RAM ROBOTICS AND MECHATRONICS

DESIGN AND IMPLEMENTATION OF A NONLINEAR MODEL PREDICTIVE CONTROLLER FOR PRELIMINARY AERIAL PHYSICAL INTERACTION APPLICATIONS

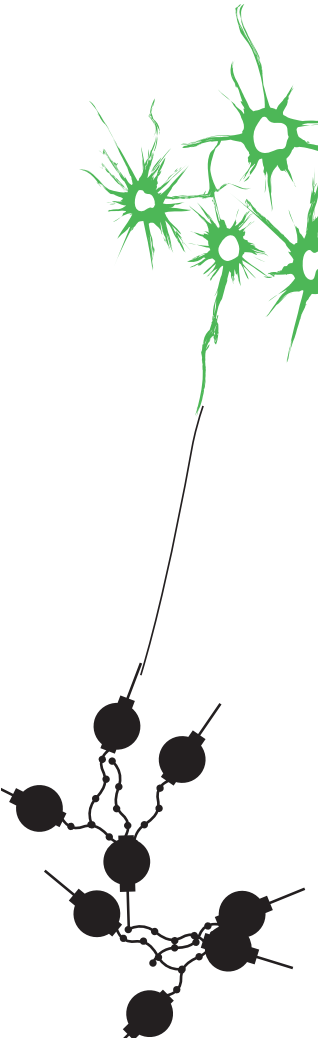
B. (Bogdan) Ganea

MSC ASSIGNMENT

Committee:
A. Franchi, Ph.D HDR
dr. D. Bicego
dr. ir. D.C. Mocanu

July 2021

038RaM2021
Robotics and Mechatronics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands



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

Summary

Traditionally, multi-rotor aerial vehicles have been used in a variety of contact-less civil applications, ranging from aerial photography, visual inspection of infrastructures or crop monitoring. In the last years they have been started however to be used, both in research and applications, for in-contact operations which involve an exchange of forces and torques with the environment in order to perform physical work.
Starting from a framework targeted for trajectory following applications, current thesis designs and then validates through simulations the steps needed to be taken until mixed motion-control can be achieved with the same framework.
For this, it is first extended with an observer software solution that is being used for the estimation of the disturbance wrench acting on the robot (considered to be acting at the center of mass). This solution is chosen instead of a sensor because normally a sensor increases both the cost and weight of the entire platform.
Because the existence of an interaction wrench introduces errors on the robot's position and orientation tracking evolutions, the next extension is to include it in the control architecture in order to assure pose control while being in physical contact.
The last design step is to offer the possibility to control also the interaction/contact force, while performing a trajectory task.
The control framework can be used up to what level it is needed. For example, if the user just wants to use the observer together with the controller without regulating the interaction force he can use the version before the last step; if he wants to use it also for force regulation, he will use the final version.
Then, as future work, it can be extended to give the possibility to include also the interaction torque, together with more realistic contact forces. In the end, the final goal would be to target human physical-interaction applications. However, for these, the current framework would need to be augmented with a vision perception and control layer, together with intelligent algorithms for understanding the human actions and answer accordingly.

Contents

List of acronyms

OCP	Optimization Control Problem
NLP	Nonlinear Problem
SQP	Sequential quadratic problem
RTi	Real Time Iteration
CMoN	Curvature Measure of Nonlinearity
NMPC	Nonlinear Model Predictive Control
MRAV	multi-rotor aerial vehicle
UDT	Unidirectional thrust
MDT	Multidirectional thrust
VTOL	Vertical take-off and landing
OD	Omni directional
APhI	Aerial Physical
AM	Aerial Manipulation

Chapter 1

Introduction

1.1 Motivation

The multi-rotor aerial vehicle (MRAV)s have been significantly used across a wide set of real life applications thanks to, among other advantages, their Vertical take-off and landing (VTOL) and hovering capabilities, their agility, compact structure and low cost. Quadrotors are probably the most studied and used platforms in contactless civil applications such as aerial photography, visual inspection of infrastructures, crop monitoring, and urban search and rescue (USAR) missions.
In the very recent years, MRAVs have started to be used also for in-contact operations which involve an exchange of forces and torques with the environment in order to perform physical work. For this reason, they are also called UAR (Unmanned Aerial Robots). In the past decade, this led to the development of topics such as Aerial Physical Interaction Aerial Physical (APhI) and Aerial Manipulation Aerial Manipulation (AM). Examples of real-life employments are inspection and maintenance by contact of sensible sites (1.1), assembly/construction and decommissioning of structures (1.2), assistance robotics in industrial/urban surroundings (1.3), removal of debris after natural catastrophes, delivery and transportation (1.4), etc.
Many laboratories and also companies have directed their related research towards it. Different collaborative projects emerged in the European Union, from which we can mention the following:

- AERIAL-CORE[1] (01.12.2019 - 30.11.2023) : development of core technology modules and an integrated aerial cognitive robotic system that will have unprecedented capabilities on the operational range and safety in the interaction with people, or Aerial Co-Workers (ACW), for applications such as the inspection and maintenance of large infrastructures.

[1] https://aerial-core.eu/

- The flying coworker[2] (04.2019 - 11.2023) : this project addresses the flying coworker, an aerial manipulator robot that act as a teammate of a human worker to transport a long bar or to realise complex tasks.
- AEROARMS[3] (1.6.2015-31.8.2019): design and build Aerial Robots with high manipulation capabilities for industrial inspection and maintenance;
- AEROWORKS[4] (1.1.2015 - 31.12.2017) : provide heterogeneous and collaborative aerial robotic workers for inspection and maintenance tasks in infrastructure environments;
- AIROBOTS [5] (1.2.2010 - 31.1.2013): design ARs for remote inspection by contact and to support human beings in applications which require interaction capabilities;

Figure 1.1: Inspection and maintenance use case

[2]https://anr.fr/Project-ANR-18-CE33-0001
[3]https://aeroarms-project.eu/
[4]https://aeroworks2020.eu/
[5]http://airobots.dei.unibo.it/

Figure 1.2: Assembly/construction and decommissioning of structures

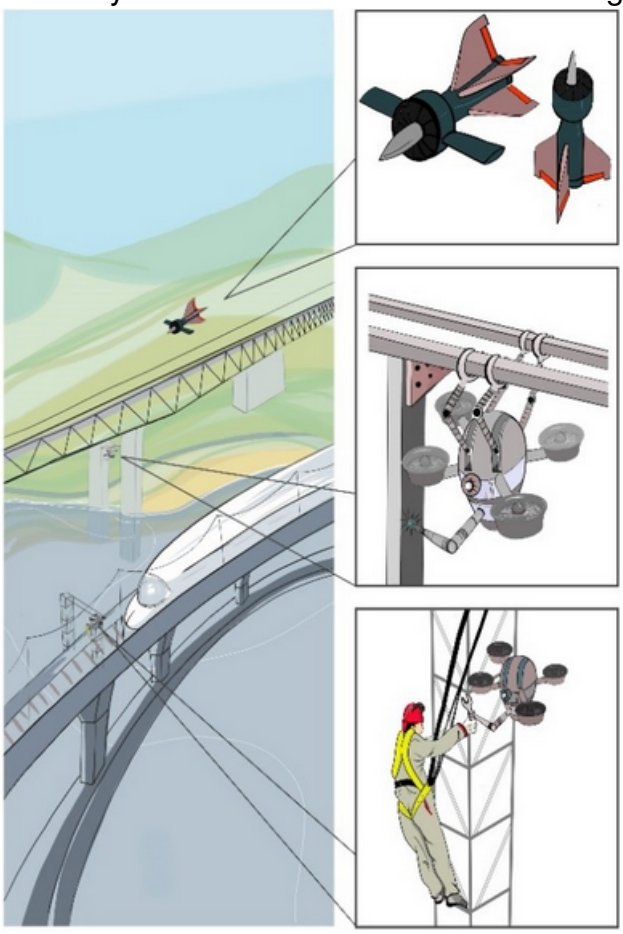


Figure 1.3: Assistance robotics in industrial/urban surroundings

Figure 1.4: Delivery and transportation

An aerial robot used for aerial physical interaction tasks has to react to forces/torques arising from the interaction with the environment in an active fashion. The control input is typically the velocity of rotating propellers, which in turn generate, thanks to the air interaction, the thrust forces and moments. Because the control of this wrench is quite complex due to the aerodynamic effects, in order to avoid substantial actuation errors, the rotor velocity can be controlled separately in a closed-loop ([2]).

The measurement of the interaction wrench is probably one of the most important aspects in aerial physical interaction. A reliable solution is the use of force/torque sensor. However, this solution increases the cost and weight of the platform. This is why solutions based on wrench estimators were proposed in the recent years. One solution (which has been used also in the current thesis is) [3], which uses an acceleration-based estimation for the external forces and a momentum-based estimator for the interaction torques. The mentioned solutions, together with the corresponding papers, are grouped in table 1.1.

Secondly, the Aerial platforms frequently possess interactive tools in order to enhance their manipulation capability and improve their dexterity. The simplest solution is to attach a rigid tool to the drone. The main drawback to this solution is the fact that it is impossible to control the full 6D (position and orientation) dynamics when operated together with an under-actuated MRAV, with limits in potential applications and stability issues. The second solution is the use of cables. However, in the case of this solution, there are again stability issues involved. The third used solution involves attaching an n **DoF!** (**DoF!**) robotic arm to the drone, thus overcoming

List of publications	
Publication	solution
[Antonelli et al 2016] [4]	use of a force/torque sensor; wrench is measured from a wrist mounted sensor and fed to an admittance filter; this solution increases however the cost and weight of the aerial platform
[Gioioso et al 2014b] [5]	sensor is placed on the interaction surface; may not be viable
[Yüksel et al 2014a] [6]	Lyapunov-based nonlinear observer for estimation of the external wrenches
[Tomic et al 2014] [3]	a hybrid estimation: linear acceleration for the interaction forces and a momentum based observer for the interaction torques
[Tomic et al 2017] [7]	a more refined hybrid estimation: the estimated forces are not simply computed by the model but through a first-order stable filter
[Rajappa et al 2017] [8]	wrench estimation and ring of eight contact sensors; the control is able to separate human interaction forces from additional disturbances such as wind and parameter uncertainties
[Augugliaro et al 2013] [9]	Kalman filters used for the external wrench estimation
[Mc Kinnon et al 2016] [10]	the external wrenches are estimated by an algorithm based on the unscented quaternion estimator

Table 1.1: Papers with solutions to wrench measurement.

the under-actuated property of typical quadrotors. This solution has drawbacks as well, like being more expensive to build and needing more maintenance across its operational life. The options that have been implemented, together with the paper sources, advantages and drawbacks are listed in tables 1.2 and 1.3.

In the following chapters, the attention is focused on a platform that uses a rigid tool because of its simplicity, in the same time representing a starting point for preliminary human-robot interactions. Furhermore, as already mentioned, for the measurement of the wrench the solution of the observer is chosen, having the benefits of weight and cost reduction.

1.2 Comparison between UDT and MDT MRAVs

In this thesis, special attention is devoted to VTOL MRAVs.
In typical VTOL platforms all the propellers spin about parallel directions, i.e., they

List of publications			
Publication	solution	Advantage	Disadvantage
[Nguyen et al 2013] [11] [Gioioso et al 2014a] [12] [Gioioso et al 2014b] [5] [Augugliaro et al 2014] [13] [Yuksel et al 2014] [14] [Staub et al 2017] [15]	tool fixed to the airframe	enables exchange of forces/torques with the environment, like pushing surfaces or objects	becasue typical VTOL MRAVs are underactuated, it is impossible to control all 6D position and orientation; thus it affects the stability of the platform and limits the potential applications
[Sreenath et al 2013] [16] [Tagliabue et al 2016] [17]	one or more cables with the load attached to the drone	allow to partially decouple the rotational dynamics of the vehicle from the one of the load	the control authority of the load pose might result limited; in addition control has to be as precise as possible in order to prevent undesired load oscillations that could make the system unstable

Table 1.2: Papers with solutions with static tool attachment to MRAVs 1.

List of publications			
Publication	solution	Advantage	Disadvantage
[Fumagalli et al 2012] [18] [Kim et al 2013] [19] [Kondak et al 2014] [20] [Suarez et al 2015] [21] [Baizid et al 2016] [22] [Muscio et al 2016] [23] [Muscio et al 2017] [24] [Tognon et al 2017] [25]	attach an n-Degree of Freedom (DoF) articulated robotic arm to the aerial platform	overcomes the under-actuation of the end-effector dynamics through the actuators provided by the arm; the load can be manipulated independently from the motion of the platform; if the total number of D.O.F is higher than the dimension of the load configuration space, the robot redundancy can be used to better compensate external disturbances or for other tasks	the payload and flight time are decreased because of the arm's weight; the final system is more expensive to build and also requires more maintenance work during its operational life due to its increased complexity; lateral forces can be generated through the dynamical/inertial coupling between the arm and the aerial robot, which requires the knowledge of the precise dynamical model and a very accurate measurement of the system inputs and states (extremely hard to achieve in real-world conditions).

Table 1.3: Papers with solutions with static tool attachment to MRAVs 2.

are collinear.The total force is exerted along the unique fixed direction in body frame. They are the so-called Uni-Directional Thrust (UDT) platforms. This configuration is the most efficient in terms of energy consumption, but it also provides disadvantages : rotation dynamics cannot be decoupled from translation, which is a problem if for example the MRAVs are required to resist a wind gust while keeping a desired attitude, or more in general if they need to exert a fully-decoupled force/torque wrench. In order to follow an arbitrary 3D position trajectory, they have to modify their orientation for the needed thrust that assures the reference linear acceleration. This fact makes UDT MRAVs under-actuated systems. They cannot follow desired 6D (position and orientation) trajectories. This property influences also their capability to exert specific interaction wrench with the environment, while keeping a reference position and orientation. Consequently, physical interaction with such platforms is challenging.

In the literature, the major solution has been to mount the rotors either in a fixed tilted way or in an actively-tilting fashion such that the thrusts of the propellers are not collinear anymore. In this way, by modifying each propeller's force, the direction of the total force can be changed arbitrarily, independently from the one of the total torque, to a certain extent, provided that the propellers arrangement is properly designed ([26] and [27]). These vehicles are identified as Multi-Directional Thrust (MDT) vehicles (1.5). In view of the aforementioned considerations, MDT MRAVs can also resist external disturbances while completing a manipulation task without the need to change their orientation.

In [28] and [29] the authors designed a special octo-rotor platform,where the four co-planar propellers traditionally used to stabilize the vehicle are supplemented by four perpendicular ones, used for the lateral movements. However, the design was limited in the set of body attitudes that it can attain.

The authors in [30] and [31] came up with an alternative arrangement, in which the six propellers are arranged in three distinct rotor planes, which gave the possibility to maneuver in confined spaces and provided the ability to land and take-off from different attitudes.

The dynamic capabilities of MDT designs can be further improved either by using varying-pitch, uni-directional thrust actuators or fixed-pitch, bi-directional ones. Thus, the body force and torque can be controlled inside a 6D ball, within the operational conditions of the actuators. This sub-class of MDT platforms are called Omni directional (OD) aerial vehicles.They can exert a force /torque in all directions and also hovering with every orientation in SO (3). This is constructively done. as pointed above, through change in direction of the force produced by each actuator either by inverting the spinning velocity of the rotor or by keeping the spinning velocity constant and using variable-pitch propellers to revert the rotor geometry. If

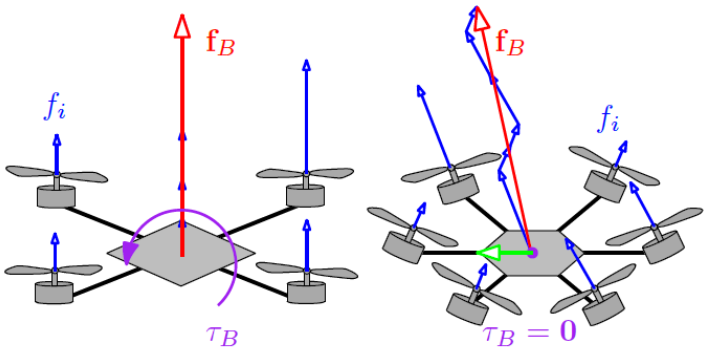


Figure 1.5: UDT vs MDT MRAV platforms (figure taken from [33])

uni-directional actuators are used for an OD purpose, the minimum number is seven [32].

1.3 Research questions and goals of the assignment

As already stated, while traditionally flying robots have been used both commercially and in research mainly for contact-less applications, recent developments targeted contact-based applications. Motivated from this, the main goal of this assignment is to study and investigate the capabilities of the MPC (Model Predictive Control) strategy to tackle aerial physical applications. In addition, contact wrench control is also desired,, i.e. the capability of actively regulating the interaction wrench.
As model-based predictive control (MPC) is already one of the most popular advanced control technology in the chemical processing industries, it has been started to be also used in safety and time-critical applications with fast dynamics, e.g., in the automotive and robotic fields. There are many variants, both in academia and in industry, but they all share the common trait that an explicitly formulated process model is used to predict and optimize future process behavior of the system. This type of controllers are able to account for constraints both in controller commands and states/inputs/outputs through the formulation of the optimization problem.
Furthermore, MPC is able to optimize, in a predictive fashion, the system behavior on a given future time horizon based on the system model. In addition, since the related Optimization Control Problem (OCP) is solved at each sampling instant as new state measurements get available, it is able to mitigate for possible model perturbations.
The Non-linear Model Predictive Control is the extension of the linear Model Predictive Control to deal with non-linear systems. It assumes the existence of a non-

linear model in the constraints and also, for certain applications, the existence of an infinite prediction time and non-square optimization function. This strategy is also implemented in current work and more detailed in chapter 2.
Current work is organized in four main parts, in an incremental fashion. The design process is outlined in chapter 3. For the simulations, tuning is also taken into account and based in the experimental results, the best option is chosen.
In each main part, the most relevant behavior is searched for and validated. In the same time other possible unpredicted positive outcomes, but also negative ones that appear are analyzed.
In parallel, the implemented control architecture can also offer the final user the possibility to choose up to what level to use the implementation.
Ultimately, the target is to perform preliminary mixed motion/interaction control tasks and to validate them at least in realistic numerical simulations.
The integration is done incrementally. First, a solution that allows measuring the external wrench arising from the interaction is included in the framework and effects of the external wrench on the position and orientation tracking are analyzed. Then, the external wrench is included in the state vector of the Nonlinear Model Predictive Control (NMPC) controller to observe the improvement in position and orientation tracking. Afterwards, the optimization criteria is also extended in order to consider state-dependent interaction force and to set a desired reference for it. At the same time, the position tracking should remain undisturbed.
The software framework has as starting point, as already mentioned, the NMPC control strategy (more details in chapter 2).
All the simulations are implemented using MATMPC, an open-source, Matlab-based and non-linear MPC toolbox, see Appendix A. This toolbox uses another 2 open-source packages. One is qpOases, a package used for solving the optimization problems that uses linear operations defined in the second package (Appendix B) for the solver algorithms.
The software framework is general and allows testing of various platforms. However, the simulations from the current thesis have used the tilt-hex platform, a MDT and fully-actuated platform. More details about its constructive characteristics and resulting model parameters, together with the theoretical model used in the software framework are in Appendix C.

1.4 Related work

In [26] it is presented the MPC control framework used to track the full 6D pose (position and orientation), the starting point of this thesis, with the same state, input,reference and output vectors considered at the start of current thesis, together

with the definition of the objective function. It is shown the effectiveness of the control architecture for both an under-actuated platform (quadrotor) and the fully-actuated one (tilthex) used in the thesis. Furthermore, the tests are done for different types of trajectories (a chirp trajectory and a discontinuous one), together with non-consistent references for all the states in order to test both the stability of the drone and its capability to re-generate the trajectory while considering the actuator constraints.
In [27], the authors present a control architecture used for both autonomous trajectory following and also for physical interaction capabilities. In this architecture it is used the observer algorithm used also in the current thesis. In this paper, it is integrated into a cascaded control architecture (admittance filter + geometric controller) and validated in a number of real scenarios (sliding on a tilted surface, sliding on a surface with multiple contacts, mass-pulling use case, etc.). This architecture type is not used in the thesis, as it is preferred the use of the full-state one from the previous paper. This is due to multiple advantages, among them being the fact that it can be avoided the setting of low-level controller references in terms of high-level controller dynamics and the fact that considering as inputs the propellers' forces derivatives give the possibility to include the realistic limitations of both angular speeds and their derivatives in the constraints section of the controller.
In [34], a robust MPC controller is designed. Its focus is to obtain stable,safe and efficient trajectory following. It is designed to obtained the minimum possible deviation from the reference for the worst-case disturbance, while being also augmented with obstacle avoidance capabilities. The state-space representation incorporates the effect of external disturbances, just like in the current thesis, but it differs from the framework used here through the definition of the optimization criteria, which is linear and through the fact that it doesn't consider control of the contact wrench, the focus being on trajectory following applications. Its formulation could be however used as a reference point for the extension of the current framework towards rejection of undesired external wrench, for example the one produced by the wind (with the added problem of separating the useful disturbance/contact wrench from the undesired one).
In [35], the paper describes trajectory generation based on numerical optimal control. As implementation, it uses multiple shooting method for the discretization of the continuous-time optimization problem and Sequential quadratic problem (SQP) to solve the resulting (condensed) problem. The same methodology is also used in the current thesis. The authors show that it can be safely used for a variety of situations and with a multitude of systems (quadrotor, quadrotor with pendulum and quadrotor with aerial manipulator), both using optimal control, but also using MPC strategy.
In [36], a cascaded control using an inner NMPC attitude controller and an outer

LQRi controller for controlling the linear position, speed and acceleration is used, which is differently from the current work, which is based on a full-state NMPC controller. The input vector is defined differently, as the propellers forces, while the state vector is also defined differently, containing the three components of the MRAV's body-frame, together with the angular speed of the platform. The optimization criteria is also different, containing terms on the attitude error, angular speed, desired thrust magnitude and control action, respectively. The paper also details a method to overcome a propeller failure, which is different than the framework presented here, which can intrinsically handle this kind of situation. The same Real Time Iteration (RTi) scheme is used as in the current framework, but the whole framework is not targeted to any kind of physical interaction applications.
In [37], a method based on Real-Time Iteration is proposed for long prediction horizons, highly non-linear and fast changing systems, ftb-RTI (fixed time-block). This method improves ml-RTI and adj-RTI schemes by computing the Curvature Measure of Nonlinearity (CMoN) nonlinear sensitivity measure at each sampling instant for all prediction time samples, reorganizing them in terms of this measure and only M out of the N prediction instants being computed (M ¡¡ N), where M is a tuning variable and N the prediction horizon. The algorithm requires at each sampling instant the integration of the system dynamics in order to compute the sensitivities and the CMoN measure. This technique also improves the overall computation time of the control commands. This methodology is also used in the current thesis and included in Appendix A.
In [38], the authors use NMPC strategy for both perception and control objectives. The perception objectives require that (projected) interest points should remain as close as possible in the center of the image and also their projected speed to be minimal (if the interest points are moving w.r.t. the World Frame). The perception objectives are needed in case of object detection : landmarks features useful for pose estimation algorithms or points belonging to an object for obstacle detection. The action objectives are related to the input saturation limits (commands limited by physical limitations, considered also in current thesis), but also contain components derived from the underactuated nature of the quadrotor (used in the paper). The state used includes only the linear position, linear speed and orientation of the drone, different than the state vectors used in the thesis. Also the input vector is different, containing the mass-normalized thrust vector and the angular speed vector. The objective function contains explicit terms on the state vector, input vector and the perception vector (image coordinates and image projected speed). The constraints are defined in terms of the input vector components and the linear speed component of the state vector. Although it doesn't consider physical interaction applications, this paper validates the usage of NMPC for applications that involve also perception ob-

jectives. This approach can be used in the extension of the final control architecture of current thesis towards human physical-interaction control applications.
In [39], the authors use a combined hexarotor platform and an arm system. The hybrid platform+arm is modelled taking into consideration also the interaction force with a whiteboard. The model of the contact force is similar with the one used in the current work, i.e. a spring model. The choice of the orientation is represented by quaternions , different than the Euler angles used in the current work. The propellers forces are not included in the state vector, like in current work. The input vector is also different, represented by the collective thrust, torque and position of the end effector. While the first 2 components are translated into propellers forces through the use of a control allocation block, the third component is needed in order to find the manipulator joint angles (through inverse kinematics). The output vector is also different, containing also the position of the end effector and the control input and no linear acceleration and angular acceleration, compared to the final one of the current work (linear position, linear velocity, linear acceleration, Euler angles, angular speed, angular acceleration, contact forces). The control strategy that is being used is also NMPC. However, in its OCP formulation, in the constraints part, there is only an inequality on the control input, as there is a second optimization problem defined for the control allocation block and which has a constraint on the propellers' forces. The implementation doesn't use any observer, but a sensor. The 9 D.O.F. of the system are split into 6 (force and torque) that control the aerial platform, while the remaining 3 are used for an Inverse Kinematics block that sets the angle references for the manipulator. The results show that the controller can handle both position control and force tracking, while writing on a whiteboard with varying velocities and dimensions of the text.
In [40], NMPC technique is used for a fully actuated aerial manipulator to track a hybrid force and pose (position and orientation) at the end-effector. The state vector contains the same elements as the one used in the current thesis, but it has 2 slight modifications : because the geometrical arrangement is colinear, there is only one propeller force in the state vector; in addition, the state vector contains also the propeller torque. The NMPC's commands are composed of the propeller's force derivative (similar to the ones from the thesis), but it contains also the derivative of the propeller torque. The cost function doesn't take into consideration also the error terms on the linear acceleration and angular acceleration, compared to the final cost function from this thesis. Also, for the contact force there is used a sensor and not a software observer. In the Newton-Euler model appear additional terms on the external disturbance wrench, different than the contact one, in contrast with the modelling considered in the thesis and in the other referenced papers. An Extended Kalman Filter is used for the estimation of the disturbance force and torque. Three operation

modes which reflect the state of contact constraints are defined:free flight and two modes for force control based on static or dynamic friction at the end-effector. The architecture is validated again through a number of experiments.

