that:

$$\boldsymbol{\omega} = \mathbf{T}\dot{\boldsymbol{\eta}} \tag{2.4}$$

where $\mathbf{T}$ depends on the order of rotation of the three Euler angles. For the famous *yaw-pitch-roll* order:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & s_\phi c_\theta \\ 0 & -s_\phi & c_\phi c_\theta \end{bmatrix} \tag{2.5}$$

The error between two sets of Euler angles, $\boldsymbol{\eta}_1$ and $\boldsymbol{\eta}_2$, can be derived in two definitions. First, the direct difference between the two sets:

$$\boldsymbol{\eta}_1 - \boldsymbol{\eta}_2 = \begin{bmatrix} \phi_1 - \phi_2 \\ \theta_1 - \theta_2 \\ \psi_1 - \phi_2 \end{bmatrix} \tag{2.6}$$

This definition ignores the cyclic nature of rotation such that an angle of 2π is equal to the angle of 0. The second definition of the error solves this issue such that:

$$\boldsymbol{\eta}_1 \ominus \boldsymbol{\eta}_2 = \begin{bmatrix} \min\left(|\phi_1 - \phi_2|,\ 2\pi - |\phi_1 - \phi_2|\right) \\ \min\left(|\theta_1 - \theta_2|,\ 2\pi - |\theta_1 - \theta_2|\right) \\ \min\left(|\psi_1 - \psi_2|,\ 2\pi - |\psi_1 - \psi_2|\right) \end{bmatrix} \tag{2.7}$$

where $\min(a,\ b)$ is the minimum function that outputs the minumum value between a and b.

Unit quaternions

For a unit quaternion $\mathbf{q} = \begin{bmatrix} q_w & q_x & q_y & q_z \end{bmatrix}^\top$, the time derivative of the quaternion $\dot{\mathbf{q}}$ w.r.t the angular velocity can be written as:

$$\dot{\mathbf{q}} = \frac{1}{2} \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \otimes \mathbf{q} \tag{2.8}$$

where the $\otimes$ operator denotes the Hamilton product of two quaternions (Cf. Appendix B.1).

On the other hand, the error between two quaternions, $\mathbf{e}_q \in \mathbb{R}^4$, can be defined by the geodesic distance on the manifold of unit quaternions [30] such that:

$$\mathbf{e}_q = \left|\left|\log(\mathbf{q}_1 \otimes \mathbf{q}_2^*)\right|\right| \tag{2.9}$$

where the operator $*$ denotes the conjugate quaternion (Cf. Appendix B.1).

Another approach to calculate the error between two quaternions is proposed in [31], where the error $\mathbf{e}_{q_3}$ is approximated in $\mathbb{R}^3$ such that:

$$\mathbf{e}_{q_3} = \mathrm{Im}(\mathbf{q}_1 \otimes \mathbf{q}_2^{-1}) \tag{2.10}$$

where Im denotes the imagenry part of the quaternion, i.e. $\mathrm{Im}(\mathbf{q}) = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}^\top$, and $\mathbf{q}^{-1}$ is the inverse of $\mathbf{q}$ (Cf. Appendix B.1).

The most important advantage of using Euler angles is that it only uses three parameters to represent the 3D rotation. However, it is well-known that the Euler-angles representation suffers from singularities, such as when $\theta = \pi/2$ in the *yaw-pitch-roll* rotation order. But also, the Euler angles representation of 3D rotations is not unique. For example, when using the *yaw-pitch-roll* rotation order, the Euler angles $\boldsymbol{\eta}_1 = \begin{bmatrix} \pi & \pi & 0 \end{bmatrix}^\top$ and $\boldsymbol{\eta}_2 = \begin{bmatrix} 0 & 0 & \pi \end{bmatrix}^\top$ represent the same rotation, even though they are different, i.e. their error is not zero. One solution to avoid these problems of Euler angles is to constrain the angles to certain boundaries where Euler angles form a perfect chart on SO(3).

Rotation matrices on the other hand are in SO(3), and naturally, they do not suffer any singularities. However, a rotation matrix has 9 parameters to represent the 3D rotation, which is not

ideal when implementing a recursive algorithm on resources-limited computers. Unit quaternions need only 4 parameters to describe the 3D rotation while covering all of SO(3) without any singularities. In [30], it is also found that unit quaternions are more computationally efficient when implemented digitally, as in the applications of ARs.

In this thesis, Euler angles, with the transformation matrix of Eq. (2.5), are used to represent the orientation regardless of their limitations and singularities, mainly because they are more intuitive to understand and interpret, and because their implementation is easier w.r.t the recursive dynamical model that will be used in the NMPC controllers. However, this choice will restrict the rotational motion of the AR so that the attitude does not approach the singularities of the Euler angles, but for the subset of APhI tasks that this thesis is interested in, these restrictions are reasonable.

2.1.3 MRAVs actuation

MRAVs are usually actuated by n rotors, which are propellers attached to motors. A rotor $i \in \{1,\dots,n\}$ can generate a *lift force*, i.e. thrust, denoted as γ_i, that is mainly dependent on the spinning velocity of rotor w_i and the propeller properties, such as geometry, while its direction is along $\mathbf{z}_{A_i}$ of the actuator frame $\mathscr{F}_{A_i}$. A by-product of this spinning is that the rotor will experience friction with the air, which will generate a *drag torque* around $\mathbf{z}_{A_i}$, the drag torque, denoted τ_{t_i}, is also dependent on the spinning velocity of the rotor and the propeller properties.

MRAVs are usually equipped with Brushless DC motors, which can rotate in the two directions, **CCW** and **CW**, but propellers are usually manufactured to optimally operate in one of the two directions, depending on the pitch angle of the propeller. Therefore, the standard designs of MRAVs are equipped with mono-directional rotors, which are assumed to be rotating in one direction only, and therefore, the direction of the thrust force and drag torque is constant, where the thrust direction is always along the positive $\mathbf{z}_{A_i}$, while the drag torque direction depends on the rotation direction.

A well-established and experimentally verified model [18] of the thrust generation is:

$$\gamma_i = c_{f_i} \mathrm{w}_i^2 \tag{2.11}$$

where $c_{f_i} \in \mathbb{R}_{>0}$ is the thrust coefficient of the i-th rotor, which is a propeller-dependent parameter that is affected by, among others, the propeller diameter, pitch angle, sweeping area, and it can be identified experimentally for each propeller type.

The total force contribution of the generated thrusts by all the actuators, expressed in the body frame, $\mathbf{f}_\mathrm{a}$ can be calculated as:

$$\mathbf{f}_\mathrm{a} = \mathbf{R}_B \sum_{i=1}^{n} \mathbf{R}_{A_i}^B \mathbf{e}_3 \gamma_i \tag{2.12}$$

where the individual thrusts are transformed from their actuator frame $\mathscr{F}_{A_i}$ to the body frame $\mathscr{F}_B$.

On the other hand, the total torque that is applied to the MRAV's CoM from the actuators ,and expressed in the body frame, $\boldsymbol{\tau}_\mathrm{a}$ is the sum of two main quantities, **1)** the thrust contribution to the torque due to the leverage arms $\boldsymbol{\tau}_\gamma$; **2)** the drag torque $\boldsymbol{\tau}_t$, such as:

$$\boldsymbol{\tau}_\mathrm{a} = \sum_{i=1}^{n} \boldsymbol{\tau}_{\gamma_i} + \boldsymbol{\tau}_{t_i} \tag{2.13}$$

First, the individual thrust contribution to the torque depends on the thrust intensity of the rotor γ_i, the position of the rotor w.r.t the CoM represented by $\mathbf{o}_{B,A_i}$, and the orientation of the

rotor w.r.t the body frame $\mathbf{R}_{A_i}^B$ such that:

$$\boldsymbol{\tau}_{\gamma_i} = \mathbf{o}_{B,A_i} \times \mathbf{R}_{A_i}^B \mathbf{e}_3 \gamma_i \tag{2.14}$$

Second, the drag torque of each individual motor is modelled as:

$$\tau_{t_i} = \beta_i c_{t_i} \gamma_i \tag{2.15}$$

where c_{t_i} is the thrust-to-drag coefficient because the drag torque τ_{t_i} is proportional to the generated thrust γ_i such as $|\frac{\tau_{t_i}}{\gamma_i}| = c_{t_i}$. The rotor parameter $\beta_i \in \{-1, 1\}$ indicates the drag torque direction around $\mathbf{z}_{A_i}$, such that its value is -1 if the rotor is spinning in **CCW** direction around $\mathbf{z}_{A_i}$, or $+1$ if it is spinning **CW**. This indicates that the drag torque direction is always opposite to the spinning direction, therefore, when the rotor is rotating **CCW**, the drag torque will be **CW** about $\mathbf{z}_{A_i}$, and vice versa.

Hence, the drag torque of each individual motor expressed in the body frame can be written as:

$$\boldsymbol{\tau}_{B,t_i} = \mathbf{R}_{A_i}^B \mathbf{e}_3 \beta_i c_{t_i} \gamma_i \tag{2.16}$$

Substituting Eqs. (2.14) and (2.16) into Eq. (2.13), and after refactoring:

$$\boldsymbol{\tau}_{\mathrm{a}} = \sum_{i=1}^{n} \left([\mathbf{o}_{B,A_i}]_\times + \beta_i c_{t_i} \mathbf{I}_3 \right) \mathbf{R}_{A_i}^B \mathbf{e}_3 \gamma_i \tag{2.17}$$

By combining Eqs. (2.12) and (2.17):

$$\begin{bmatrix} \mathbf{f}_{\mathrm{a}} \\ \boldsymbol{\tau}_{\mathrm{a}} \end{bmatrix} = \begin{bmatrix} \mathbf{G}_1 \\ \mathbf{G}_2 \end{bmatrix} \boldsymbol{\gamma} = \mathbf{G}\boldsymbol{\gamma} \tag{2.18}$$

where $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_n]^\top$, and $\mathbf{G} \in \mathbb{R}^{6\times n}$ is the *wrench map* matrix which maps the thrusts of the rotors from its $\mathbb{R}^n$ thrusts space to the body wrench space of $\mathbb{R}^6$, and its components $\mathbf{G}_1, \mathbf{G}_2$ represent the mapping from of the rotors' thrust to the 3D forces and torques, respectively, expressed in body frame. This mapping is defined by several platform/hardware dependent parameters, namely: the rotors position and orientation w.r.t the CoM; the propeller's thrust and thrust-to-drag coefficient; the rotor spinning direction. Therefore, the i-th column of $\mathbf{G}$ which corresponds to the contribution of the i-th rotor in the body wrench is defined as:

$$\mathbf{G}(:, i) = \begin{bmatrix} \mathbf{R}_{A_i}^B \mathbf{e}_3 \\ \left([\mathbf{o}_{B,A_i}]_\times + \beta_i c_{t_i} \mathbf{I}_3 \right) \mathbf{R}_{A_i}^B \mathbf{e}_3 \end{bmatrix} \tag{2.19}$$

Finally, the model in Eq. (2.2) can be re-written more explicitly, including the Euler angles dynamics, such as:

$$\begin{aligned} m\mathbf{I}_3 \ddot{\mathbf{p}} &= -mg\mathbf{e}_3 + \mathbf{R}_B \mathbf{G}_1 \boldsymbol{\gamma} + \mathbf{R}_S \mathbf{f}_{\mathrm{c}} + \mathbf{f}_{W,\mathrm{ext}} + m\mathbf{I}_3 \mathbf{R}_B \left(\dot{\boldsymbol{\omega}} \times \mathbf{o}_{B,E} + [\boldsymbol{\omega}]_\times^2 \mathbf{o}_{B,E} \right) \\ \mathbf{J}\dot{\boldsymbol{\omega}} &= -\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \mathbf{G}_2 \boldsymbol{\gamma} + \mathbf{o}_{B,E} \times \left(\mathbf{R}_B^\top \mathbf{R}_S \mathbf{f}_{\mathrm{c}} \right) + \boldsymbol{\tau}_{B,\mathrm{ext}} \\ \dot{\boldsymbol{\eta}} &= \mathbf{T}^{-1} \boldsymbol{\omega} \end{aligned} \tag{2.20}$$

2.1.4 MRAVs actuator limits

It is important to discuss the physical constraints of the system and to take them into account while designing the controllers. As far as the MRAVs are concerned, the actuators limits are the only physical constraints to the system. A standard actuator consists of an electric motor, typically a brushless DC motor, and attached to it is a propeller. Naturally, the electric motor rotational speed w has an upper limit $\overline{\mathrm{w}}$, that mainly depends on the maximum current that

the motor can draw, and thus an upper limit to the thrust $\overline{\gamma}$ that the actuator can generate. However, there is also a lower limit $\underline{w}$ to the motor's rotational speed, which is usually associated with the capabilities of the Electronic Speed Controller (ESC) that is controlling the motor rotational speed [32].

Additionally, the rotational acceleration of electric motors is also bounded, and therefore there will be an upper and lower limits to the time derivative of the rotational velocities, $\overline{\dot{w}}$ and $\underline{\dot{w}}$, leading to an upper and lower limits on the rate of the change of the generated thrust, $\overline{\dot{\gamma}}$ and $\underline{\dot{\gamma}}$. This was investigated thoroughly in [18], and it was found that these limits are functions of the rotational velocity, where the motor acceleration/deceleration limits are functions of the rotational velocity, $\overline{\dot{w}}(w)$ and $\underline{\dot{w}}(w)$, hence, these limits should be experimentally identified throughout the range of operation to have proper understanding of the real actuator limits.

2.2 Hybrid force/position control

When a robotic system is interacting with a rigid environment, its motion is typically constrained in one or more DoFs. These constraints mainly depend on the environment geometry, and they are called *natural constraints*. In a naturally constrained DoF, either the translation/rotation is constrained, where the environment obstructs the translation/rotation along/about this DoF, or the force/torque is constrained along/about that DoF, such that the environment does not allow for the application of force/moment along/about that DoF.

On the other hand, in the DoFs that are not naturally constrained, the robotic system can control either the motion or the force/torque in that DoF, and drive them to desired values. These desired values are called *artificial constraints*, because they do not depend on the nature of the environment's geometry, but rather depend on the specified task of the system. This means that, when the velocity/angular velocity is naturally constrained in a DoF, an artificial constraint can be imposed on the corresponding force/torque, and vice versa.

A simple example is depicted in Fig. 2.2, where an end-effector of a robot is approaching a rigid board to write on it. To simplify the description of the constraints, a local reference frame that is rigidly attached to the contact surface at the contact point between the end-effector and the surface, is defined such that $\mathscr{F}_S = O_S\{\mathbf{x}_S, \mathbf{y}_S, \mathbf{z}_S\}$. It can be noticed that the end-effector is free to move on the planar axes $\mathbf{y}_S, \mathbf{z}_S$, but its motion is constrained in the axis that is normal to the board surface $\mathbf{x}_S$, meaning that the environment poses the following natural constraints:

1. The translation along the normal axis ($\mathbf{y}_S$)
2. The forces along the planar axes ($\mathbf{x}_S, \mathbf{z}_S$)
3. The torques about the planar axes ($\mathbf{x}_S, \mathbf{z}_S$)
4. The torque about the normal axis ($\mathbf{y}_S$)

Whereas, the complementary artificial constraints that can be imposed are:

1. The force along the normal axis ($\mathbf{y}_S$)
2. The motion along the planar axes ($\mathbf{x}_S, \mathbf{z}_S$)
3. The rotation about the planar axes ($\mathbf{x}_S, \mathbf{z}_S$)
4. The rotation about the normal axis ($\mathbf{y}_S$)

Therefore, when the robotic system is interacting with a rigid environment, the system can be decoupled into, and treated as, two subsystems. The first subsystem represents the naturally constrained variables that the robot has no control over, and therefore can be excluded from control consideration. The second subsystem represents the artificially constrained DoFs,

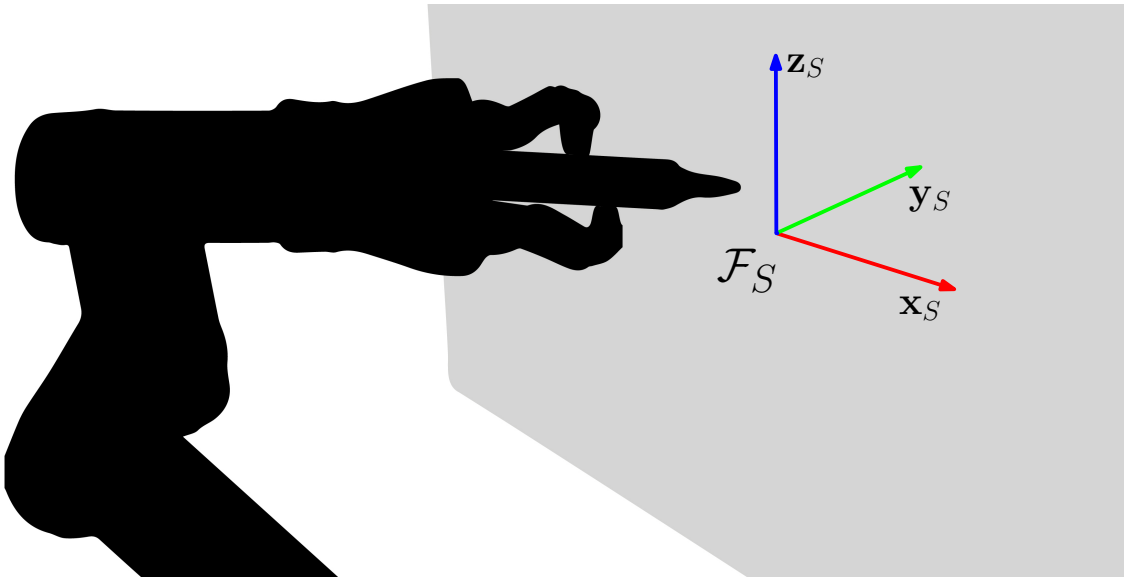


Figure 2.2: Robot approaching a rigid board to write on it, an example of hybrid pose/force control.

which the robot can directly control, and therefore, separate control actions can be designed to control those variables. This control paradigm is called, Hybrid Force/Position Control (HFPC), since the controlled variables will be a mixture of position/orientation and forces/torques variables. Position and force can be controlled separately with two different control actions and approaches.

For an AR, it is easier to express the constrained and unconstrained translational DoFs in the surface frame $\mathcal{F}_S$ that is attached to the contact surface. And since the interaction forces are applied on the end-effector, and the motion of the end-effector is constrained/unconstrained in certain directions, it is more convenient to derive the translational dynamics of the robot in the end-effector frame $\mathcal{F}_E$ w.r.t the world frame $\mathcal{F}_W$. Then, following [21], the dynamics of Eq. (2.2) can be rewritten as:

$$\ddot{\mathbf{p}}_W = \mathbf{S}_\kappa \Big(m^{-1} \big(\mathbf{R}_B \mathbf{f}_\mathrm{a} + \kappa \mathbf{R}_S \mathbf{f}_\mathrm{c} \big) - g\mathbf{e}_3 + \mathbf{R}_B \big(\dot{\boldsymbol{\omega}} \times \mathbf{o}_{B,E} + [\boldsymbol{\omega}]_\times^2 \mathbf{o}_{B,E} \big) \Big) \tag{2.21}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1} \Big(\boldsymbol{\tau}_a + \mathbf{o}_{B,E} \times \big(\kappa \mathbf{R}_B^\top \mathbf{R}_S \mathbf{f}_\mathrm{c} \big) - \boldsymbol{\omega} \times \big(\mathbf{J}\boldsymbol{\omega} \big) \Big) \tag{2.22}$$

where, $\mathbf{S}_\kappa$ is the selection matrix which selects the unconstrained translational dynamics and truncates the constrained translational dynamics, such that:

$$\mathbf{S}_\kappa = \mathbf{R}_S \mathrm{diag}(1,\ 1-\kappa,\ 1) \mathbf{R}_S{}^\top \tag{2.23}$$

which means that during interaction, the translational dynamics along the $\mathbf{y}_S$ axis is naturally constrained.

2.3 Indirect force control

The most prominent approach for indirect force control, and physical interaction control is the framework of impedance control. This framework was first introduced by Neville Hogan in his famous three-part paper [33]. In impedance control, the goal of the controller is neither to steer the system to the desired position, nor to apply a certain force to the environment, but rather to achieve a desired dynamical behaviour. To understand that, Hogan suggests, given that the environment has inertial objects, that the environment should be considered as a mechanical admittance, and therefore, the robotic system should be thought of as a mechanical impedance.

The desired dynamical behaviour is usually described as a second-order mechanical system, i.e. a spring-mass-damper system, which can be expressed in a one-dimensional case as:

$$\underbrace{m\ddot{p}}_{\substack{\text{Inertia}\\ \text{force}}} + \underbrace{d\dot{p}}_{\substack{\text{Damping}\\ \text{force}}} + \underbrace{kp}_{\substack{\text{Spring}\\ \text{force}}} = f \tag{2.24}$$

where f represents the combined forces applied to the mass, while p, $\dot{p}$, $\ddot{p}$ are the position, velocity, and acceleration of the mass, and m, d, k are the mass, damping coefficient, and spring coefficient, respectively. The three quantities on the left represent the three components of the forces that the system will exert if perturbed from its equilibrium point, $\ddot{p} = \dot{p} = p = 0$.

In terms of robotic systems control, the impedance control framework can be used in two different settings, called *impedance control* (sometimes referred to as force-based impedance control), and *admittance control* (position-based impedance control), which will be discussed in Sections 2.3.1 and 2.3.2, respectively.

2.3.1 Impedance control

In impedance control, the controller will behave as a spring-mass-damper system that is trying to reach its equilibrium point, i.e. $\ddot{p} = \dot{p} = p = 0$ in Eq. (2.24). For a nonzero equilibrium points, and in 3D, this can be re-written as:

$$\mathbf{M}(\ddot{\mathbf{p}}_r - \ddot{\mathbf{p}}) + \mathbf{D}(\dot{\mathbf{p}}_r - \dot{\mathbf{p}}) + \mathbf{K}(\mathbf{p}_r - \mathbf{p}) = \mathbf{f}_{\text{u}} \tag{2.25}$$

where $\mathbf{M}$, $\mathbf{D}$, $\mathbf{K} \in \mathbb{R}^{3\times3}_{>0}$ are the mass, damping, and stiffness diagonal matrices, $\ddot{\mathbf{p}}_r$, $\dot{\mathbf{p}}_r$, $\mathbf{p}_r \in \mathbb{R}^3$ are the reference acceleration, velocity, and position, $\ddot{\mathbf{p}}$, $\dot{\mathbf{p}}$, $\mathbf{p} \in \mathbb{R}^3$ are the acceleration, velocity, and position of the system, and finally $\mathbf{f}_{\text{u}} \in \mathbb{R}^3$ is the control force that the controller commands to steer the system to the reference states with the dynamical behaviour that is defined by $\mathbf{M}$, $\mathbf{D}$, $\mathbf{K}$, as shown in Fig. 2.3a.

This can be extended to the rotational DoFs, where the impedance can be defined on SO(3) to have a rotational spring, and rotational damper, and apparent inertia. Since this topic falls outside the scope of this thesis, the interested reader is referred to [13; 34] for more details.

2.3.2 Admittance control

Admittance control is, as the name suggests, the reciprocal of impedance control. In admittance control, the interaction forces are used to modify the reference signal as shown in Fig. 2.3b, where admittance control is used as an interaction controller in the *outer-loop*, with another controller in the *inner-loop* that is dedicated for motion control (pose tracking) as in [8; 35].

This type of *cascaded control* separates the motion control from the interaction control, which allows for robust trajectory tracking behaviour, that rejects disturbances better, in the absence of interaction, while a compliant behaviour can be achieved during interaction with the environment. The admittance control tries to achieve the desired dynamical behaviour during the interaction not by affecting the control action directly, but by modifying the reference signal, assuming that perfect reference tracking can be achieved by the inner-loop motion controller. Mathematically, this can be written as:

$$\mathbf{M}\,\delta\ddot{\mathbf{p}} + \mathbf{D}\,\delta\dot{\mathbf{p}} + \mathbf{K}\,\delta\mathbf{p} = \mathbf{f}_{W,\text{c}} \tag{2.26}$$

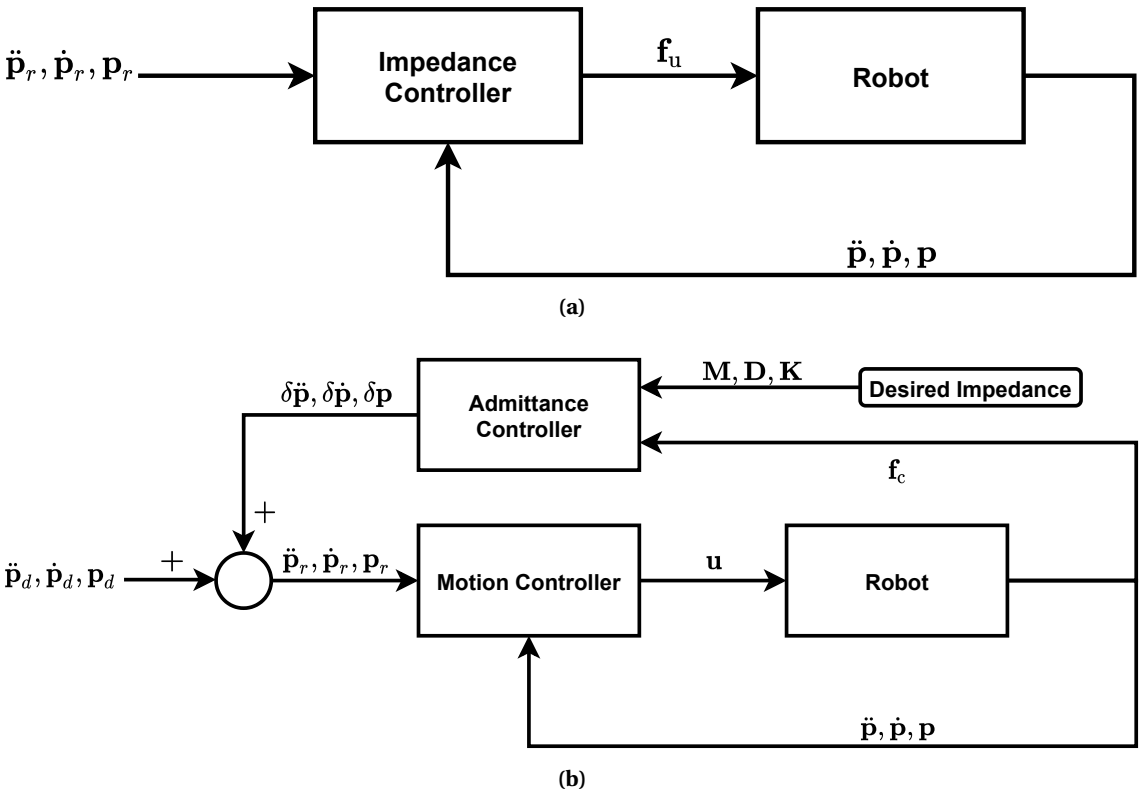


Figure 2.3: Indirect interaction control block diagrams. **(a)** Impedance control block diagram. **(b)** Admittance control block diagram.

where $\mathbf{f}_{W,c} \in \mathbb{R}^3$ is the interaction contact force that is applied to the system expressed in $\mathcal{F}_W$, and $\delta\ddot{\mathbf{p}}$, $\delta\dot{\mathbf{p}}$, $\delta\mathbf{p} \in \mathbb{R}^3$ are the acceleration, velocity, and position admittance modification, such as:

$$\begin{aligned} \ddot{\mathbf{p}}_r &= \ddot{\mathbf{p}}_d + \delta\ddot{\mathbf{p}} \\ \dot{\mathbf{p}}_r &= \dot{\mathbf{p}}_d + \delta\dot{\mathbf{p}} \\ \mathbf{p}_r &= \mathbf{p}_d + \delta\mathbf{p} \end{aligned} \tag{2.27}$$

The admittance controller will modify the desired trajectory to a reference trajectory that will achieve the desired dynamical behaviour defined by $\mathbf{M}$, $\mathbf{D}$, $\mathbf{K} \in \mathbb{R}^{3\times3}_{>0}$. For example, in a simple 1D interaction with an environment, the admittance controller will try to minimize $|\mathbf{f}_c|$, by moving the reference trajectory *away* from the environment. It is noted that admittance control can also be used for force tracking, such as [36], where Proportional-Integral (PI) controller is added to the right-hand side of Eq. (2.26) so that the admittance controller will modify the reference trajectory — moving it *towards* or *away* from the environment, to simultaneously achieve the desired dynamical behaviour, and track the desired interaction force.

In the previous two sub-sections, impedance and admittance control have been presented and explained mathematically for the 3D translational dynamics, however, the rotational dynamics can also be included in the impedance and admittance control framework, to achieve a desired dynamical behaviour in the rotational DoFs as well. The mathematical representation of this 6D impedance/admittance controller depends on the chosen representation of the three rotational DoFs — such as rotation matrix, unit quaternion, Euler angles, or others. For example, in [8], where the orientation is expressed using rotation matrices, the 6D admittance controller

is defined as:

$$\mathbf{M}_6 \begin{bmatrix} \ddot{\mathbf{p}}_d - \ddot{\mathbf{p}}_r \\ \dot{\boldsymbol{\omega}}_d - \dot{\boldsymbol{\omega}}_r \end{bmatrix} + \mathbf{D}_6 \begin{bmatrix} \dot{\mathbf{p}}_d - \dot{\mathbf{p}}_r \\ \boldsymbol{\omega}_d - \boldsymbol{\omega}_r \end{bmatrix} + \mathbf{K}_6 \begin{bmatrix} \mathbf{p}_d - \mathbf{p}_r \\ \frac{1}{2}(\mathbf{R}_d\mathbf{R}_r^\top - \mathbf{R}_r\mathbf{R}_d^\top)^\vee \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{W,\mathrm{c}} \\ \boldsymbol{\tau}_{B,\mathrm{c}} \end{bmatrix} \tag{2.28}$$

where $\mathbf{M}_6$, $\mathbf{D}_6$, $\mathbf{K}_6 \in \mathbb{R}^{6\times6}_{>0}$ are the mass, damping, and stiffness diagonal matrices, and the term $\frac{1}{2}(\mathbf{R}_d\mathbf{R}_r^\top - \mathbf{R}_r\mathbf{R}_d^\top)^\vee$ describes the orientation error, which can be different depending on the used representation of the rotation as discussed in Section 2.1.2. Including the rotational DoFs in the admittance control is a design choice that is dependent on the interaction task. For example, for tasks like peg-in-the-hole, it is desired to have a compliant behaviour from the rotational DoFs in order to adapt to the hole orientation uncertainties.

2.4 Contact force model

The contact force $\mathbf{f}_\mathrm{c}$ representing the environment reaction to the physical interaction can be modeled based on Newton's third law of motion. Assuming that the environment is a mechanically grounded rigid body, $\mathbf{f}_\mathrm{c}$ can be estimated based on the force generated by the actuators $\mathbf{f}_\mathrm{a}$, where the generated force will be applied to the environment in the constrained DOFs and therefore the environment reaction (in that constrained DoF) will be equal to the applied force and in the opposite direction. Hence the contact force is:

$$\mathbf{f}_\mathrm{c} = -\bar{\mathbf{S}}_\kappa \mathbf{R}_S^\top \mathbf{R}_B \mathbf{G}_1 \boldsymbol{\gamma} \tag{2.29}$$

where $\mathbf{G}_1$ is a map from the thrusts $\boldsymbol{\gamma}$ to the 3D forces acting on the body frame, as defined in Eq. (2.18). Then, its time derivative is:

$$\dot{\mathbf{f}}_\mathrm{c} = -\bar{\mathbf{S}}_\kappa \left(\mathbf{R}_S^\top \mathbf{R}_B [\boldsymbol{\omega}]_\times \mathbf{G}_1 \boldsymbol{\gamma} + \mathbf{R}_S^\top \mathbf{R}_B \mathbf{G}_1 \dot{\boldsymbol{\gamma}} \right) \tag{2.30}$$

Alternatively, following [21], the contact force can be modelled as:

$$\mathbf{f}_\mathrm{c} = \bar{\mathbf{S}}_\kappa \mathbf{R}_S^\top \left(m\ddot{\mathbf{p}}_r - mg\mathbf{e}_3 - \mathbf{R}_B \mathbf{f}_\mathrm{a} \right) \tag{2.31}$$

which means that the contact force can be calculated based on the residual force.

It is important to note that this contact model is only suitable for modelling the normal force of the interaction, and therefore, this model does not describe the friction forces, because friction forces arise due to motion, and by definition, the motion should be constrained for the model in Eq. (2.29) to be valid. Also, this model only captures the steady-state response of the environment, while the collision dynamics at the beginning of the interaction are not described by this model.

2.5 Nonlinear Model Predictive Control

A brief introduction to MPC is presented in Appendix A, however, this section will introduce the mutual building blocks for the NMPC-based controllers that will be proposed in the next chapter, to avoid repetition.

First, the NonLinear Programming (NLP) problem that will be used throughout the next chapter is defined as:

$$\min_{\substack{\mathbf{x}_0 \dots \mathbf{x}_N \\ \mathbf{u}_0 \dots \mathbf{u}_{N-1}}} \sum_{k=0}^{N} \left\| \mathbf{J}(\mathbf{r}_k, \mathbf{x}_k) \right\|^2_{\mathbf{Q}_k} \tag{2.32}$$

$$\text{s.t.} \quad \mathbf{x}_0 = \hat{\mathbf{x}}(t) \tag{2.33}$$

$$\mathbf{x}_{k+1} = \boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k), \quad k\in\{0,N-1\} \tag{2.34}$$

$$\underline{\boldsymbol{\gamma}} \le \boldsymbol{\gamma}_k \le \overline{\boldsymbol{\gamma}}, \quad k\in\{0,N\} \tag{2.35}$$

$$\underline{\dot{\boldsymbol{\gamma}}}_k(\boldsymbol{\gamma}_k) \le \mathbf{u}_k \le \overline{\dot{\boldsymbol{\gamma}}}_k(\boldsymbol{\gamma}_k), \quad k\in\{0,N-1\} \tag{2.36}$$

where $\mathbf{J}(\mathbf{r}_k, \mathbf{x}_k) \in \mathbb{R}_{>0}^{n_y}$ is the objective function that describes the controller tasks, and $\mathbf{Q}_k \in \mathbb{R}_{>0}^{n_y \times n_y}$ is the weights matrix, and $n_y \in \mathbb{N}_{>0}$ is the number of the outputs in the objective function $\mathbf{J}(\mathbf{r}_k, \mathbf{x}_k)$. The subscript k denotes the time step kT such that $\mathbf{x}_k = \mathbf{x}(kT)$ where T is the sampling time. The objective function, together with the discrete states vector $\mathbf{x}_k \in \mathbb{R}^{n_x}$ and references vector $\mathbf{r}_k \in \mathbb{R}^{n_r}$ will be defined for each control approach separately in the next chapter depending on the control objective and approach.

The state-feedback is defined in Eq. (2.33), while the prediction model is defined by the discrete map $\boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k)$ which will be defined for each control approach separately in the next chapter. While, the symbols $\overline{\bullet}$ and $\underline{\bullet}$ denote the upper and lower bounds, respectively. Therefore the actuator thrusts limits are $\underline{\boldsymbol{\gamma}}, \overline{\boldsymbol{\gamma}} \in \mathbb{R}^{n_u}$. Note that the control input is defined as the actuators thrusts derivatives, as discussed in Section 2.1.4:

$$\mathbf{u} := \dot{\boldsymbol{\gamma}} \tag{2.37}$$

and therefore the constraints on the control inputs are $\underline{\dot{\boldsymbol{\gamma}}}, \overline{\dot{\boldsymbol{\gamma}}} \in \mathbb{R}^{n_u}$, noting that the constraints are not constant because the upper and lower bounds are functions of $\boldsymbol{\gamma}_k$ as discussed in Section 2.1.4.

To summarize, the NLP problem defined in Eqs. (2.32) to (2.36) will be used in the next chapter with different objective functions $\mathbf{J}(\mathbf{r}_k, \mathbf{x}_k)$, prediction models $\boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k)$, states vectors $\mathbf{x}_k$, and references vectors $\mathbf{r}_k$.

3 NMPC design for APhI control

In this chapter, three different NMPC-based approaches for APhI control are proposed. As presented in Section 1.1.1, three categories of **(N)**MPC for physical interaction control can be recognized: Direct pose/force control with **(N)**MPC; Indirect interaction control with **(N)**MPC; and Cascaded control with **(N)**MPC. First, the three categories are introduced with more details in Section 3.1, and then in Section 3.2 the NMPC cascaded control design is presented. In Section 3.3 the design of NMPC impedance control is presented, and finally, the NMPC hybrid control design is proposed in Section 3.4.

3.1 Introduction

As there are different approaches for physical interaction control, **(N)**MPC can be deployed in different fashions for APhI control. As shown in Section 1.1.1, the use of **(N)**MPC for APhI control is always limited to hybrid pose/force tracking in the state of the art, but there are many contributions in the field of interaction control with ground robots that take different approaches to address the problem, such as indirect interaction control, or cascaded control architecture.

As discussed in Section 2.3.2, the motion control and the interaction control can be separated into two nested control loops, an inner-loop dedicated for motion control, and an outer-loop responsible for interaction control. In this architecture, **(N)**MPC can be implemented for motion control or interaction control, i.e. in the inner-loop or outer-loop. The most important advantage of using **(N)**MPC as an inner-loop motion controller is the ability to define constraints that the robot should satisfy, additionally, the motion control can prioritize the tracking of some states over the others, which can be critical for the stability of an ARs.

On the other hand, using **(N)**MPC for interaction control in the outer-loop provides many options to improve the interaction behaviour, for instance, a sub-objective can be added to the control problem to maintain contact with the environment to avoid the well-known problem of bouncing at the beginning of the interaction task, as done in [28]. Also, constraints can be defined on the interaction wrench to guarantee safe and smooth interaction. However, this configuration will not be further discussed in this thesis due to time constraints, and attention will be given to the aforementioned cascaded configuration where NMPC is in the inner loop, as will be presented in Section 3.2.

Indirect interaction control can also be done without the cascaded control structure in the form of impedance control. In impedance control, the controller will try to drive the system with the desired dynamical behaviour, thus, neither the motion nor the force is controlled directly. This concept will be implemented in Section 3.3 as the NMPC impedance control, where the controller will drive the system to minimize the error between its dynamic behaviour and the desired behaviour.

Finally, in Section 3.4, the NMPC hybrid control is presented, where the controller will directly control the normal force that is applied to the contact surface, and the pose of the AR in the unconstrained DoFs during physical interaction, but in free-flight when there is no physical interaction with the environment, the controller will simply control the motion of the AR in the 6D space. To do that, the concepts of hybrid force/position control that were discussed in Section 2.2 will be utilized, and a control-mode switching strategy is devised.

3.2 NMPC cascaded control

In this section, a cascaded control approach is proposed for APhI indirect control. In this approach, a trajectory tracking NMPC controller is used in the inner-loop, while the outer-loop is

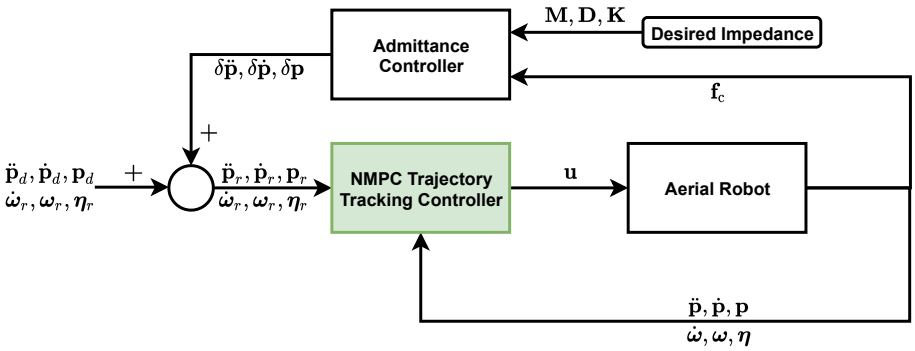


Figure 3.1: NMPC cascaded control block diagram.

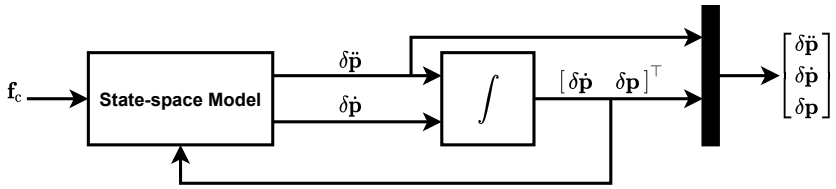


Figure 3.2: Admittance controller block diagram.

controlled by an admittance controller which will modify the reference trajectory of the inner-loop, as shown in the block diagram in Fig. 3.1.

3.2.1 Admittance controller

As discussed in Section 2.3.2, the admittance controller modifies the reference acceleration, velocity, and position by $\ddot{\mathbf{p}}, \dot{\mathbf{p}}, \mathbf{p} \in \mathbb{R}^3$, respectively, to steer the system with the desired impedance, which is defined by $\mathbf{M}$, $\mathbf{D}$, $\mathbf{K} \in \mathbb{R}^{3\times3}_{>0}$, the mass, damping, and stiffness diagonal matrices, respectively, such as:

$$\mathbf{M}\,\delta\ddot{\mathbf{p}} + \mathbf{D}\,\delta\dot{\mathbf{p}} + \mathbf{K}\,\delta\mathbf{p} = \mathbf{R}_S\mathbf{f}_c \tag{3.1}$$

where $\mathbf{f}_c \in \mathbb{R}^3$ is the contact forces applied to the end-effector and expressed in the surface frame. The trajectory modifiers $\ddot{\mathbf{p}}, \dot{\mathbf{p}}, \mathbf{p}$ can be solved as a state-space model, as shown in Fig. 3.2 as:

$$\begin{bmatrix}\delta\dot{\mathbf{p}}\\ \delta\ddot{\mathbf{p}}\end{bmatrix} = \begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{I}_3\\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{D}\end{bmatrix}\begin{bmatrix}\delta\mathbf{p}\\ \delta\dot{\mathbf{p}}\end{bmatrix} + \begin{bmatrix}\mathbf{0}_{3\times3}\\ \mathbf{M}^{-1}\end{bmatrix}\mathbf{R}_S\mathbf{f}_c \tag{3.2}$$

with the initial conditions of $\delta\mathbf{p}(0) = \delta\dot{\mathbf{p}}(0) = 0$.

3.2.2 NMPC trajectory tracking controller

The design of NMPC for trajectory tracking is based on [18], but it will be summarized here for completeness, where the NMPC's prediction model, states vector, references vector, and objective function will be presented.

Naturally, the states vector will include the motion states of position, velocity, orientation, and angular velocity. However, the actuators thrusts are also included in the states since the control inputs are their derivatives, and constraints should be defined on them. Therefore, the states vector can be defined as:

$$\mathbf{x} := \begin{bmatrix}\mathbf{p}^\top & \dot{\mathbf{p}}^\top & \boldsymbol{\eta}^\top & \boldsymbol{\omega}^\top & \boldsymbol{\gamma}^\top\end{bmatrix}^\top \tag{3.3}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ such that $n_x = 12 + n_u = 18$, where n_u is the number of control inputs, and it is equal to six when a fixed-tilt hexarotor is used, which is the assumption from now on. And the states vector can be discretized such that $\mathbf{x}_k = \mathbf{x}(kT)$

The dynamic prediction model of the AR for the linear and rotational DoFs, using Euler angles to represent orientation, as derived in Section 2.1, is:

$$\begin{aligned} m\mathbf{I}_3\ddot{\mathbf{p}} &= -mg\mathbf{e}_3 + \mathbf{R}_B\mathbf{G}_1\boldsymbol{\gamma} + m\mathbf{I}_3\mathbf{R}_B\left(\dot{\boldsymbol{\omega}} \times \mathbf{o}_{B,E} + \boldsymbol{\omega} \times \boldsymbol{\omega} \times \mathbf{o}_{B,E}\right) \\ \mathbf{J}\dot{\boldsymbol{\omega}} &= -\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \mathbf{G}_2\boldsymbol{\gamma} \\ \dot{\boldsymbol{\eta}} &= \mathbf{T}^{-1}\boldsymbol{\omega} \\ \dot{\boldsymbol{\gamma}} &= \mathbf{u} \end{aligned} \tag{3.4}$$

noting that this model does not consider any external forces or torques, so the NMPC is not aware of any interaction forces or disturbances. This can be written in terms of the map $\mathbf{f}(\bullet)$ as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{3.5}$$

which represents the map from $\mathbf{x}$ and $\mathbf{u}$ to $\dot{\mathbf{x}}$, and can be discretized to the corresponding discrete-time map $\boldsymbol{\phi}(\bullet)$, which can be written as:

$$\dot{\mathbf{x}}_{k+1} = \boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k), \quad k = 0, 1, \ldots, N-1 \tag{3.6}$$

The references vector will include motion reference for the linear and rotational motion, and it is defined as:

$$\mathbf{r}(t) = \begin{bmatrix} \mathbf{p}_r^\top & \dot{\mathbf{p}}_r^\top & \ddot{\mathbf{p}}_r^\top & \boldsymbol{\eta}_r^\top & \boldsymbol{\omega}_r^\top & \dot{\boldsymbol{\omega}}_r^\top \end{bmatrix} \tag{3.7}$$

which can be discretized such that $\mathbf{r}_k = \mathbf{r}(kT) \in \mathbb{R}^{18}$.

The objective function should reflect the trajectory tracking objectives that the NMPC controller is trying to achieve, and hence, it will depend on the errors of position, orientation, velocity, angular velocity, acceleration, and angular acceleration. Therefore, the discrete objective function is defined as:

$$\mathbf{J}(\mathbf{r}_k, \mathbf{x}_k) = \begin{bmatrix} \mathbf{p}_k - \mathbf{p}_{r,k} \\ \dot{\mathbf{p}}_k - \dot{\mathbf{p}}_{r,k} \\ \ddot{\mathbf{p}}_k - \ddot{\mathbf{p}}_{r,k} \\ \boldsymbol{\eta}_k \ominus \boldsymbol{\eta}_{r,k} \\ \boldsymbol{\omega}_k - \boldsymbol{\omega}_{r,k} \\ \dot{\boldsymbol{\omega}}_k - \dot{\boldsymbol{\omega}}_{r,k} \end{bmatrix} \tag{3.8}$$

where $\boldsymbol{\eta} \ominus \boldsymbol{\eta}_r \in \mathbb{R}^3$ is the attitude error of the Euler angles, and $\mathbf{J}(\mathbf{r}_k, \mathbf{x}_k) \in \mathbb{R}^{18}$.

Finally, the defined prediction model (Eq. (3.6)), objective function (Eq. (3.8)), states vector (Eq. (3.3)), and references vector (Eq. (3.7)) are implemented in the NLP problem defined in Eqs. (2.32) to (2.36).

3.3 NMPC impedance control

In this section, an NMPC controller will be developed to control the APhI indirectly through impedance control. The basic concepts of indirect interaction control, and impedance control, in particular, were presented in Section 2.3.1. The designed NMPC controller will drive the system to match the desired dynamical behaviour specified by the desired damping, stiffness, and apparent inertia, as shown in Fig. 3.3.

To control the interaction indirectly, the desired impedance behaviour should be included in the objective function on the NMPC. To do this, an *impedance error* will be defined, which will

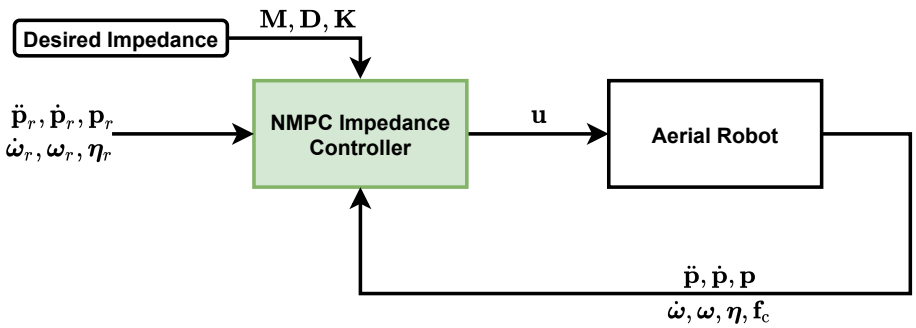


Figure 3.3: NMPC impedance block diagram.

represent the error between the current dynamical behaviour and the desired behaviour. For the three translational DoFs, the impedance error can be written as:

$$\mathbf{e}_Z = \mathbf{M}(\ddot{\mathbf{p}}_r - \ddot{\mathbf{p}}) + \mathbf{D}(\dot{\mathbf{p}}_r - \dot{\mathbf{p}}) + \mathbf{K}(\mathbf{p}_r - \mathbf{p}) + \mathbf{R}_S\mathbf{f}_c \tag{3.9}$$

where $\mathbf{e}_Z \in \mathbb{R}^3$ is the impedance error, $\mathbf{M}$, $\mathbf{D}$, $\mathbf{K} \in \mathbb{R}^{3\times3}_{>0}$ are the mass, damping, and stiffness diagonal matrices, $\ddot{\mathbf{p}}_r$, $\dot{\mathbf{p}}_r$, $\mathbf{p}_r \in \mathbb{R}^3$ are the reference acceleration, velocity, and position, $\ddot{\mathbf{p}}$, $\dot{\mathbf{p}}$, $\mathbf{p} \in \mathbb{R}^3$ are the acceleration, velocity, and position of the system, and finally $\mathbf{f}_c \in \mathbb{R}^3$ is the contact force transformed to the world frame by $\mathbf{R}_S$.