1.5 Thesis structure

The remainder of this thesis is organized as follows. In Chapter 2, detailed information about NMPC control strategy is provided. Then, in Chapter 3, the incremental steps taken for the design of the controller and the reasoning behind them are explained, together with the choosing of state vector, reference/output one and optimization criteria. In chapter 4 all the simulations are detailed, together with the most important conclusions from each part. Finally, in Chapter 5, the main conclusions are drawn, together with the hints of future work : more general and realistic contact force and torque, possible contact torque control, extension to human-aerial physical interaction applications through usage of an actuated manipulation arm, vision-based instrumentation and intelligent control algorithms for a broad range of situations.

Chapter 2

Theoretical background

2.1 NMPC for MRAVs with generic designs

Inspired by [26], this section presents the mathematical formalism of the MPC controller introduced in the previous chapter (in the nonlinear form, specific to the current framework), together with its particular properties.
The 2 main goals of the controller architecture are to simultaneously address the problem of local reference trajectory planning and that of stabilizing the vehicle dynamics, giving in the same time the possibility for general platform models to be tested.
The reference trajectory denoted $(p_r(t), \eta_r(t))$,considered twice continuously differentiable and given by a generic global planner, is desired to be followed. Towards the end of the thesis, this reference trajectory (together with the optimization cost function) will be extended with the reference contact force with a surface that is also desired to be followed.
Stability on the other hand is fulfilled if the reference trajectory is compatible with the model of the platform and the considered constraints. However, considering a general global planner it is possible to not take the model of the MRAV into consideration when generating the reference trajectory. This property will be actually used and unfeasible trajectories w.r.t. the robot model will be generated in order to test the capability of the controller to locally re-generate the trajectory (in terms of the model dynamics and constraints), while preserving in the same time the stability of the system.
In this way the algorithm can deal with arbitrarily designed MRAVs, without the need for a preliminary analysis on the system dynamics.
For modelling the platform it has been chosen the Newton-Euler formalism. The equations are grouped into a translation part and a rotational part, as detailed in 2.1. To the right of the equations the terms are grouped into 3 main types : the first term is due to the gravity, the second term is due to the influence of the propellers'

forces and the third term represents the forces and torques felt at the body-level due to the interaction wrench of the environment on the end-effector. The main quantities present here are the mass m, inertia J, the linear acceleration $\ddot{p}$, angular acceleration $\dot{omega}_R^R$ and gravitational acceleration g. Furthermore, the force f and torque τ^R are the ones due to the propellers' forces and f_R and τ_R^R is the external wrench due to the interaction. The complete model , together with its mathematical derivation and the form that takes into consideration the allocation matrix are detailed in Appendix C.

$$\begin{bmatrix} m\ddot{p}_R \\ J\dot{\omega}_R^R \end{bmatrix} = - \begin{bmatrix} m * g * e_3 \\ \omega_R^R X J * \omega_R^R \end{bmatrix} + \begin{bmatrix} f \\ \tau^R \end{bmatrix} + \begin{bmatrix} f_R \\ \tau_R^R \end{bmatrix} \tag{2.1}$$

The principle of the control strategy is best outlined in figure 2.1. Its main objective is to minimize the error between the reference trajectory and the predicted output on the prediction horizon (defined in terms of its length, t_h).

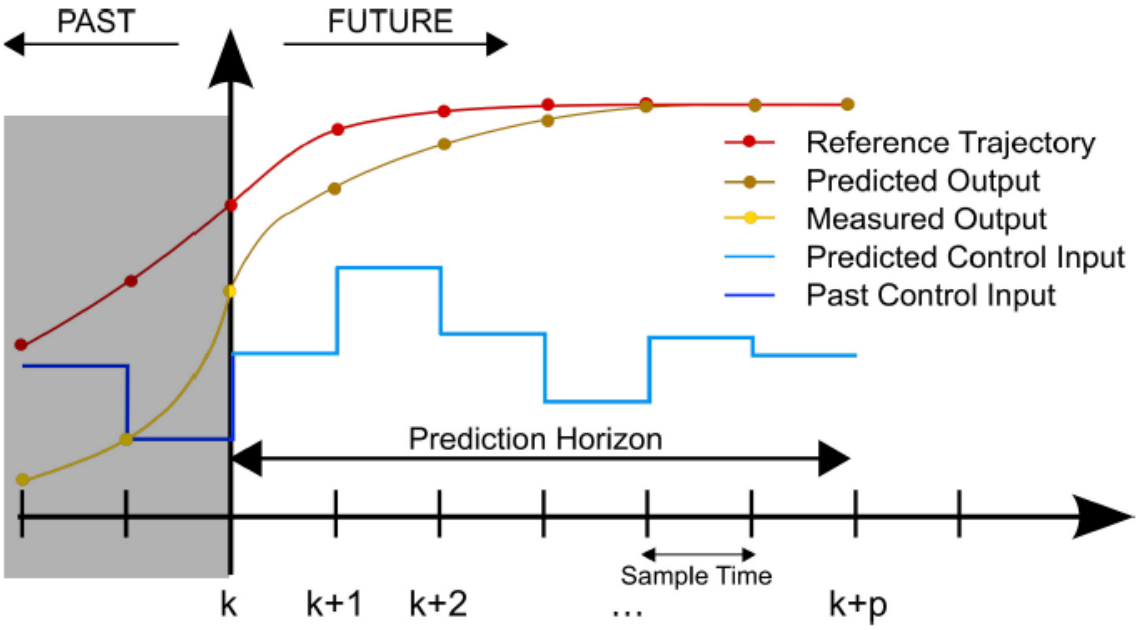


Figure 2.1: MPC control strategy

The initial state vector that is being used contains the position p, the linear velocity $\dot{p}(t)$, the orientation $\eta(t)$, the angular velocity $\dot{\omega}$ and the propellers' forces $\gamma = [f_1 ... f_6]$ (2.2), with the mention that it considers also, different that the common choice used in the literature, the propellers forces. This allows the inclusion of their real physical limitations in the constraints part of the controller. In this way, it is possible to actually impose them to the control strategy, in comparison with typical reactive controllers.

$$x(t) = \begin{bmatrix} p(t) \\ \dot{p}(t) \\ \eta(t) \\ \omega(t) \\ \gamma \end{bmatrix} \tag{2.2}$$

In order to guarantee smoothness properties of the desired trajectory, it is given the possibility to drive also the derivatives of the state vector.
Under these considerations, the initial reference signal (and thus also the output vector) will be defined as follows:

$$y_r(t) = [p_r^T(t) \quad \eta_r^T(t) \quad \dot{p}_r^T(t) \quad \omega_r^T(t) \quad \ddot{p}_r^T(t) \quad \dot{\omega}_r^T(t)]^T \tag{2.3}$$

and the corresponding output vector will be defined as:

$$y(t) = \begin{bmatrix} p(t) \\ \eta(t) \\ \dot{p}(t) \\ \omega(t) \\ \ddot{p}(x(t), u(t)) \\ \dot{\omega}(x(t), u(t)) \end{bmatrix} \tag{2.4}$$

with the mention that, while the position, speed, orientation, angular velocity can be measured directly from the state vector, the linear acceleration and angular acceleration are dependent on both state and input vectors.
Both state and output vectors will be further expanded in the following 2 chapters in order to accomplish the desired goals.
The general OCP optimization problem defined in continuous-time is defined in 2.2. The controller's command is calculated at each sampling instance by minimizing the optimization function that contains the final term and the integral of a general, usually non-linear function on the whole prediction horizon, subject to equality and inequality constraints.

$$\begin{aligned} \mathbf{u}^* = \arg\min_{\mathbf{x},\mathbf{u}} \quad & \mathbf{l}_f(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} \mathbf{l}(\mathbf{x}(t), \mathbf{u}(t))\, dt \\ \text{s.t.} \quad & \mathbf{r}(\mathbf{x}, \mathbf{u}) = \mathbf{0}, \\ & \mathbf{s}(\mathbf{x}, \mathbf{u}) \leq \mathbf{0} \end{aligned}$$

Figure 2.2: General MPC optimization criteria

In order to be able to be implemented on a computer, the continuous time OCP criteria is modified into its discrete-time equivalent. For this, the discretized versions of the $y_r(t)$ and $y(t)$ are defined as $y_{r,k} = y_r(kT)$ and $y_k = y(kT)$.
The Nonlinear Problem (NLP) to be solved at current time kT, given the current state x_k, is generically formulated as :

$$\begin{aligned} min_{\substack{\hat{x_0}...\hat{x_N} \\ \hat{u_0}...\hat{u_{N-1}}}} \; & \Sigma_{h=0}^{N-1} \|\hat{y_h} - y_{r,k+h}\|_{Q_h}^2 + \|\hat{u_h}\|_{R_h}^2 + \|\hat{y_N} - y_{r,k+N}\|_{Q_N}^2 \\ s.t. \quad & \hat{x_0} = x_k \\ & \hat{x}_{h+1} = \Phi(\hat{x_h}, \hat{u_h}), h = 0, 1...N-1 \\ & \hat{y}_h = h(\hat{x_h}, \hat{u_h}), h = 0, 1...N \\ & \underline{\gamma} <= M * x_h <= \overline{\gamma}, h = 0, 1...N \\ & \underline{\dot{\gamma}_{k+h}} <= u_h <= \overline{\dot{\gamma}_{k+h}}, h = 0, 1, ..N \end{aligned} \tag{2.5}$$

where it is also used the discrete version of the continuous-time dynamic model (with its mathematical equation given in equation 2.6 where there can be observed explicit terms related to the allocation matrix and the inertia matrix, together with the Euler angles $(\phi, \theta and \psi)$ and the corresponding rates $(p, q and r)$). This form is quite general, it will be then particularized in the following chapters for the particular cases/needs.
First, it contains an additional element in the optimization expression, the one dependent on the commands, which will not be taken into account in the current framework. This is because the purpose was to test the potentiality of the system up to the actuators' saturation. What should be understood from this is that it was wanted to actually see that either the upper of lower limits were sought to be reached and see that the tracking is still accomplished and also stability maintained.
Second, the M matrix is introduced to be able to extract the desired states (x,y and z positions, roll and pitch angles, propellers' forces).
Third, an additional inequality appears in terms of the inputs/commands.This allows explicitly imposing constraints on them, experimentally or not, as will be detailed later in this chapter.
Fourth, in order to simplify the problem, the propellers' forces limits are considered as constant on the whole prediction horizon and equal to γ_k,i.e. equal to the one from the first prediction step.
Fifth, regarding the robustness stability requirement, after several simulations with different platforms for different trajectories and different values of the prediction horizon the chosen value for it was 1.5 s.
Q_h and R_h are semidefinite positive matrices that represent weights through which the tuning of the controller is done. The bigger the weights, the more the related

output term is considered for determining the optimal values of the commands.

$$x_{dot}(t) = \begin{bmatrix} \dot{p}(t) \\ 0 + G1(1,:)[f_1; f_2; f_3; f_4; f_5; f_6]/m \\ 0 + G1(2,:)[f_1; f_2; f_3; f_4; f_5; f_6]/m \\ -g + G1(3,:)[f_1; f_2; f_3; f_4; f_5; f_6]/m \\ p + rcos(\phi)tan(\theta) + qsin(\phi)tan(\theta) \\ qcos(\phi) - rsin(\phi) \\ rcos(\phi)/cos(\theta) + qsin(\phi)/cos(\theta) \\ qr(I_y - I_z)/I_x + G2(1,:)[f_1; f_2; f_3; f_4; f_5; f_6]/I_x \\ rp(I_z - I_x)/I_y + G2(2,:)[f_1; f_2; f_3; f_4; f_5; f_6]/I_y \\ qp(I_x - I_y)/I_z + G2(3,:)[f_1; f_2; f_3; f_4; f_5; f_6]/I_z \\ \dot{f}_1 \\ \dot{f}_2 \\ \dot{f}_3 \\ \dot{f}_4 \\ \dot{f}_5 \\ \dot{f}_6 \end{bmatrix} \tag{2.6}$$

The solution to the OCP, at a given time step k will be the optimal values $x_{0|k}, ..., x_{N|k}$, $u_{0|k}, ..., u_{N1|k}$. According to the receding horizon principle, it is enough to apply the input value $u_k = u_{0|k}$, and the procedure is then repeated at the subsequent time step k + 1.

As already mentioned, the above framework is general enough to work with arbitrarily trajectories and general platforms, as it was already validated in practical experiments on multiple platforms (under-actuated and fully-actuated) for various trajectories ([26]). An important mention to be made here is that, while fully-actuated platforms can follow a decoupled position + orientation trajectory, in the case of under-actuated platforms, the trajectory needs to follow the differential flatness property, i.e. it needs to be written in terms of the flat output(s) and a number of their derivatives (usually 4) [41].

Another small mention is to be made regarding stability. If the reference trajectory is compatible with the system dynamics, it has been shown that stability is assured if the prediction horizon (N) is sufficiently long. However, the focus here will not be on the ”optimal” horizon length, as previous results regarding it will be used ([26]).

2.2 Implementation details of the OCP resolution

Current framework uses multiple shooting method to discretize the continuous-time OCP problem into its discrete equivalent (A). The resulting NLP problem is then linearized using SQP method. From the available options, it is chosen then to condense the obtained linear problem and solve it using qpOases ([42]). The derivatives needed to perform the optimization are obtained from the toolbox CasADi4 (B).
qpOases is an open-source library that uses the active-set method to solve the optimization problem. Briefly developing on the subject, the 2 most used methods in solving optimization problems are the Interior Point Method and the Active-Set Method. While the first one tries to reach the optimal solution through a number of successive Newton-like steps obtained by solving a system of linear equations, the second one identifies the active set of its solution through redefining an initial guess by adding or deleting constraints to it.
Finally, the solution to the initial non-linear problem can be normally obtained using the so-called l1 merit function. However, in order not to wait until the SQP solver converges to its solution, there is given also the option to use the solution after one interation. This strategy is called Real-Time Iteration (RTI). More details about its mathematical form in A. Furthermore, the partial sensitivity update measurement (CMoN) mentioned in the same appendix, is used in the simulation to reduce the computational complexity. In addition, the current framework offers the possibility for simulations to be carried out both in Matlab and Simulink, which eases testing and development.
The sampling time of the simulation or real experiment should be chosen as the result of a trade-off between very small (necessary to make the system as reactive as possible) and larger than the average computational time of the NMPC solution in order to assure the existence of at least one solution at each sampling time.
The final used sampling time was 0.004 s.
If the solution of the optimal problem is not available within any sample time, it will be used the computed one from the previous sample time, as a back-up solution.

2.3 Simulations setup

The general architecture of the system used throughout real experiments is shown in figure (2.3)[1].The main components are the NMPC controller, which periodically computes the input of the actuator low-level controllers (the ESCs) [2], the physical

[1] general architecture from reference [26]

aerial robot to be controlled, and the sensors.
However, for the simulations presented in this thesis, only the details related to the NMPC implementation are important to be mentioned. The predictive controller is implemented using MATMPC (Appendix A). Its algorithmic routines are written using MATLAB C API and available as MEX functions. They are important for the time-critical operations of the NMPC controller in order to assure performance comparable with pure .c routines.
An important mention to be done at this moment is related to the choice of the controller outputs/commands. They are chosen as the derivative of the propeller forces in order to comply with a more detailed and representative model of the real physical limits of the system. These realistic limitations mainly refer to being able to consider velocity-dependent propellers acceleration limits, rather than considering them constant. Limitations on rotor velocities and accelerations (which are each regulated by a separate ESC), further determine limitations in the force derivatives, also the actuators commands.
The control input u is then integrated and then converted into a rotor velocity command w (according to the chosen thrust model). In experimental conditions, the velocity setpoints are then transferred to the module of the low-level controllers onboard the aerial platform, but in the simulation they are transferred directly to the model of the platform. The sensors then measure the necessary variables and provide them to the feedback loop in order to close it. A MoCap (motion capture) sensor for obtaining the position and orientation w.r.t. the World Frame and a gyroscope for the angular speeds of the drone. Furthermore, for the propellers' angular speeds the rotor spinning velocities were measured by computing the time elapsed between two phase switches (which is further used in the thrust generation model to generate the propellers' forces).

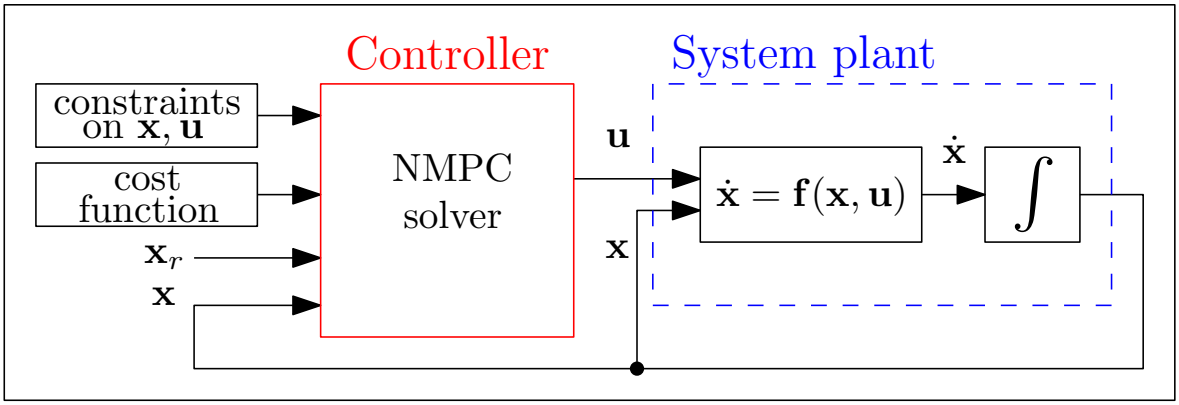


Figure 2.3: General architecture of NMPC

2.4 Implementation details

First, in the case of NMPC technique there is a series of time parameters that needs to be considered when determining the prediction horizon and corresponding value of one interval of it. The general rule is given in the following inequation :

$$t_{solv} \leq T_{ctrl} \leq T \leq t_h, \tag{2.7}$$

where t_{solv} is the time necessary to compute the solution at every time sample, T_{ctrl} is the sampling time of the control algorithm, T is the duration of one sample of the prediction horizon and t_h is the duration of the prediction horizon.
Generally, the sample period of the controller needs to be higher than or equal with the average solver period (in order to assure the existence of the solution, as already pointed before) and smaller or equal with one sample of the prediction horizon, in order to include at least one term in the cost function.
Second, in order to represent the 3D orientation of a rigid body several choices can be made. These are the Euler angles, the Rotation Matrix and the Quaternion Representation. Transforms between them are also used for easily using the desired one. For this work it is used the representation with the 3 Euler angles - roll, pitch, yaw in the 3-2-1 configuration, with the final rotation matrix given as :

$$R = \begin{bmatrix} cos(\theta)cos(\psi) & sin(\phi)sin(\theta)cos(\psi) - cos(\phi)sin(\psi) & sin(\phi)sin(\psi) + cos(\phi)sin(\theta)cos(\psi) \\ cos(\theta)sin(\psi) & cos(\phi)cos(\psi) + sin(\phi)sin(\theta)sin(\psi) & cos(\phi)sin(\theta)sin(\psi) - sin(\phi)cos(\psi) \\ -sin(\theta) & sin(\phi)cos(\theta) & cos(\phi)cos(\theta) \end{bmatrix}$$

(2.8)

Furthermore, using the same convention, a relationship between the Euler rates and angular velocity can be expressed using a matrix transformation :

$$\omega = T * \dot{\eta} \tag{2.9}$$

where the matrix T has the form :

$$T = \begin{bmatrix} 1 & 0 & -sin(\theta) \\ 0 & cos(\phi) & sin(\phi)cos(\theta) \\ 0 & -sin(\phi) & cos(\phi)cos(\theta) \end{bmatrix} \tag{2.10}$$

Equation 2.9 gives also the possibility to express the derivative of the angles in terms of the angular velocity by inverting it.
One mention to be done here is that if the platform is to be span the whole SO(3)

manifold, the angle $\pi/2$ must be avoided as it represents a singularity. This can be achieved also, as in all current simulations, by setting a constraint in the OCP resolution in which the roll, pitch and/or yaw to be between $-\pi/2$ and $\pi/2$.
The second mention would be that the choice of orientation is not strict, it can be done as how the user wants, the framework being able to adapt to it.
The last thing to mention here is the fact that the limits of the propellers' forces have been chosen to be constant on the time horizon and equal with the first one $f_{k,h}$.

2.4.1 State-dependent actuator bounds

As already mentioned, the purpose of the selected commanded inputs is related to the desire to have a more detailed and representative model of the real physical limits of the system. They are determined by the limitations on the propellers angular speeds (which have effect also on the system stability), but also by limitations on their angular accelerations. Constraining propellers' speeds is needed in order to prevent the risk of damaging the motors, while constraining their acceleration is needed in order to guarantee an accurate force tracking.
Second, proper constraints for the actuator angular speeds and forces are determined offline, in relation to the particular model of the drone considered in the simulations. If desired, any other thrust generation model can be adopted. The angular speeds limits are taken from a lookup table determined off-line, dependent on the current angular speed and so, they are eventually determined online. Departing from them, the propellers' thrust derivatives are also determined online, in terms of the current limits on the angular accelerations.
More details about the whole procedure can be found in [26].
For the Tilt-Hex platform that was used in the simulations, the identified limits of the angular speeds were identified as 16Hz and 102Hz. These lead to the values of 0.25N and 10.3N for the propellers' forces limits, needed to constrain the OCP resolution in the MPC algorithm.
The entire span of propellers' thrusts and thrusts derivatives are given in figure 2.4 (which contains also the span corresponding to an under-actuated platform used in the simulations, a quadrotor), where the area corresponding to the tilt-hex is determined by the bold colored lines, both for thrust forces and force derivatives. It should be noted that they are dependent on a particular motor/propeller choice.
Furthermore, the limits of the rotor accelerations of the Tilt-Hex are listed in table 2.1, again being dependent on a particular motor/propeller choice. This table is used by the controller to calculate at each sampling instance, through mathematical interpolation, the acceleration limits corresponding to the current angular speed. This is calculated individually for each propeller, as it will be seen in the simulations.