In the absence of interaction forces, i.e. free flight where $\mathbf{f}_c = \mathbf{0}_3$, the aforementioned impedance error will behave like a simple impedance controller for motion control, where the errors between the references and the current states are multiplied with their respective desired impedance behaviour. In case of interaction, i.e. $\mathbf{f}_c \neq \mathbf{0}_3$, the impedance error will represent the deviation between the current dynamic behaviour and the desired behaviour, and therefore, minimizing this error will drive the system to the desired dynamics.

3.3.1 Prediction Model

To include this impedance error in the designed NMPC controller, the contact forces are added to the states vector, which can be defined as:

$$\mathbf{x} := \begin{bmatrix} \mathbf{p}^\top & \dot{\mathbf{p}}^\top & \boldsymbol{\eta}^\top & \boldsymbol{\omega}^\top & \boldsymbol{\gamma}^\top & \mathbf{f}_c^\top \end{bmatrix}^\top \tag{3.10}$$

where the interaction contact force is denoted as $\mathbf{f}_c \in \mathbb{R}^3$, and the state vector $\mathbf{x} \in \mathbb{R}^{n_x}$ such that $n_x = 21$, assuming that a fixed-tilt hexarotor is used.

The prediction model of the AR will be similar to the derived model in Section 2.1, but it will also include the contact force model that was developed in Section 2.4, which can be written as:

$$\begin{aligned} m\mathbf{I}_3\ddot{\mathbf{p}} &= -mg\mathbf{e}_3 + \mathbf{R}_B\mathbf{G}_1\boldsymbol{\gamma} + \mathbf{R}_S\mathbf{f}_c + m\mathbf{I}_3\mathbf{R}_B\left(\dot{\boldsymbol{\omega}} \times \mathbf{o}_{B,E} + \boldsymbol{\omega} \times \boldsymbol{\omega} \times \mathbf{o}_{B,E}\right) \\ \mathbf{J}\dot{\boldsymbol{\omega}} &= -\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \mathbf{G}_2\boldsymbol{\gamma} + \mathbf{o}_{B,E} \times \left(\mathbf{R}_B^\top\mathbf{R}_S\mathbf{f}_c\right) \\ \dot{\boldsymbol{\eta}} &= \mathbf{T}^{-1}\boldsymbol{\omega} \\ \dot{\boldsymbol{\gamma}} &= \mathbf{u} \\ \dot{\mathbf{f}}_c &= -\mathbf{R}_S^\top\mathbf{R}_B[\boldsymbol{\omega}]_\times\mathbf{G}_1\boldsymbol{\gamma} - \mathbf{R}_S^\top\mathbf{R}_B\mathbf{G}_1\dot{\boldsymbol{\gamma}} \end{aligned} \tag{3.11}$$

which can be written in terms of the map $\mathbf{f}(\bullet)$ as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{3.12}$$

which represents the map from $\mathbf{x}$ and $\mathbf{u}$ to $\dot{\mathbf{x}}$, and can be discretized to the corresponding discrete-time map $\boldsymbol{\phi}(\bullet)$, which can be written as:

$$\dot{\mathbf{x}}_{k+1} = \boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k), \quad k = 0, 1, \ldots, N-1 \tag{3.13}$$

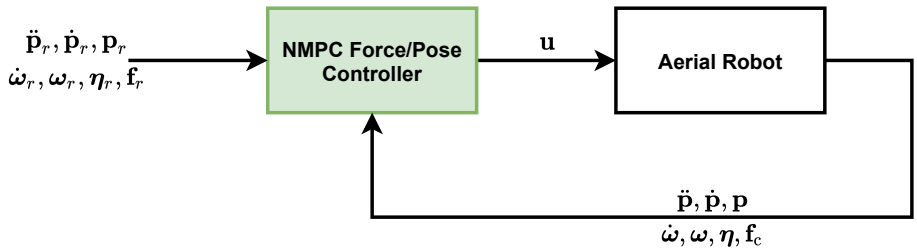


Figure 3.4: NMPC hybrid control block diagram.

3.3.2 NMPC design

After defining the states vector $\mathbf{x}$ in Eq. (3.10), the discrete prediction model in Eq. (3.13), and the impedance error $\mathbf{e}_Z$ in Eq. (3.9), the references vector will be a motion trajectory reference identical to Eq. (3.7). Then, the discrete objective function for the NMPC impedance control is defined as:

$$\mathbf{J}(\mathbf{r}_k, \mathbf{x}_k) = \begin{bmatrix} \mathbf{e}_Z \\ \boldsymbol{\eta}_k \ominus \boldsymbol{\eta}_{r,k} \\ \boldsymbol{\omega}_k - \boldsymbol{\omega}_{r,k} \\ \dot{\boldsymbol{\omega}}_k - \dot{\boldsymbol{\omega}}_{r,k} \end{bmatrix} \tag{3.14}$$

where $\mathbf{e}_Z \in \mathbb{R}^3$ is the translational impedance error and $\mathbf{J}(\mathbf{r}_k, \mathbf{x}_k) \in \mathbb{R}^{12}$.

In this objective function, there are no explicit position/velocity/acceleration errors, compared to the attitude/angular velocity/angular acceleration error terms. This choice is motivated by the fact that the impedance error implicitly includes the errors of the position, velocity, and acceleration, and therefore the system will be driven to track the linear trajectory motion with the desired dynamical behaviour in the absence of interaction.

Note that the impedance error is for the translational DoFs only, and it does not include the rotational DoFs, this choice is motivated by the desire to have separate robust control over the rotational DoFs to keep the AR attitude stable and resilient to disturbances. However, if the APhI task requires a compliant behaviour in the attitude as well, e.g. peg-in-the-hole task, the impedance error can be developed to include the rotational dynamics, as in Eq. (2.28).

Finally, the prediction model (Eq. (3.13)), objective function (Eq. (3.14)), states vector (Eq. (3.10)), and references vector (Eq. (3.7)) are implemented in the NLP problem defined in Eqs. (2.32) to (2.36).

3.4 NMPC hybrid control

In this section, an NMPC controller will be developed to directly control the APhI in the form of HFPC. Building on the basic concepts of HFPC which were presented in Section 2.2, two different *control modes* will be defined, namely, Free-flight Mode (FM), and Contact Mode (CM). In FM, the system's motion is not constrained in any DoF, and in CM the system's motion is constrained along at least one DoF. The NMPC controller, as shown in Fig. 3.4, will only track the position reference trajectory in FM, but in CM, the NMPC controller will track both the position and force references.

As presented before in Section 2.2, it is important to explicitly represent the constrained DoFs in the prediction model and control design, and it is more intuitive to define the constrained DoFs in $\mathcal{F}_S$, where $\mathbf{y}_S$ is pointing towards the contact surface, which is assumed to be planar, as shown in Fig. 2.1. It can be noticed that when the system is in CM, the end-effector is free to move on the planar axes $\mathbf{x}_S, \mathbf{z}_S$, but its motion is constrained in the axis that is normal to the contact surface $\mathbf{y}_S$. Furthermore, the selection factor $\kappa \in \{0, 1\}$ is defined such that $\kappa = 0$

when the AR is in FM, and $\kappa = 1$ when the AR is in CM. This factor will be used in the derivation of the system dynamics in the two control modes. The computation of κ will be discussed later in Section 3.4.3.

The selection matrix $\mathbf{S}_\kappa$, which selects the unconstrained translational dynamics and truncates the constrained translational dynamics when the system is in FM, is defined as:

$$\mathbf{S}_\kappa = \mathbf{R}_S \text{diag}(1,\ 1-\kappa,\ 1)\mathbf{R}_S{}^\top \tag{3.15}$$

and its complement $\bar{\mathbf{S}}_\kappa$, which selects the constrained translational DoFs and truncates the unconstrained translational DoFs when the system is in CM, is defined as:

$$\bar{\mathbf{S}}_\kappa = \mathbf{R}_S \text{diag}(0,\ \kappa,\ 0)\mathbf{R}_S{}^\top \tag{3.16}$$

note that regardless of the control mode, FM or CM, $\mathbf{S}_\kappa \bar{\mathbf{S}}_\kappa = 0_{3\times 3}$, which means that the position control and force control will exclude each other for all the translational DoFs.

3.4.1 Prediction Model

As discussed in Section 2.2, the dynamics of the AR can be derived directly in the end-effector frame, where the states vector is defined as:

$$\mathbf{x} := \begin{bmatrix} \mathbf{p}^\top & \dot{\mathbf{p}}^\top & \boldsymbol{\eta}^\top & \boldsymbol{\omega}^\top & \boldsymbol{\gamma}^\top & \mathbf{f}_c^\top \end{bmatrix}^\top \tag{3.17}$$

where the state vector $\mathbf{x} \in \mathbb{R}^{n_x}$ such that $n_x = 21$.

The prediction model of the AR, as derived in Section 2.2, is:

$$\begin{aligned}
\ddot{\mathbf{p}} &= \mathbf{S}_\kappa \Big(m^{-1}\big(\mathbf{R}_B \mathbf{f}_a + \kappa \mathbf{R}_S \mathbf{f}_c\big) - g\mathbf{e}_3 + \mathbf{R}_B\big(\dot{\boldsymbol{\omega}} \times \mathbf{o}_{B,E} + \boldsymbol{\omega} \times \boldsymbol{\omega} \times \mathbf{o}_{B,E}\big)\Big) \\
\dot{\boldsymbol{\omega}} &= \mathbf{J}^{-1}\Big(-\boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}) + \boldsymbol{\tau}_a + \mathbf{o}_{B,E} \times \big(\kappa \mathbf{R}_B^\top \mathbf{R}_S \mathbf{f}_c\big)\Big) \\
\dot{\boldsymbol{\eta}} &= \mathbf{T}^{-1}\boldsymbol{\omega} \\
\dot{\boldsymbol{\gamma}} &= \mathbf{u} \\
\dot{\mathbf{f}}_c &= -\bar{\mathbf{S}}_\kappa \left(\mathbf{R}_S^\top \mathbf{R}_B [\boldsymbol{\omega}]_\times \mathbf{G}_1 \boldsymbol{\gamma} + \mathbf{R}_S^\top \mathbf{R}_B \mathbf{G}_1 \dot{\boldsymbol{\gamma}} \right)
\end{aligned} \tag{3.18}$$

note that the contact force dynamics are equal to zero in FM, while the full 3D translational dynamics are included in the model. While in CM, the selection matrix will exclude the translational dynamics of $\mathbf{y}_S$ from the prediction model, and the contact force model in the $\mathbf{y}_S$ will be included by the complement matrix. The prediction model can be written in terms of the map $\mathbf{f}(\bullet)$ as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{3.19}$$

which represents the map from $\mathbf{x}$ and $\mathbf{u}$ to $\dot{\mathbf{x}}$.

The continuous-time map $\mathbf{f}(\bullet)$ is discretized to the corresponding discrete-time map $\boldsymbol{\phi}(\bullet)$ can be written as:

$$\dot{\mathbf{x}}_{k+1} = \boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k), \quad k = 0, 1, \ldots, N-1 \tag{3.20}$$

The subscript k denotes the time step kT such that $\mathbf{x}_k = \mathbf{x}(kT)$ and T is the sampling time.

3.4.2 NMPC design

After defining the states vector $\mathbf{x}$ in Eq. (3.17), and the discrete prediction model in Eq. (3.20), the references vector is defined as:

$$\mathbf{r}(t) = \begin{bmatrix} \mathbf{p}_r^\top & \dot{\mathbf{p}}_r^\top & \ddot{\mathbf{p}}_r^\top & \boldsymbol{\eta}_r^\top & \boldsymbol{\omega}_r^\top & \dot{\boldsymbol{\omega}}_r^\top & \mathbf{f}_r^\top \end{bmatrix} \tag{3.21}$$

where $\mathbf{f}_r \in \mathbb{R}^3$ represents the reference contact forces expressed in $\mathscr{F}_S$. Then, the references vector can be discretized such that $\mathbf{r}_k = \mathbf{r}(kT) \in \mathbb{R}^{21}$.

Then, the discrete objective function is defined as:

$$\mathbf{J}(\mathbf{r}_k, \mathbf{x}_k) = \begin{bmatrix} \mathbf{S}_\kappa(\mathbf{p}_k - \mathbf{p}_{r,k}) \\ \mathbf{S}_\kappa(\dot{\mathbf{p}}_k - \dot{\mathbf{p}}_{r,k}) \\ \mathbf{S}_\kappa(\ddot{\mathbf{p}}_k - \ddot{\mathbf{p}}_{r,k}) \\ \boldsymbol{\eta}_k \ominus \boldsymbol{\eta}_{r,k} \\ \boldsymbol{\omega}_k - \boldsymbol{\omega}_{r,k} \\ \dot{\boldsymbol{\omega}}_k - \dot{\boldsymbol{\omega}}_{r,k} \\ \bar{\mathbf{S}}_\kappa(\mathbf{f}_{c,k} - \mathbf{f}_{r,k}) \end{bmatrix} \tag{3.22}$$

note that in objective function is also dependent on the control model such that in FM the force tracking term is always equal to zero, i.e. excluded from the objective function, while in CM the position, velocity, and acceleration error in $\mathbf{y}_S$ is excluded from $\mathbf{J}(\mathbf{r}_k, \mathbf{x}_k)$, and the force tracking in the $\mathbf{y}_S$ is included. Finally, $\mathbf{J}(\mathbf{r}_k, \mathbf{x}_k) \in \mathbb{R}^{21}$, and the NLP problem is defined as in Eqs. (2.32) to (2.36).

3.4.3 Contact mode switching

Since switching from one control mode to another in hybrid force/pose control is dependent on the selection factor κ, it is important to devise a switching strategy of this factor that can reflect the real physical situation of the robot. A clear distinction between FM and CM is the presence of the interaction control force, therefore, a trivial choice for contact mode switch is to make it dependant on $\mathbf{f}_c$, such that when the contact force is bigger than a certain threshold a, the contact mode is CM, and therefore $\kappa = 1$, and vice versa, which can be written as:

$$\kappa = \begin{cases} 0 & |\mathbf{f}_c| \leq a \\ 1 & |\mathbf{f}_c| > a \end{cases}$$

However, this choice is not suitable for some interaction tasks such as pushing, and push-and-slide tasks with a static environment, because the robot will not be able to retract from interaction once the task is completed, since the controller does not control the position in the same DoF of the interaction force control, i.e. the contact mode can not be switched back to FM, and therefore it can not retract from interaction.

Since this thesis is mainly concerned with contact-based APhI with static environments, the aforementioned scenario is not suitable, therefore, other switching logic should be proposed. Another switching strategy is to include the position error in the normal direction e_n, i.e. the direction of controlling the force in CM, in the switching logic, such that if the position error is larger than a certain threshold b, the control mode switches to FM even if the contact force condition is still true. This can be written as:

$$\kappa = \begin{cases} 0 & |\mathbf{f}_c| \leq a \\ 1 & |\mathbf{f}_c| > a \quad \textbf{AND} \quad |e_n| \leq b \end{cases}$$

this switching logic will allow the controller to switch back to FM when the reference trajectory in the normal direction retracts away from the surface, which will make e_n larger than b, and thus the controller switches to FM. But it also applies a restriction on the reference of the normal position during interaction, where it should not be set to penetrate the surface by a larger penetration than b.

4 MATLAB Simulations

In this chapter, the proposed NMPC control approaches are implemented in MATLAB/Simulink simulation environment, and preliminary tests are conducted to analyse and select the proper parameters for the NMPC controllers. This chapter is not intended to present simulation results that validate the proposed controllers in interaction tasks, but rather it will present the implementation details of the proposed controllers in MATLAB using MATMPC tool, and the simulation setup that was used to test the feasibility of the proposed controllers, and the analysis of the NMPC parameters effects on the system behaviour.

This chapter will be organized as follows, first, the simulation setup and models are introduced in Section 4.1, and then the NMPC implementation details is presented in Section 4.2. After that an analysis of NMPC parameters is presented in Section 4.3, and the presented results are discussed in Section 4.4.

4.1 Simulation setup

The simulation block diagram is shown in Fig. 4.1, where the controller block is dependent on the control approach, and was discussed in Chapter 3. The trajectory generator is a simple waypoints-based trajectory generator where it generates the required $N+1$ motion trajectories points for the prediction horizon t_H. The generated trajectory consists of position and orientation references, while the other references, such as velocity, acceleration, angular velocity, and angular acceleration, are set to zero. These motion trajectories are sent to the controller in the form of two matrices, references matrix $h_{ref} \in \mathbb{R}^{n_y \times N}$, and parameters matrix $\mathbf{a} \in \mathbb{R}^{n_p \times N+1}$ depending on the corresponding control approach, as will be discussed in Section 4.2.

The robot model that will be used throughout these simulations is a model of the fully-actuated hexarotor aerial robot *FiberThex*, shown Fig. 4.2, that was developed in the University of Twente, and it is equipped with a fixed end-effector that is rigidly attached to the robot mechanical frame. The robot parameters such as mass m, and inertia matrix $\mathbf{J}$, and the actuators parameters such as the thrust and thrust-to-drag coefficients c_f, c_t, respectively, have been experimentally validated in previous work to prompt the accuracy of this model. Also, the actuators minimum and maximum thrusts were experimentally identified, however, due to time constraints, the control inputs constraints (minimum and maximum acceleration of the actuators) were not experimentally identified. Instead, the constraints of a similar AR, *Tilt-Hex*, which were identified in [18], will be used in these simulations, since FiberThex is equipped with the same motors as Tilt-Hex. The model parameters of FiberThex are listed in Table 4.1. Based on the identified parameters and rotors positions, the wrench map $\mathbf{G}$ is calculated as described in Section 2.1.3.

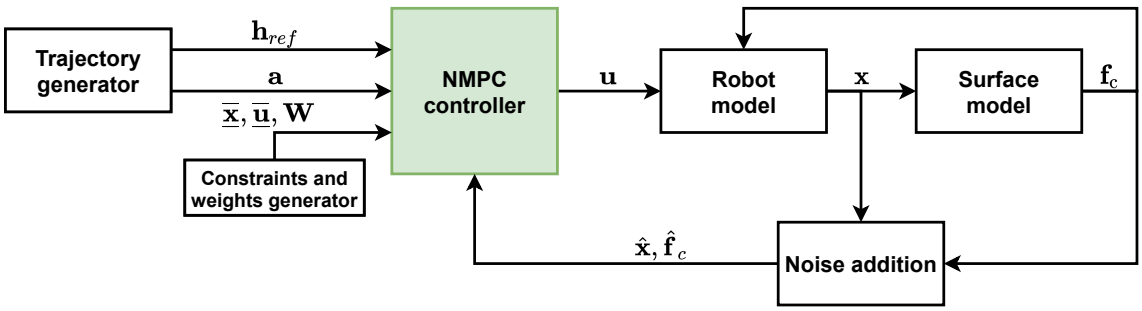


Figure 4.1: MATLAB/Simulink simulations block diagram.

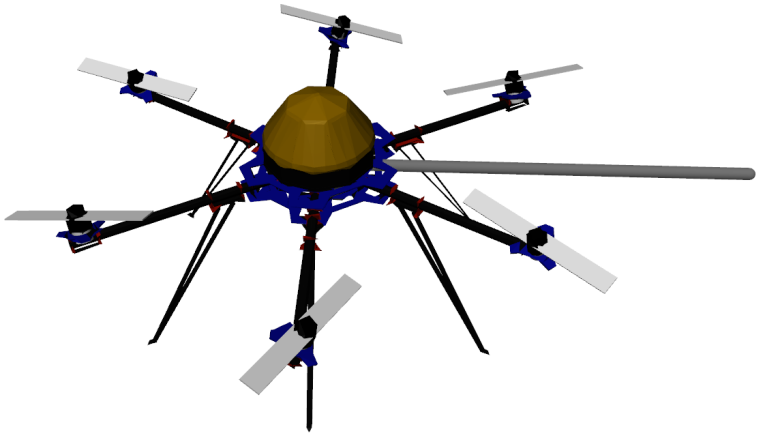


Figure 4.2: Photo of FiberThex CAD model.

In these simulations, the control input $\mathbf{u}$ is integrated using a fixed-step integrator to calculate $\boldsymbol{\gamma}$ which is considered the input to the AR dynamical model, together with the contact forces $\mathbf{f}_c$ that are generated by the surface model, such that the dynamics of the robot are described by Eq. (2.20). With proper integration, the states of the AR $\mathbf{x}$ are calculated such that:

$$\mathbf{x} := \begin{bmatrix} \mathbf{p}^\top & \dot{\mathbf{p}}^\top & \boldsymbol{\eta}^\top & \boldsymbol{\omega}^\top & \boldsymbol{\gamma}^\top \end{bmatrix}^\top \tag{4.1}$$

which corresponds also to the output of the robot model.

The surface model, which simulates the contact surface that the end-effector will interact with, is divided into two parts: the normal force model, and the friction model. The normal force f_n model is based on Hunt-Crossley model [37], which is a collision model representing the contact dynamics of viscoelastic systems. Essentially, the model can be seen as a spring-damper system that is activated when contact between the wall and the object is detected, and deactivated when there is no contact. The contact is detected based on a positive penetration model, such that if the position of the end-effector w.r.t the surface frame $\mathcal{F}_S$ is greater than or equal to zero in the normal direction ($\mathbf{y}_S$ in Fig. 2.1) then contact is detected, otherwise there is no contact. The model can be written as:

$$f_n = -k_h\sigma - d_h\sigma\dot{\sigma} \tag{4.2}$$

where $k_h, d_h \in \mathbb{R}_{>0}$ are the stiffness and damping coefficients, respectively, σ is the surface penetration, i.e. the position of the end-effector w.r.t the surface frame in the normal direction, and $\dot{\sigma}$ the velocity of the end-effector in the normal direction expressed in the surface frame. Note that the normal force is expressed in the surface frame. In the presented simulations, the surface model parameters are $k_h = 100, d_h = 100$. However, it was noted that higher values for the stiffness and damping coefficients would deteriorate the stability of contact with the model, and therefore, this model can not represent a fully rigid contact surface, but it was suitable for preliminary tests that are not intended to validate the controllers, but rather to study the feasibility of their respective OCPs.

On the other hand, the model of the friction force f_f along the two dimensional surface is based on the Coulomb friction model, such that:

$$f_f = -\mu \tanh(qv)\,|f_n| \tag{4.3}$$

Table 4.1: FiberThex physical parameters.

Parameter	Symbol	Value
Mass	m	3.061[Kg]
Inertia matrix	$\mathbf{J}$	$\text{diag}(0.115, 0.114, 0.194)[\text{Kg}\cdot m^2]$
Rotors tilt angle	α	20°
Rotor thrust coefficient	c_f	$12.5e-4$ [N/Hz^2]
Rotor thrust-to-drag coefficient	c_t	$2.338e-5$ [m]
Rotor maximum thrust	$\overline{\gamma}$	13[N]
Rotor minimum thrust	$\underline{\gamma}$	0.32[N]
Arm length (from CoM to rotor)		$0.38998m$
Angle between rotors		60°

where μ is the Coulomb friction coefficient, q is the friction slop, and v is the velocity in the friction direction. The output of the surface model is the contact force $\mathbf{f}_c = [f_{f_x} \quad f_n \quad f_{f_z}]^\top$ which is expressed in the surface frame.

Additionally, white noise is generated and added to the states and contact forces $\mathbf{x}, \mathbf{f}_c$ to simulate the real noisy measurements of the on-board sensors that the controller should be able to deal with, and to test the limits of which the controller can handle this noise. In impedance and hybrid control, $\hat{\mathbf{f}}_c$ will be included in the states vector of the NMPC controller.