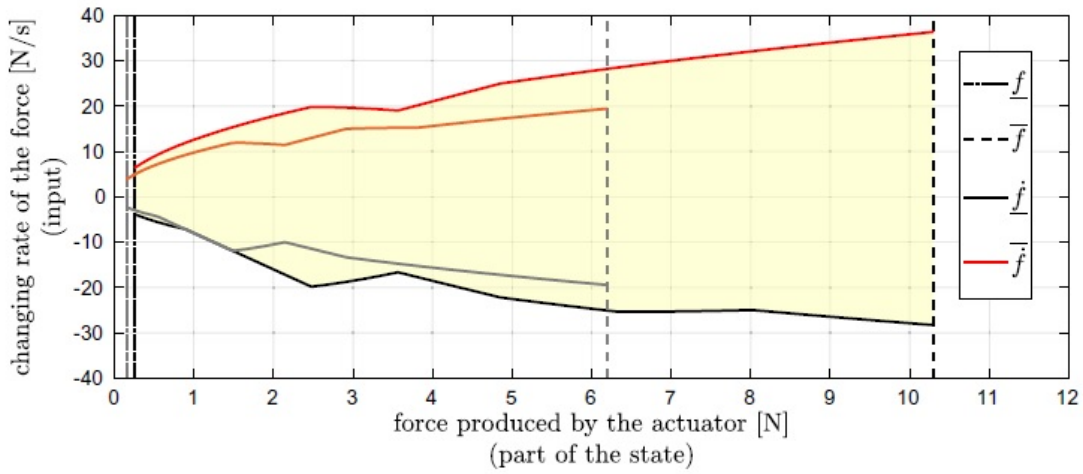


Figure 2.4: Span of admissible set of thrusts and thrust derivatives for the TiltHex MRAV and quadrotor MRAV

$\omega[Hz]$	$\underline{\dot{\omega}}[Hz/s]$	$\overline{\dot{\omega}}[Hz/s]$
30	-127	209
40	-121	208
50	-114	244
60	-118	208
70	-128	149
80	-111	156
90	-95	135

Table 2.1: Limits on angular accelerations depending on the angular speed setpoint, for the Tilthex MRAV

Tu sum up, the main advantages of the current framework are first related to the more representative, realistic and detailed model used that takes into consideration the realistic physical limits of the actuators. In this way, by embedding the constraints on the propellers' angular speed and their accelerations, it is possible to determine also in a realistic way the ones belonging to the propellers' forces and their derivatives.

Then, by using the derivative of the propellers' forces as controller command, it is assured the generality of the framework, basically any other platform being able to be used/simulated. In addition, the generality is also demonstrated by giving the possibility to use any particular thrust model.

Nevertheless, in terms of the computational aspects, the framework is flexible enough

to give the possibility to use alternative routines at each intermediate step in the solving process, starting from the discrete equivalent of the continuous-time optimization problem up to the final step of solving, using either a condensed or non-condensed form.

Chapter 3

From motion control to mixed motion/interaction control, controller design

This chapter provides an overview of all the intermediate steps taken from the starting point up to the final validation of both trajectory and force tracking.
First, it will present the chosen wrench measurement solution used in the current work and validate it. It will continue then with the design that allows the improvement of the tracking performance in the presence of the external disturbance wrench. Finally, the design of the complete solution that allows both 6D pose and force tracking will conclude the chapter. Where necessary, if the state or output vector are altered and/or also the cost function is modified, they will be explicitly shown.The whole framework is designed and tested only in Matlab simulation. The goal was to test it also in real experiments with the Tilt-Hex platform, but due to the situation related to the Coronavirus pandemic, it was not possible to perform also these real tests.

3.1 External wrench sensing/estimation

Departing from pure trajectory control, the interest was to expand the capability of the NMPC controller in order to also be able to sense the environment with which the platform interacts.

3.1.1 Use of a force/torque (f/τ) sensor

To this aim, a force/torque sensor could be mounted on the robot's tooltip, which is usually capable to provide a reliable measure.

The three main types of force sensors are :

1. Load cells are a type of force sensor/force transducer that converts an applied force into an output signal that can be used to measure forces such as compressive forces, most commonly weight. Load cells can use different technologies to produce an output.

2. Strain gauges (also spelled as gages) are a type of sensor element whose electrical resistance varies as a result of an applied force. Stress is the term used to describe the internal resistance force that an object will exhibit to the external application of force, while a strain is the measure of the amount of deformation and displacement that the object will experience as a result of the applied external force.

3. Force Sensing Resistors (FSRs), also known as printed force sensors or force-sensitive resistors, are a type of piezoresistive sensing technology that consists of a semi-conductive material or ink which is sandwiched between two substrates that are separated by a spacer. When a force is applied to the device, a conductive film is deformed and presses against the conductive ink. With zero force applied, the sensor exhibits a very high resistance (on the order of Meg-ohms). The resistance drops inversely proportional to the applied force. The FSRs exhibit a linear increase in conductance with increased force.

As for the torque measurement, there are also two main ways in which it can be done:

1. Dynamic Torques Sensors :

    - Rotary torques sensors - used in applications on rotating shafts
    - Non-contacting torque sensors - that use magnetic or inductive technology to provide accurate measurements at high rotational speeds.

2. Static torque sensors are well suited to industrial applications. They offer long term reliability as they have non-moving parts. Static Torque sensors are used in applications where angular motion is limited and in-line torque measurements are required.

3.1.2 Use of a force/torque (f/τ) observer

However, as usage of physical transducers increases both cost and weight of the platform, the alternative option is the usage of a wrench observer, which can provide an accurate enough measure of the external wrench applied to the point in

which the force/torque sensor is placed $\hat{\omega}_E = [f_E^T, \tau_E^T]^T$, if there are accurate measurements of position, velocities and, if applicable, accelerations.
As is already mentioned in chapter 1, in the literature are found different implementations. From these it can be mentioned a Lyapunov-based non-linear observer for the external wrench ([6]), while in ([8]), the authors propose a mixed sensor-observer solution through which the human interaction forces can be separated from additional disturbances, like the wind for example.
In this work, as in [27], the hybrid approach from [7] is to be followed: the acceleration based observer from [6] for the estimation of the external interaction forces on the robot CoM, f_R, while the external torques, τ_R^R are estimated from a momentum-based observer [43].

External force estimation

The first disturbance observer uses the vehicle acceleration in order to estimate the contact forces. It was firstly proposed for aerial robots in [6].

$$\dot{\hat{f}}_R = L * (f_R - \hat{f}_R) = -L * \hat{f}_R - L * (m * \ddot{p}_R + m * g * e_3 - R_R * G_1 * u), \tag{3.1}$$

where L $\epsilon R^{3\times3}$ is the gain matrix to be designed, $\hat{f}_R$ is an estimate of f_R, F_1 contains all physical and geometrical properties of the platform and u contains the squared of all propellers' angular speeds.
If we define $e_f = f_R - \hat{f}_R$ then, under the assumption that the external force is constant or slowly varying, the error dynamics can be written as:

$$\dot{e}_f + L * e_f = 0 \tag{3.2}$$

This proves that, for any positive definite matrix L, the error dynamics is exponentially convergent to 0, so the force estimate converges to the real force value.

External torque estimation

For the estimation of the external torques, exerted by the external environment on the tool-tip, the already mentioned momentum-based observer ([**?**]) has been used. The starting point is the equation of the angular momentum q^R:

$$q^R = J * w_R^R \tag{3.3}$$

From the same system dynamics equations, the time derivative of 3.3 is :

$$\dot{q}^R = J * \dot{w}_R^R = -w_R^R X J * w_R^R + G_2 * u + \tau R^R \tag{3.4}$$

From 3.4, the estimate of the torque can be written as the following residual array:

$$\dot{\hat{\tau}}_R^R = K * [(q^R(t) - q^R(t_0)) + \int_{t_0}^{t} (w_R^R X J * w_R^R - G_2 * u - \hat{\tau}_R^R) d\tau] \tag{3.5}$$

K represents the observer's gain matrix, G2 is the part of the allocation matrix corresponding to the orientations part and u is the same as in the force estimator case. Taking the time derivative of 3.5 and also considering 3.4, the dynamic equation for the estimation of the torque will be:

$$\dot{\hat{\tau}}_R^R + K * \hat{\tau}_R^R = K * \tau_R^R \tag{3.6}$$

It represents a first order low-pass dynamic system. The torque estimation will converge to the real value of the torque for any positive definite gain matrix K. For greater values in K the estimation will convergence faster while smaller values will filter the high-frequency noise.

Wrench acting on the tool-tip

After both force and torque estimates are determined, the estimated wrench on the tool tip will be:

$$\hat{\omega}_E = H_E^{-T} * \begin{bmatrix} \hat{f}_R \\ R_R * \hat{\tau}_R^R \end{bmatrix}, H_E = \begin{bmatrix} I_3 & -[R_R * p_E^R]_\times \\ 0_3 & I_3 \end{bmatrix} \tag{3.7}$$

The numerical values of the observers' gain matrices that have been used starting from this point are detailed in the table 3.1. Furthermore, in order to have conditions

Observer	Gain Matrix	value
Force	L	$\begin{pmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 25 \end{pmatrix}$
Torque	K	$\begin{pmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 50 \end{pmatrix}$

Table 3.1: Gain matrices for Force and Torque observers

closer to real situations, noise on the state variables was added as white noise with the standard deviations detailed in table 3.2.

Validation of the chosen observer

As a side effect of the external wrench acting on the drone, it was expected to disturb the position and orientation tracking, in the case when the NMPC controller was not

state name x	value of the standard deviation
position	0.005 m
linear velocity	0.02 m/s
orientation angles	pi/180 radians
angular velocities x-y	0.15 rad/s
angular velocities z	0.05 rad/s

Table 3.2: Standard deviations for the state measurement noise

aware of this external wrench. This assumption is shown to be actually true in the following chapter.
However, at this point the main goal is to validate the chosen observer and an example simulation 4.3.4 will be borrowed from next chapter, where the external wrench was a combination between a sinus-shaped force and pulse-based torque. Both force and torque components are estimated correctly (3.1 and 3.2), which validates the correctness of the observer implementation.
One detail to mention is that there is an offset that appears for all the force components and this is explained by the presence of the noise components in the linear speed, which is then derived and used as the acceleration in the force estimation. Further details regarding it and also when it vanishes are outlined in the following chapter.

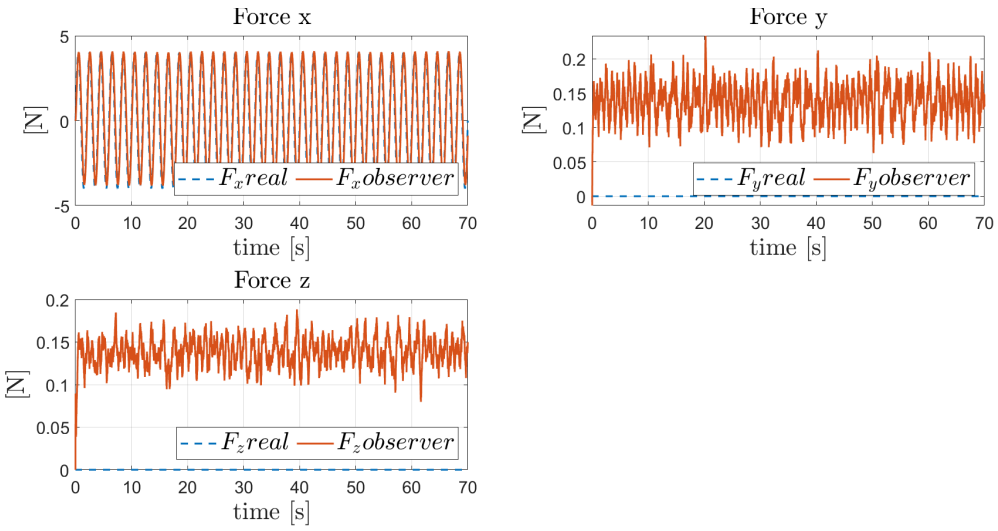


Figure 3.1: Disturbance force estimation

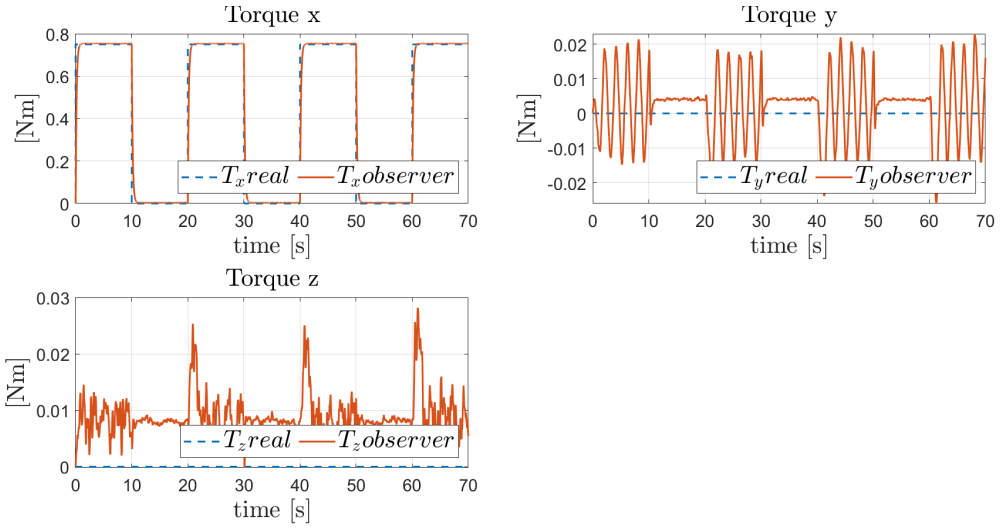


Figure 3.2: Disturbance torque estimation

3.2 Improving tracking performance

With the wrench observer validated at the previous sub-point, the next logical step was to make the NMPC controller aware of the external wrench. This was needed in order to improve the position and orientation tracking in the presence of the disturbance wrench because the goal in this step of the design process was to be able to regulate the pose of the drone while being in contact with a surface.
Naturally, this has been implemented by including it in the equations of motion that appear in the model of the plant used in the constraints' part of the OCP problem, as detailed in 3.8, where the added components are outlined in magenta. The controller assumes that the external force and torque are constant. Consequently, the values that appear in x_{dot} corresponding to the derivatives of these are 0. This represents the case when both external force and torque are included in the state vector, with the corresponding force and torque terms appearing in the equations when either the force or torque (or both) are not included in the state vector. In the simulations part it will be seen that the controller can handle quasi-static contact wrench (sinus-shaped, pulse-shaped), while still considering it as constant in its model part. The limitations will be tested and outlined while increasing the frequency of the sinus-shaped disturbance wrench.
Similarly, as already stated, the state vector was changed, depending on the inclusion of either the force (3.9), torque (3.10) or both components (3.11) in it. The

objective function, as well as the output vector, remained the same as initially.

$$x_{dot}(t) = \begin{bmatrix} \dot{p}(t) \\ 0 + (G1(1,:) * [f_1; f_2; f_3; f_4; f_5; f_6] + \hat{f_{R\_x}})/m \\ 0 + (G1(2,:) * [f_1; f_2; f_3; f_4; f_5; f_6] + \hat{f_{R\_y}})/m \\ -g + (G1(3,:) * [f_1; f_2; f_3; f_4; f_5; f_6] + \hat{f_{R\_z}})/m \\ p + r * cos(\phi) * tan(\theta) + q * sin(\phi) * tan(\theta) \\ q * cos(\phi) - r * sin(\phi) \\ r * cos(\phi)/cos(\theta) + q * sin(\phi)/cos(\theta) \\ q * r * (I_y - I_z)/I_x + (G2(1,:) * [f_1; f_2; f_3; f_4; f_5; f_6] + \hat{\tau_{R\_x}})/I_x \\ r * p * (I_z - I_x)/I_y + (G2(2,:) * [f_1; f_2; f_3; f_4; f_5; f_6] + \hat{\tau_{R\_y}})/I_y \\ q * p * (I_x - I_y)/I_z + (G2(3,:) * [f_1; f_2; f_3; f_4; f_5; f_6] + \hat{\tau_{R\_z}})/I_z \\ \dot{f}_1 \\ \dot{f}_2 \\ \dot{f}_3 \\ \dot{f}_4 \\ \dot{f}_5 \\ \dot{f}_6 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.8}$$

$$x = [p; \dot{p}; \eta; \omega; f_1; f_2; f_3; f_4; f_5; f_6; \hat{f_{R\_x}}; \hat{f_{R\_y}}; \hat{f_{R\_z}}] \tag{3.9}$$

$$x = [p; \dot{p}; \eta; \omega; f_1; f_2; f_3; f_4; f_5; f_6; \tau_{R\_x}; \tau_{R\_y}; \tau_{R\_z};] \tag{3.10}$$

$$x = [p; \dot{p}; \eta; \omega; f_1; f_2; f_3; f_4; f_5; f_6; \hat{f}_{R\_x}; \hat{f}_{R\_y}; \hat{f}_{R\_z}; \hat{\tau}_{R\_x}; \hat{\tau}_{R\_y}; \hat{\tau}_{R\_z}] \tag{3.11}$$

The current design objective is validated also in the following chapter. Taking as example the case where an external wrench is simulated consisting of both force and torque, it is seen that including both of them in the NMPC's state vector will lead

to both position and orientation improvement. Position is regulated (3.3) on all axes. The Euler angles are also regulated (3.4).
The

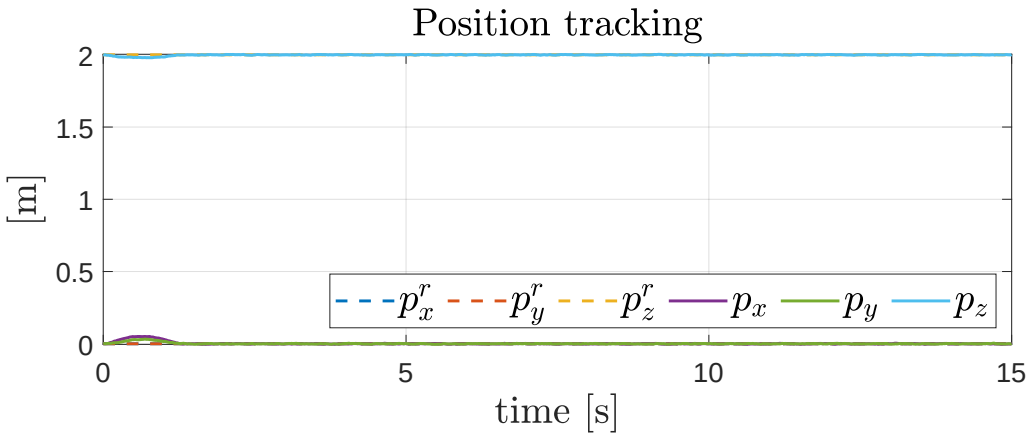


Figure 3.3: Position tracking

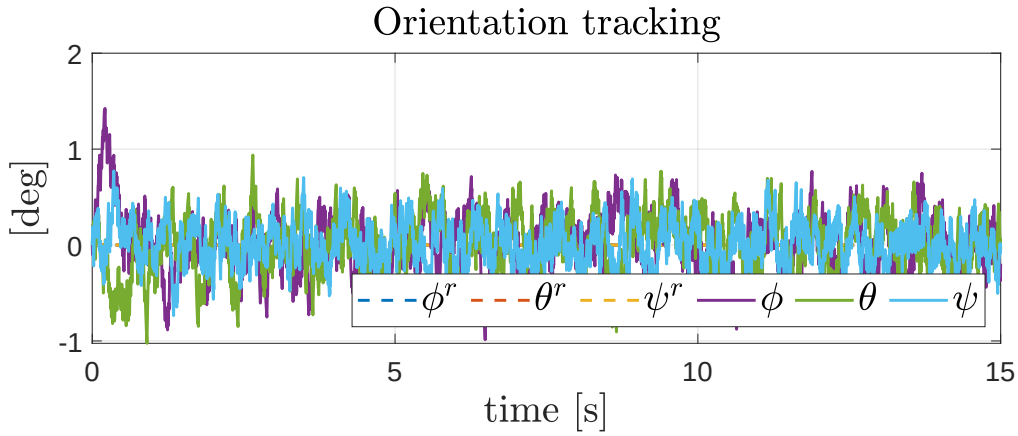


Figure 3.4: Orientation tracking

3.3 Introducing wrench regulation

The last and final step of the thesis was related to the possibility to control also the force while performing a certain trajectory. The force and position cannot be regulated with the same accuracy on the same dimension because they are interdependent. A certain steady-state error for the force tracking will determine a certain error in the position tracking, while the same behavior will be observed vice-versa (a certain position steady-state error will determine a certain force steady-state error). This is why the final goal of the thesis (mixed motion/interaction control) was split into

force tracking on one dimension and position tracking on the other 2 dimensions.The state function was kept the same as in the previous part (with its particularities depending on the inclusion of force or torque, as it was already explained), with the observation that the external force was now simulated as a state-dependent one.
In the following paragraph the topic of contact forces will be shortly expanded before explicitly giving the equation for the one chosen for this thesis.
The two main components of contact forces can be broken down into

- Normal forces: Push objects away from each other when they collide. As their name suggests, these forces are normal to the point(s) of contact.
- Friction forces: Prevent objects in contact from perfectly sliding off each other. These forces act along the contact surface, and are related to the smoothness or roughness of contact surfaces.

Normal forces can be modeled in several ways, including:

- Impulse-based: This is a one-time event that occurs when two objects collide. Before collision, each object has a certain momentum (or rotational equivalent). After the collision, the magnitude and direction of each object's momentum will change based on the type of collision; for example, elastic vs. inelastic.
- Force-based: Normal forces are applied to objects based on a force law. A common approach is to apply a penalty force; for example, treating contact surfaces as springs and dampers whose coefficients approximate real world behavior. Penalty forces allow objects to overlap, which can approximate the deformation (or "squishing") of these objects when they collide.
- Motion constraint: Here, we assume that objects are always in contact, so we do not have to worry about collision dynamics. For example, if we assume that a wheel is always on the ground, we do not need to model the normal forces; we only need a friction model to relate the rotation of the wheel with its linear motion.

Friction forces typically consist of a force law with two distinct regions.

- Static friction represents the initial force needed by an object at rest to begin sliding along its contact surface.
- Kinetic friction represents the resistive force of an object when it is moving along its contact surface.