As discussed in Chapter 3 and Section 2.1.4, the constraints on the control input $\mathbf{u}$ are functions of the rotors thrusts $\boldsymbol{\gamma}$, therefore, these constraints are calculated online based on the experimentally identified limits of the modeled AR, and then they are sent to the controller in the matrices $\overline{\mathbf{u}}, \underline{\mathbf{u}} \in \mathbb{R}^{n_{bu} \times N}$, denoted in Fig. 4.1 as $\overline{\underline{\mathbf{u}}}$ for compactness. Moreover, the constraints on the states are also generated and sent to the controller in the matrices $\overline{\mathbf{x}}, \underline{\mathbf{x}} \in \mathbb{R}^{n_{bx} \times N}$, which is also denoted in Fig. 4.1 as $\overline{\underline{\mathbf{x}}}$ for compactness. In these simulations, the only states constraints were the upper and lower bound of the thrusts $\boldsymbol{\gamma}$. Finally, the weights of the objective function are sent to the controller in the matrix $\mathbf{W} \in \mathbb{R}^{n_y \times N}$.

4.2 NMPC implementation

The designed NMPC controllers are implemented in MATMPC, a MATLAB-based NMPC toolbox [38]. After defining the discretized dynamical model, and the sizes of the relevant vectors (e.g. states, control inputs, and outputs vectors), MATMPC uses MATLAB to generate C code routines that will be called at run-time through MEX functions. This strategy increases the time efficiency of solving the NLP problem, which allows MATMPC to run in real-time applications.

The prediction models are discretized using a fixed step 4$^{\text{th}}$ order explicit Runge-Kutta integrator which is supported by MATMPC for multiple shooting [39], while the derivatives that are required for solving the OCP are obtained using CasADi [40], an open-source tool for symbolic algorithmic differentiation. Furthermore, qpOASES [41], which is a C++-based open-source Quadratic Programming (QP) solver, is used to solve the dense QP problem using Real Time Iteration (RTI) scheme [42].

RTI is a state-of-the-art fast algorithm for solving NMPC problems, where a single Sequential Quadratic Programming (SQP) iteration is carried out to solve the NLP problem, which means that the QP problem that corresponds to the NLP is solved through one iteration only. Thanks to the initial value embedding strategy, RTI scheme can quickly solve a sequence of similar QP problems online with varying initial conditions. The interested reader is referred to [16] for more details about RTI in NMPC, and its implementation in MATMPC [43].

Table 4.2: MATMPC vetors sizes for NMPC cascaded control.

Symbol	Size	Meaning
n_x	18	No. of states
n_u	6	No. of controls
n_y	18	No. of outputs (and references)
n_{yN}	18	No. of outputs at terminal stage
n_p	0	No. of parameters
n_c	0	No. of general constraints
n_{bx}	6	No. of bounds on states
n_{bu}	6	No. of bounds on controls
n_{bx}_idx	13-18	indexes of bounded states
n_{bu}_idx	1-6	indexes of bounded controls

To implement the proposed NLPs in MATMPC, these problems should be adapted to the standard NLP problem that is defined in MATMPC, which is described in details in Appendix C, such that for each one of the three proposed controller, a reference vector $\mathbf{h}^k_{ref}$, an inner objective vector function $\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k)$, and a parameters vector $\mathbf{a}_k$ (if necessary) is defined. The choice of these vectors depends on the objective function $\mathbf{J}(\mathbf{r}_k, \mathbf{x}_k)$ of each controller (which was designed in Chapter 3), and on the standard NLP problem in MATMPC, cf. Appendix C.

4.2.1 NMPC cascaded control implementation

For the NMPC cascaded control, the reference vector is defined as:

$$\mathbf{h}^k_{ref} = \begin{bmatrix} \mathbf{p}_{r,k} \\ \dot{\mathbf{p}}_{r,k} \\ \ddot{\mathbf{p}}_{r,k} \\ \boldsymbol{\eta}_{r,k} \\ \boldsymbol{\omega}_{r,k} \\ \dot{\boldsymbol{\omega}}_{r,k} \end{bmatrix} \tag{4.4}$$

While the vector function $\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k)$ is defined as:

$$\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} \mathbf{p}_k \\ \dot{\mathbf{p}}_k \\ \ddot{\mathbf{p}}_k \\ \boldsymbol{\eta}_k \\ \boldsymbol{\omega}_k \\ \dot{\boldsymbol{\omega}}_k \end{bmatrix} \tag{4.5}$$

This definition of the vectors will render an objective function in MATMPC that is identical to the defined objective function in Eq. (3.8).

A summary of the sizes of the vectors and matrices that MATMPC will use is presented in Table 4.2.

4.2.2 NMPC impedance control implementation

In the NMPC impedance control, the reference vector is defined as:

$$\mathbf{h}^k_{ref} = \begin{bmatrix} \mathbf{0}_{3\times 1} \\ \boldsymbol{\eta}_{r,k} \\ \boldsymbol{\omega}_{r,k} \\ \dot{\boldsymbol{\omega}}_{r,k} \end{bmatrix} \tag{4.6}$$

Table 4.3: MATMPC vetors sizes for NMPC impedance control.

Symbol	Size	Meaning
n_x	21	No. of states
n_u	6	No. of controls
n_y	12	No. of outputs (and references)
n_{yN}	12	No. of outputs at terminal stage
n_p	9	No. of parameters
n_c	0	No. of general constraints
n_{bx}	6	No. of bounds on states
n_{bu}	6	No. of bounds on controls
$n_{bx\_}idx$	13-18	indexes of bounded states
$n_{bu\_}idx$	1-6	indexes of bounded controls

While the vector function $\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k)$ is defined as:

$$\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} \mathbf{M}(\ddot{\mathbf{p}}_r - \ddot{\mathbf{p}}_k) + \mathbf{D}(\dot{\mathbf{p}}_r - \dot{\mathbf{p}}_k) + \mathbf{K}(\mathbf{p}_r - \mathbf{p}_k) + \mathbf{R}_S \mathbf{f}_{c,k} \\ \boldsymbol{\eta}_k \\ \boldsymbol{\omega}_k \\ \dot{\boldsymbol{\omega}}_k \end{bmatrix} \tag{4.7}$$

where the $\ddot{\mathbf{p}}_r$, $\dot{\mathbf{p}}_r$, $\mathbf{p}_r$ will be passed to the NMPC controller as *MATMPC parameters* in the vector $\mathbf{a} \in \mathbb{R}^9$ such that:

$$\mathbf{a}_k = \begin{bmatrix} \ddot{\mathbf{p}}_{r,k}^\top & \dot{\mathbf{p}}_{r,k}^\top & \mathbf{p}_{r,k}^\top \end{bmatrix}^\top \tag{4.8}$$

This definition of the vectors will allow the required references to be passed to the vector function through the parameters vector, and eventually, the objective function in MATMPC will be identical to the defined objective function in Eq. (3.14).

A summary of the sizes of the vectors and matrices that MATMPC will use is presented in Table 4.3.

4.2.3 NMPC hybrid control implementation

The reference vector for the NMPC hybrid control is defined as:

$$\mathbf{h}_{ref}^k = \begin{bmatrix} \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times1} \\ \boldsymbol{\eta}_{r,k} \\ \boldsymbol{\omega}_{r,k} \\ \dot{\boldsymbol{\omega}}_{r,k} \\ \mathbf{0}_{3\times1} \end{bmatrix} \tag{4.9}$$

And the vector function $\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k)$ is defined as:

$$\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} \mathbf{S}_\kappa(\mathbf{p}_k - \mathbf{p}_{r,k}) \\ \mathbf{S}_\kappa(\dot{\mathbf{p}}_k - \dot{\mathbf{p}}_{r,k}) \\ \mathbf{S}_\kappa(\ddot{\mathbf{p}}_k - \ddot{\mathbf{p}}_{r,k}) \\ \boldsymbol{\eta}_k \\ \boldsymbol{\omega}_k \\ \dot{\boldsymbol{\omega}}_k \\ \bar{\mathbf{S}}_\kappa(\mathbf{f}_{c,k} - \mathbf{f}_{r,k}) \end{bmatrix} \tag{4.10}$$

Table 4.4: MATMPC vetors sizes for NMPC hybrid control.

Symbol	Size	Meaning
n_x	21	No. of states
n_u	6	No. of controls
n_y	21	No. of outputs (and references)
n_{yN}	21	No. of outputs at terminal stage
n_p	12	No. of parameters
n_c	0	No. of general constraints
n_{bx}	6	No. of bounds on states
n_{bu}	6	No. of bounds on controls
n_{bx}_idx	13-18	indexes of bounded states
n_{bu}_idx	1-6	indexes of bounded controls

where the $\ddot{\mathbf{p}}_r$, $\dot{\mathbf{p}}_r$, $\mathbf{p}_r$, $\mathbf{f}_r$ will be passed to the NMPC controller as *MATMPC parameters* in the vector $\mathbf{a} \in \mathbb{R}^{12}$ such that:

$$\mathbf{a}_k = \begin{bmatrix} \ddot{\mathbf{p}}_{r,k}^\top & \dot{\mathbf{p}}_{r,k}^\top & \mathbf{p}_{r,k}^\top & \mathbf{f}_{r,k}^\top \end{bmatrix}^\top \tag{4.11}$$

This definition of the vectors will allow the required references to be passed to the vector function $\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k)$ through the parameters vector $\mathbf{a}$, and eventually, the objective function in MATMPC will be identical to the defined objective function in Eq. (3.22).

A summary of the sizes of the vectors and matrices that MATMPC will use is presented in Table 4.4

4.3 NMPC parameters tuning

Choosing suitable NMPC parameters is vital for the feasibility of the OCP, and stability of the system, therefore this section is dedicated to studying the effects of these parameters on the stability and feasibility of the proposed controllers. The parameters that will be addressed are sampling time T_s, shooting interval T_{st}, and number of shooting points N. It is worth noting that the prediction horizon T_h is determined by the number of shooting points and shooting interval such that $T_h = N\,T_{st}$.

4.3.1 Sampling time

In the field of aerial vehicles control, it is favourable to run the controller at relatively high frequencies to match the fast nonlinear dynamics of these systems, especially the rotational dynamics [44]. However, the frequency of optimization-based controllers is always constrained by the time-efficiency of solving the optimization problem, which can increase the feasible sampling time. Therefore, there is a clear relationship between the computational time that is required to solve the NLP problem, denoted as T_{solv}, and T_s, where $T_s \geq T_{\text{solv}}$ such that the optimization problem is solved and the optimal solution is available on time, at each time step.

A series of simulations have been conducted to estimate the average T_{solv} for the three control approaches, where the controllers are tested in various free-flight maneuvers (such as hovering and square trajectory tracking), and interaction tasks (pushing and push-and-slide tasks). The computational time of the NMPC solver during the interaction tasks is presented in Fig. 4.3, where the preliminary NMPC parameters in all of these tests are: sampling time $T_s = 5ms$, shooting interval $T_{st} = 0.1$, and number of shooting points $N = 15$. It can be noticed that the average computational time is always smaller than $2ms$, however, numerous outliers spread between $2ms$ and $5ms$, while other outliers are greater than $5ms$ (which are not shown in Fig. 4.3), and they represent 0.32% of the sampled data. Furthermore, similar results were

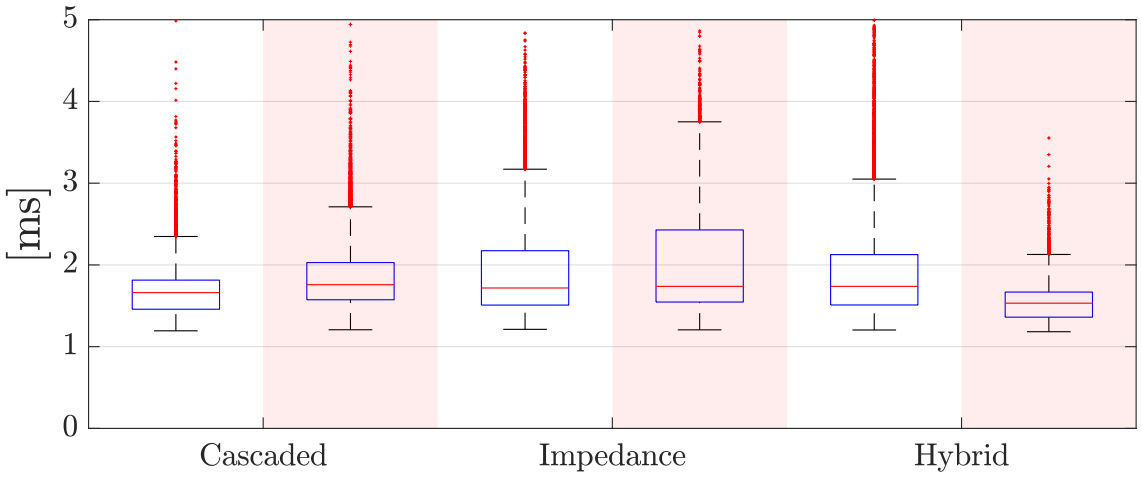


Figure 4.3: Box-plots for the computational time T_{solv} of the respective NMPC controllers in a pushing and push-and-slide (highlighted) task.

observed when N and T_{st} were varied to different combinations, which decouples the computational time from the other parameters.

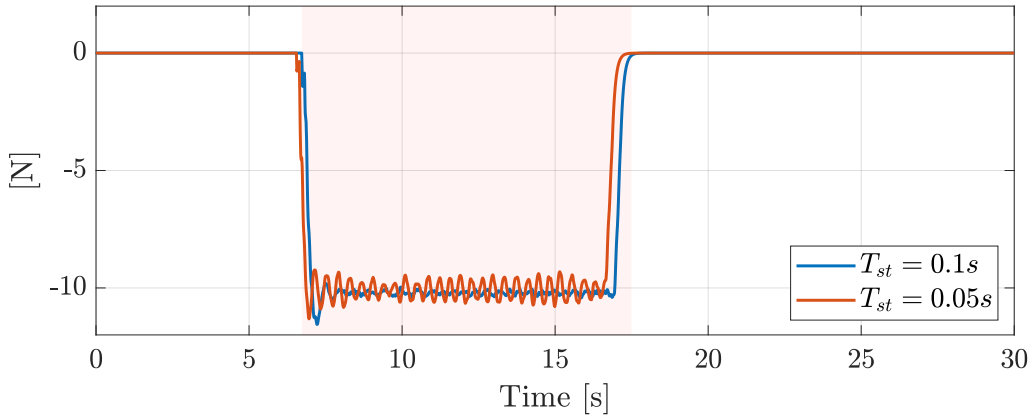
Nonetheless, the average T_{solv} in all the experiments combined is equal to $1.85ms$, and therefore the sampling time can be chosen to be within the range of $2ms$ and $5ms$. Statistically, a sampling rate of $5ms$ will satisfy the inequality $T_s \geq T_{\text{solv}}$ in 99.68% of the sampled data, while a sampling rate of $4ms$ will satisfy the aforementioned inequality in 99.1% of the sampled data. After multiple iterations of tests and comparisons, a sampling time of $4ms$ was selected, which is a trade-off between a higher sampling frequency, and a higher statistical guarantee of satisfying the time constraints.

However, there is no guarantee that at any time step kT, the solution of the OCP $\mathbf{u}_k$ will be ready before the next time step $(k+1)T$. These cases can be tackled in different approaches, for instance, a zero-order hold model can be used to hold the solution of the previous time step $\mathbf{u}_{k-1}$ when the new solution is not ready yet, and hence, the newest available solution is always held as a control input. Another approach is to use the solution of the second shooting point from the previous time step $k-1$. Keeping in mind that the solution of the OCP at time kT will be a trajectory of the control inputs for the future prediction horizon, denoted as ${}^{k}\mathbf{u}^{\star}_{1,\dots,N} \in \mathbb{R}^{n_u \times N}$, and in NMPC the optimal control input is chosen to be the control input of the first shooting point $\mathbf{u}_k = {}^{k}\mathbf{u}^{\star}_1$. Hence, when the solution of the time step kT is not ready yet, the control input can be set to $\mathbf{u}_k = {}^{k-1}\mathbf{u}^{\star}_2$.

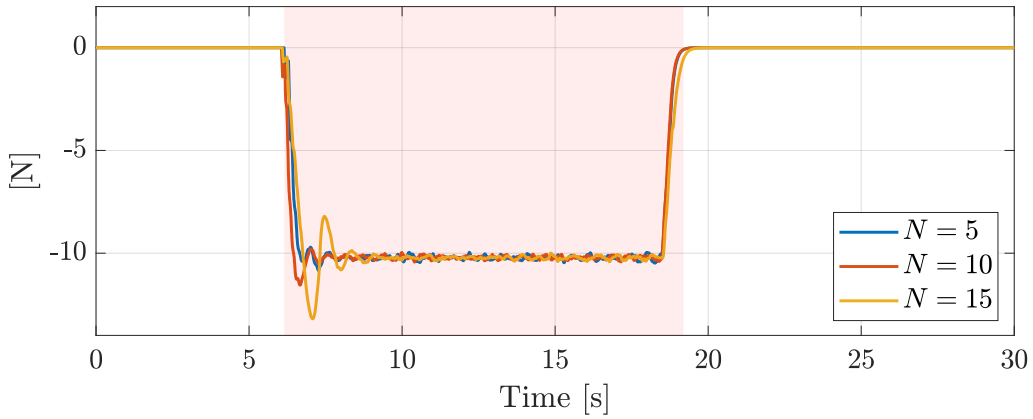
The two proposed approaches are suboptimal and can be considered ad-hoc, but after testing both of them, it was noted that they both handled the mentioned rare cases well, and no significant difference was observed in the response of the system when using either of them. Thus, a clear superiority of one approach over the other was not be observed experimentally, therefore, a zero-order hold approach was selected.

4.3.2 Prediction horizon

As mentioned before, the prediction horizon is determined by the shooting interval T_{st}, and the number of shooting points N, therefore, it is important to analyse the effect of both of them on the NMPC behaviour. To do that, a series of tests have been conducted where one of the two parameters, T_{st} and N, is varied while the other parameters are constant, to observe the effects of this parameter on the NMPC behaviour.



(a) $T_s = 4ms$ and $N = 10$



(b) $T_s = 4ms$ and $T_{st} = 0.1s$

Figure 4.4: The normal force response during an identical pushing task with NMPC hybrid control and a reference force of $-10N$. In **(a)** several shooting intervals T_{st} is compared, and in **(b)** several number of shooting points N is compared.

For these tests, the NMPC hybrid control was chosen to simulate a pushing task against a rigid static wall, where the reference of the normal force was chosen to be a step function of $-10N$. While similar results were observed when smaller force references were commanded, the effects of the NMPC parameters were more clear to observe and visualize when the force reference was set to a high value like $-10N$.

The normal force $f_{c,y}$ response for the pushing task is presented in Fig. 4.4a, where $T_s = 4ms$, $N = 10$, and the shooting interval is tested for the values $0.05s$ and $0.1s$. It is noticed that more oscillation is present when $T_{st} = 0.05s$, while the response and reference tracking is better and smoother when $T_{st} = 0.1s$. While a clear interpretation of these results is out of the scope of this thesis and was not thoroughly investigated, a shooting interval of $0.1s$ has been chosen by trial-and-error, after analysing different combinations of T_{st} and N for the three control approaches, and observing similar results to what was shown in Fig. 4.4a.

On the other hand, the effect of the number of shooting points has also been studied in a similar manner. Where a pushing task has been tested with the NMPC hybrid control, and a reference force of $-10N$, such that $T_s = 4ms$, and $T_{st} = 0.1$, and N is varied between the values $\{5, 10, 15\}$. The response of the normal force $f_{c,y}$ is plotted in Fig. 4.4b, where it is clear that a larger number of shooting points leads to more aggressive behaviour, with a larger overshoot. By trial-and-error, $N = 10$ was chosen after testing different combinations with the three control approaches, and observing similar results.

It is worth noting that many combinations of N and T_{st}, that would render an identical prediction horizon T_h, were not feasible, and the OCP was not solvable considering the defined constraints, such as the combination of $N = 20$, $T_{st} = 0.05s$. Moreover, other combinations that would render different horizon was also not feasible, such as $N = 20$, $T_{st} = 0.1s$, or $N = 10$, $T_{st} = 0.15s$.

4.4 Discussion

In the literature, a longer prediction horizon with a larger number of shooting points is usually associated with better performance [15; 16], however, this was not always the case in the shown simulations, where it was noticed that when $N > 5$, an increase in the number of shooting points was associated with larger overshoots in the pushing task, this observation might be understood and interpreted in the light of the contact force prediction model which was introduced in Section 2.4. This model does not account for the collision dynamics, instead, it depicts the steady-state contact after the initial collision, therefore, the observed overshoots can be a result of the unmodeled dynamics. The effects of these dynamics would be less significant when the prediction horizon is shorter, because the divergence between the predicted states and the actual states is smaller in shorter prediction horizons. This interpretation is also supported by the fact that these overshoots were not observed in free-flight, instead, they were only observed during the physical interaction, and they were very clear when an aggressive force reference should be tracked.

This result might also be interpreted as a bi-product of using RTI scheme, where a longer prediction horizon will yield a *less optimal* solution using only one SQP iteration. However, both of the aforementioned correlations should be analysed thoroughly before drawing conclusions.

At the end, the number of shooting points was chosen to be $N = 10$, where the shooting interval time is set to $T_{st} = 0.1s$ which renders a prediction horizon of $T_h = 1s$ and an internal NMPC prediction frequency of $10Hz$. On the other hand, the NMPC controller runs at a frequency of $250Hz$ with a sampling time of $T_s = 4ms$, and the average computation time to solve the NLP problems T_{solv} was found to be equal to $1.85ms$, where it should be less than or equal to the sampling time T_s, and the sampling time should be less than or equal to the shooting interval time T_{st}, and finally, the latter should be less than or equal to the prediction horizon T_h [18; 44], which renders the following chain of inequalities:

$$T_{\text{solv}} \leq T_s \leq T_{st} \leq T_h \tag{4.12}$$

and the chosen parameters satisfy these inequalities.

While this chapter focused on the details of the NMPC implementation in MATLAB/Simulink, and the NMPC parameters tuning and analysis, it did not present any simulation results to validate the capabilities and limitations of the proposed controllers. This will be carried out in the next chapter, where the NMPC controllers are validated through real-time simulations of two APhI tasks.

5 Gazebo Simulations

In this chapter, the proposed NMPC control approaches are further validated with real-time simulations of interaction tasks in Gazebo simulation environment [45]. Gazebo is an open-source real-time multi-robot simulator, where simulation worlds can be built around the simulated robots to test and develop the robots in different scenarios, tasks, and configurations. Thanks to its robust physics engine, robots can physically interact with each other and with other objects that are part of the simulated world.

Each control approach will be tested in two physical interaction tasks between the AR and a rigid wall with flat surface, where the two tasks are: *pushing*, and *push-and-slide*. In the pushing task, the AR will interact with the wall by applying normal force to the contact surface, and since the wall is static this pushing force will not change its position. This resembles tasks such as data sampling for Non-Destructive Testing (NDT), where a certain sensor should be pushed against the tested surface for a certain period of time to collect data. Comparatively, the push-and-slide task includes two phases, where the first phase is similar to the pushing task, but the second phase, sliding, includes sliding on the surface while maintaining contact with it. The Push-and-slide task also resembles a wide variety of practical tasks, such as surface cleaning, polishing, and continuous data sampling for NDT.

Testing the proposed control approaches in Gazebo will provide more insights into the capabilities and limitations of the controllers in a relatively realistic simulation environment. Also, the real-time software components that are used to connect the controller with the simulator are the same components that are used to connect with the real robot, which moves the controllers one more step towards real physical experiments.

This chapter will be organized as follows, first, the Gazebo simulation setup, models, and interfaces are introduced in Section 5.1. Then, the simulation results of the NMPC cascaded control are presented in Section 5.2, while Section 5.3 includes the results of the NMPC impedance control, and Section 5.4 presents the NMPC hybrid control results. In Section 5.5, the NMPC response to constraints is validated, and finally, the simulation results are discussed in Section 5.6.

5.1 Simulation setup

While the controller is still running in MATLAB/Simulink, the physical models of the robot and the wall are defined and simulated in Gazebo. The simulation models of the robot and the wall are defined in Simulation Description Format (SDF), which is an XML format developed by *Open Source Robotics Foundation* as part of Gazebo simulator. SDF describes objects and environments for robot simulators, where any object can be characterized as a series of links connected by joints.

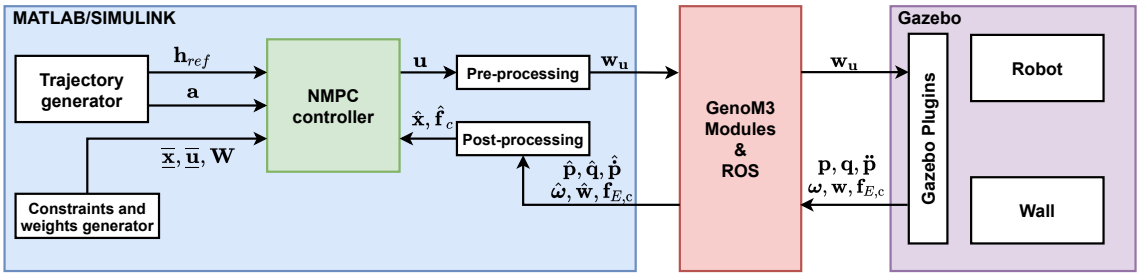
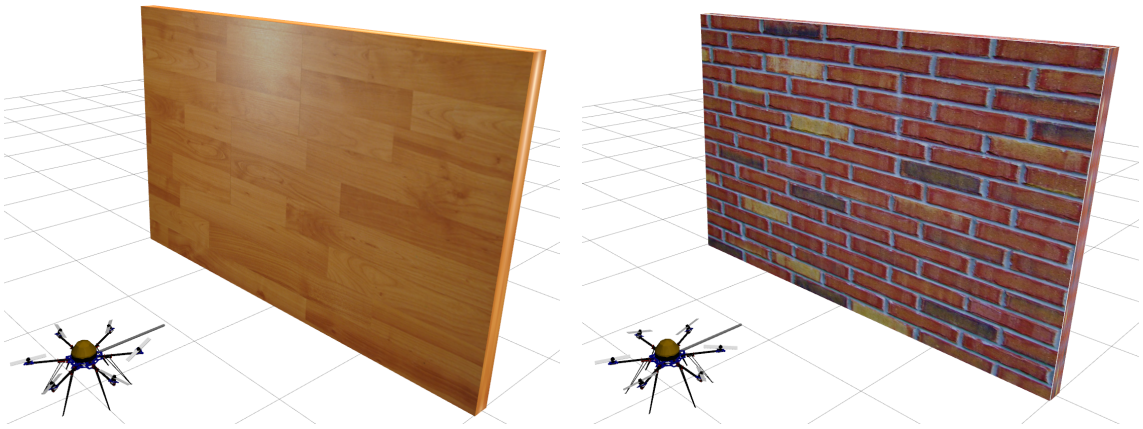


Figure 5.1: Gazebo simulations block diagram.



(a) Low friction wall model. **(b)** High friction wall model.

Figure 5.2: Pictures of FiberThex with the two wall models in Gazebo.

As in Chapter 4, the robot model is based on the aerial robot FiberThex with the physical parameters listed in Table 4.1. The robot is modelled as one rigid body, while the rotors thrust/-torque generation is modelled using the open-source Gazebo plugin *mrsim-gazebo*, where each rotor is specified by its position, orientation, thrust, and drag coefficients. The robot is also equipped with a Force/Torque (F/T) sensor which is mounted on the end-effector.

In Gazebo, a collision between two rigid bodies is modelled based on Conservation of Momentum, and therefore, the wall is modelled as a mechanically-grounded rigid body, i.e. it cannot be moved or deformed under the influence of any force, and hence the contact model in Gazebo is in line with the assumptions that was made while developing the prediction contact model in Section 2.4. On the other hand, the friction forces of the contact surface is characterized by its Coulomb friction coefficient μ. Two wall models have been developed, a *low-friction wall* with $\mu = 0.1$ shown in Fig. 5.2a, and a *high-friction wall* with $\mu = 1$ shown in Fig. 5.2b, to test the controllers in interaction scenarios with different friction conditions. When the robot and the wall are in contact, the contact forces are measured using the aforementioned F/T sensor.

A detailed description of the MATLAB/Simulink interface with Gazebo is depicted in Fig. 5.3, where GenoM3[1] modules and Robot Operating System (ROS) are used in the interface. The motivation to use Genom3 modules is that the same modules will be used to communicate with the real robot in a real physical experiment, which makes it easier to move from Gazebo simulation to real experiments with minimal modifications to the software architecture.

The *Rotorcraft* Genom3 module takes the rotors velocities commands $\mathbf{w_u}$ that are calculated after integrating the control inputs $\mathbf{u}$, and send them to the low-level ESCs that control the rotors velocities. In simulation, those commands are passed to *mrsim* Genom3 module which simulates the work of the ESCs, and passes the commands to Gazebo plugin *mrsim-gazebo*, that calculates the corresponding generated forces and torques which will be applied to the robot model in Gazebo. mrsim-gazebo plugin also exports the acceleration $\ddot{\mathbf{p}}$ and angular velocity $\boldsymbol{\omega}$ of the robot from Gazebo, to simulate the measurements of the Inertial Measurement Unit (IMU) in the real experiments.

The *Optitrack* Genom3 module exports the data of position $\mathbf{p}$ and orientation $\mathbf{q}$, represented by unit quaternions, from Gazebo through the Gazebo plugin *optitrack-gazebo*. In real experiments, the Optitrack module will export the position and orientation data from a real Optitrack motion capture system. After that, the measurements from the simulated IMU and the simulated Optitrack are fused in the *POM* Genom3 module to estimate the full motion states of the

[1] The Generator of Modules GenoM is a tool to design real-time software architectures developed by CNRS-LAAS: https://git.openrobots.org/projects/genom3

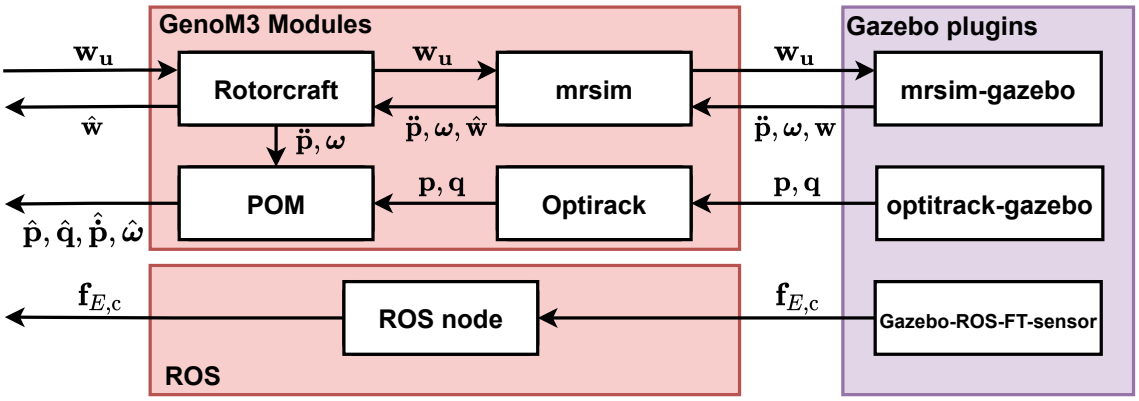


Figure 5.3: Block diagram of GenoM3 and ROS interface between MATLAB/Simulink and Gazebo.

robot, i.e. position, orientation, velocity, and angular velocity $\hat{\mathbf{p}}, \hat{\mathbf{q}}, \hat{\dot{\mathbf{p}}}, \hat{\omega}$. While the estimated velocity of the rotors $\hat{\mathbf{w}}$ is exported from the Rotorcraft module.

The F/T sensor measures the contact forces (in the end-effector frame) $\mathbf{f}_{E,\mathrm{c}}$ which are exported from Gazebo to a ROS network through the Gazebo plugin *Gazebo-ROS-FT-sensor*, then the measurements are received in MATLAB/Simulink using the ROS Toolbox in Simulink, through a buffer ROS node, and they are properly transformed to the surface frame to acquire the state $\mathbf{f}_{\mathrm{c}}$.

Finally, it is worth noting that the NMPC parameters that have been selected in Section 4.3 were tested and validated again in Gazebo. The simulation results showed similar results to those presented in Section 4.3, therefore, the same NMPC parameters were used in all the simulations in this chapter.

5.2 NMPC cascaded control results

This section will present the simulation results of the NMPC cascaded controller for two interaction tasks, a pushing task, and a push-and-slide task, against two wall models, a low friction wall, and a high friction wall. In these simulations, the desired impedance behaviour is defined by the matrices $\mathbf{M} = 1.5\,\mathbf{I}_3$, $\mathbf{D} = 10\,\mathbf{I}_3$, $\mathbf{K} = 6\mathbf{I}_3$, while the weights of the different terms in the objective function are listed in Table 5.1, where $\mathbf{Q_p}, \mathbf{Q_{\dot{p}}}, \mathbf{Q_{\ddot{p}}} \in \mathbb{R}^3$ are the weights of the position, velocity and acceleration errors, respectively, and $\mathbf{Q_\eta}, \mathbf{Q_\omega}, \mathbf{Q_{\dot{\omega}}} \in \mathbb{R}^3$ are the weights of the orientation, angular velocity, and angular acceleration, respectively. These weights will be used for all the simulations in this section.

Table 5.1: Weights of the objective function for NMPC cascaded control during the pushing and push-and-slide simulations.

Weight	$\mathbf{Q_p}$	$\mathbf{Q_{\dot{p}}}$	$\mathbf{Q_{\ddot{p}}}$	$\mathbf{Q_\eta}$	$\mathbf{Q_\omega}$	$\mathbf{Q_{\dot{\omega}}}$
Value	$0.1\,\mathbf{I}_3$	$0.01\,\mathbf{I}_3$	$0\,\mathbf{I}_3$	$1\,\mathbf{I}_3$	$0.01\,\mathbf{I}_3$	$0\,\mathbf{I}_3$

The results of the pushing task are shown in Fig. 5.4, where Fig. 5.4a represents the results of pushing against a low friction wall, while Fig. 5.4b is the high friction wall results. The same motion trajectory was used in the two experiments, and it can be noticed that the response is almost identical in the two experiments regardless of the friction of the wall, mainly because the AR is interacting with a constant point on the wall surface, and therefore, the end-effector is not experiencing any significant friction forces.

The position reference $p_{r,y}$ was set to reach $2.6m$, however, the admittance controller in the outer-loop modified the position reference $p_{r,y}$ and the velocity reference $\dot{p}_{r,y}$ to match the desired impedance behaviour. It is also noted that at the beginning of the interaction (highlighted section) between $9-15s$, there was bouncing against the wall that resulted in losing contact multiple times as seen in $f_{c,y}$ plot, after that when the penetration was larger in the surface, the interaction was maintained.

The push-and-slide task results are shown Fig. 5.5, where Fig. 5.5a represents the results of interacting with a low friction wall, while Fig. 5.5b is the high friction wall results. Because of the sliding action in this task, a clear difference can be noticed between the interaction with the low and the high friction wall, where a smooth trajectory tracking was achieved when sliding on the smooth surface as shown in the plot of p_x during the sliding phase (highlighted in green), while the orientation tracking was not affected by the sliding action, because the friction forces were not significant compared to the rough surface.

On the contrary, the high friction wall caused relatively significant friction forces as shown in Fig. 5.5b, where the friction force $f_{c,x}$ obstructed the trajectory tracking of p_x, and caused large deviations in the yaw angle ψ. Nonetheless, the NMPC motion controller was able to maintain stability, while exploiting the full-actuation of FiberThex, as shown in the plots of $\boldsymbol{\gamma}$, to simultaneously maintain a stable interaction and motion, especially attitude stability which was prioritized as a response to its higher relative weight in Table 5.1 compared to the other objectives. It is also noted that the reference position $p_{r,x}$ and velocity $\dot{p}_{r,x}$ were modified by the admittance controller, compared to the low friction wall case, corresponding to the higher friction forces.

The plots of the control inputs $\dot{\boldsymbol{\gamma}}$ for the cascaded control simulations are shown in Fig. 5.6, where the upper and lower limits are time-varying, and the controller keeps the control inputs within the defined constraints.

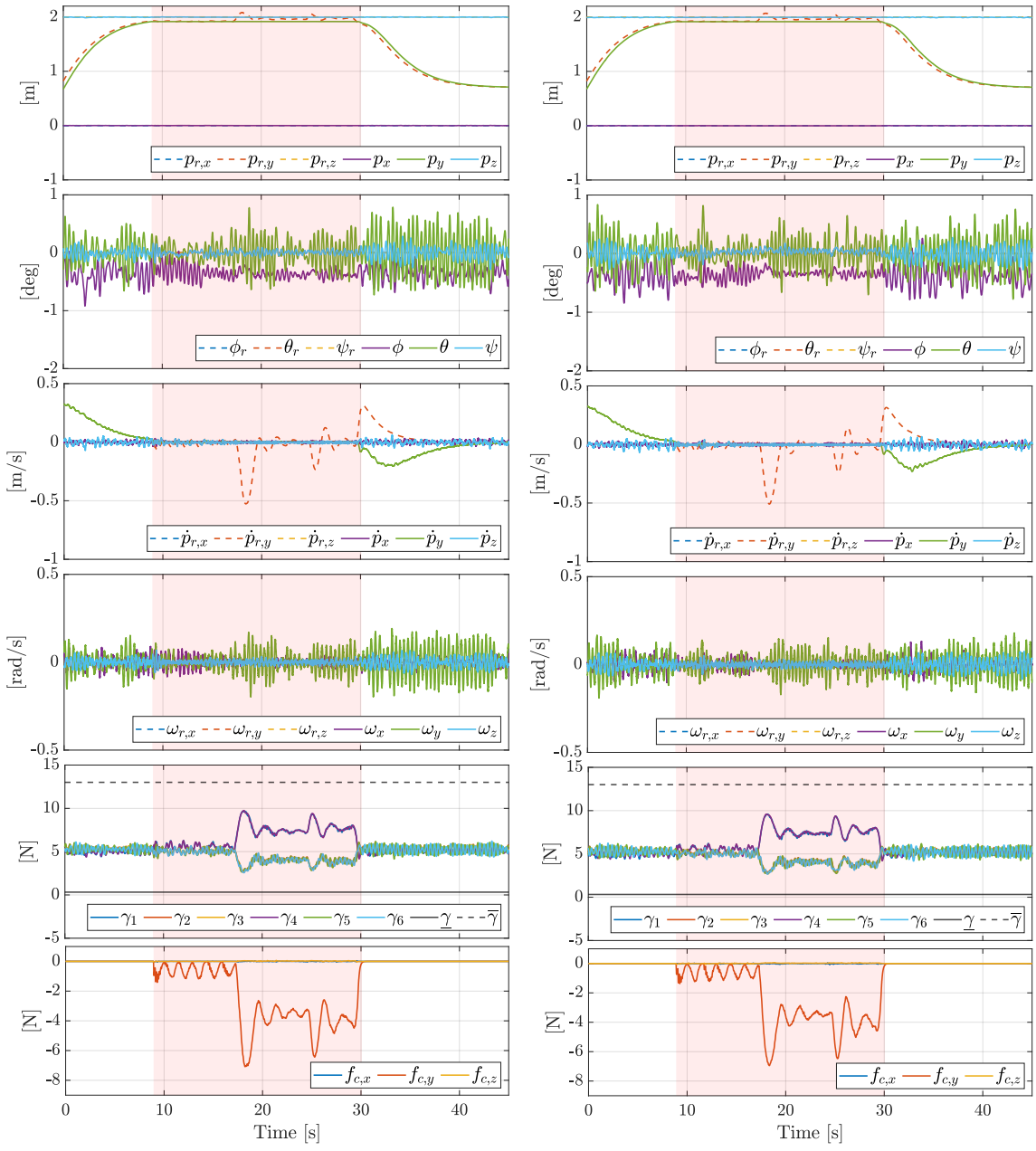
5.3 NMPC impedance control results

Similar to the previous section, this section will present the simulation results of the NMPC impedance controller for the two interaction tasks against the two wall models with low and high friction surfaces. In these simulations, the desired impedance behaviour, and the motion trajectories are identical to their counterparts from the NMPC cascaded control simulations in the previous section. This symmetry between the two sets of simulations will allow a preliminary comparison between the responses of the two controllers for the same task and environment. Additionally, the weights of the different terms in the objective function are listed in Table 5.2, where $\mathbf{Q}_{\mathbf{e}_Z} \in \mathbb{R}^3$ is the weight of the impedance error, and these weights will be used for all the simulations in this section.

Table 5.2: Weights of the objective function for NMPC impedance control during the pushing and push-and-slide simulations.

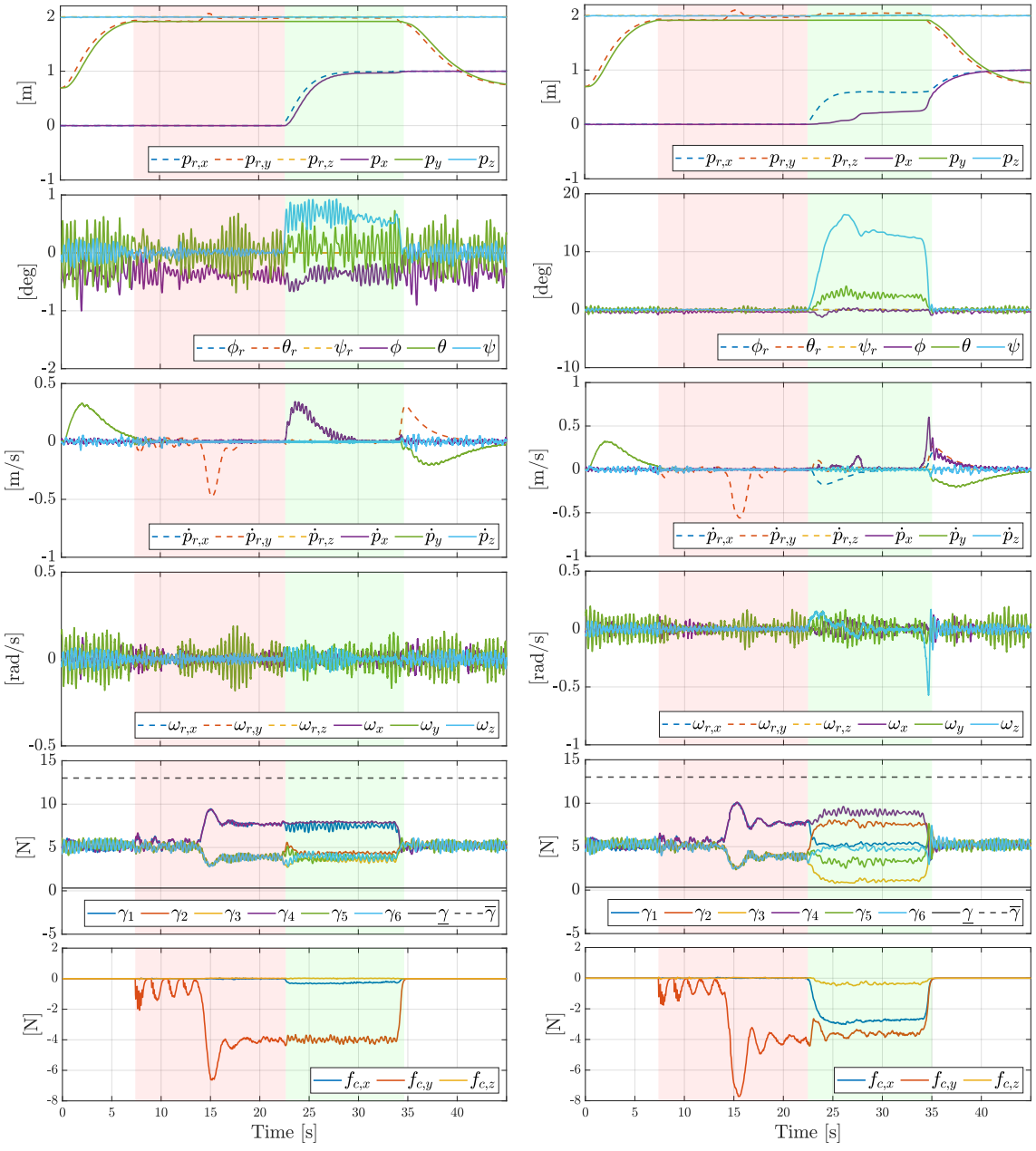
Weight	$\mathbf{Q}_{\mathbf{e}_Z}$	$\mathbf{Q}_{\eta}$	$\mathbf{Q}_{\omega}$	$\mathbf{Q}_{\dot{\omega}}$
Value	$0.1\,\mathbf{I}_3$	$1\,\mathbf{I}_3$	$0.01\,\mathbf{I}_3$	$0\,\mathbf{I}_3$

The results of the pushing task are shown in Fig. 5.7, where Fig. 5.7a represents the results of pushing against a low friction wall, while Fig. 5.7b is the high friction wall results. The two simulations were commanded with the same motion trajectory, and it can be noticed that the behaviour in the two experiments, regardless of the friction of the wall, is almost identical, as noticed in the previous section. However, compared to the cascaded control simulations, it is noted that the interaction forces are relatively smaller, even though the motion trajectory is identical. This can be interpreted by the different NMPC objectives, because the objective of



(a) Low friction wall. **(b)** High friction wall.

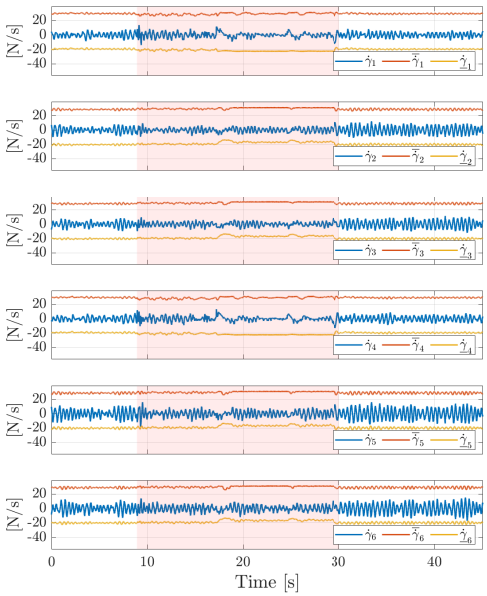
Figure 5.4: Plots of FiberThex performing a pushing task against a low **(left)** and a high friction wall **(right)** with NMPC cascaded control. From top to bottom, the position, orientation, linear velocity, and angular velocity tracking, actuators thrusts, and contact forces. The red highlight represents the pushing phase.



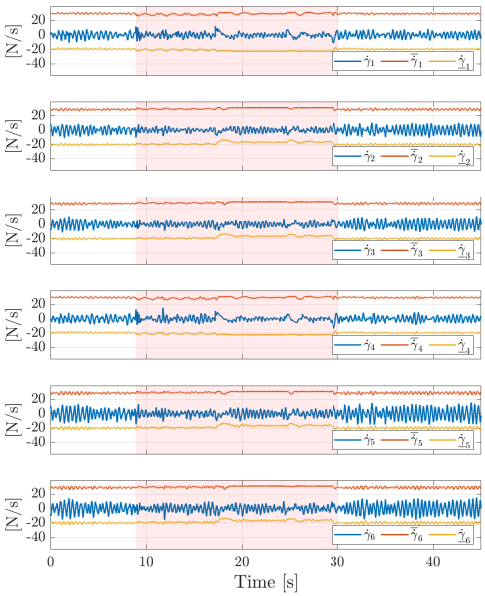
(a) Low friction wall.

(b) High friction wall.

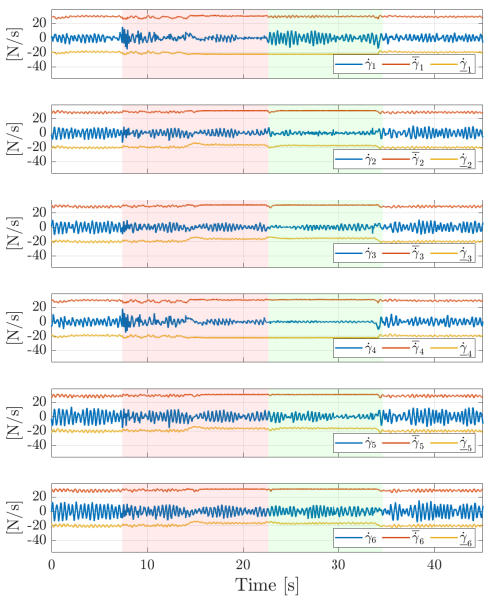
Figure 5.5: Plots of FiberThex performing a push-and-slide task against a low **(left)** and a high friction wall **(right)** with NMPC cascaded control. From top to bottom, the position, orientation, linear velocity, and angular velocity tracking, actuators thrusts, and contact forces. The red highlight represents the pushing phase, while the green highlight represents the sliding phase.