Here it was chosen to represent a force-based normal contact force, as it was the best approximation out of the types that were outlined above. The damper component was not included in the model because it was assumed that the platform has

not a deformation or energy loss due to the contact with the contact surface. Thus, the final model was that of a spring (3.12), where K_s is the spring constant and Δx is the position error and equal to the difference between the position of the MRAV's COM and the rest point assumed to be in the world-frame origin. This also caused a slight modification in the x_{dot} vector, in the position corresponding to the derivative of this force, that now we can call a contact one (3.13). This modification was needed because it was observed during simulations that the assumption of being constant led to both wrong force estimation and the impossibility to be regulated.
Furthermore, in order to be able to actually regulate it, the difference between the contact force and its reference has been added to the optimization vector that has been used until this part (3.14), where again the added component is shown in magenta.
Another important observation to be mentioned at this point is that the spring constant plays the role of a tuning parameter. In the moment when it is increased, it will lead to lower steady-state force error and also better position regulation on the same dimension on which the contact force is applied.
Choosing a representative simulation from the following chapter (4.5.2) it is actually seen that the main objectives are achieved.
Position is regulated (3.3) on y and z, as desired.The steady-state error on x axis is due to the imposed contact force on the same axis, as already mentioned.
The second objective, which is the force tracking, is also achieved (4.177).

$$\hat{f_R}\_x = K_s \Delta x \tag{3.12}$$

$$x_{dot}(t) = \begin{bmatrix} \dot{p}(t) \\ 0 + (G1(1,:)[f_1; f_2; f_3; f_4; f_5; f_6] + f_{R\_}x)/m \\ 0 + (G1(2,:)[f_1; f_2; f_3; f_4; f_5; f_6] + f_{R\_}y)/m \\ -g + (G1(3,:)[f_1; f_2; f_3; f_4; f_5; f_6] + f_{R\_}z)/m \\ p + rcos(\phi)tan(\theta) + qsin(\phi)tan(\theta) \\ qcos(\phi) - rsin(\phi) \\ rcos(\phi)/cos(\theta) + qsin(\phi)/cos(\theta) \\ qr(I_y - I_z)/I_x + (G2(1,:)[f_1; f_2; f_3; f_4; f_5; f_6] + \tau_{R\_}x)/I_x \\ rp(I_z - I_x)/I_y + (G2(2,:)[f_1; f_2; f_3; f_4; f_5; f_6] + \tau_{R\_}y)/I_y \\ qp(I_x - I_y)/I_z + (G2(3,:)[f_1; f_2; f_3; f_4; f_5; f_6] + \tau_{R\_}z)/I_z \\ \dot{f}_1 \\ \dot{f}_2 \\ \dot{f}_3 \\ \dot{f}_4 \\ \dot{f}_5 \\ \dot{f}_6 \\ settings.K_s u \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.13}$$

$$y(t) = \begin{bmatrix} p(t) \\ \eta(t) \\ \dot{p}(t) \\ \omega(t) \\ \ddot{p}(x(t), u(t)) \\ \dot{\omega}(x(t), u(t)) \\ f_{R\_}x \\ f_{R\_}y \\ f_{R\_}z \\ \tau_{R\_}x \\ \tau_{R\_}y \\ \tau_{R\_}z \end{bmatrix} \tag{3.14}$$

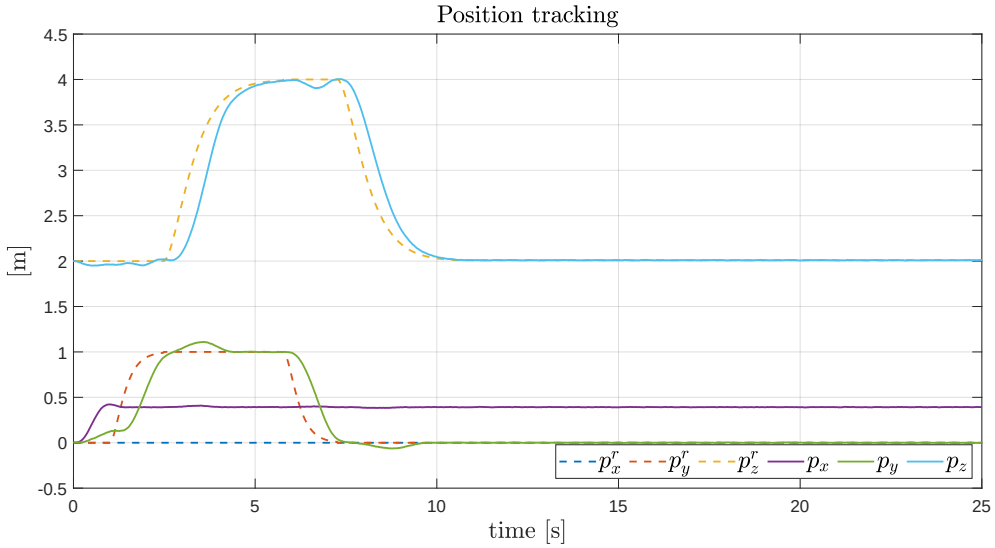


Figure 3.5: Position tracking

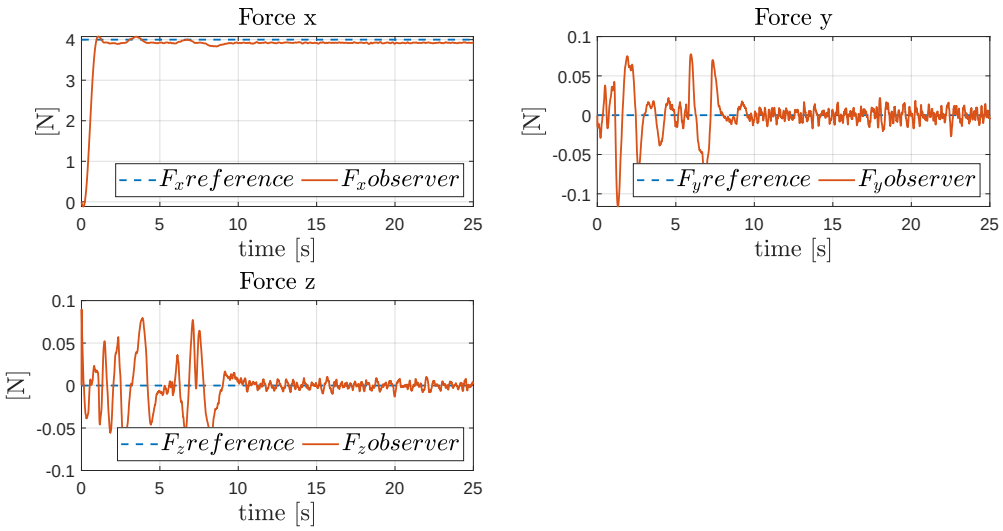


Figure 3.6: Reference force tracking

3.4 Final discussion

This chapter outlined gradually the 4 main steps that lead the improvement of the initial controller (2.3) which targeted only trajectory following applications to the final version of the controller (3.7), which is able to track mixed trajectory and force references, with application also in aerial-physical interaction.

For this it was explicitly shown, where applicable, the modifications/additions that were made for the state vector, reference/output vector and cost/optimization function in order to achieve every intermediate goal of the thesis: meaningful quantities were introduced (contact wrench components, spring constant, etc.) and their pur-

pose was explained. In addition, plots from representative simulations were used from the following chapter in order to validate these intermediate goals: validation of the external observer algorithm, position and orientation tracking improvement when the external wrench is included in the state used by the controller or the regulation of the external force when an additional term containig it appears in the optimization/cost function. The detailed simulations with more conclusions are detailed in the following chapter.

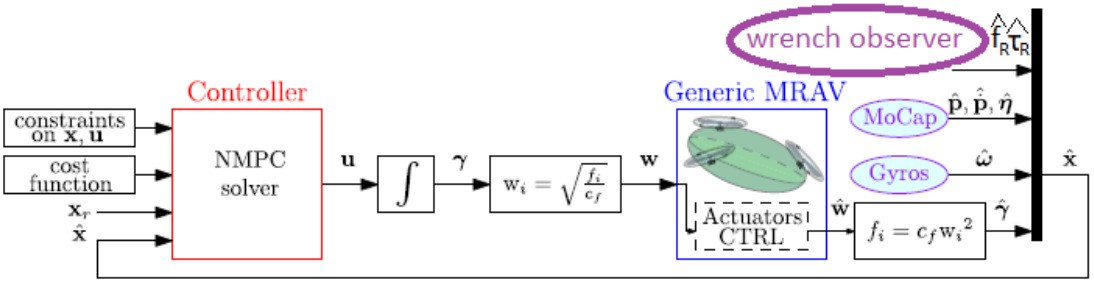


Figure 3.7: Final architecture of the controller - trajectory following + force tracking simulations

Chapter 4

Simulations

4.1 Chapter overview

Current chapter groups all the simulations that have been performed in the scope of the thesis goal, purpose and motivation. From each incremental step a number of representative simulations have been chosen.
In the first subsection an overview of the settings used, together with a brief motivation for the reasoning will be also provided.
Then, in the following 4 subsections the sets of simulations related to the main steps of the design development will be detailed.

4.2 Trajectory following simulations

4.2.1 Square trajectory tracking

The first simulation from this part will assume only tracking the position's trajectory. The weights are detailed in table 4.2.
The state constraints (which are valid for almost all simulations) are detailed in table 4.1. Besides the constraints on the positions, which are dependent on the considered application, the ones on the roll and pitch maintain the stability of the platform. The limits on the propellers' forces, which were detailed in the previous chapter, take into consideration for this set of simulations also conservative limits, which make them span the interval [0.3...8.3N].

The position is regulated (4.1 and (4.2)), but it has tracking error and also a disturbed behavior for the y axis before the final regulated value.
The linear speed (4.3), which has lower weights, is followed with little perturbations on the z component, while on the x and y axes it has a maximum of 1.9 m/s in the

State Parameter	Upper value	Lower value
x	5	-5
y	5	-5
z	5	0.3
$roll$	$pi/2$	$-pi/2$
$pitch$	$pi/2$	$-pi/2$
$propellers' forces$	$0.3N$	$8.3N$

Table 4.1: State constraints

Measurement	x	y	z
pos	300	300	300
att	500	500	500
vel	5	5	5
$avel$	1	1	1
acc	$1e-4$	$1e-4$	$1e-4$
$aacc$	$1e-4$	$1e-4$	$1e-4$

Table 4.2: Weights

middle of each motion, behavior which is also seen in the position error figure (4.2). This shows that the MRAV is accelerating and then decelerating while going from one corner of the square to the following one.
Roll and pitch angles have maximum errors of 4-5 degrees, while the yaw angle has one of less than 2 degrees (4.4 and 4.5). While moving on x, the biggest influence is on pitch angle, but however there are some extra influences on the roll and yaw also. Similar corresponding behavior is present also for the movement on y (main roll influence, parasitic behavior on pitch and yaw). The attitude errors are caused mainly by the dynamic couplings caused by the saturation of the propellers, which is seen in the thrust evolution figure (4.7).
Some of the thrusts derivatives (4.8) reach their limits during the simulation, which prove the effectiveness of the controller, the stability of the platform being maintained.

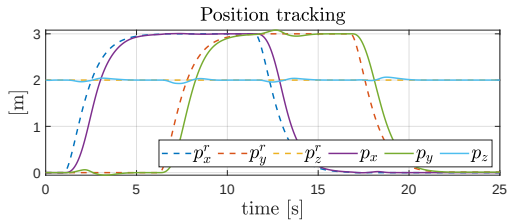


Figure 4.1: Position tracking

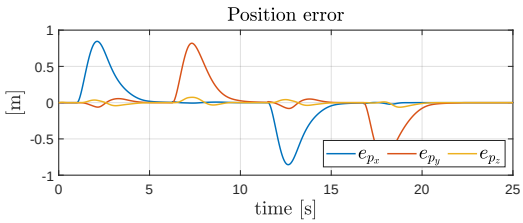


Figure 4.2: Position error

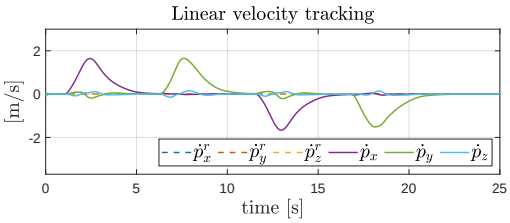


Figure 4.3: Linear velocity tracking

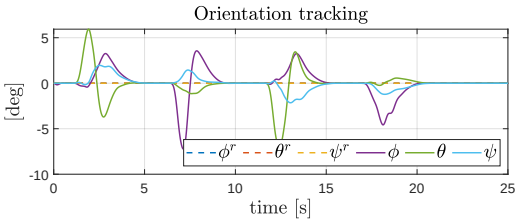


Figure 4.4: Attitude tracking

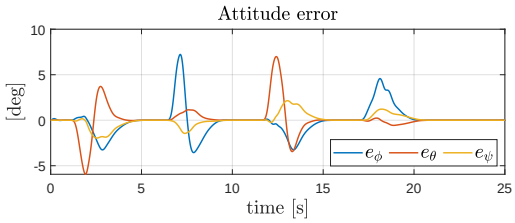


Figure 4.5: Attitude error

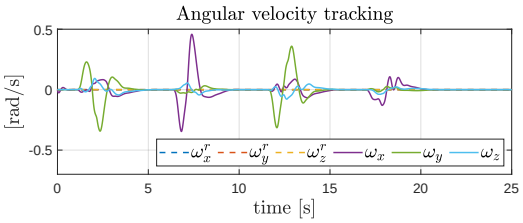


Figure 4.6: Angular speed

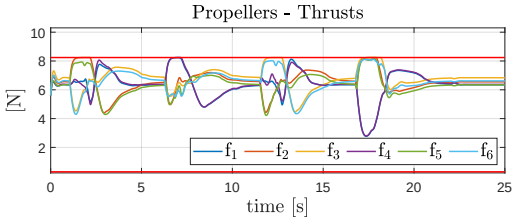


Figure 4.7: Propellers thrusts

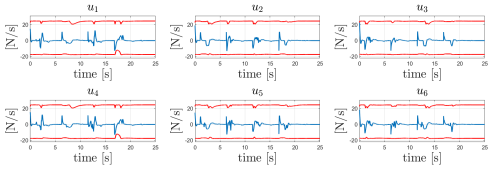


Figure 4.8: Propellers thrusts derivatives

We have seen that the chosen weights assure position control. However, the propellers' speeds get saturated because the trajectory is unfeasible, which further introduce dynamic couplings that cause the orientation angles to have non-zero errors during the motion. Furthermore, the most disturbed angle is the one on the opposite coordinate w.r.t. the one on which the drone is currently moving. However, the stability of the platform is assured, even though the controller commands/ process inputs are also reaching their limits during the simulation .

4.2.2 Square trajectory and sinus orientation tracking

The second simulation outlined for this part introduces also sine signals for the roll (while moving on the y axis) and pitch (while moving on the x axis). The weight on position is taken as 500, while the weight on attitude is taken as 300 (4.3).
In the position tracking figure, it can be seen that it is similar to the previous one. The increased weights don't contribute to an improvement in the position tracking in this case, possibly due to change in the orientation reference. The same behavior as in the previous simulation is also seen in the position error figure (4.10) and linear velocity (4.11).
The pitch and roll angles have maximum errors of 10-12 degrees, and this happens around the peaks of the sinus signals. This may be due to the impossibility of the drone to follow those portions of the signals, because the reference signal varies quickly in that region. In rest, the pitch and roll angles follow quite well the reference signals.The yaw angle has a maximum error of approx. 3 degrees. The angular velocity has also acceptable errors, with maximum of approx. 0.5-0.6rad/s,while the yaw rate has a maximum of 0.2 rad/s.
The propellers' thrusts are again saturated, while the thrusts derivatives reach their (lower) limits again. Again, even if saturation of the thrust derivatives is reached , stability is maintained.

Measurement	x	y	z
pos	500	500	500
att	300	300	300
vel	5	5	5
$avel$	1	1	1
acc	$1e-4$	$1e-4$	$1e-4$
$aacc$	$1e-4$	$1e-4$	$1e-4$

Table 4.3: Weights

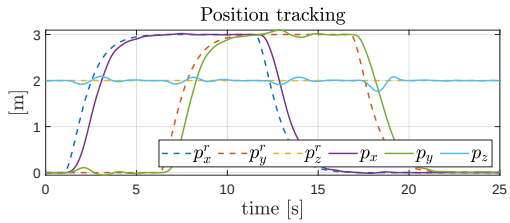


Figure 4.9: Position tracking

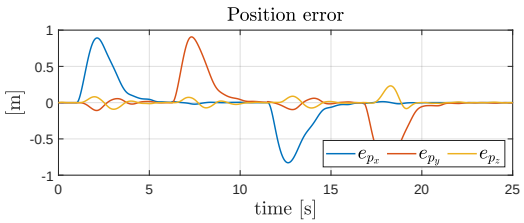


Figure 4.10: Position error

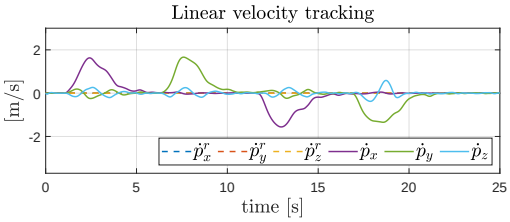


Figure 4.11: Linear velocity tracking

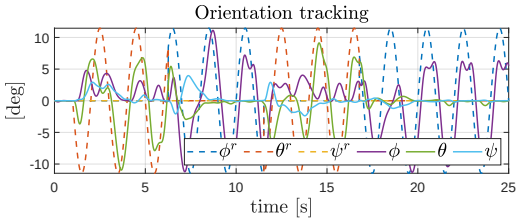


Figure 4.12: Attitude tracking

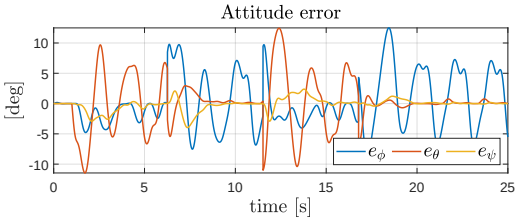


Figure 4.13: Attitude error

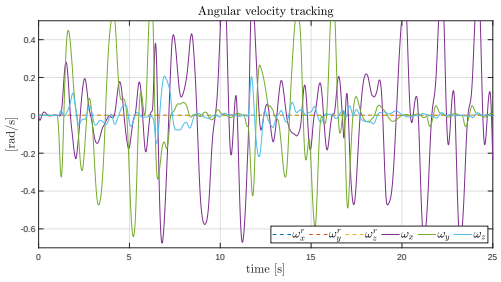


Figure 4.14: Angular speed

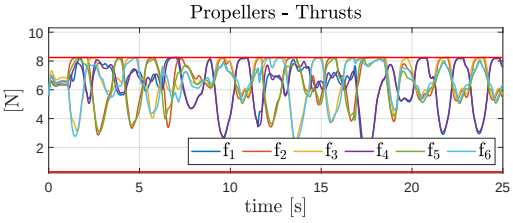


Figure 4.15: Propellers thrusts

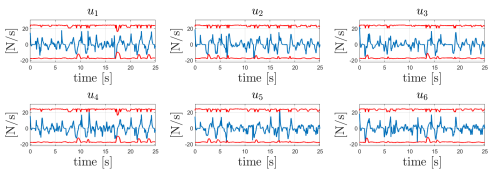


Figure 4.16: Propellers thrusts derivatives

max error position x [m] 0.891803	max error position y[m] 0.906203	max error position z[m] 0.231106
max error roll x [rad] 0.218010	max error pitch[rad] 0.217008	max error yaw[rad] 0.069195
RMS error position x[m] 0.891803	RMS error position y[m] 0.906203	RMS error position z[m] 0.231106
RMS error roll[rad] 0.218010	RMS error pitch[rad] 0.217008	RMS error yaw[rad] 0.069195

Table 4.4: max and RMS for position and attitude errors subsection 4.2.2.

In this simulation the chosen weights assure again the 6D position and orientation regulation. In this case however a different reference is given for the attitude and bigger maximum errors are observed, possible due to its shape and/or as a consequence of the dynamic couplings introduced by the propellers' saturations. Stability of the platform is still assured, even though the controller commands/ process inputs are also reaching their limits during the simulation.Starting with this simulation, for the interested reader, but also in order to outline certain observations, also tables with maximum and RMS error values for the position and orientation will be provided. The one corresponding to this simulation is table 4.4.

4.2.3 Square trajectory and sinus orientation tracking 2

In the last simulation of trajectory following the weights on the orientation were increased and the ones on the position were decreased: the ones on position are now taken as 100, while the weights on attitude are taken as 500 (table 4.3).
Although the position and orientation seem to be similar with the ones from previous simulation, the improvement in orientation vs. slightly decrease in position tracking is seen in the max and RMS values of the position and orientation errors (table 4.6 vs. table 4.4) and also in the position and orientation error figures (4.18 and (4.21)).
The angular velocity profile is similar to the previous one, as the weights are unchanged (4.22).
The propellers' thrusts are again saturated (4.23), while the thrusts derivatives reach their (lower) limits again (4.24), proving again that stability is maintained.

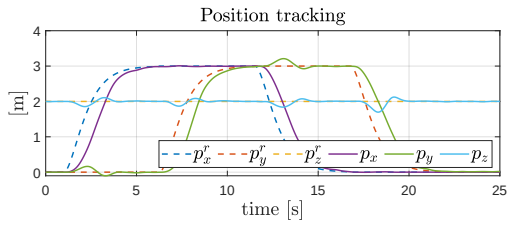


Figure 4.17: Position tracking

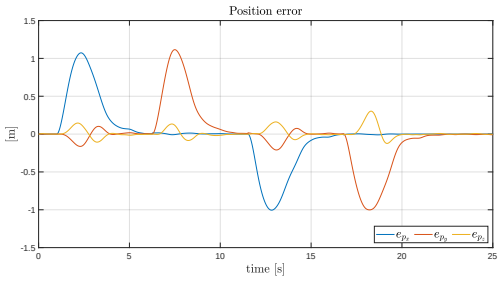


Figure 4.18: Position error

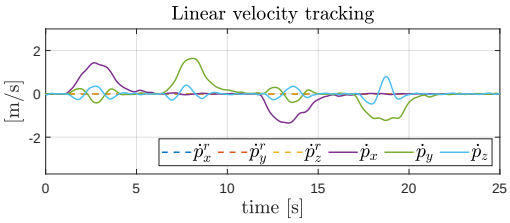


Figure 4.19: Linear velocity tracking

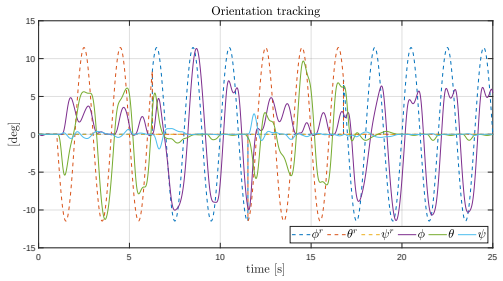


Figure 4.20: Attitude tracking

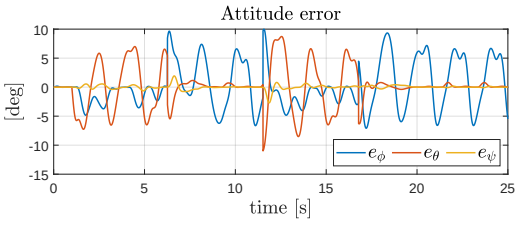


Figure 4.21: Attitude error

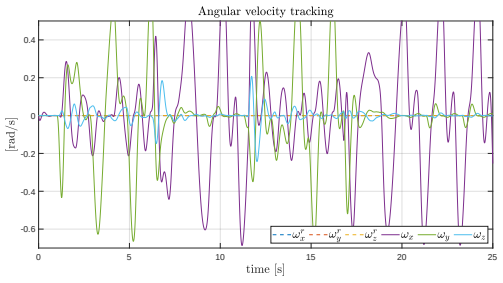


Figure 4.22: Angular speed

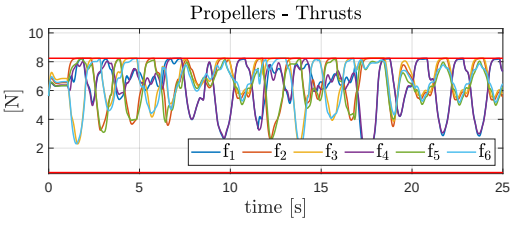


Figure 4.23: Propellers thrusts

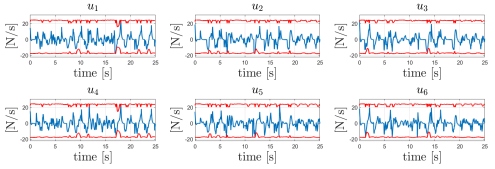


Figure 4.24: Propellers thrusts derivatives

Measurement	x	y	z
pos	100	100	100
att	500	500	500
vel	5	5	5
$avel$	1	1	1
acc	$1e-4$	$1e-4$	$1e-4$
$aacc$	$1e-4$	$1e-4$	$1e-4$

Table 4.5: Weights

max error position x [m] 1.073269	max error position y [m] 1.115232	max error position z [m] 0.302865
max error roll x[rad] 0.174078	max error pitch[rad] 0.191661	max error yaw[rad] 0.047213
RMS error position x[m] 1.073269	RMS error position y[m] 1.115232	RMS error position z [m] 0.302865
RMS error roll [rad] 0.174078	RMS error pitch[rad] 0.191661	RMS error yaw[rad] 0.047213

Table 4.6: max and RMS for position and attitude errors 4.2.3.

In this simulation the chosen weights on position were decreased, while the ones on attitude were increased. While the 6D position and orientation was still assured, the real purpose was to give more importance to the attitude tracking and in the same time obtaining a set of weights to be used in the following simulations. The reference 6D was the same as in the previous simulation and all the other conclusions hold : stability maintained, controller commands and propellers' forces saturated.

4.3 External wrench disturbance. Validation of the observer and tracking performance of an agnostic NMPC

In this part, as it was already stated in chapter 3, the Observer algorithm was added into the main Trajectory Following simulation.
The purpose of the simulations in this part is mainly to test the capability of the observer to correctly provide the values of the simulated contact wrench (forces and/or torques), while also analyzing the effect of the introduced contact wrench on the position and attitude tracking, when the NMPC controller is agnostic about it, i.e. a model of the contact wrench is not included in the NMPC's state or OPC criteria for

these simulations.
For this, a number of simulations were performed with the disturbance force and torque considered as constant, sinus-shaped and pulse-type. Finally, a combination between a sinus-shaped force and a pulse-typed torque was also tried, without and with noise on the state variables.
The scenario used in these simulations is the hovering one.
The constraints are kept the same as in the previous trajectory simulations, while the weights for position and attitude are given in table 4.7.
As already pointed, the weights are derived from the results that were obtained from previous simulations. As the conclusion from previous part was that position weights should be between 50 and 100, here they will be set for 100, while keeping the attitude weights at 500. For the final simulations however, further tuning may be needed.
The observer gains, both for force observer (L) and torque observer (K) are detailed in table 4.8, same as already presented in previous chapter, together with the filtering ones used for each variable of the acceleration-based observer. This represents an additional filter in order to further process the acceleration-based force and torque observer values. In real experiments, low-pass filtering is needed to reduce the disturbances due to noise measurements. The benefit of the filter may be seen after additional noise will be added to the state measurements before feeding them back to the optimization-based controller.
The choice for the diagonal form of the observer gains was made in order not to have influence between different components of the force/torque observers. Furthermore, in the Observer sub-model there is a different way of determining the contact force and torque that doesn't use the information from the accelerometer, useful in the case when the vehicle doesn't have a sensor for acceleration measurement. In that case, both force and torque are determined using only the momentum formula and this is the reason why K matrix has dimension 6*6. In the acceleration-based observer case however, only the last 3*3 matrix from K is used for the torque observer.
It is supposed that the external wrench is acting on the Center of Mass.

4.3.1 Sinus-shaped disturbance force

The first selected simulation takes into account a sinus-shaped disturbance force with the amplitude of 4N on the x dimension.
In the force tracking figures (4.33), the estimated contact force quickly converges to the real value. Both y and z components have components with the negligible amplitudes.

Measurement	x	y	z
pos	100	100	100
att	500	500	500
vel	5	5	5
$avel$	1	1	1
acc	$1e-4$	$1e-4$	$1e-4$
$aacc$	$1e-4$	$1e-4$	$1e-4$

Table 4.7: Weights controller

Observer	Gain Matrix	value
Force	L	$\begin{pmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 25 \end{pmatrix}$
Torque	K	$\begin{pmatrix} 50 & 0 & 0 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 \\ 0 & 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 0 & 50 \end{pmatrix}$
Filter values		$A = 0.96$ $B = 0.004$ $C = 10$ $D = 0$

Table 4.8: Gain matrices for Force and Torque observers and acceleration-based filter value

The torques (4.34) also follow periodic behaviors, with negligible amplitudes.
The position (4.25 and 4.26) is again perturbed, as a consequence of the contact force. The x dimension has a sinusoidal shaped error with the amplitude of approx. 0.2 m. The linear velocity (4.27) follow exactly the same pattern, with the speed on x having the biggest amplitude, of about 0.2 m/s, while the ones of y and z have again negligible amplitudes.
The Euler angles (4.28 and 4.29) follow also sinusoidal evolutions. The yaw has the smallest amplitude, while pitch has one of approx. 4 deg and roll one of approx. 2 deg. The angular velocity (4.30) follows the reference of 0 rad/s for all angles.
The MRAV has non-zero steady-state errors on x and the biggest error on the pitch as a consequence of the presence of the disturbance force.
The propellers' thrusts' (4.31) are again saturated and the thrusts derivatives (4.32)

reach their limits.

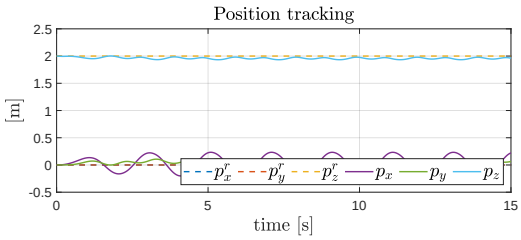


Figure 4.25: Position tracking

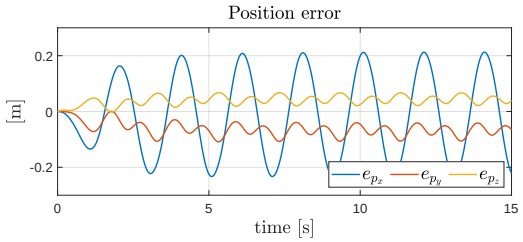


Figure 4.26: Position error

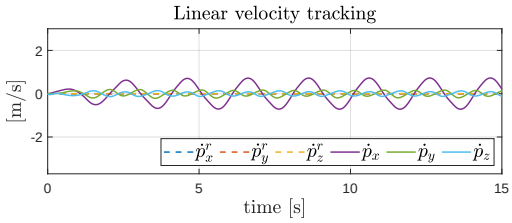


Figure 4.27: Linear velocity tracking

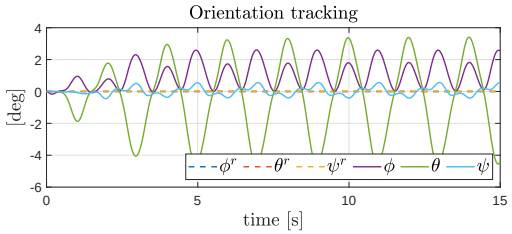


Figure 4.28: Attitude tracking

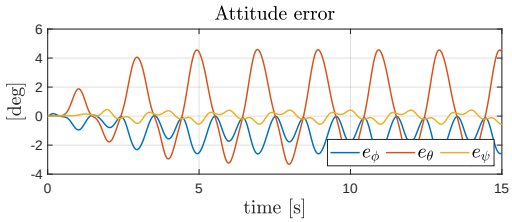


Figure 4.29: Attitude error

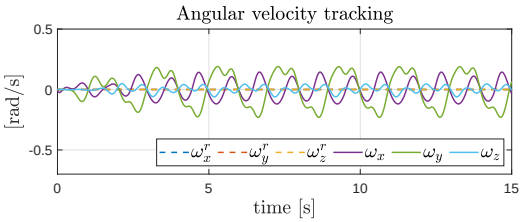


Figure 4.30: Angular speed

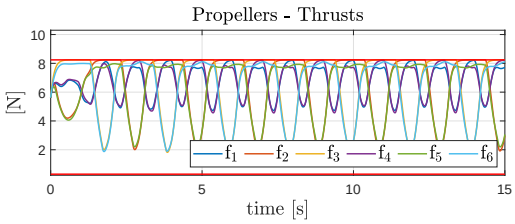


Figure 4.31: Propellers thrusts

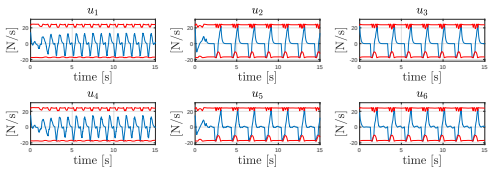


Figure 4.32: Propellers thrusts derivatives

In this simulation we have seen that a disturbance force simulated on one dimension of the Cartesian space causes the biggest error on the same position

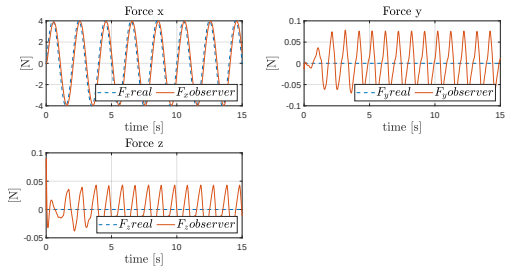


Figure 4.33: Force estimated by Observer

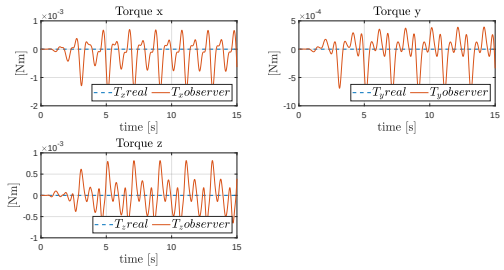


Figure 4.34: Torque estimated by Observer

dimension and possibly on the opposite attitude axis (in the x-y plane), depending on the considered performance criteria, chosen weights and the full actuated property (the result of all three mentioned criteria determine the second outcome). Furthermore, because in this case there are dynamic couplings due to the saturation of propellers, there are extra disturbances on the position opposite axis corresponding to the force's axis (y in this case) and on the attitude of the same axis (roll in this case). Force is correctly estimated, stability is maintained.

4.3.2 Pulse-shaped disturbance torque

In this simulation, a pulse shaped contact torque with a period of 10 s for 0.75 Nm and another 10 s for 0 Nm is simulated on the x direction given in the Body Frame. In the torque tracking figures (4.44), the contact torque is estimated very fast, while the y and z components are negligible.
The contact forces (4.43) have spikes of negligible maximum value, so it can be inferred that the estimations of 0N is correct.
The position (4.35) is not perturbed because the propellers are not saturated. It can be seen however in the position error figure (4.36) that the y dimension is the most disturbed, as a confirmation of the general rule that it was inferred already from previous simulation. The linear velocity (4.37) tracks the reference without error.
The roll has a maximum error of approx. 11 degrees, while the pitch and yaw are tracked without error (4.38). In the angular velocity figure, the roll rate error is the biggest, with spikes between 0.7 and 0.9 rad/s, while the other 2 angular rates are not perturbed.
The roll angle and rate are disturbed as a consequence of the disturbance torque on x dimension).
The propellers' thrusts (4.41) are not saturated, as it was already mentioned in the chain of causality of position tracking.
Thrusts derivatives 2 and 3 reach their upper limits at the beginning of the motion

(4.42).

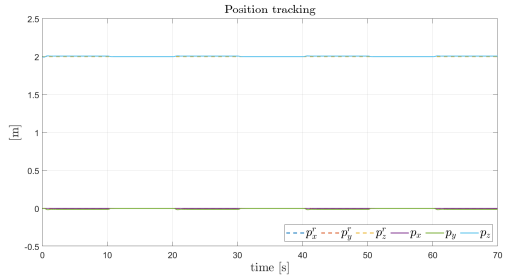


Figure 4.35: Position tracking

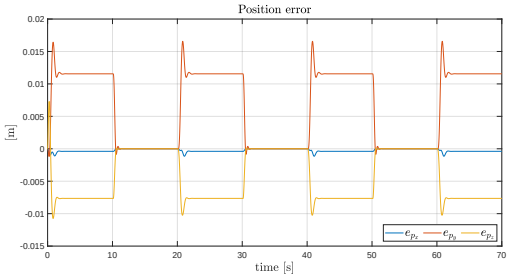


Figure 4.36: Position error

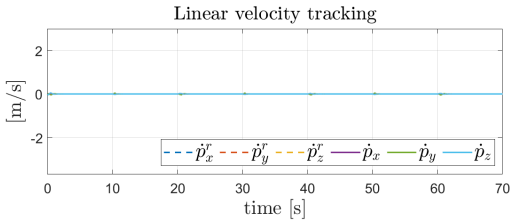


Figure 4.37: Linear velocity tracking

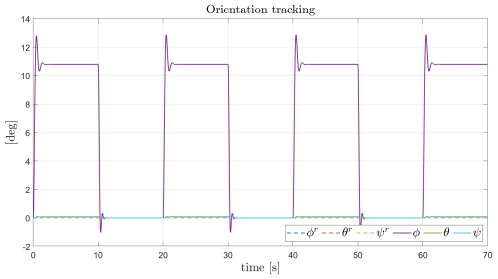


Figure 4.38: Attitude tracking

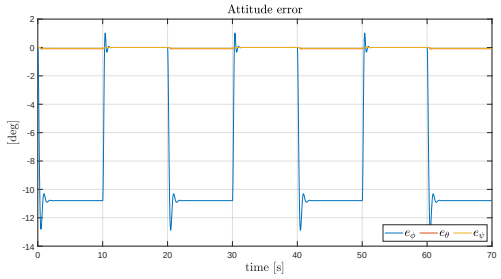


Figure 4.39: Attitude error

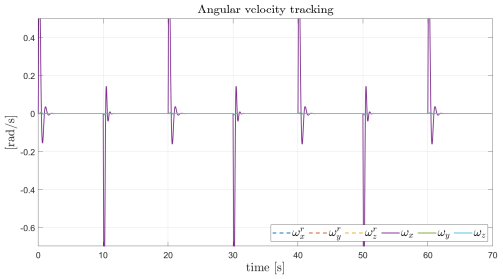


Figure 4.40: Angular speed

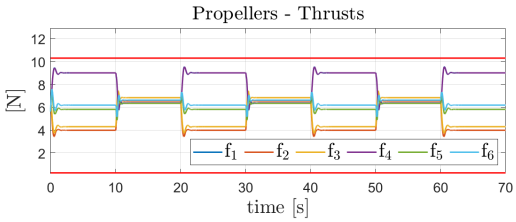


Figure 4.41: Propellers thrusts

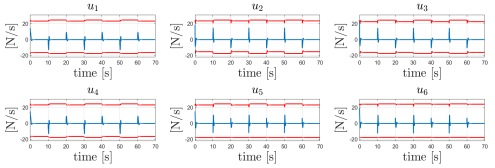


Figure 4.42: Propellers thrusts derivatives

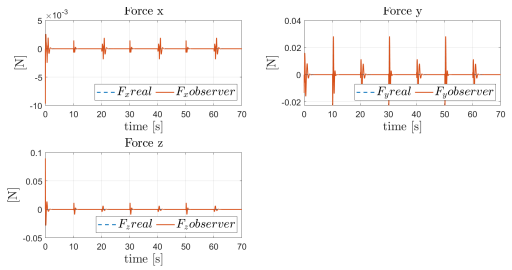


Figure 4.43: Force estimated by Observer

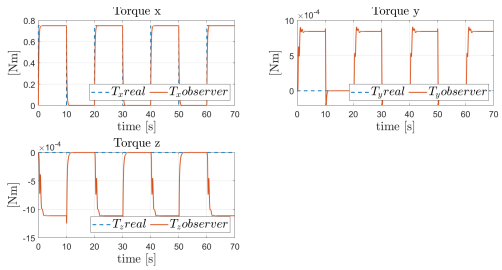


Figure 4.44: Torque estimated by Observer

In this simulation the conclusions from previous simulation were validated in the case of a pulse-shaped disturbance torque. Analog to previous case, the disturbance torque causes the biggest error on the same orientation dimension (roll in this case) and possibly on the opposite position axis (in the x-y plane). This second outcome is not observed here, due to the (quite) big weights and/or full-actuated property. As the propellers are not saturated for this simulation, the extra disturbance mentioned before (for this case it should have been pitch and position on x, i.e. attitude on the opposite dimension and position on the same axis) also do not appear. Torque is correctly estimated, stability is maintained.

4.3.3 Sinus-shaped disturbance force and pulse-shaped disturbance torque

A combination between previous 2 conditions for the external wrench has been tried out (a sinus-shaped contact force of 4N and a pulse-shaped contact torque of 0.75Nm).
The contact force (4.53)is calculated instantly. On the y component appear some influences of the torque, but overall the y and z components are negligible.
Similar observations can be drawn for torque tracking simulations (4.54). The one on x is estimated correctly,while the other ones are negligible.
In the position figure (4.48) the x component is the most perturbed, as it was expected. Dynamic couplings introduce extra disturbances on y and z components, which can be also considered negligible (dependent however on the desired performance criteria).
The roll has a maximum error of approx. 12 degrees, and this applies in the moment when the torque pulse is simulated. It is coupled with bigger yaw errors, while in the moment when the torque is 0 Nm, the pitch angle has the biggest error, of 2 degrees, followed by the roll angle (4.48). The angular velocities have, except spikes for roll rate, the amplitudes of maximum 0.1 rad/s (4.50).

The expected measurements are affected the most, the position on the x dimension and the roll angle. Moreover, position on y dimension, the pitch and yaw angles are more disturbed than in previous simulations because of the dynamic couplings induced by the actuators' saturation(4.51).
The thrusts derivatives reach their bounds (4.54).

max error position x [m] 0.242898	max error position y [m] 0.143291	max error position z [m] 0.074533
max error roll x[rad] 0.243314	max error pitch[rad] 0.024607	max error yaw[rad] 0.035846
RMS error position x[m] 0.242898	RMS error position y[m] 0.143291	RMS error position z [m] 0.074533
RMS error roll [rad] 0.243314	RMS error pitch[rad] 0.024607	RMS error yaw[rad] 0.035846

Table 4.9: max and RMS for position and attitude errors subsection 4.3.3.

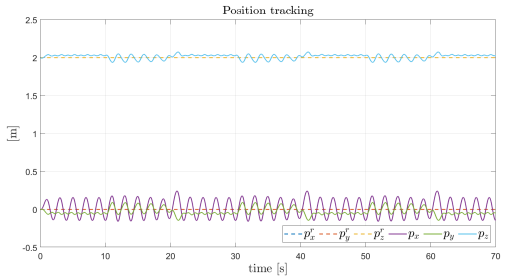


Figure 4.45: Position tracking

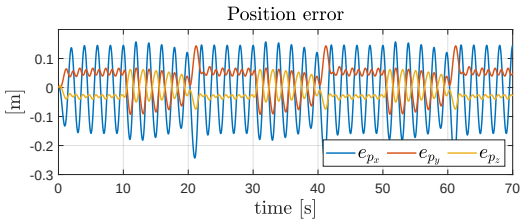


Figure 4.46: Position error

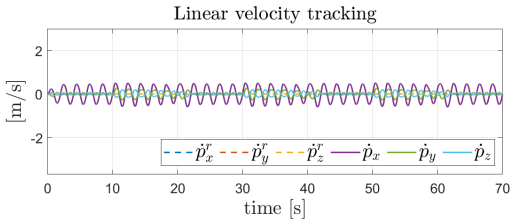


Figure 4.47: Linear velocity tracking

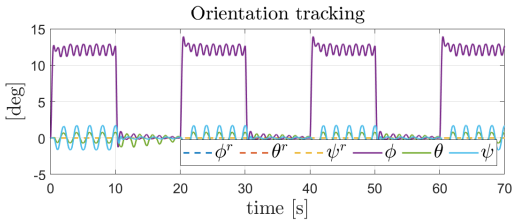


Figure 4.48: Attitude tracking

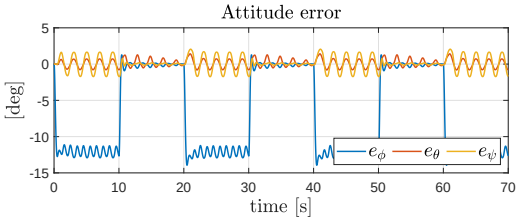


Figure 4.49: Attitude error

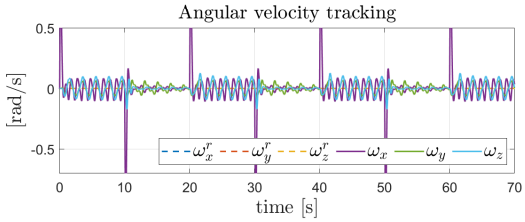


Figure 4.50: Angular speed

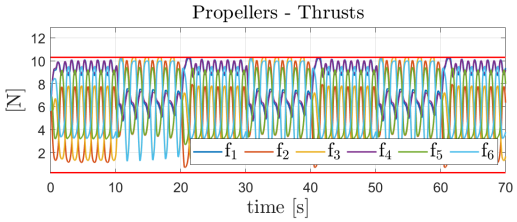


Figure 4.51: Propellers thrusts

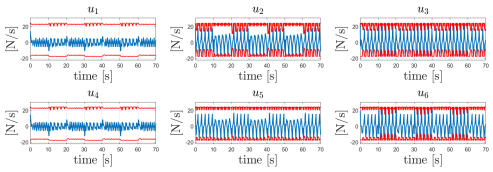


Figure 4.52: Propellers thrusts derivatives

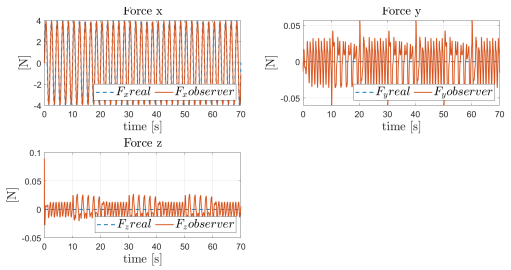


Figure 4.53: Force estimated by Observer

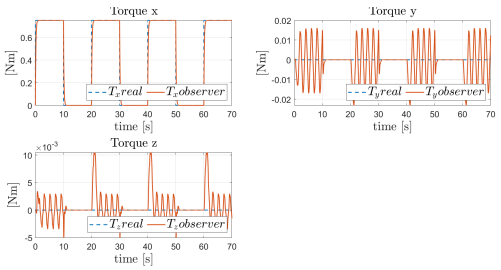


Figure 4.54: Torque estimated by Observer

In this simulation the external disturbance wrench was represented as the composition of the ones presented in the previous 2 simulations. Again, the same conclusions were again validated. The most disturbed components were position on x and roll, while the their disturbed quantities, together with the effect on pitch and y position are increased because they contain the effect from both disturbance sources. The disturbance wrench is correctly estimated, stability is maintained.