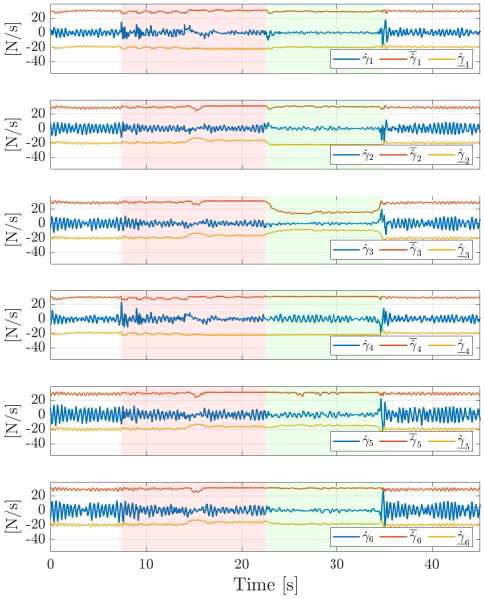
(a) Pushing against low friction wall.



(b) Pushing against high friction wall.



(c) Push-and-slide on low friction wall.



(d) Push-and-slide on high friction wall.

Figure 5.6: Plots of the control inputs for the NMPC cascaded control simulations.

the NMPC cascaded control is to track the motion references, while the objective of the NMPC impedance control is to track the desired dynamical behaviour, which generates different responses for the same trajectory. Additionally, the weights of the OCP play a major role in prioritizing certain tasks over others, and therefore different weights will render different behaviours.

The push-and-slide task results are shown in Fig. 5.8, where Fig. 5.8a represents the results of interacting with a low friction wall, while Fig. 5.8b is the high friction wall results. As expected, the sliding action in this task shows that a smooth trajectory tracking can be achieved when sliding on the smooth surface as shown in the plot of p_x in the sliding phase, and the orientation tracking was not affected by the sliding action, because the friction forces were not significant compared to the high friction surface.

In contrast, when sliding on the rough surface, the high friction forces deteriorated the trajectory tracking of p_x, as shown in Fig. 5.8b, where the friction force $f_{c,x}$ also caused large deviations in the yaw angle ψ. However, the NMPC controller was able to simultaneously maintain a stable interaction and motion, especially attitude stability, while respecting the control inputs $\dot{\boldsymbol{\gamma}}$ constraints as shown in Fig. 5.9.

5.4 NMPC hybrid control results

This section will present the simulation results of the NMPC hybrid controller in the two interaction tasks against the two wall models. In these simulations, a motion reference trajectory is defined for the 6 DoFs (translation and rotational) which will be tracked during FM, and a force reference $f_{r,y}$ is defined for the motion-constrained DoF during CM, which is the translational $y-$axis in these simulations. The force reference is set to be a step function, to test the step response of the force tracking. The weights of the different terms in the objective function are listed in Table 5.3, where $\mathbf{Q_f} \in \mathbb{R}^3$ are the weights of the force tracking errors. The same weights will be used for all the simulations in this section.

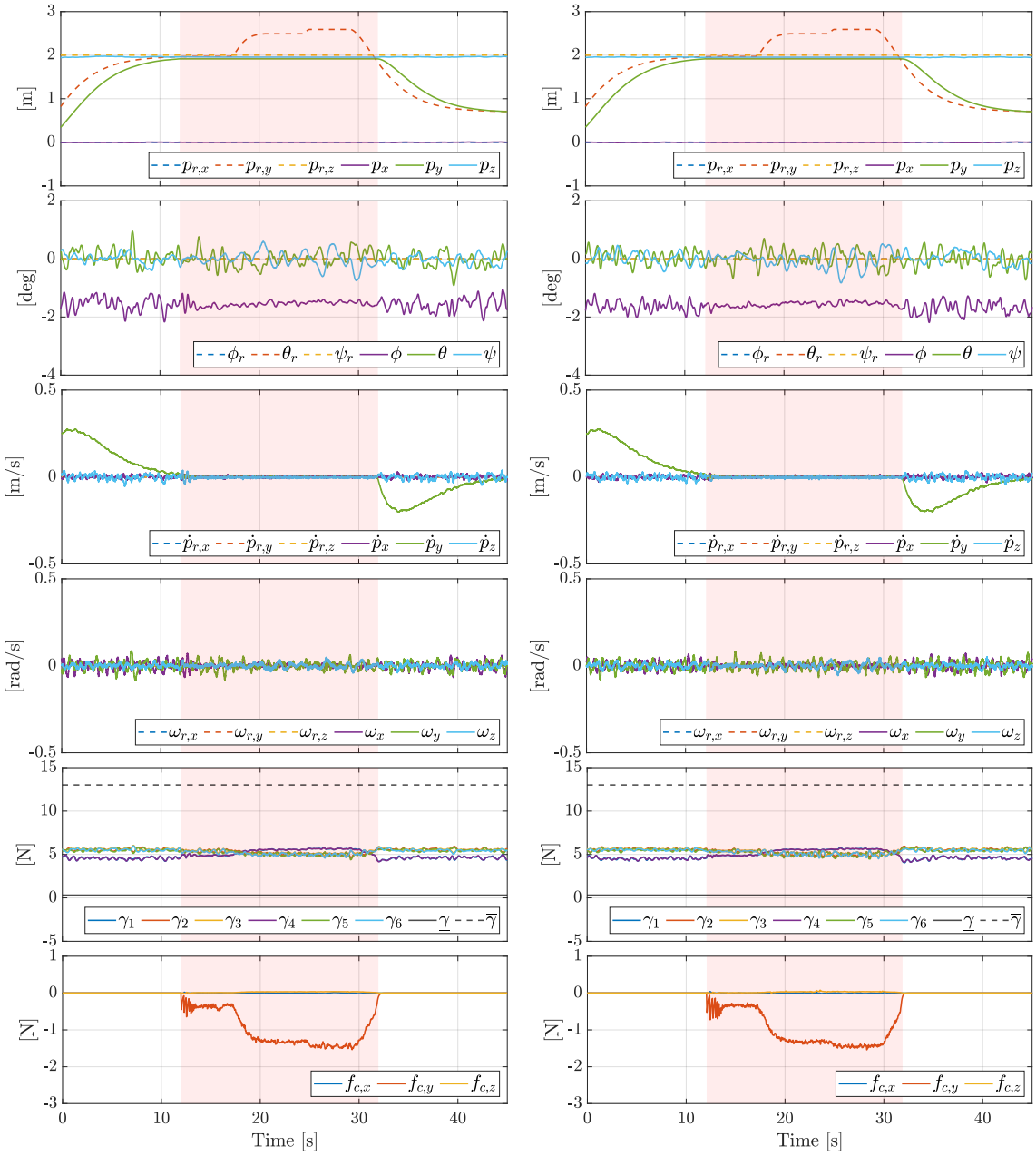
Table 5.3: Weights of the objective function for NMPC hybrid control during the pushing and push-and-slide simulations.

Weight	$\mathbf{Q_p}$	$\mathbf{Q_{\dot{p}}}$	$\mathbf{Q_{\ddot{p}}}$	$\mathbf{Q_\eta}$	$\mathbf{Q_\omega}$	$\mathbf{Q_{\dot{\omega}}}$	$\mathbf{Q_f}$
Value	$0.1\,\mathbf{I}_3$	$0.01\,\mathbf{I}_3$	$0\,\mathbf{I}_3$	$1\,\mathbf{I}_3$	$0.01\,\mathbf{I}_3$	$0\,\mathbf{I}_3$	$0.1\,\mathbf{I}_3$

The results of the pushing task are shown in Fig. 5.10, where Fig. 5.10a represents the results of pushing against a low friction wall, and Fig. 5.10b shows the high friction wall results. The same motion and force references were used in the two experiments, and it can be noticed that the response is almost identical in the two experiments regardless of the friction of the wall. As noticed in the previous two sections, this similarity is expected as long as the direction of actuators forces $\mathbf{f}_{W,\mathrm{a}}$ is normal to the contact surface, but if the generated forces have a component in the lateral directions of the surface, this lateral component will cause a sliding effect, unless there is sufficient friction to counteract this sliding.

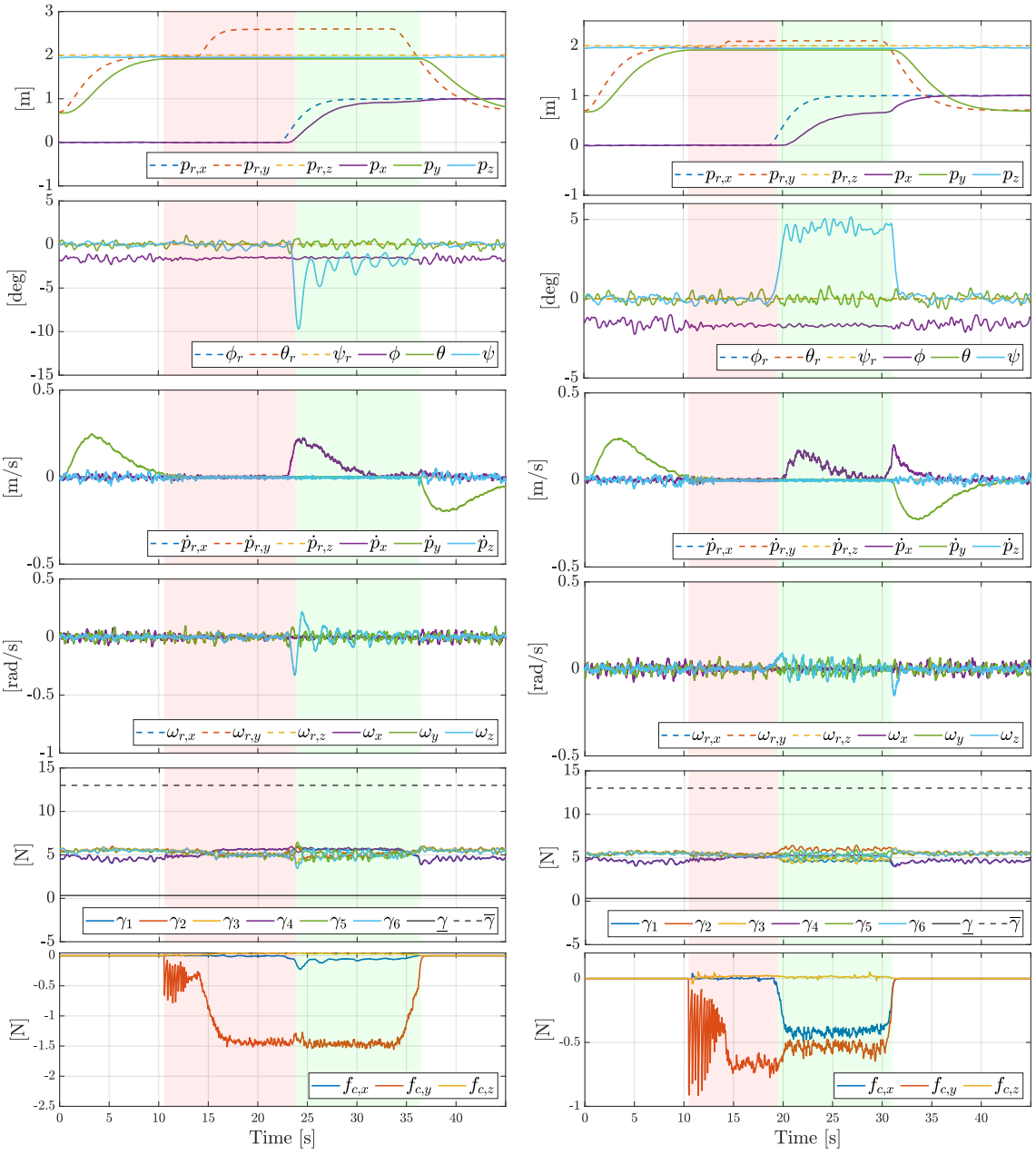
In other words, if $\mathbf{y}_S$ is a vector that is normal to the contact surface, it is favourable that the vector $\mathbf{f}_{W,\mathrm{a}}$ is parallel to $\mathbf{y}_S$, to maintain a constant contact point without sliding. It is important to mention that FiberThex is a fully-actuated AR, it is theoretically possible to generate a certain $\mathbf{f}_{W,\mathrm{a}}$ regardless of AR orientation, however, it is recommended that the vector $\mathbf{y}_E$ of the end-effector frame should be parallel to $\mathbf{y}_S$, to minimize the contact torques $\boldsymbol{\tau}_\mathrm{c}$ that will be generated around the $\mathbf{x}_B$ axis of the body-frame.

It can also be noted that during the interaction in CM (the highlighted part), the force reference is activated and tracked in the $\mathbf{y}_S$ direction only, while in FM the motion reference is tracked in that direction, which is the essence of hybrid control.



(a) Low friction wall. **(b)** High friction wall.

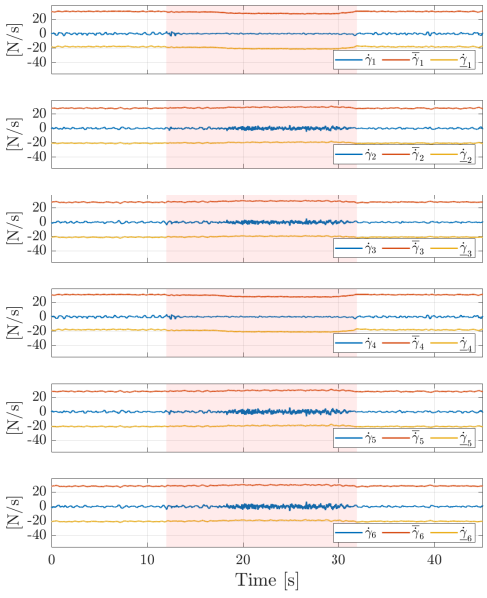
Figure 5.7: Plots of FiberThex performing a pushing task against a low **(left)** and a high friction wall **(right)** with NMPC impedance control. From top to bottom, the position, orientation, linear velocity, and angular velocity tracking, actuators thrusts, and contact forces. The red highlight represents the pushing phase.



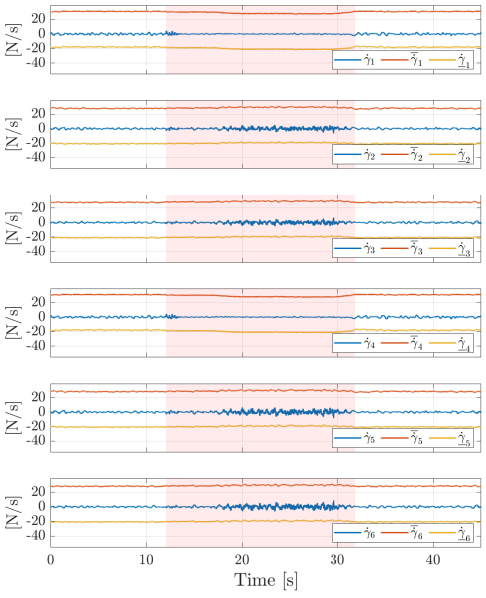
(a) Low friction wall.

(b) High friction wall.

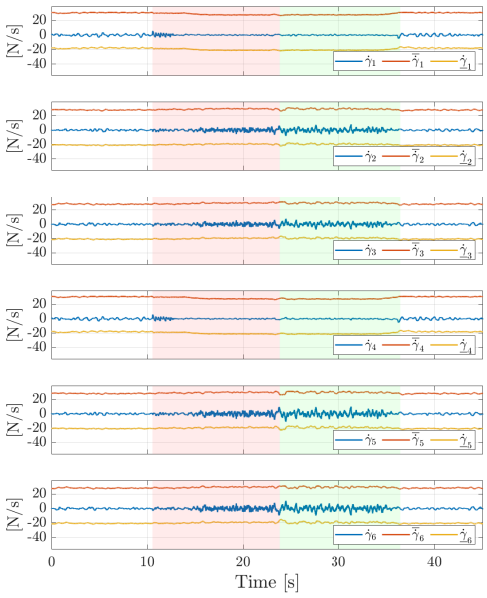
Figure 5.8: Plots of FiberThex performing a push-and-slide task against a low **(left)** and a high friction wall **(right)** with NMPC impedance control. From top to bottom, the position, orientation, linear velocity, and angular velocity tracking, actuators thrusts, and contact forces. The red highlight represents the pushing phase, while the green highlight represents the sliding phase.



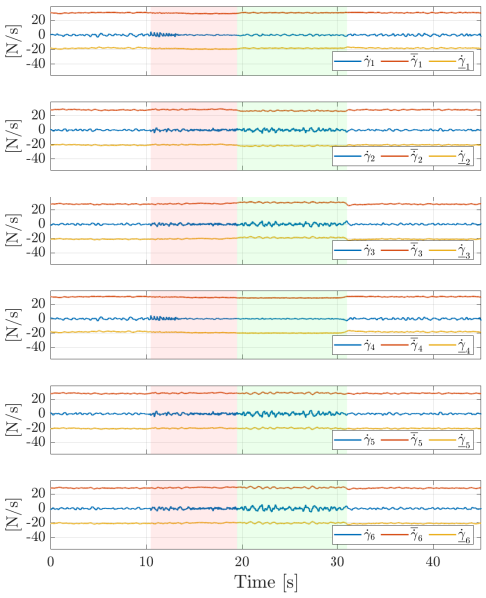
(a) Pushing against low friction wall.



(b) Pushing against high friction wall.

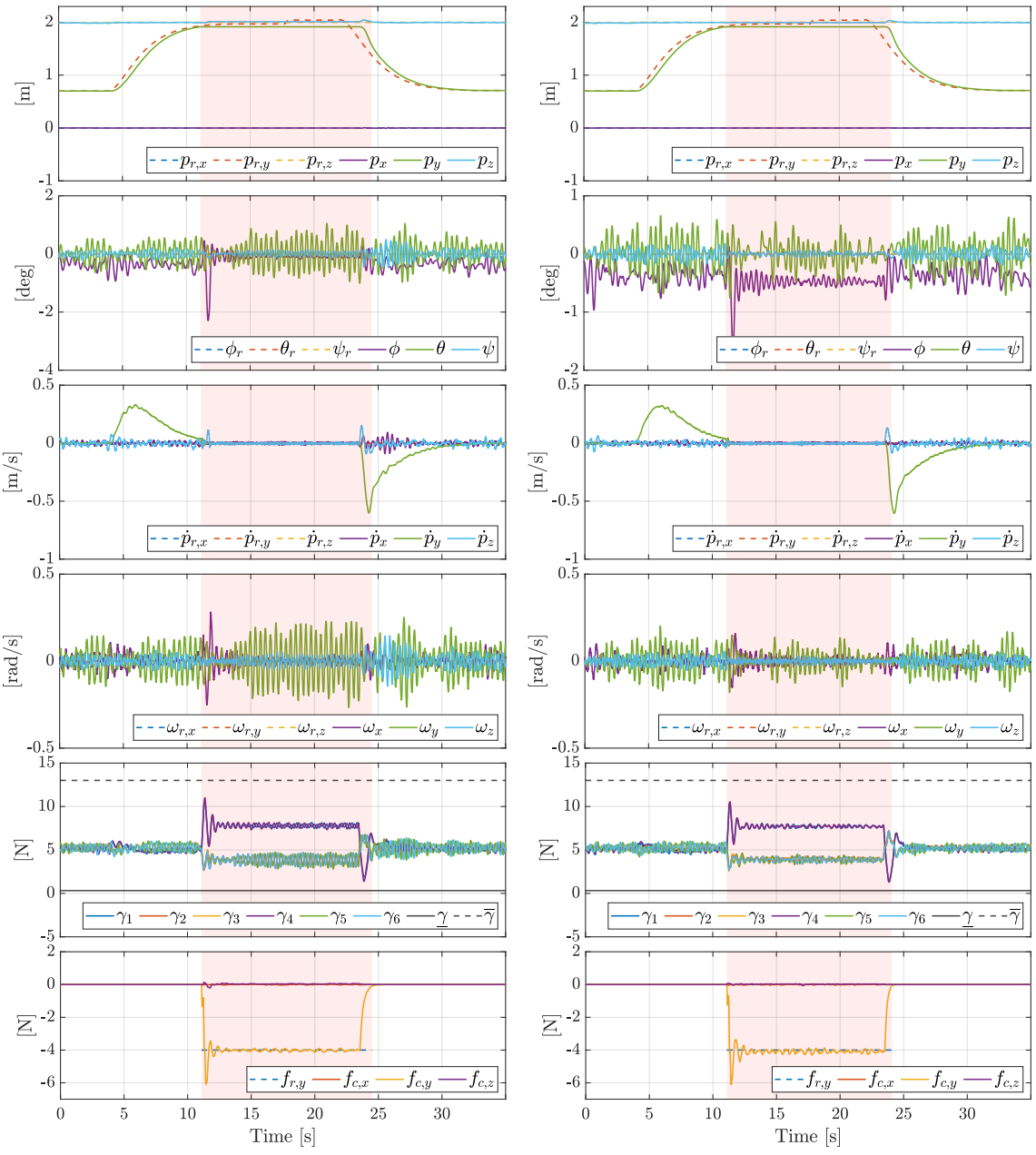


(c) Push-and-slide on low friction wall.



(d) Push-and-slide on high friction wall.

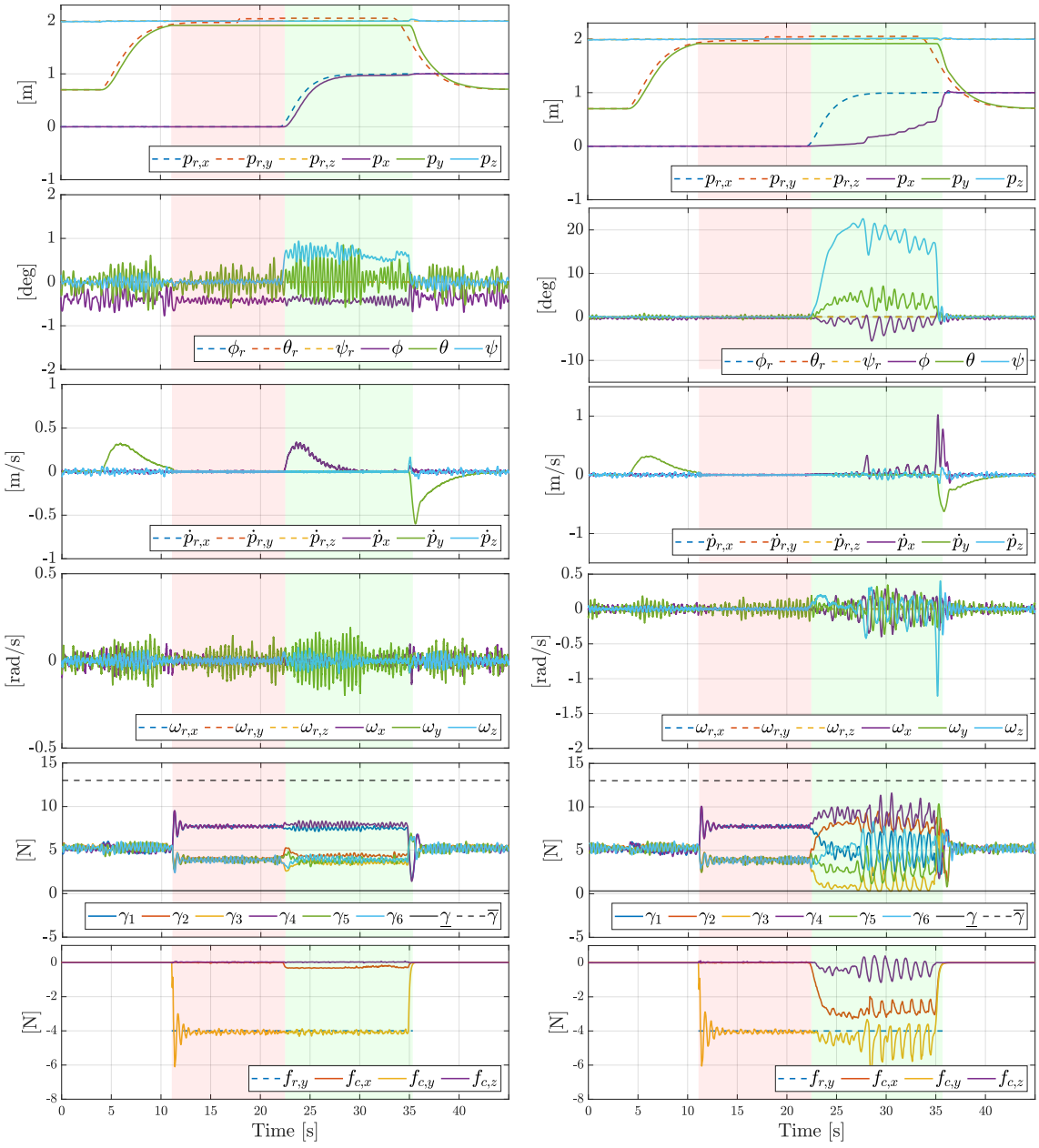
Figure 5.9: Plots of the control inputs for the NMPC impedance control simulations.



(a) Low friction wall.

(b) High friction wall.

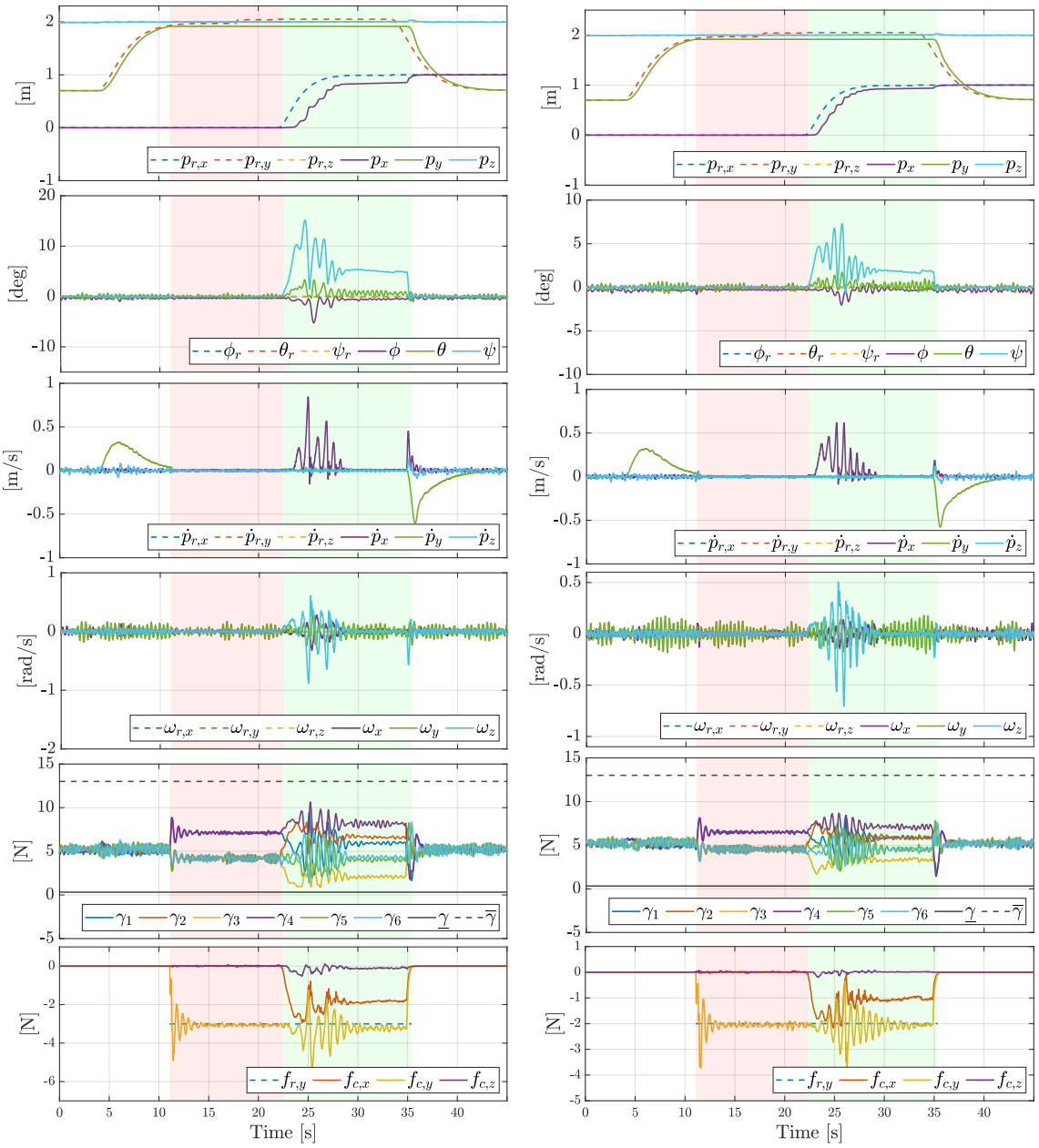
Figure 5.10: Plots of FiberThex performing a pushing task against a low **(left)** and a high friction wall **(right)** with NMPC hybrid control, and a reference contact force of $-4N$. From top to bottom, the position, orientation, linear velocity, and angular velocity tracking, actuators thrusts, and contact force tracking. The red highlight represents the pushing phase.



(a) Low friction wall.

(b) High friction wall.

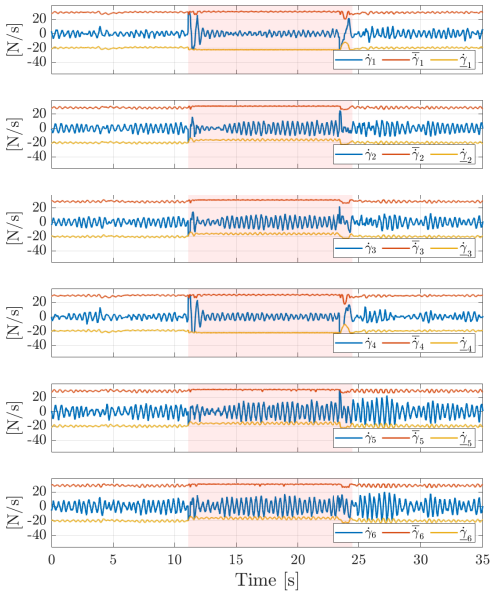
Figure 5.11: Plots of FiberThex performing a push-and-slide task against a low **(left)** and a high friction wall **(right)** with NMPC hybrid control, and a reference contact force of $-4N$. From top to bottom, the position, orientation, linear velocity, and angular velocity tracking, actuators thrusts, and contact force tracking. The red highlight represents the pushing phase, while the green highlight represents the sliding phase.



(a) Reference contact force of $-3N$.

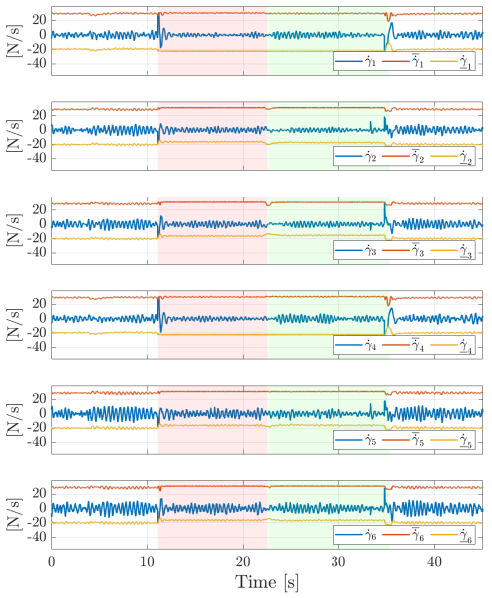
(b) Reference contact force of $-2N$.

Figure 5.12: Plots of FiberThex performing a push-and-slide task against a high friction wall with NMPC hybrid control, and a reference contact force of $-3N$ and $-2N$. From top to bottom, the position, orientation, linear velocity, and angular velocity tracking, actuators thrusts, and contact force tracking. The red highlight represents the pushing phase, while the green highlight represents the sliding phase.

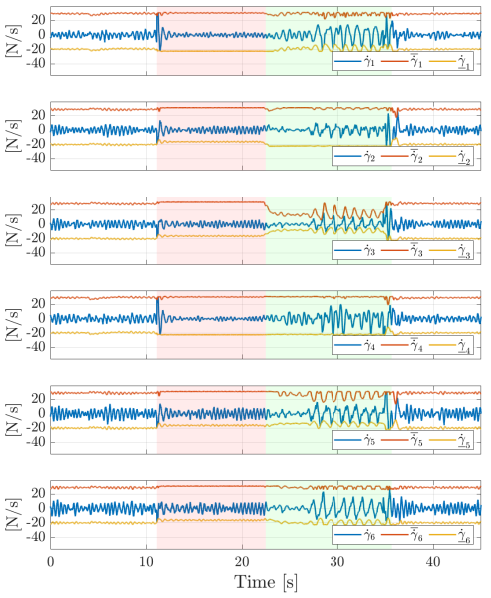


(a) Pushing against low friction wall.

(b) Pushing against high friction wall.



(c) Push-and-slide on low friction wall.



(d) Push-and-slide on high friction wall.

Figure 5.13: Plots of the control inputs for the NMPC hybrid control simulations with a reference force of $-4N$.

The push-and-slide task results are shown Fig. 5.11, where Fig. 5.11a represents the results of sliding on a low friction wall, and the high friction wall results are shown in Fig. 5.11b. As expected, sliding on a low friction surface is a much easier and smoother task than sliding on a high friction surface. It can be seen that when sliding on a smooth surface, there are no significant position or orientation tracking errors as shown in Fig. 5.11a, and the force tracking performance is also acceptable.

On the contrary, sliding on the high friction wall was a major challenge for the AR, as it suffered significant friction forces as shown in Fig. 5.5b, which deteriorated the trajectory tracking of the position p_x, the yaw angle ψ, and the contact force f_y. However, the controller was still able to maintain stability during these harsh circumstances, while exploiting the full-actuation of FiberThex, as shown in the plots of $\boldsymbol{\gamma}$, to simultaneously maintain a stable interaction and motion.

The behaviour of the AR during the sliding phase is greatly influenced by the reference contact force $f_{r,y}$, because the normal contact force will indirectly determine how large the friction forces will be, since the friction forces are functions of the normal force, and therefore smaller $f_{r,y}$ values will improve the behaviour during sliding. To further verify this, the simulation in Fig. 5.11 is replicated with smaller reference forces, $-3N$ in Fig. 5.12a and $-2N$ in Fig. 5.12b, to test the sliding response of the AR with relatively smaller friction forces. It can be seen that the position and orientation tracking are improved when the force reference is smaller, while the force tracking always suffers some oscillations during the sliding phase. It is also noted that the controller was able to handle the stick-slip effect during the sliding phase, where the jerky motion of p_x is clearly visible in Figs. 5.12a and 5.12b.

The control inputs $\dot{\boldsymbol{\gamma}}$ during the first 4 simulations are plotted in Fig. 5.13, where it is noted that even under the very harsh conditions in the push-and-slide task on the high friction surface with a reference force of $-4N$, shown in Fig. 5.13d, the controller respected the constraints, and the control inputs were kept within the defined limits. Also, it can be noticed in the 4 sub-figures of Fig. 5.13 that during the initial collision with the surface ($12s$) almost all the thrusts derivatives in the 4 presented simulations reached their upper or lower limits when the controller started tracking the force reference. This saturation did not cause any noticeable effect on the other states such as position or orientation, and it definitely did not destabilize the system.

5.5 Constraints response analysis

As discussed before, one of the main advantages of using NMPC is its ability to optimize a performance metric considering the defined constraints. The main constraints that were considered in this thesis were the actuators thrusts $\boldsymbol{\gamma}$, and the thrusts derivatives $\dot{\boldsymbol{\gamma}}$, both of them can be identified experimentally. In this section, the NMPC ability to handle constraints is further validated by forcing the controller to operate at the limits of those constraints.

To do that, NMPC hybrid controller was chosen to perform a push-and-slide task on a high friction surface, where the upper limits of actuators thrusts are set to less than 50% of its real limits. This means that instead of $13N$, the upper limits of the actuator's thrust will be equal to $6.1N$, noting that the estimated hovering thrusts are equal to $5.3N$. This task was chosen because it shows clearly how the NMPC can priorities some tasks over the other, corresponding to the tasks weights, while considering the system's constraints. The AR will get in contact with the wall, where it is commanded to track a reference force of $-4N$, and after that is it commanded to move $1m$ in the positive $x-$axis, identical to the tests shown in Section 5.4 with the same weights.

The results of this simulation are shown in Fig. 5.14, where it is noted that at the beginning of the pushing phase (highlighted in red) two actuators thrusts (γ_1&γ_4) quickly saturate in an

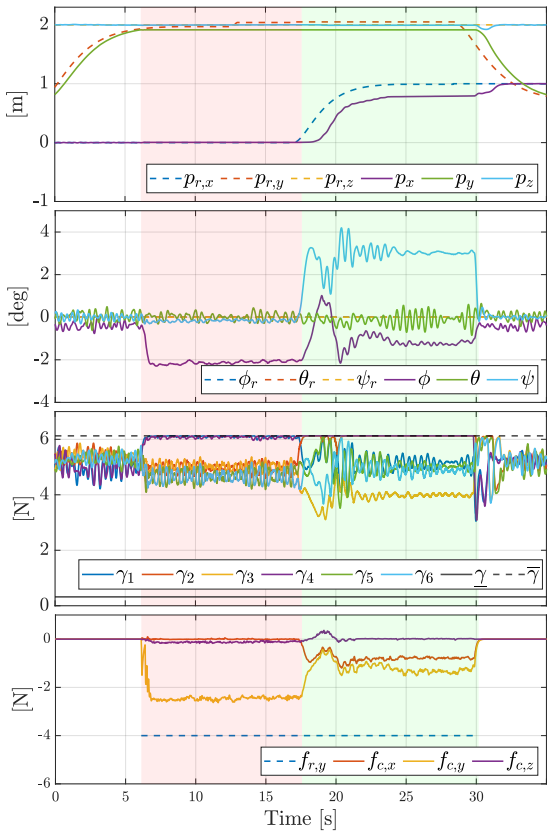


Figure 5.14: Plots of FiberThex performing a push-and-slide task against a high friction wall with NMPC hybrid control, and a reference contact force of $-4N$, while using less than 50% of its actuators thrusts. From top to bottom, the position, orientation tracking, actuators thrusts, and contact force tracking. The red highlight represents the pushing phase, while the green highlight represents the sliding phase.

attempt to track the force reference, however, the AR is unable to reach the reference force while keeping the attitude stable, therefore, it priorities the attitude stabilization over the force tracking, because the attitude tracking weights are larger than those of the force tracking. It is also noted that the NMPC controller has, to a certain limit, exploited the attitude to apply more force to the surface, where controller tilted the AR towards the wall, $\phi \approx -2$, to generate more force towards the normal direction of the wall.

At the beginning of the sliding phase (highlighted in green), γ_2 quickly saturates, while γ_1 is unsaturated and γ_5&γ_6 are close to the upper limit. Still, the controller again chooses to stabilize the attitude, which was disturbed by the friction-induced torques, over force, or lateral motion tracking within the limits of the actuators.

Additionally, in the NMPC hybrid control push-and-slide simulation on the rough surface with a reference force of $-4N$, shown in Fig. 5.10b, the actuator γ_3 reached its lower limit multiple times during the sliding phase, and its derivative $\dot{\gamma}_3$, shown in Fig. 5.13d, also reached its upper and lower limit multiple times during the sliding phase , which clearly illustrates the NMPC capability to comply with the system constraints in harsh and critical conditions.

5.6 Discussion

This chapter presented the results of the preliminary validation simulations for the three NMPC controllers in two interaction tasks with different friction conditions. It was clear that different control approaches render different behaviour even with identical trajectories and parameters, the common advantage of the three controllers is their ability to cope with the constraints of the system, and prioritize certain tasks over others corresponding to their relative weights.

The variations in the behaviour of the controllers provide multiple options that correspond to the needs of different interaction tasks, while the essence of the three control approaches that are addressed in this thesis was preserved, their NMPC implementation provided optimized solutions that are implicitly accounting for the defined constraints, while exploiting the predictive nature of NMPC. Additionally, the system's robustness towards external unmodelled disturbances was validated through its response to the unmodelled friction forces, while sliding on different surfaces, that the controller is agnostic about its condition.

6 Conclusion

In this chapter, the thesis is concluded with an overview of the presented work, where the research questions of the thesis are revisited, and an outlook for future work is presented.

6.1 Overview

In this section, the research questions that are proposed in Section 1.1.3 are reviewed, and a summary of the thesis work and conclusions that are related to each question is presented. The research questions that were addressed in this thesis are:

RQ1 How to exploit NMPC capabilities for APhI control?

After surveying the APhI control literature using NMPC, it was noted that all the contributions are focused on the hybrid force/pose control approach, while other approaches such as impedance control or admittance control with NMPC were not explored. This thesis proposed, implemented, and validated three NMPC-based control approaches for APhI in Chapter 3, which reflects the three main approaches in classical physical interaction control, namely, direct control, and indirect control with impedance, or admittance control as discussed in Chapter 2.

The main advantage of using NMPC for APhI control is the ability to implement the system's constraints in the controller design, where these constraints will be accounted for when solving the optimization problem, while also considering the predicted future evolution of the system using the prediction model. It was shown that the proposed controllers can optimize the performance while satisfying the physical constraints of the system, even when performing complicated tasks with unmodeled disturbances, very restrictive constraints, and harsh environments. It was also shown that the NMPC controllers are able to priorities certain tasks over the others, corresponding to the relative weight of each task, such as prioritizing the attitude stability over position tracking, which is vital during APhI.

However, designing NMPC-based controllers is a major challenge from an implementation and validation stand-point, where the feasibility of the proposed controller cannot be guaranteed, and it is determined for the most part by the NMPC parameters, such as prediction horizon length, and sampling time. Tuning the NMPC parameters is not a trivial task, and it is affected by many variants such as the task definition, prediction model, states, and control inputs, and there is no systematic approach for tuning these parameters, to the best of the author's knowledge, but rather broad guidelines with trial-and-error iterations, which can be a challenging and time-consuming task.

RQ2 How to choose an NMPC-based control approach for a specific task?

It was noticed that the proposed NMPC controllers have different characteristics that correspond to the different objectives of controllers. For instance, since the admittance controller in the NMPC cascaded control is responsible for the interaction control, the controller response during the interaction was shaped by the admittance response, where the controller tries to minimize the impedance error by modifying the motion trajectory. For contact-based APhI that requires maintaining contact with the environment, this admittance behaviour might be unfavorable, because sometimes it leads to losing contact with the environment or diverging from the initial trajectory corresponding to external forces, such as friction. However, for other APhI tasks, such as physical interaction with humans, this kind of compliant behaviour might be favourable from a safety and reli-

ability point of view. Nonetheless, the ability to embed the AR physical and operational constraints in the controller design will improve the reliability of the physical interaction.

In contrast, the NMPC impedance control takes a different approach to indirect interaction control, where the controller is always trying to minimize the error between the AR dynamical behaviour and the desired behaviour, without modifying the motion trajectory. Instead, the NMPC controller exploits the prediction model of the AR to minimize the impedance error during free-flight, while during contact, the controller also exploits the contact force model for minimizing the error. When the interaction is with an unknown environment, the contact force prediction model can not be included in the NMPC prediction model, still, the controller was able to operate under such uncertainties, which means that the NMPC impedance controller can imitate a reactive impedance controller while complying with the constraints of the system. It was also noted that this approach resulted in better contact maintenance, which can be essential for contact-based APhI tasks.

Finally, it is clear that the NMPC hybrid control is suitable for the tasks that require accurate tracking of motion and force trajectories during the physical interaction. The separation between the force tracking problem and the motion tracking problem allows for prioritizing a certain task over the other, or to shape the response of each task individually. However, HFPC requires prior knowledge of the contact surface geometry and shape, such that the constrained DoFs can be excluded from the motion tracking problem during the APhI. While this requirement can be easily satisfied for some tasks, such as interacting with simple surfaces of uniform geometry, it might be restrictive to others where there are uncertainties about the environment.

RQ3 Can NMPC be combined with Admittance control to create a "Constrained Admittance Control"?

It was shown that in a cascaded structure, admittance control can be combined with a trajectory tracking NMPC controller, where the admittance controller will control the APhI. In this architecture, a compliant behaviour can be achieved during the interaction, and constraints can be defined to satisfy physical, operational, or safety limits. Another advantage of this approach is that it does not require a prediction model for the environment because of the reactive nature of the admittance controller, which means that the system can interact with unknown environments.

6.2 Future work

This line of work can be extended and developed in many aspects, such as:

Experimental validation of the control approaches

Validation of the proposed control approaches through an experimental campaign will provide more realistic insights into the capabilities and limitations of these control approaches in the real world.

Contact prediction

Currently, switching from one contact mode to another is dependent on feedback, and therefore it is a reactive action that cannot be predicted in the simulated future horizon in NMPC. If the position of the contact surface is known, either by visual feedback from an onboard camera, or from a priori knowledge, contact can be predicted internally, and therefore transitioning from free-flight to interaction can be better regulated and controlled.

Momentum-based contact force prediction model

It was noticed during the simulations that the contact force prediction model, which ignores the collision dynamics and models the steady-state interaction only, affects the AR behaviour during the initial collision phase of the interaction because of the unmodelled dynamics. A more descriptive contact model that takes into account these dynamics can improve the performance during that phase.

Friction prediction models

So far, the contact forces model that is included in the NMPC prediction model was for the normal force only, while the friction forces that are experienced during sliding on a surface are not modelled in the prediction model of the controller, and they are treated as external disturbances that the system should coup with. However, a friction model can be developed and included in the prediction model, and therefore benefiting from the NMPC prediction to optimize the interaction task.

Disturbance observer

During real experiments, the unmodelled dynamics might cause the predictive states and the real states to diverge quickly, which can destabilize the system. One solution for that is to implement a disturbance observer that can estimate the external disturbances that might be caused by the unmodelled dynamics, or other external sources, such as wind. The interaction forces and torques can be separated from the external disturbances using force feedback from a force sensor.

Online model learning/adaptation

The parameters of the prediction model that is implemented in each NMPC design are constant, and they are determined a priori. However, these parameters can be estimated and updated online, so that the controller can deal with adapt to the change of these parameters online, or get a better estimate of these parameters to minimize the uncertainty and the deviation between the prediction model and the actual physical system. Also, different models can be used to describe the normal and friction forces, and the parameters of these models can also be learned/updated online.

Admittance-based NMPC

An NMPC cascaded control was proposed in this thesis as an admittance-based interaction control with NMPC. However, an interesting idea is to embed the admittance controller in the NMPC which will allow the admittance controller to benefit from the advantages of NMPC, such as prediction and constraints, while maintaining the admittance control properties which are desirable in some APhI tasks, such as interaction with humans.

Transition to unit quaternions

While the Euler angles are suitable for the interaction scenarios that are considered in this thesis, they are not suitable for a global definition of the attitude. Therefore, it is better to abandon Euler angles representation and transition to a unit quaternion representation.

Implementation of interaction constraints

One of the main advantages of using NMPC is its ability to comply with system constraints and optimize the control action to achieve an objective while respecting the defined constraints. However, this advantage has not been fully explored during this thesis, and therefore, constraints can be defined on the interaction behaviour to regulate it to a certain desirable be-

haviour. These constraints can be defined to solve some of the common problems such as the unintended loss of contact.

Interaction with complex surfaces

The interaction tasks presented in this thesis were limited to simple flat structures which is a common assumption in the physical interaction control field. However, the ability to interact with non-flat surfaces with complex geometries is very intriguing for many practical applications, and should be explored.

A Brief introduction to Model Predictive Control

This chapter is dedicated to introduce the reader to the basic concepts of MPC.