4.3.4 Sinus-shaped disturbance force and pulse-shaped disturbance torque with added noise on the state components

In this simulation, extra noise on the state variables was added in conditions of previous simulations : disturbance force of 4N and disturbance torque of 0.75Nm. The extra noise was considered as white noise with particular standard deviations,

obtained from experimental evaluation, as detailed in table (4.10). Furthermore, the noise was limited and then low-pass filtered with the period of the filter equal to 1/25 s. Except the following subsection, the extra noise was considered for all remaining simulations.
The contact force (4.63)is calculated instantly, like in previous simulation. However, the y and z components are different. They are periodic signals, with offset values between 0.1 and 0.2 N. Their appearance may be explained by the acceleration usage in the force estimation formulas, which is influenced by the added noise on the velocity.
The contact torque (4.64) is estimated without error and the y and z components are similar to the ones from previous simulation.
The position evolution (4.55) seems to not be disturbed more from the noise, as it is similar with the one from previous simulation. Consequently, the linear velocity (4.57) seems to be also similar with the one from previous simulation.
Likewise, the attitude angles and angular velocities seem to follow also the same behavior as the previous ones (4.58 and 4.59).
While examining the data with max and RMS errors, it will be seen however that the introduction of noise contribute to more disturbance in position and attitude. The propellers' thrusts saturate again, introducing dynamic couplings.
The NMPC algorithm is able to handle extra process noise, introducing an extra offset on all force dimensions, between 0.1 and 0.2 N. And also the position and orientation, together with their rates, are more disturbed than in the case without noise.

state name x	value
position	0.005 m
linear velocity	0.02 m/s
orientation angles	pi/180 rad / 1 degree
angular velocities x-y	0.15 rad/s
angular velocities z	0.05 rad/s

Table 4.10: Standard deviations for the included noise state

max error position x [m]	max error position y [m]	max error position z [m]
0.277384	0.168465	0.086883
max error roll x[rad]	max error pitch[rad]	max error yaw[rad]
0.266040	0.040942	0.059133
RMS error position x[m]	RMS error position y[m]	RMS error position z [m]
0.277384	0.168465	0.086883
RMS error roll [rad]	RMS error pitch[rad]	RMS error yaw[rad]
0.266040	0.040942	0.059133

Table 4.11: max and RMS for position and attitude errors subsection 4.3.4

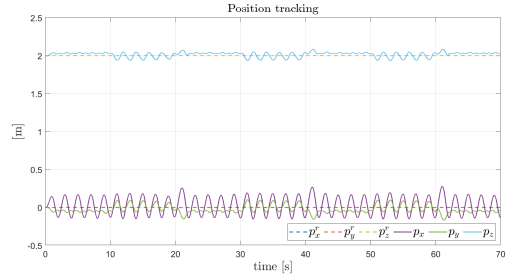


Figure 4.55: Position tracking

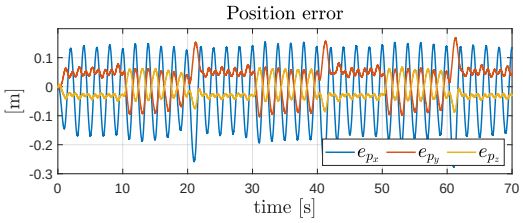


Figure 4.56: Position error

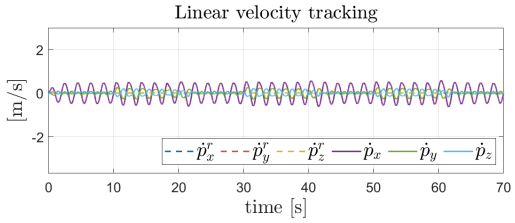


Figure 4.57: Linear velocity tracking

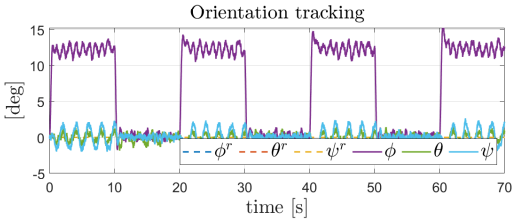


Figure 4.58: Attitude tracking

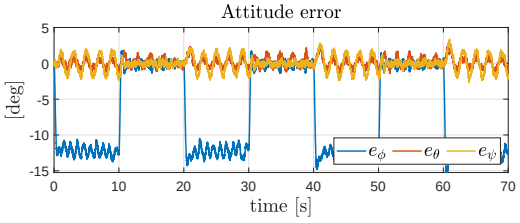


Figure 4.59: Attitude error

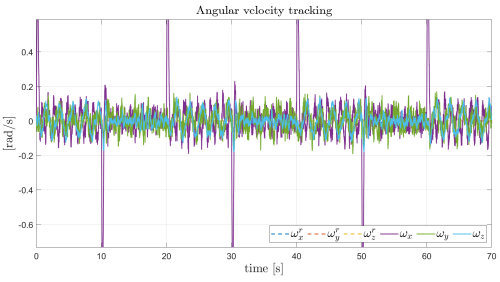


Figure 4.60: Angular speed

In this simulation it was experimentally validated the expected behavior, i.e. when there is noise present in the system, position and orientation will be more disturbed compared to the ideal case. A side effect was the presence of the offset on the external force estimation, explained by the usage of the acceleration (as derivative of the speed, containing thus its noise quantity) in its estimation. Stability is

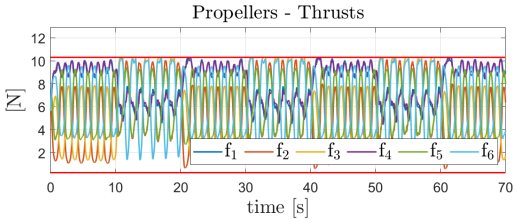


Figure 4.61: Propellers thrusts

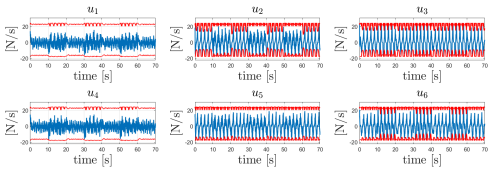


Figure 4.62: Propellers thrusts derivatives

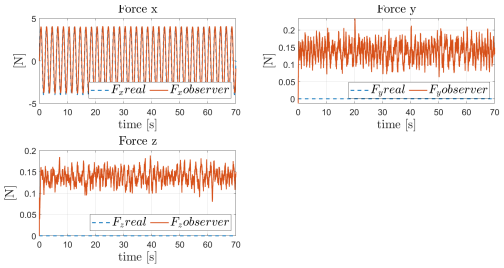


Figure 4.63: Force estimated by Observer

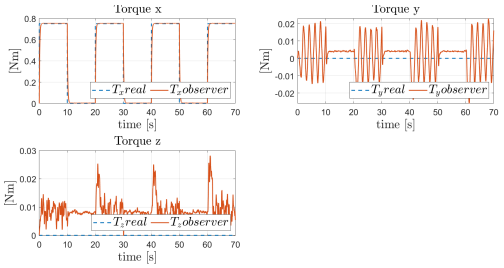


Figure 4.64: Torque estimated by Observer

maintained.

4.3.5 Robustness analysis

Simulations from this subsection have as purpose testing the robustness of the Tilt Hex platform towards model uncertainties. The parameter that was varied for the robustness check was the mass, as it has the biggest influence on the dynamics of the MRAV, out of all the parameters (mass, inertia, c_f and c_t parameters).
It was first increased and then decreased with 50% of its nominal value.
The trajectory used was a more dynamic one, with the y component being varied to 1m and then back to 0m, while the simulated disturbance wrench was set as 0.
In the results it can be observed that in both cases the position is regulated (4.65 and 4.66), in the second case the tracking being better than in the first case, seen clearer in the position error figures (4.67 vs 4.68).
The linear velocity has the biggest error on the y component during motion, as expected (4.69 and 4.70).
The orientation has an error on the roll component during motion, in the first case (4.71). It is caused by the fact that the propellers are saturated in this case (4.77), while in the second case, the orientation has only the noise component (4.72), as the propellers are not saturated (4.78). The same behavior is seen also in the

angular velocity figures (4.75 and 4.76).
The influence of the mass change is mostly seen in the observer's force tracking figures. With the mass increased, the value of the force is centered around 0.2N (4.81), while in the second case it is centered at approx. 0.09N (4.82). This behavior is proportionally to the mass, according to the component that appears in the force estimator and which contains the mass component.
The controller outputs (thrusts derivatives) appear to reach their limits in both cases, assuring however the stability of the platform in both cases (4.79 and 4.80).
Robustness analysis shows that with the uncertainty in mass between -50% and approx. +20/+30%, there will be no actuator saturation and resulting dynamic couplings seen in the orientation behavior.
However, even with the mass in the range +/- 50%, the stability is maintained and thus it provides a reasonably wide safety interval.

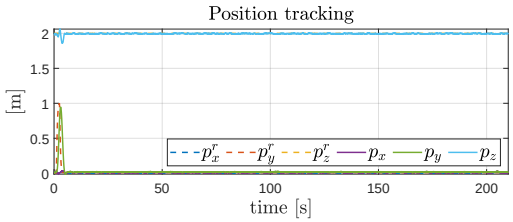


Figure 4.65: Position tracking mass + 50%

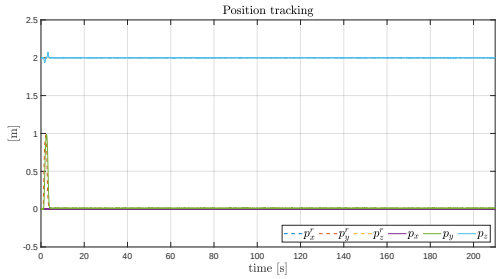


Figure 4.66: Position tracking mass - 50%

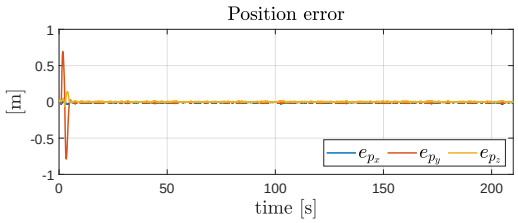


Figure 4.67: Position error mass + 50%

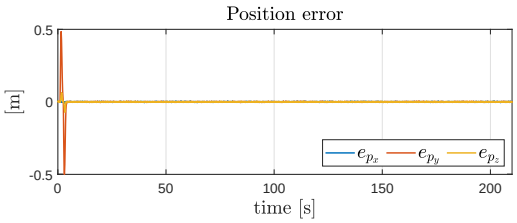


Figure 4.68: Position error mass - 50%

4.4 External wrench disturbance: tracking performance of an aware NMPC

4.4.1 External disturbance force

In the next step, the external disturbance wrench (force and/or torque) was added in the state vector used by the NMPC algorithm. As it was already stated in the

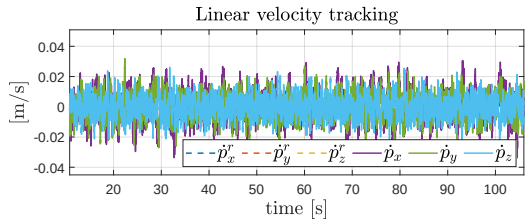


Figure 4.69: Linear velocity tracking mass + 50%

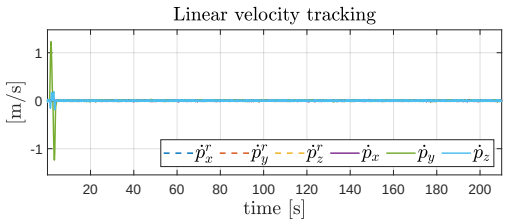


Figure 4.70: Linear velocity tracking mass - 50%

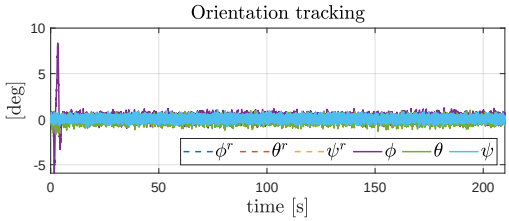


Figure 4.71: Attitude tracking mass + 50%

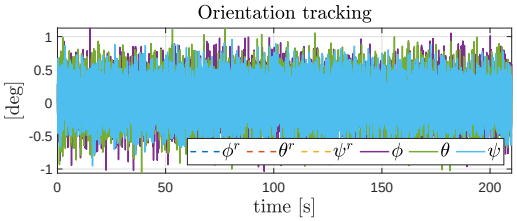


Figure 4.72: Attitude tracking mass - 50%

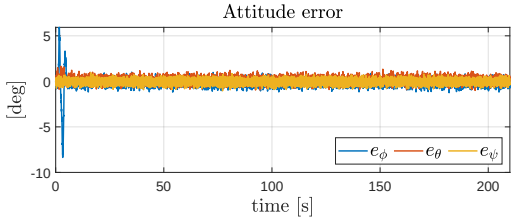


Figure 4.73: Attitude error mass + 50%

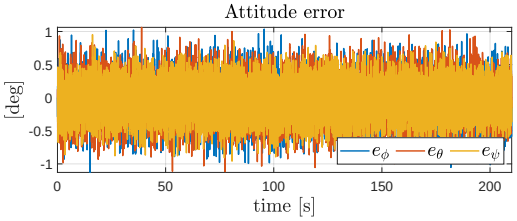


Figure 4.74: Attitude error mass - 50%

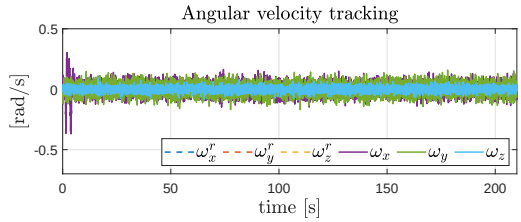


Figure 4.75: Angular speed mass + 50%

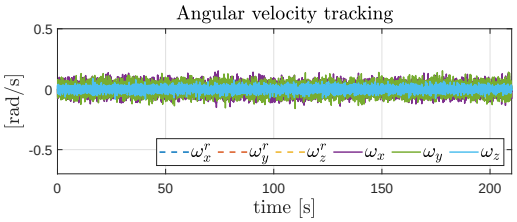


Figure 4.76: Angular speed mass - 50%

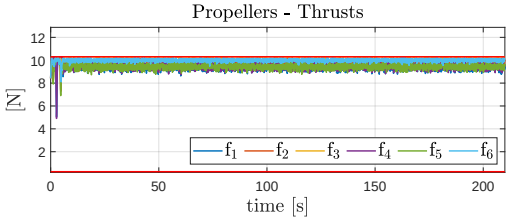


Figure 4.77: Propellers thrusts mass + 50%

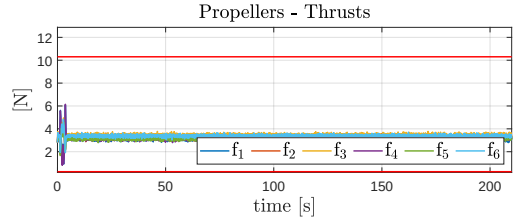


Figure 4.78: Propellers thrusts mass - 50%

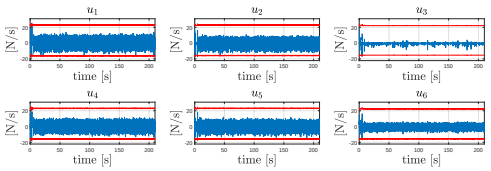


Figure 4.79: Propellers thrusts derivatives mass + 50%

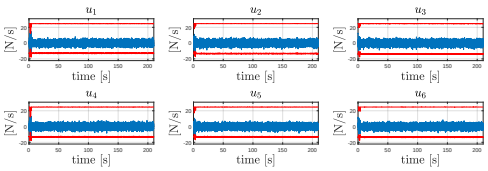


Figure 4.80: Propellers thrusts derivatives mass - 50%

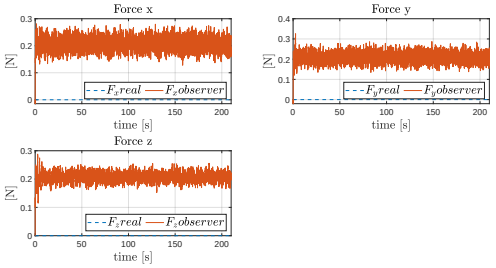


Figure 4.81: Force estimated by Observer, mass + 50 %

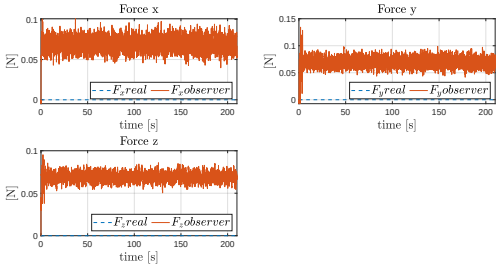


Figure 4.82: Force estimated by Observer, mass - 50 %

introduction of current chapter, but also in the previous chapter, the main goal was to verify that through this inclusion, the position and orientation tracking are improved, that they are actually regulated. In addition, other consequences were observed throughout the process. The values of the weights were the same as the ones used in the previous set of simulations. Furthermore, all the simulations considered the noise on the state vector, as already pointed.

First, only an external force of 7N on one dimension (x in this case) was simulated as a disturbance force. In order to show the effect of inclusion in the state vector of NMPC, both cases, with and without the inclusion, will be shown.

In figures 4.83 and 4.86 it can be easily observed that through the inclusion of the disturbance force in the state, after a very short transition the position is regulated. On contrary, already concluded from previous set of simulations, when it is not included, the force causes steady-state errors on all position components, the biggest being on the same dimension on which it is imposed (4.84 and 4.86). Furthermore, in tables 4.12 and 4.13 it can be observed that in both cases the biggest max/RMS deviation remains on the x dimension, validating once more the main rule drawn from previous part. Also, important to point for this simulation is the fact that the position can be regulated even though the propellers get saturated. The linear velocities stabilize in the end to the desired reference values of 0 m/s (4.87 and 4.88), containing only the noise values.

In addition, as a validation of the general rule, the biggest error on x position is coupled with the pitch, the second ones being the ones on roll and the y component of the position (due to dynamic couplings because of actuator saturation) (4.89, 4.91, 4.90 and (4.92)). In addition, it seems that the introduction of the force in the state vector, even when the propellers get saturated, will improve the angle that is disturbed the most when it is not included (pitch angle in this case), as steady-state value.
The angular velocities, after short fluctuations at the beginning, stabilize in the end to contain only the values of the simulated noise.
The force and torque are estimated correctly. The offset of 0.1-0.2 N that appears when the external force is not included in the state vector disappears when the external force is included.
One propeller gets saturated in both situations. Moreover, at least at the beginning of the motion, in each case there are commands that reach their limits, stability being maintained.

max error position x [m] 0.0996	max error position y [m] 0.0516	max error position [m] z 0.0267
max error roll x [rad] 0.0819	max error pitch [rad] 0.1347	max error yaw [rad] 0.0277
RMS error position x [m] 0.0272	RMS error position y [m] 0.0137	RMS error position z [m] 0.0089
RMS error roll [rad] 0.0267	RMS error pitch [rad] 0.0426	RMS error yaw [rad] 0.0085

Table 4.12: max, mean, RMS for position and attitude errors simulation with force included in the state vector of NMPC

max error position x [m] 0.6187	max error position y [m] 0.2806	max error position z [m] 0.1417
max error roll x [rad] 0.0599	max error pitch [rad] 0.1156	max error yaw [rad] 0.0138
RMS error position x [m] 0.4593	RMS error position y [m] 0.1956	RMS error position z [m] 0.1160
RMS error roll [rad] 0.0241	RMS error pitch [rad] 0.0523	RMS error yaw [rad] 0.0042

Table 4.13: max, mean, RMS for position and attitude errors simulation without the force included in the state vector of NMPC

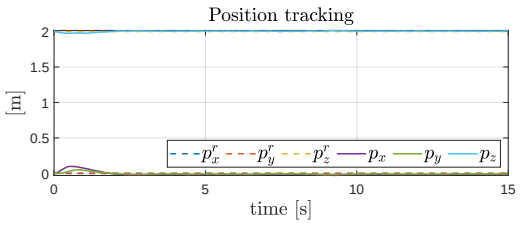


Figure 4.83: Position tracking with force included

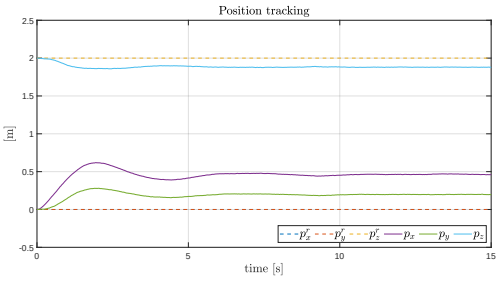


Figure 4.84: Position tracking without force included

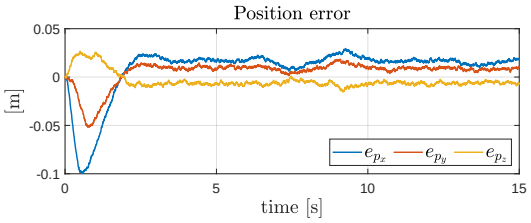


Figure 4.85: Position error with force included

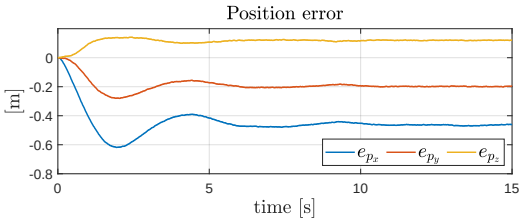


Figure 4.86: Position error without force included

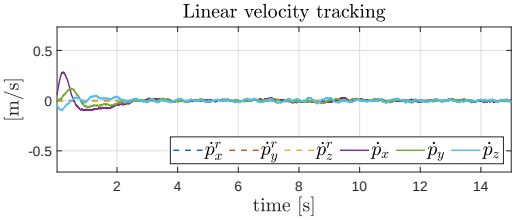


Figure 4.87: Linear velocity tracking with force included

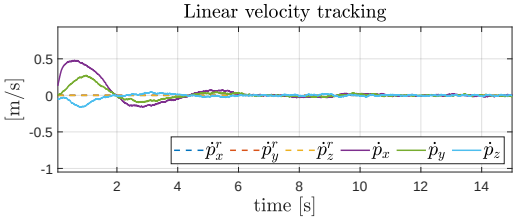


Figure 4.88: Linear velocity tracking without force included

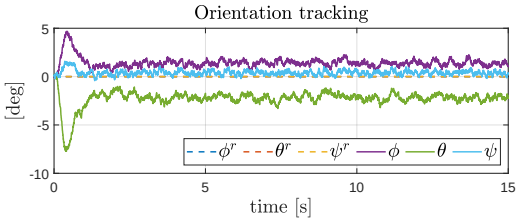


Figure 4.89: Attitude tracking with force included

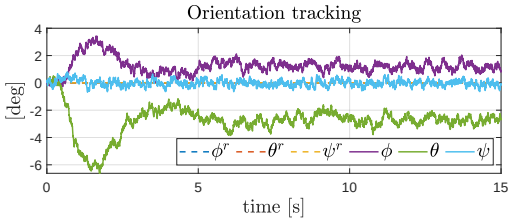


Figure 4.90: Attitude tracking without force included

These first 2 simulations from current section took into account only the existence of an external disturbance force and its inclusion in the state vector. While it was seen that position gets regulated when the external force is included in the state vector, notable to be mentioned is that this behavior seems to happen even if the propellers get saturated. Another quite important mention to be made here is that

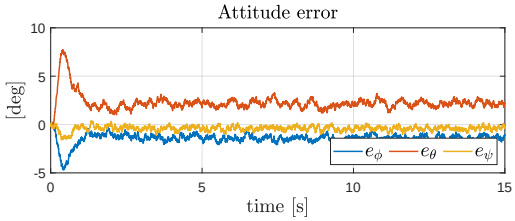


Figure 4.91: Attitude error with force included

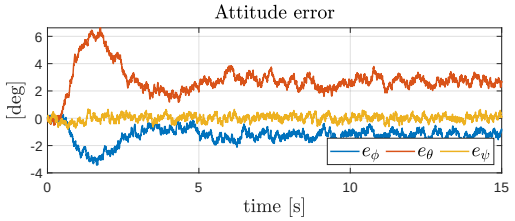


Figure 4.92: Attitude error without force included

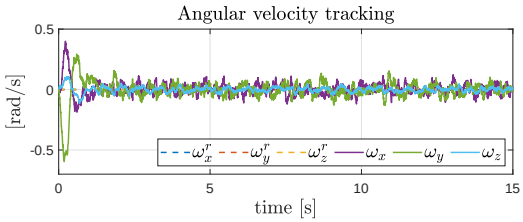


Figure 4.93: Angular speed with force included

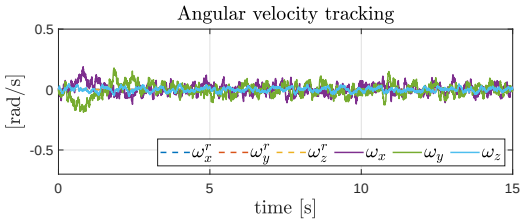


Figure 4.94: Angular speed without force included

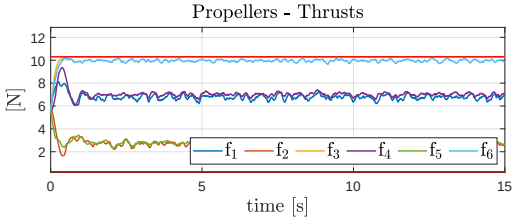


Figure 4.95: Propellers thrusts with force included

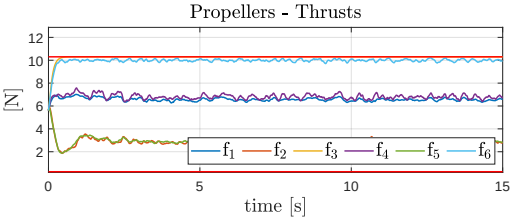


Figure 4.96: Propellers thrusts without force included

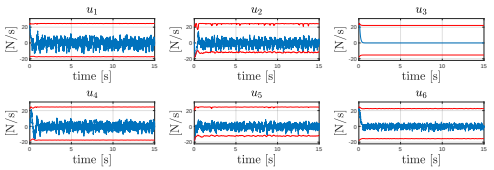


Figure 4.97: Propellers thrusts derivatives with force included

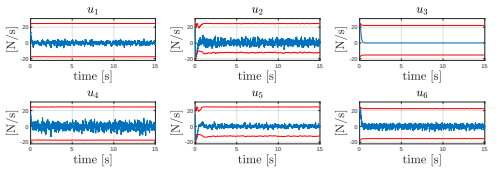


Figure 4.98: Propellers thrusts derivatives without force included

through the inclusion in the state vector, also the angle that is initially disturbed the most (the one on the opposite axis in the x-y plane) will be less disturbed, as a steady-state value. The last important thing to mention here is that the offset on the disturbance force estimation (due to the noise in the state vector) is eliminated when it is included in the state vector, as the acceleration now contains this force

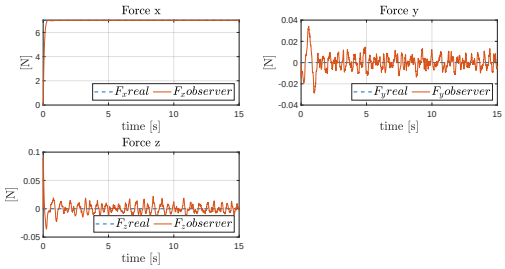


Figure 4.99: Estimated disturbance forces with force included

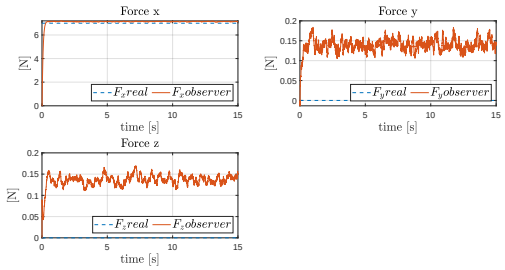


Figure 4.100: Estimated disturbance forces without force included

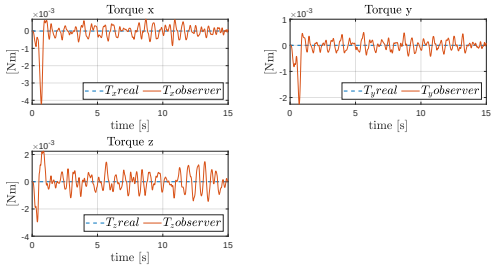


Figure 4.101: Estimated disturbance torques with force included

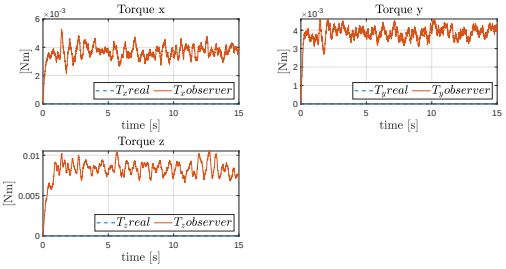


Figure 4.102: Estimated disturbance torques without force included

and is considered in the minimization criteria of the controller. The disturbance force remains correctly estimated and also stability is maintained.

4.4.2 External disturbance wrench : force and torque

Next, both disturbance force (4N) and torque (0.3 Nm) were simulated on the same Carterian coordinate (x dimension here). Again, in order to show the effectiveness of the inclusion in the state vector, all possible cases were tried (both force and torque included, only torque included, only force included, none included).
An important observation to point here is the fact that a series of simulations were tried in order to find the combination between maximum perturbation force and maximum perturbation torque that don't lead to saturation of actuators (when they are not included in the state vector). This saturation needs to be avoided in order to show that when the torque is included in the state vector, the orientation will be regulated. This is in contrast with the position tracking conclusions from previous 2 simulations, where it was seen that even when the actuators get saturated, if the external force is included in the state vector, the position will be regulated (when there is only disturbance force simulated).

As expected, the position has a steady-state error on x component in cases 4.104 and 4.106, compared to the ones in which only the external force or both force and torque are included in the state vector (4.103 and 4.105). Position on x is improved when the torque is included in the state vector (4.108 vs 4.110). Introduction of torque in the state vector improves the evolution of the position on the same dimension where it appears, while it doesn't degrade the evolution for the other dimensions. In the same time, as expected, introduction of the force in the state vector guarantees 0 steady-state error for the position.
Linear velocity remains in the limits +/- 0.02 m/s for all simulations, corresponding to the noise. This shows that the desired value of 0 m/s is fulfilled throughout the motion (4.111, 4.112, 4.113 and 4.114).
In the case of the orientation angles, it is seen that the introduction of the external force in the state vector leads in improvement of the roll angle (4.121 vs 4.122), while in the cases when only the torque or both disturbance force and torque are included the orientation is regulated, it contains only the noise, as it was expected (4.115 and 4.116).
The angular velocity remains for all simulations in the limits +/- 0.15 rad/s, the value of the noise which, like in the case of the linear velocity shows that the reference value of 0 rad/s is fulfilled (4.123, 4.124, 4.125 and 4.126).
From the propellers' thrusts (4.127, 4.128, 4.129 and 4.130), only one gets saturated in the beginning of the motion, but it doesn't cause additional dynamic couplings, which was already confirmed in the evolution of the position and orientation, i.e. for example if the force is included in the state vector, it will lead to both position regulation and roll angle improvement, the dual observations holding when only the torque is included.
In all simulations it can be observed that at the beginning of motion, a number of propellers reach their limits, stability being fulfilled.
The external force and torque estimations are correct, with the mention that again, with the inclusion of either the torque, force or both of them in the state vector, the offset of the estimated forces disappear (4.137, 4.136 and 4.135). While the cases which imply the force are somehow logical to lead to vanishing of this offset (through the minimization of the acceleration in the NMPC criteria), in the case when only the torque is included the conclusion is not very straightforward. The only reasonable explanation is the fact that the NMPC tries to minimize the angular acceleration, which now contains in the model of the MRAV the external torque and this influences the propellers' speeds, which influences the term ($\tau_v[1:3]$) in the force estimation.

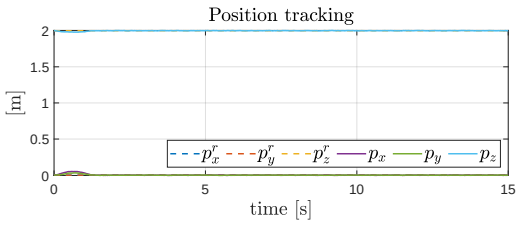


Figure 4.103: Position tracking with force and torque included

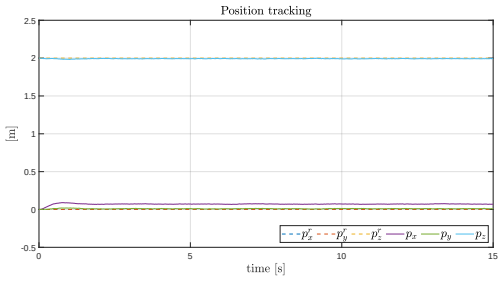


Figure 4.104: Position tracking with torque included

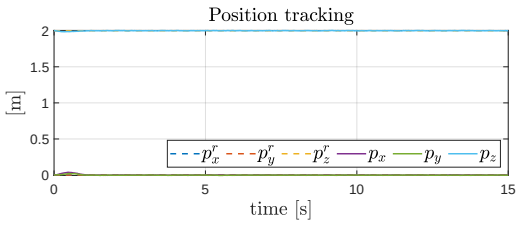


Figure 4.105: Position tracking with force included

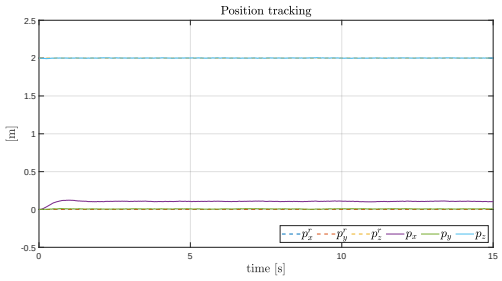


Figure 4.106: Position tracking without force and torque included

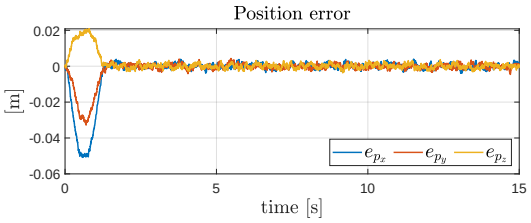


Figure 4.107: Position error with force and torque included

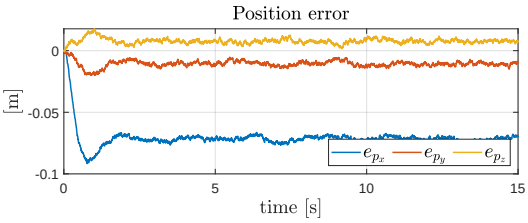


Figure 4.108: Position error with torque included

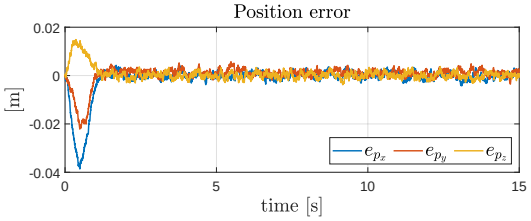


Figure 4.109: Position error with force included

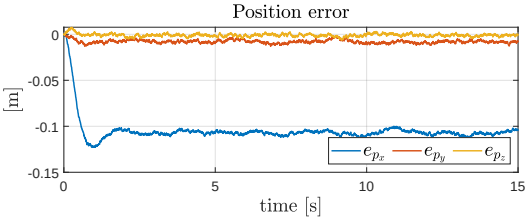


Figure 4.110: Position error without force and torque included

The last 4 simulations from current section took into account also the existence of an external disturbance torque besides the initial disturbance force, and compared all the possible inclusion cases in the state vector. Previous conclusions were

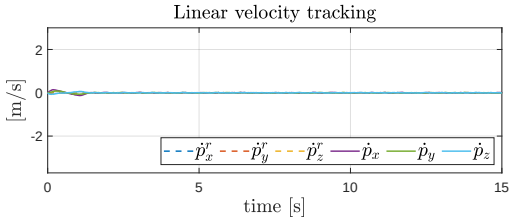


Figure 4.111: Linear velocity tracking with force and torque included

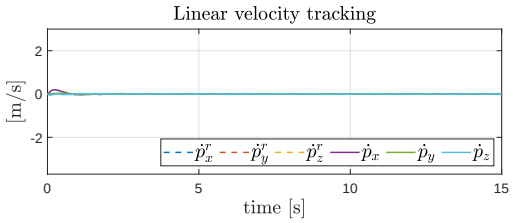


Figure 4.112: Linear velocity tracking with torque included

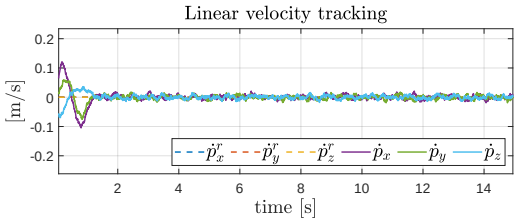


Figure 4.113: Linear velocity tracking with force included

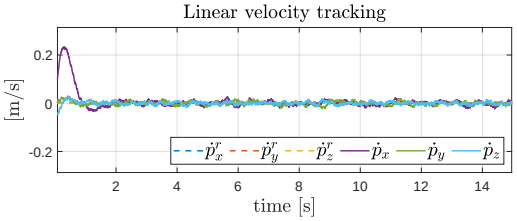


Figure 4.114: Linear velocity tracking without force and torque included

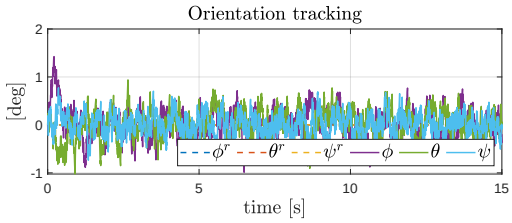


Figure 4.115: Attitude tracking with force and torque included

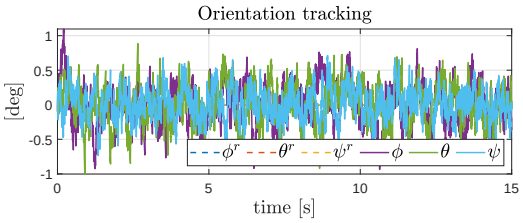


Figure 4.116: Attitude tracking with torque included

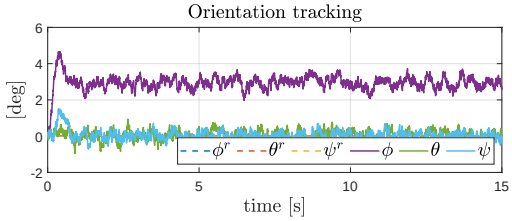


Figure 4.117: Attitude tracking with force included

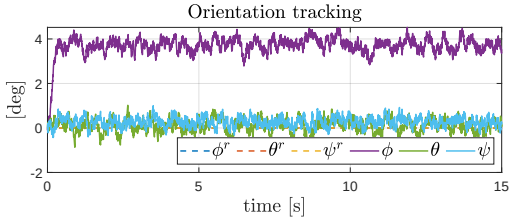


Figure 4.118: Attitude tracking without force and torque included

changed in a certain level and extended. The first important condition is that the propellers must not be saturated. With this condition fulfilled, when the disturbance torque will be included in the state vector, the attitude will be regulated and, similar to the inclusion of the external force in the state vector, it will improve the position

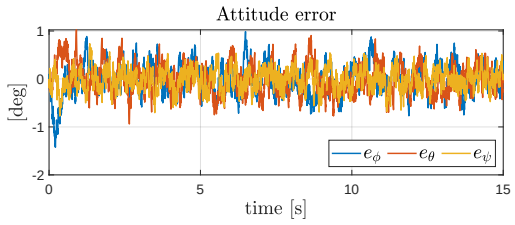


Figure 4.119: Attitude error with force and torque included

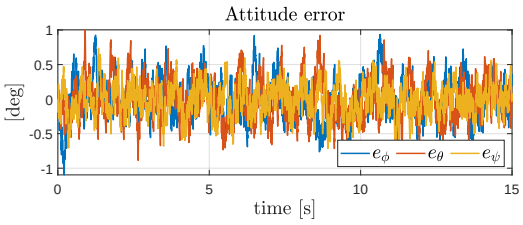


Figure 4.120: Attitude error with torque included

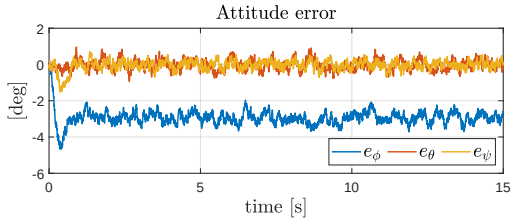


Figure 4.121: Attitude error with force included

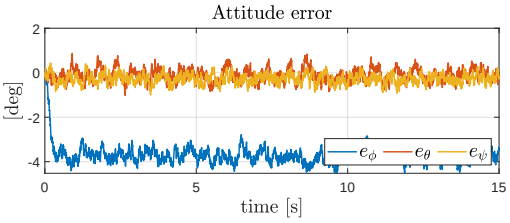


Figure 4.122: Attitude error without force and torque included

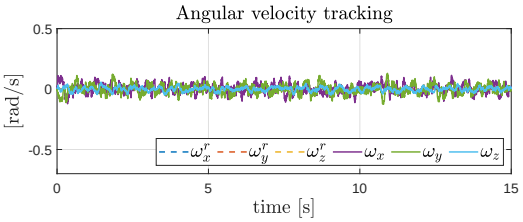


Figure 4.123: Angular speed with force and torque included

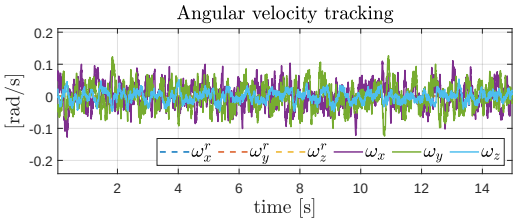


Figure 4.124: Angular speed with torque included

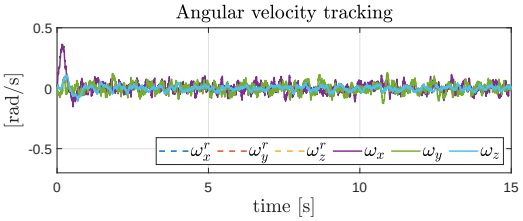


Figure 4.125: Angular speed with force included

that it disturbs the most when not included (in this case, the one on the same axis in the x-y plane). Second, the observations regarding the inclusion of the force in the state vector are similar : position regulated and the maximum deviated angle when not included is improved when the force is included (here, the roll angle). Third, the offset on the disturbance force (due to the noise in the state vector) is eliminated when either component is included in the state vector, due to the minimization of either the linear or angular accelerations. If both disturbance force

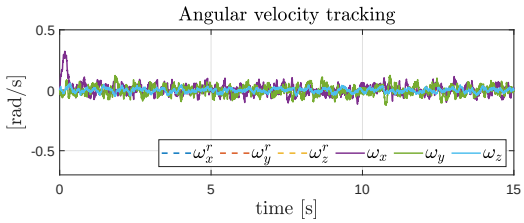


Figure 4.126: Angular speed without force and torque included

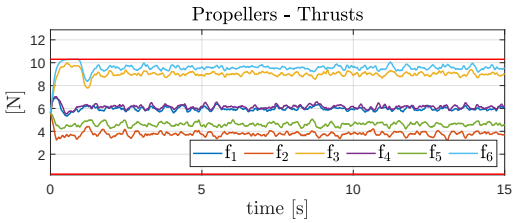


Figure 4.127: Propellers thrusts with force and torque included

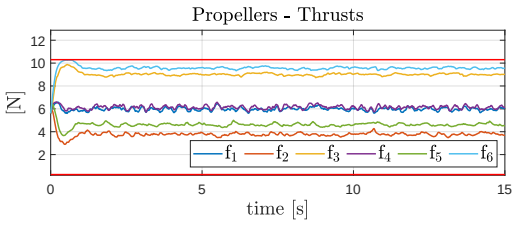


Figure 4.128: Propellers thrusts with torque included

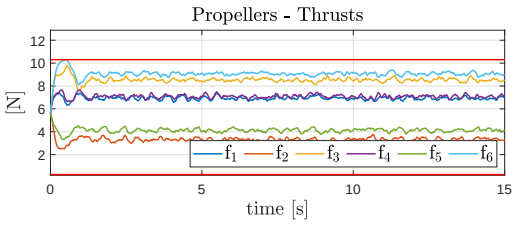


Figure 4.129: Propellers thrusts with force included

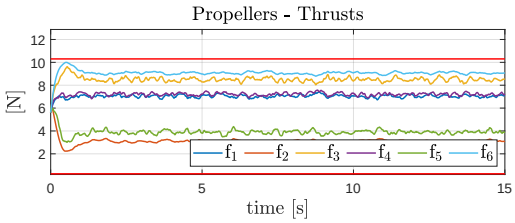


Figure 4.130: Propellers thrusts without force and torque included

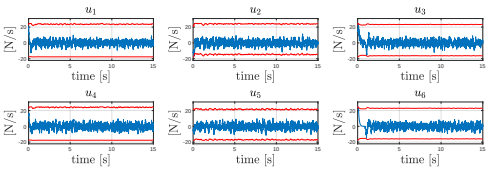


Figure 4.131: Propellers thrusts derivatives with force and torque included

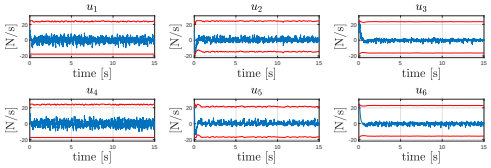


Figure 4.132: Propellers thrusts derivatives with torque included

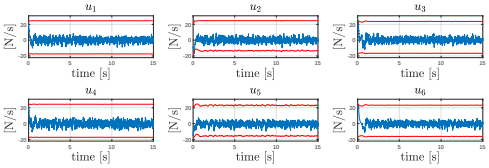


Figure 4.133: Propellers thrusts derivatives with force included

and torque are included in the state vector, both position and orientation will be regulated. The disturbance wrench remains correctly estimated and also stability

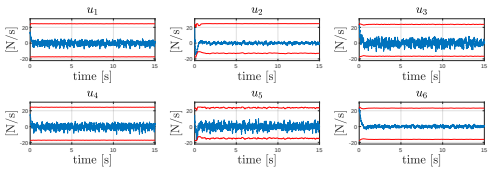


Figure 4.134: Propellers thrusts derivatives without force and torque included

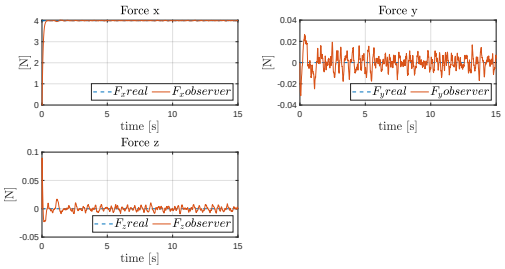


Figure 4.135: Estimated force, with force and torque included

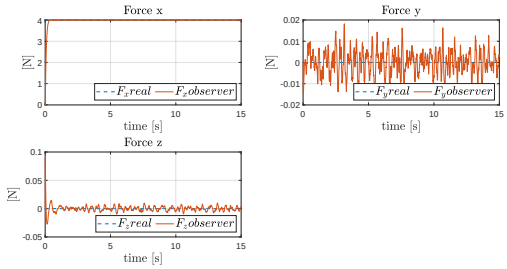


Figure 4.136: Estimated force, with torque included

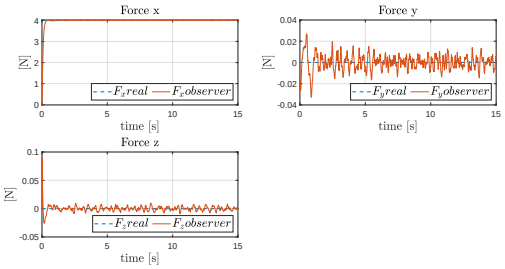


Figure 4.137: Estimated force, with force included

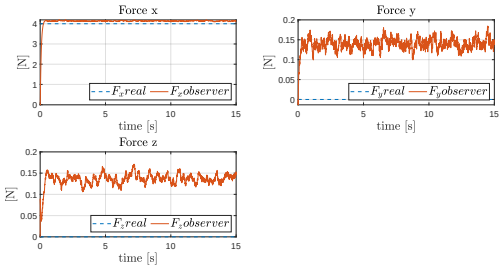


Figure 4.138: Estimated force, without force and torque included

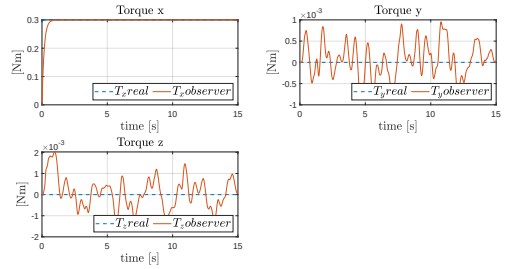


Figure 4.139: Estimated torque, with force and torque included

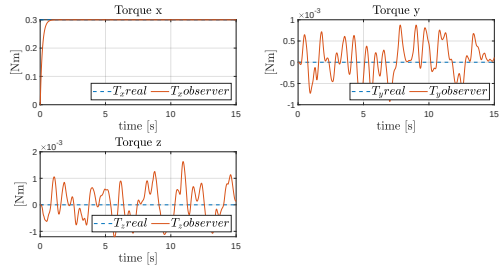


Figure 4.140: Estimated torque, with torque included

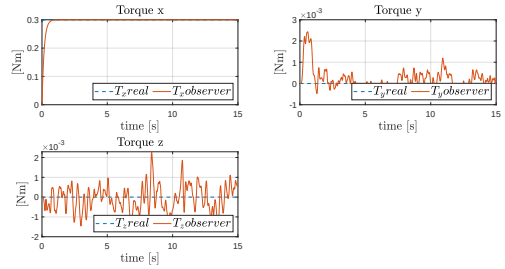


Figure 4.141: Estimated torque, with force included

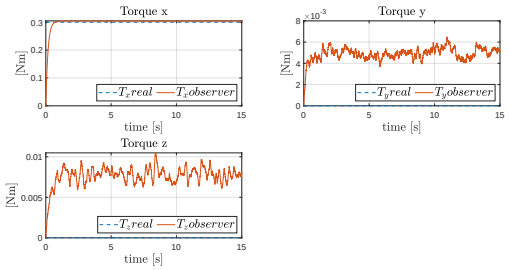


Figure 4.142: Estimate torque, without force and torque included

remains maintained.