A.1 Background

MPC refers to a family of control methods which share a common framework, in which a dynamical model of the plant is used to find the optimal control signal that would minimize a cost function, and hence maximizing certain performance metric(s) [46]. The general structure of a system that is controlled by MPC can be seen in Fig. A.1. Where at each time step k :

1. The states (outputs) of the plant $x(kT)$ are measured (estimated) from the actual plant at time kT.
2. MPC will solve the OCP to find the optimal control sequence $\{u_0, ..., u_{N-1}\}$ over the future prediction horizon of N steps. This step includes:

Algorithm 1: Solving optimization problem over a prediction horizon of N steps

Result: $\{u_0, ..., u_{N-1}\}$
$h = 0$;
$x_h = x(kT)$;
while $h < N$ **do**
  Solve OCP to find u_h;
  Find the predicted state x_{h+1} from the plant model;
  h++;
end

3. Only the first optimal control u_0 is sent as control signal to the plant, while the others $\{u_1, ..., u_{N-1}\}$ are discarded, such that $u(kT) = u_0$

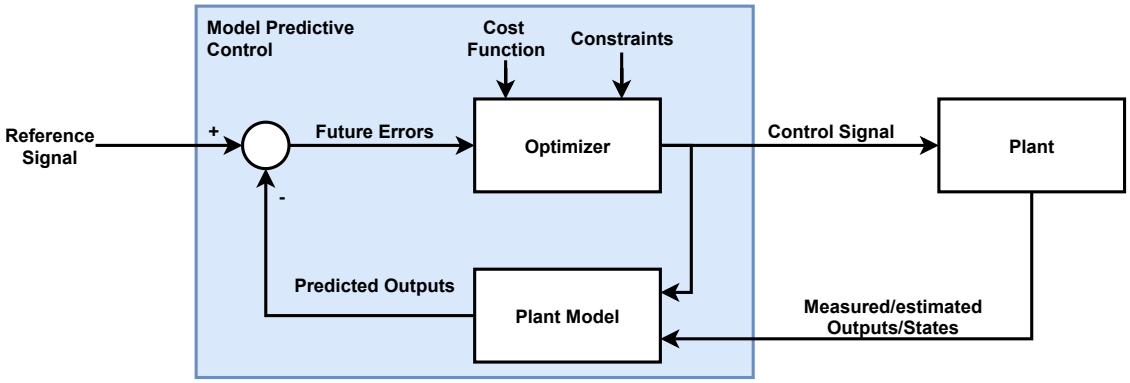


Figure A.1: Basic MPC structure

A.2 Optimal Control Problem

The MPC's OCP is usually characterized by a cost function (Eq. (A.1)), the prediction model (Eqs. (A.2) and (A.3)), a set of constraints (Eqs. (A.5) to (A.7)), and the state feedback definition (Eq. (A.8)). The OCP can be written as:

$$\min_{u_0,\dots,u_{N-1}} \quad J(u,x) \tag{A.1}$$
$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k) \tag{A.2}$$
$$y_k = g(x_k, u_k) \tag{A.3}$$
$$x_{min} < x_k < x_{max} \tag{A.4}$$
$$u_{min} < u_k < u_{max} \tag{A.5}$$
$$\Delta u_{min} < \Delta u_k < \Delta u_{max} \tag{A.6}$$
$$y_{min} < y_k < y_{max} \tag{A.7}$$
$$x_0 = x(kT) \tag{A.8}$$

A.2.1 Cost Function

The cost function represents a performance metric(s) that MPC will try to minimize. For example, in a position trajectory tracking task, the performance metric to be minimized is the position error between the reference and the actual position. The cost function usually consists of two main parts, namely, the running cost, and the terminal cost.

The terminal cost represents the cost at the end of the optimization horizon. In the example of position trajectory tracking task, the terminal cost would depend on the error at the end of the horizon, i.e. at the final step N. On the other hand, the running cost is the summation of the certain costs at each optimization step in the horizon except the final step, i.e. $[0,\dots,N-1]$.

An example of a standard weighted quadratic cost function is:

$$J(u,x) = ||T(y_N - r(t))||_2^2 + \sum_{h=0}^{N-1} ||Q(y_h - r(t))||_2^2 + ||Ru_h||_2^2 \tag{A.9}$$

where:

- $r(t)$ is the reference signal at time t
- h is the simulated time step in the future horizon $[0, N-1]$
- y_h is the predicted output at the simulated time step (hT)
- u_h is the optimal control signal at the simulated time step (hT)
- $||a||_2$ is the euclidean norm of vector $a \in \mathbb{R}^n$, such that: $||a||_2 = \sqrt{a_1^2 + \cdots + a_n^2}$
- $T, Q, R \succeq 0$ are the weights matrices which are all positive semi-definite

A.2.2 Prediction Model

Since the prediction model will be used to optimize the control inputs, it is essential that the model capture the dynamic behaviour of the plant while also keeping it simple so that it can be used efficiently in online control. Needless to say that the system evolution prediction is as precise as the prediction model, which raises a potential trade-off for complex systems, where a detailed model will increase the required time for solving the OCP, limiting the sampling rate, and the prediction horizon, and on the other hand, the prediction model should capture the main characteristics of the system such that MPC can predict the systems evolution. Additionally, a complicated prediction model might affect the feasibility of the OCP, and therefore the solver might not converge to a solution. State-space models are usually used to represent the prediction models since most toolboxes are designed to use them, and the model should be linear, or linearized, in case of a nonlinear system.

A.2.3 Constraints

One of the great advantages of using MPC is the ability to define constraints on the controlled variables (Plant output), or the control signal. Such constraints might be derived from physical limitation (e.g. actuators limits), or from a performance tuning point of view; where a desired behaviour can be enforced by these constraints (e.g. keeping a mobile robot inside a safe workspace), they can be constants, time-varying, or state-dependant.

The constraints will be considered while solving the OCP to find the optimal control action within the defined limits. This is a much better approach than clipping (saturating) the control signal. The latter approach might destabilize the system because it breaks the feedback loop continuity, while MPC can find a suitable control action to control the system [47] within the respective constraints, if it is feasible.

A.3 Nonlinear Model Predective Control

Thanks to recent developments in optimal control algorithms, and the increasingly improving computational power of embedded control computers, nonlinear prediction models of fast dynamics nonlinear systems can be implemented in MPC for real time applications. This variant of MPC can be called NMPC, where the prediction model in Eqs. (A.2) and (A.3), can be a nonlinear model, and the constraints in Eqs. (A.4) to (A.7) can also be nonlinear. The main advantage of using NMPC over MPC is that NMPC explicitly treats the nonlinear dynamics and constraints compared to MPC where linear approximations are used.

B Mathematical background

B.1 Operators

B.1.1 $\times$ operator

$$\boldsymbol{u} = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} \tag{B.1}$$

$$\boldsymbol{u}_\times = \begin{pmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{pmatrix} \tag{B.2}$$

B.1.2 $\vee$ operator

$$\boldsymbol{A} = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \tag{B.3}$$

$$\boldsymbol{A}^\vee = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \tag{B.4}$$

B.1.3 Hamilton product of quaternions $\otimes$

The Hamilton product of two unit quaternions, $\mathbf{q}$ and $\mathbf{p}$, is defined as:

$$\mathbf{q} \otimes \mathbf{p} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \otimes \begin{bmatrix} p_w \\ p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} q_w p_w - q_x p_x - q_y p_y - q_z p_z \\ q_w p_x + q_x p_w + q_y p_z - q_z p_y \\ q_w p_y - q_x p_z + q_y p_w + q_z p_x \\ q_w p_z + q_x p_y - q_y p_x + q_z p_w \end{bmatrix} \tag{B.5}$$

B.1.4 Quaternion conjugate $*$

The conjugate of quaternion $\mathbf{q}$, which is denoted as $\mathbf{q}^*$ is defined as:

$$\mathbf{q}^* = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix}^* = \begin{bmatrix} q_w \\ -q_x \\ -q_y \\ -q_z \end{bmatrix} \tag{B.6}$$

B.1.5 Quaternion inverse

The inverse of quaternion $\mathbf{q}$, which is denoted as $\mathbf{q}^{-1}$ is defined as:

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{||\mathbf{q}||^2} = \frac{\mathbf{q}^*}{\left(\sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2}\right)^2} \tag{B.7}$$

C MATMPC Implementation

MATMPC is a MATLAB-based NMPC tool where the OCP is transcribed by multiple shooting and the resulting NLP problem is solved by Sequential Quadratic Programming (SQP) method. MATMPC uses MATLAB code generators to write its routines in MATLAB C API and call them at run-time through MEX functions.

In this appendix, a brief introduction to the standard definition of the NLP problem in MATMPC and the main components of the implementation are presented. Understanding how MATMPC defines the NLP problem is crucial to adapt that definition to the different NMPC controllers that are designed in this thesis. However, the details of MATMPC internals are out of the scope of this thesis, and the interested reader is referred to [38].

C.1 Standard problem definition

The MATMPC standard NLP problem is defined as:

$$\begin{aligned}
\min_{\mathbf{x}_k,\mathbf{u}_k}\ & \frac{1}{2}\sum_{k=0}^{N-1} d(\mathbf{h}(\mathbf{x}_k,\mathbf{u}_k)-\mathbf{h}_{ref}^k)_{\mathbf{W}_k}+\frac{1}{2}d_N(\mathbf{h}_N(\mathbf{x}_N)-\mathbf{h}_{ref}^N)_{\mathbf{W}_N}\\
s.t.\quad & \mathbf{x}_0=\hat{\mathbf{x}}_0,\\
& \mathbf{x}_{k+1}=\boldsymbol{\phi}_k(\mathbf{x}_k,\mathbf{u}_k),\ k=0,1,\dots,N-1,\\
& \underline{\mathbf{x}}_k\le\mathbf{x}_k\le\overline{\mathbf{x}}_k,\ k=0,1,\dots,N,\\
& \underline{\mathbf{u}}_k\le\mathbf{u}_k\le\overline{\mathbf{u}}_k,\ k=0,1,\dots,N-1\\
& \underline{r}_k\le r_k(\mathbf{x}_k,\mathbf{u}_k)\le\overline{r}_k,\ k=0,1,\dots,N-1,\\
& \underline{r}_N\le r_N(\mathbf{x}_N)\le\overline{r}_N,
\end{aligned}\tag{C.1}$$

where $\mathbf{h}:\mathbb{R}^{n_x}\times\mathbb{R}^{n_u}\to\mathbb{R}^{n_y}, \mathbf{h}_N:\mathbb{R}^{n_x}\to\mathbb{R}^{n_{y_N}}$ are vector functions of state and control $(\mathbf{x},\mathbf{u})$, with corresponding references $\mathbf{h}_{ref}^k$ and $\mathbf{h}_{ref}^N$. Note that $\mathbf{h},\mathbf{h}_N$ can be nonlinear and nonconvex. The outer objective functions $d:\mathbb{R}^{n_y}\to\mathbb{R}, d_N:\mathbb{R}^{n_{y_N}}\to\mathbb{R}$ are assumed convex, e.g. linear sum or quadratic. $\mathbf{W}_k,\mathbf{W}_N$ are weights for each term in d for stage k. $\hat{\mathbf{x}}_0$ is the measurement of the current state. The constraints functions $r(\mathbf{x}_k,\mathbf{u}_k):\mathbb{R}^{n_x}\times\mathbb{R}^{n_u}\to\mathbb{R}^{n_c}$ and $r(\mathbf{x}_N):\mathbb{R}^{n_x}\to\mathbb{R}^{n_{c_N}}$ can be linear or nonlinear, with lower and upper bound $\underline{r}_k,\overline{r}_k$.

C.2 System dynamics

To adapt any NLP problem to this definition, the states $\mathbf{x}\in\mathbb{R}^{n_x}$ and control input $\mathbf{u}\in\mathbb{R}^{n_u}$ vectors should be defined. Then, the dynamical model of the system should be defined in the map function $\boldsymbol{\phi}_k(\mathbf{x}_k,\mathbf{u}_k)$.

C.3 Outer objective function

The outer objective functions $d(\bullet), d_N(\bullet)$ in this thesis are always chosen to be quadratic weighted sum, such that:

$$d(\bullet)=0.5*(\mathbf{h}(\mathbf{x}_k,\mathbf{u}_k)-\mathbf{h}_{ref}^k)^\top*\mathrm{diag}(\mathbf{W})*(\mathbf{h}(\mathbf{x}_k,\mathbf{u}_k)-\mathbf{h}_{ref}^k)\tag{C.2}$$

$$d_N(\bullet)=0.5*(\mathbf{h}_N(\mathbf{x}_N)-\mathbf{h}_{ref}^N)^\top*\mathrm{diag}(\mathbf{W}_N)*(\mathbf{h}_N(\mathbf{x}_N)-\mathbf{h}_{ref}^N)\tag{C.3}$$

where diag$(\bullet)$ is the diagonal matrix of the vector $\bullet$.

C.4 Inner objective function, references and parameters

The definition of the reference vectors $\mathbf{h}_{ref}^k,\mathbf{h}_{ref}^N$ and vector functions $\mathbf{h}(\mathbf{x}_k,\mathbf{u}_k),\mathbf{h}_N(\mathbf{x}_N)$ depends on the objective function definition. For example, for a states trajectory tracking objective function (states – references), the states references can be simply passed to MATMPC

Table C.1: MATMPC sizes symbols

Symbol	Meaning
n_x	No. of states
n_u	No. of controls
n_y	No. of outputs (and references)
n_{yN}	No. of outputs at terminal stage
n_p	No. of parameters
n_c	No. of general constraints
n_{bx}	No. of bounds on states
n_{bu}	No. of bounds on controls
n_{bx_idx}	indexes of bounded states
n_{bu_idx}	indexes of bounded controls

through the reference vectors $\mathbf{h}^k_{ref}, \mathbf{h}^N_{ref}$ and the corresponding states will be defined in the vector functions $\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k), \mathbf{h}_N(\mathbf{x}_N)$, and the end result will be $(\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{h}_{ref}) \rightarrow$ (states – references).

However, when the objective does not fit to a simple definition of (states – references) (such as the impedance error that was discussed in Section 3.3, where one error depends on multiple states ($\ddot{\mathbf{p}}, \dot{\mathbf{p}}, \mathbf{p}$), references ($\ddot{\mathbf{p}}_r, \dot{\mathbf{p}}_r, \mathbf{p}_r$) and gains ($\mathbf{M}, \mathbf{D}, \mathbf{K}$)), the objective should be implemented in the vector functions $\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k), \mathbf{h}_N(\mathbf{x}_N)$ where the required references can be passed to the vector functions in the *parameters* vector $\mathbf{a}_k \in \mathbb{R}^{n_p}$ which is a standard MATMPC vector that can be passed used to pass external variables to the NLP. This means that reference functions $\mathbf{h}^k_{ref}, \mathbf{h}^N_{ref}$ will have zeros in the corresponding indices.

C.5 Example

For example, to implement the impedance error *Eq.* (3.9) in MATMPC, after defining the appropriate map function $\boldsymbol{\phi}_k(\mathbf{x}_k, \mathbf{u}_k)$ and control input $\mathbf{u} \in \mathbb{R}^{n_u}$ vector, the states vector can be defined as:

$$\mathbf{x}_k = \begin{bmatrix} \ddot{\mathbf{p}} & \dot{\mathbf{p}} & \mathbf{p} \end{bmatrix}^\top \tag{C.4}$$

While the vector function $\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) \in \mathbb{R}^3$ is defined as:

$$\mathbf{h}(\mathbf{x}_k, \mathbf{a}_k) = \mathbf{M}(\ddot{\mathbf{p}}_r - \ddot{\mathbf{p}}_k) + \mathbf{D}(\dot{\mathbf{p}}_r - \dot{\mathbf{p}}_k) + \mathbf{K}(\mathbf{p}_r - \mathbf{p}_k) + \mathbf{f}_{\mathrm{c,k}} \tag{C.5}$$

where the $\ddot{\mathbf{p}}_r$, $\dot{\mathbf{p}}_r$, $\mathbf{p}_r \in \mathbb{R}^3$ will be passed to MATMPC in the parameters vector $\mathbf{a}_k \in \mathbb{R}^9$ such that:

$$\mathbf{a}_k = \begin{bmatrix} \ddot{\mathbf{p}}_r & \dot{\mathbf{p}}_r & \mathbf{p}_r \end{bmatrix}^\top \tag{C.6}$$

And the reference vector is defined as:

$$\mathbf{h}^k_{ref} = \mathbf{0}_{3\times 1} \tag{C.7}$$

C.6 Vector sizes

A summary of the sizes of the vectors that should be defined for MATMPC at compile-time is presented in Table C.1.

C.7 Run-time sizes

At run-time, some vectors should be passed to MATMPC for the entire future horizon N, therefore it will be passed as a matrix where each column is the vector corresponding to k-th step in the future horizon. A summary of the run-time sizes in presented in table Table C.2.

Table C.2: MATMPC run-time matrices sizes

Symbol	Meaning	Size
$\mathbf{h}_{ref}$	References	$[n_y\ N]$
$\mathbf{W}$	Objective function weights	$[n_y\ N]$
$\overline{\mathbf{x}}$	States upper bound constraints	$[n_{bx}\ N]$
$\underline{\mathbf{x}}$	States lower bound constraints	$[n_{bx}\ N]$
$\overline{\mathbf{u}}$	Control input upper bound constraints	$[n_{bu}\ N]$
$\underline{\mathbf{u}}$	Control input lower bound constraints	$[n_{bu}\ N]$
$\overline{r}$	General upper bound constraints	$[n_c\ N]$
$\underline{r}$	General lower bound constraints	$[n_c\ N]$
$\mathbf{a}$	Parameters	$[n_p\ N+1]$

Bibliography

[1] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019.

[2] X. Meng, Y. He, and J. Han, "Survey on aerial manipulator: System, modeling, and control," *Robotica*, vol. 38, no. 7, pp. 1288–1317, 2020.

[3] F. Huber, K. Kondak, K. Krieger, D. Sommer, M. Schwarzbach, M. Laiacker, I. Kossyk, S. Parusel, S. Haddadin, and A. Albu-Schäffer, "First analysis and experiments in aerial manipulation using fully actuated redundant robot arm," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3452–3457, IEEE, 2013.

[4] C. Korpela, M. Orsag, M. Pekala, and P. Oh, "Dynamic stability of a mobile manipulating unmanned aerial vehicle," in *2013 IEEE international conference on robotics and automation*, pp. 4922–4927, IEEE, 2013.

[5] R. Rashad, J. Goerres, R. Aarts, J. B. C. Engelen, and S. Stramigioli, "Fully actuated multirotor uavs: A literature review," *IEEE Robotics Automation Magazine*, vol. 27, no. 3, pp. 97–107, 2020.

[6] B. Yüksel, C. Secchi, H. H. Bülthoff, and A. Franchi, "Aerial physical interaction via idapbc," *The International Journal of Robotics Research*, vol. 38, no. 4, pp. 403–421, 2019.

[7] G. Gioioso, M. Ryll, D. Prattichizzo, H. H. Bülthoff, and A. Franchi, "Turning a near-hovering controlled quadrotor into a 3d force effector," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6278–6284, IEEE, 2014.

[8] M. Ryll, G. Muscio, F. Pierri, E. Cataldi, G. Antonelli, F. Caccavale, D. Bicego, and A. Franchi, "6d interaction control with aerial robots: The flying end-effector paradigm," *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1045–1062, 2019.

[9] A. Ollero, M. Tognon, A. Suarez, D. Lee, and A. Franchi, "Past, present, and future of aerial robotic manipulators," *IEEE Transactions on Robotics*, pp. 1–20, 2021.

[10] S. Park, J. Lee, J. Ahn, M. Kim, J. Her, G.-H. Yang, and D. Lee, "Odar: Aerial manipulation platform enabling omnidirectional wrench generation," *IEEE/ASME Transactions on mechatronics*, vol. 23, no. 4, pp. 1907–1918, 2018.

[11] G. Nava, Q. Sablé, M. Tognon, D. Pucci, and A. Franchi, "Direct force feedback control and online multi-task optimization for aerial manipulators," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 331–338, 2019.

[12] H.-N. Nguyen and D. Lee, "Hybrid force/motion control and internal dynamics of quadrotors for tool operation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3458–3464, IEEE, 2013.

[13] R. Rashad, F. Califano, and S. Stramigioli, "Port-hamiltonian passivity-based control on se(3) of a fully actuated uav for aerial physical interaction near-hovering," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4378–4385, 2019.

[14] E. Cataldi, G. Muscio, M. A. Trujillo, Y. Rodríguez, F. Pierri, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero, "Impedance control of an aerial-manipulator: Preliminary results," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3848–3853, IEEE, 2016.

[15] E. F. Camacho and C. Bordons, *Introduction to Model Predictive Control*, pp. 1–11. London: Springer London, 2007.

[16] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear mpc: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93,

no. 1, pp. 62–80, 2020.

[17] H. Deng and T. Ohtsuka, "Parnmpc–a parallel optimisation toolkit for real-time nonlinear model predictive control," *International Journal of Control*, pp. 1–16, 2020.

[18] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi, "Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs," *Journal of Intelligent & Robotic Systems*, vol. 100, no. 3, pp. 1213–1247, 2020.

[19] M. Brunner, K. Bodie, M. Kamel, M. Pantic, W. Zhang, J. Nieto, and R. Siegwart, "Trajectory tracking nonlinear model predictive control for an overactuated mav," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5342–5348, IEEE, 2020.

[20] H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, "Model predictive control for micro aerial vehicles: A survey," *arXiv preprint arXiv:2011.11104*, 2020.

[21] L. Peric, M. Brunner, K. Bodie, M. Tognon, and R. Siegwart, "Direct force and pose nmpc with multiple interaction modes for aerial push-and-slide operations," in *International Conference on Robotics and Automation (ICRA 2021)*, 2021.

[22] G. Darivianakis, K. Alexis, M. Burri, and R. Siegwart, "Hybrid predictive control for aerial robotic physical interaction towards inspection operations," in *2014 IEEE international conference on robotics and automation (ICRA)*, pp. 53–58, IEEE, 2014.

[23] D. Tzoumanikas, F. Graule, Q. Yan, D. Shah, M. Popovic, and S. Leutenegger, "Aerial manipulation using hybrid force and position nmpc applied to aerial writing," *arXiv preprint arXiv:2006.02116*, 2020.

[24] J. Matschek, R. Jordanowa, and R. Findeisen, "Direct force feedback using gaussian process based model predictive control," in *2020 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 8–13, IEEE, 2020.

[25] M. Bednarczyk, H. Omran, and B. Bayle, "Model predictive impedance control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4702–4708, IEEE, 2020.

[26] K. J. Kazim, J. Bethge, J. Matschek, and R. Findeisen, "Combined predictive path following and admittance control," in *2018 Annual American Control Conference (ACC)*, pp. 3153–3158, IEEE, 2018.

[27] J. Pankert and M. Hutter, "Perceptive model predictive control for continuous mobile manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6177–6184, 2020.

[28] A. Wahrburg and K. Listmann, "Mpc-based admittance control for robotic manipulators," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 7548–7554, IEEE, 2016.

[29] B. Siciliano and O. Khatib, *Springer handbook of robotics*. springer, 2016.

[30] D. Q. Huynh, "Metrics for 3d rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 155–164, 2009.

[31] J. Sola, "Quaternion kinematics for the error-state kalman filter," *arXiv preprint arXiv:1711.02508*, 2017.

[32] A. Franchi and A. Mallet, "Adaptive closed-loop speed control of bldc motors with applications to multi-rotor aerial vehicles," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5203–5208, IEEE, 2017.

[33] N. Hogan, "Impedance control: An approach to manipulation: Part i—theory," *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 1–7, 1985.

[34] S. Stramigioli, *Modeling and IPC control of interactive mechanical systems—A coordinate-free approach*. Springer-Verlag London Limited, 2001.

[35] T. Tomić and S. Haddadin, "A unified framework for external wrench estimation, interaction control and collision reflexes for flying robots," in *2014 IEEE/RSJ International Con-*

ference on Intelligent Robots and Systems, pp. 4197–4204, IEEE, 2014.

[36] A. Y. Mersha, S. Stramigioli, and R. Carloni, "Variable impedance control for aerial interaction," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3435–3440, IEEE, 2014.

[37] K. H. Hunt and F. R. E. Crossley, "Coefficient of restitution interpreted as damping in vibroimpact," *Journal of Applied Mechanics*, 1975.

[38] Y. Chen, M. Bruschetta, E. Picotti, and A. Beghi, "MATMPC - a Matlab based toolbox for real-time nonlinear model predictive control," in *2019 18th European Control Conference (ECC)*, pp. 3365–3370, IEEE, 2019.

[39] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[40] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[41] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[42] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *Journal of Process Control*, vol. 12, no. 4, pp. 577–585, 2002.

[43] Y. Chen, "Algorithms and applications for nonlinear model predictive control with long prediction horizon," *Ph.D. Thesis*, 2018.

[44] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.

[45] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, pp. 2149–2154, IEEE, 2004.

[46] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[47] S. Di Cairano and A. Bemporad, "Model predictive control tuning by controller matching," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 185–190, 2009.