4.4.3 Trajectory following with quasi-static disturbance wrench

The simulation shown below is the result of a series of simulations which had as purpose finding the maximum frequency of quasi-static disturbance wrench (force and torque) that comply with imposed performance criteria, which is found in table 4.14. Both disturbance force and torque were included in the NMPC's state for this tests.

Variable	max desired error
position	15-20 cm
Euler angles	7-10 degrees

Table 4.14: Desired performance for position and orientation for quasi-static shaped disturbance wrench

The final frequencies that comply with the required performance criteria were 0.7Hz for the disturbance force and 0.5Hz for the disturbance torque. Position is regulated with the maximum error of 15 cm (4.144 and 4.15) and also the orientation is regulated, with the maximum error of approx. 6 degrees (4.147).
Propellers 3 and 6 seem to be at the limit of saturation (4.149), however in the position and orientation evolutions additional dynamic couplings seem not to be present (position on x and roll angle can have additional disturbance due to these). The disturbance force and torque are estimated correctly(4.151 and 4.152), however it is seen that an extra delay is introduced in both estimations, while in the case of the torque the amplitude of the estimation is also lowered. From other simulations it was seen that increasing the frequency actually lead to even more lowering of the estimations, without improving the positions and/or orientation

trajectories. This is due to the limitation caused by the assumption that the wrench is constant in the state vector used by the controller, but also due to the impossibility of the observer to provide accurate estimations while the frequencies are big.

max error position x [m] 0.1467	max error position y [m] 0.0529	max error position z [m] 0.0350
max error roll [rad] x 0.0623	max error pitch [rad] 0.1184	max error yaw [rad] 0.0405
RMS error position x [m] 0.1467	RMS error position y [m] 0.0529	RMS error position z [m] 0.0350
RMS error roll [rad] 0.0623	RMS error pitch [rad] 0.1184	RMS error yaw [rad] 0.0405

Table 4.15: max and RMS for position and attitude errors; quasi-static disturbance wrench, included in NMPC state

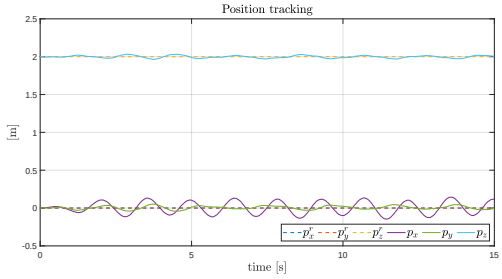


Figure 4.143: Position tracking quasi-static external/disturbance wrench

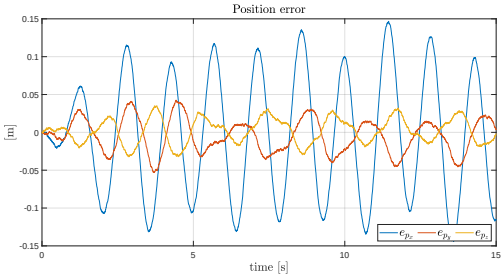


Figure 4.144: Position error quasi-static external/disturbance wrench

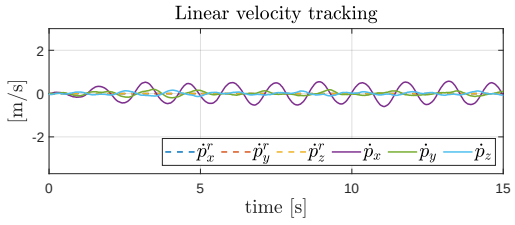


Figure 4.145: Linear velocity tracking quasi-static external/disturbance wrench

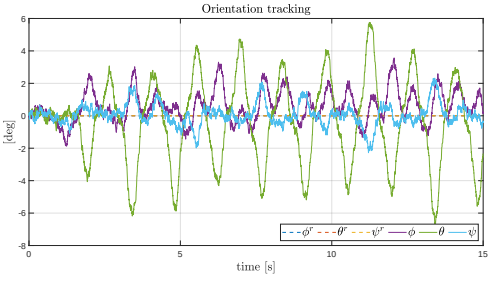


Figure 4.146: Attitude tracking quasi-static external/disturbance wrench

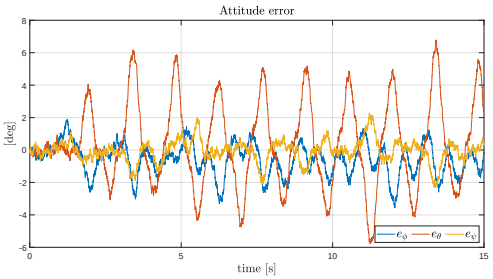


Figure 4.147: Attitude error quasi-static external/disturbance wrench

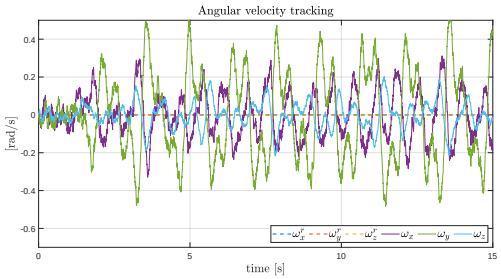


Figure 4.148: Angular speed quasi-static external/disturbance wrench

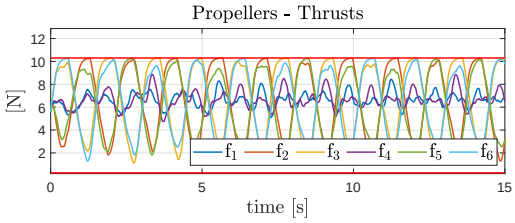


Figure 4.149: Propellers thrusts quasi-static external/disturbance wrench

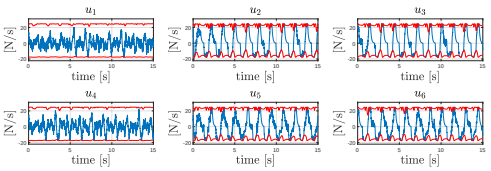


Figure 4.150: Propellers thrusts derivatives quasi-static external/disturbance wrench

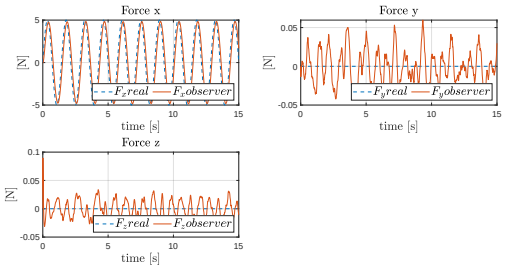


Figure 4.151: Force estimated by the Observer quasi-static external/disturbance wrench

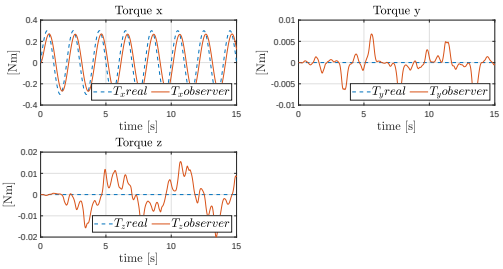


Figure 4.152: Torque estimated by the Observer quasi-static external/disturbance wrench

4.4.4 Robustness against external wrench

Another experimental analysis that has been done at this stage was related to the stability of the platform w.r.t. the disturbance wrench. Ramp signals were simulated as external force and torque until the simulation stopped, i.e. the qpOASES routines couldn't find a solution to the optimization problem. In order to find these maximum values, first the external force was increased while the external torque was kept at 0 Nm and then the external torque was increased while keeping the

external force at 0N.
As it is observed in (4.153), the maximum force that the controller can handle is approx. 40N. Likewise, from figure (4.154) it can be deduced that the maximum torque value is approx. 3.8 Nm. However, it is assumed that these values will not occur in practical situations, so the platform will not be deviated from its stable configuration because of too big disturbance force or torque.
Last, but not least, in order to find also maximum values for the steps in force and torque that lead to an unfeasible solution of the solver, experimentally it was concluded that this combination would be approx. of 8N and 2.7 Nm.
It is again underlined the fact that all these combinations cannot occur in practice because the maximum value of the force encountered in practical situations is 4N and the maximum torque is 1 Nm.

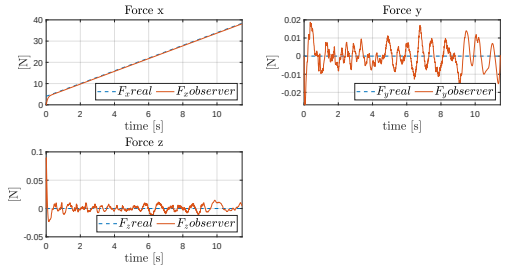


Figure 4.153: Ramp Force simulation, estimated force

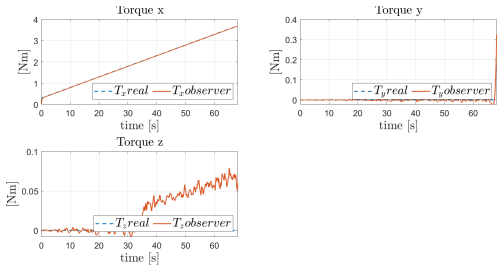


Figure 4.154: Ramp Torque simulation, estimated torque

4.4.5 Position and orientation error dependency on the magnitude of the disturbance force and torque

As an extension of the previous sub-section, there were also sought dependencies between the magnitudes of the force and torques and their influences on the position and orientation errors.
For this, several simulations were performed for different steps in disturbance force or torque and the corresponding RMS error values on position and attitude were calculated.
The effect of the disturbance force on the position error (4.155) follows the same expected behavior as already concluded from previous sub-points. Up to 20N, the error on x grows approx. in a linear fashion, followed by a growth on z as result of the x + pitch (and also roll) displacement. Position on y is also displaced, as a result of dynamic couplings. What is interesting to point here is the fact that starting with 20N, the error on y and z becomes negligible (because possibly the dynamic couplings disappear), while for the x displacement, it also starts to decay and after 30N it becomes negligible. We can assume that the interaction force looses also its stiffness property, the connection surface-MRAV is rigid so as the displacement cannot be present anymore. As for the orientation angles (4.156), they can be seen as negligible, although they also rise and fall.
The effect of the disturbance torque on the attitude (4.158)is exactly the expected one. As long as the torque is increasing, the roll error grows also, while the pitch and yaw errors are negligible. The roll error is coupled with the error on y and the one on the z component (as a result of roll and y displacement) (4.157). Here after 1.5NM, all errors on position disappear, the only disturbed measurement remaining the roll angle.

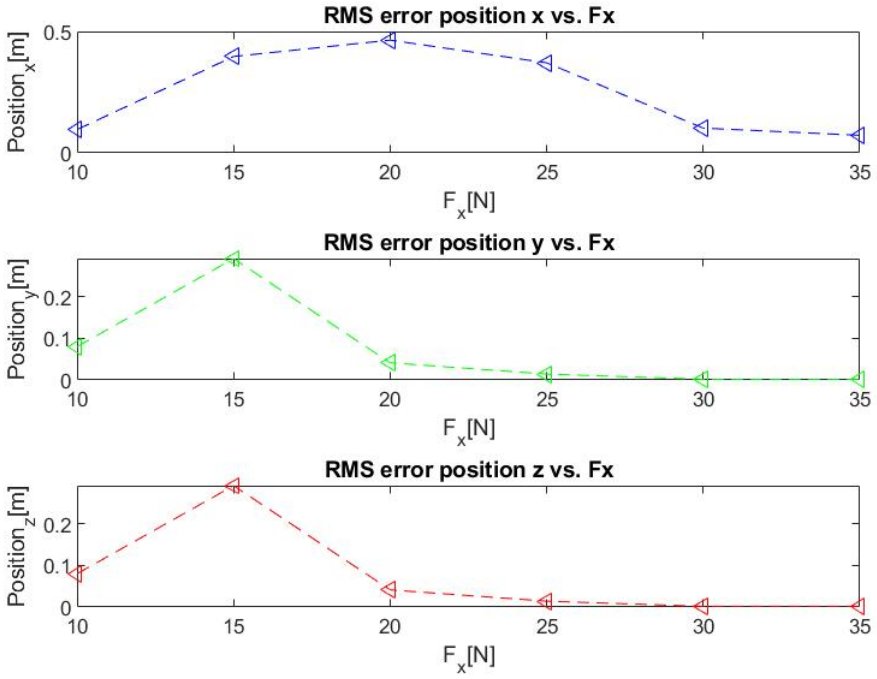


Figure 4.155: Disturbance force vs. RMS error position

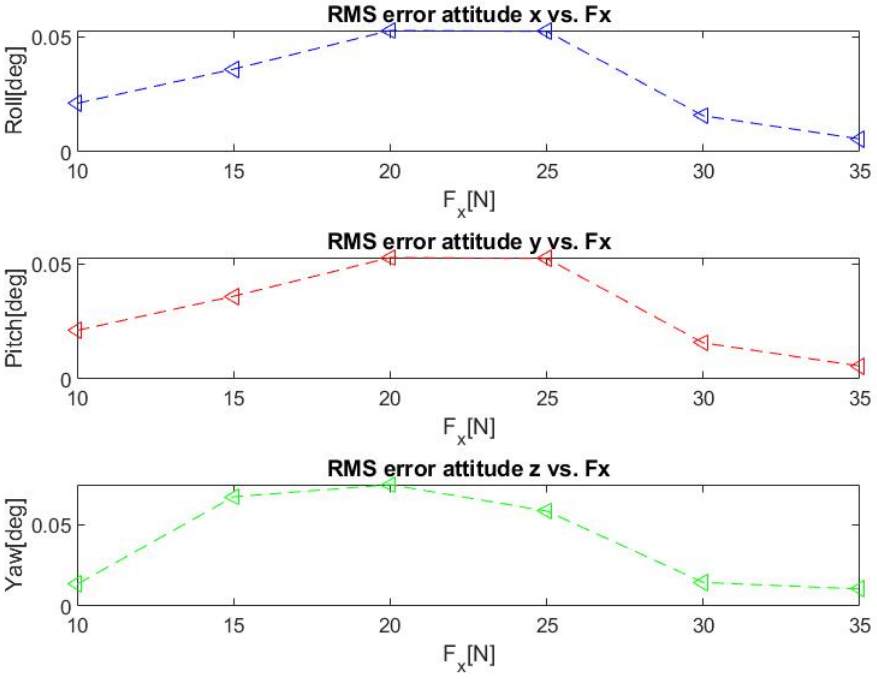


Figure 4.156: Disturbance force vs. RMS error orientation

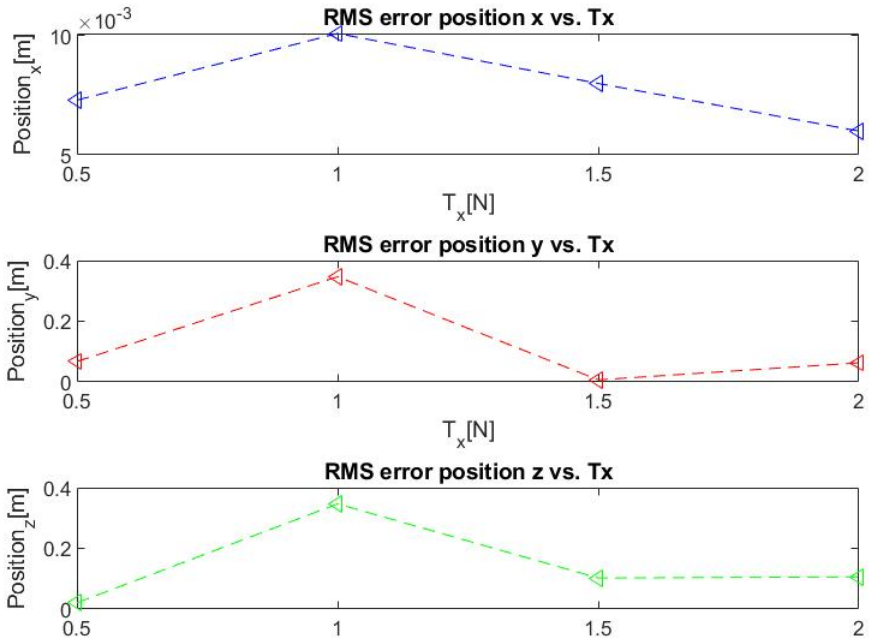


Figure 4.157: Disturbance torque vs. RMS error position

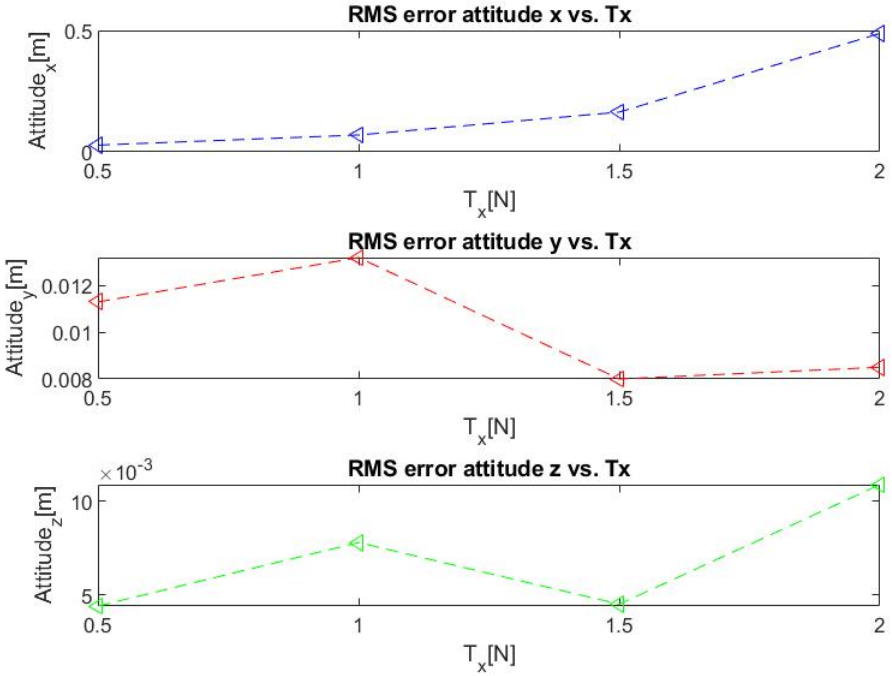


Figure 4.158: Disturbance torque vs. RMS error orientation

4.5 External wrench disturbance: tracking and regulation performance of an aware NMPC, state-dependent contact force

Finally, the final goal was to be able to control both position and contact force with the surface in the same time.
The force was intended to be controlled on one dimension (x in the current case) and the position on the other 2 dimensions (y-z in the current case), with the reasons explained in chapter 3.
Furthermore, because the external/disturbance force was in this case state-dependent, a more appropriate name for this force is a contact force.
The contact force was included in the NMPC's state vector for all simulations in this section.
As already explained in the previous chapter, the contact force was modelled as a spring force and the spring constant's value used in the end was 10, after a number of simulations with different values. Its reference magnitude is of 4N.

4.5.1 Force tracking while hovering

First, it was simulated the hovering scenario giving weights of 50 to all of the force components and keeping the default values of the other weights, as in the other 2 sub-sections.
As expected, the position is regulated, except the x dimension, on which there is a steady-state error, due to the conflict with force tracking on the same axis (4.159 and (4.160)).
The orientation is undisturbed, containing only the components of the noise (4.162 and 4.163). This behavior is caused by the fact that the dynamic couplings are actually present only at the beginning of the motion (4.165), which actually introduce errors in the orientation tracking in this part of the simulation, but which disappear afterwards.
The second objective is also fulfilled, the reference force being also regulated, almost instantly (4.167).

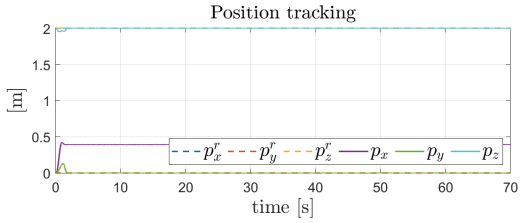


Figure 4.159: Position tracking with state-dependent force

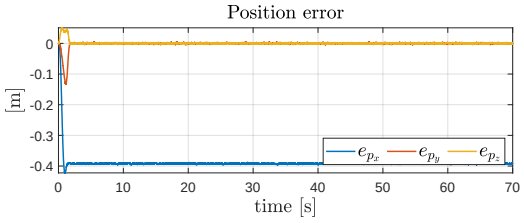


Figure 4.160: Position error with state-dependent force

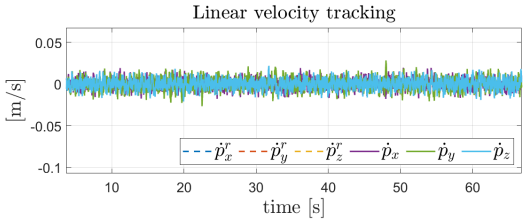


Figure 4.161: Linear velocity tracking with state-dependent force

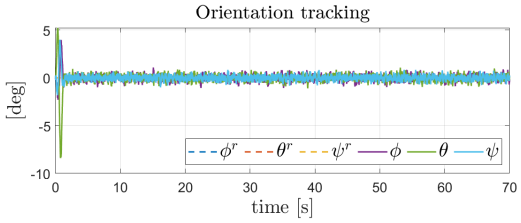


Figure 4.162: Attitude tracking with state-dependent force

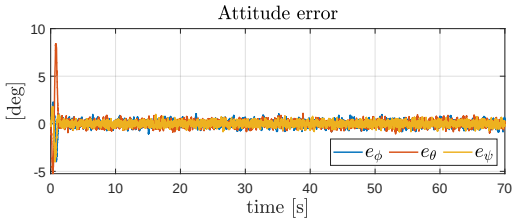


Figure 4.163: Attitude error with state-dependent force

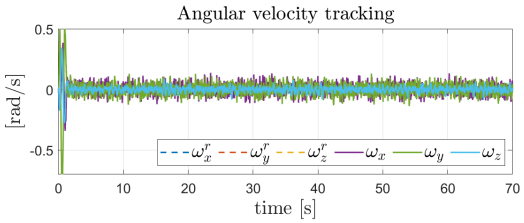


Figure 4.164: Angular speed with state-dependent force

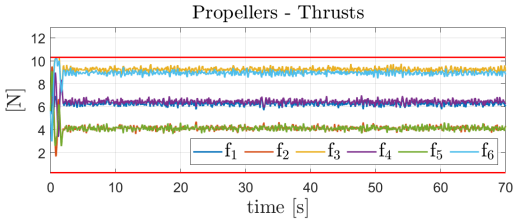


Figure 4.165: Propellers thrusts with state-dependent force

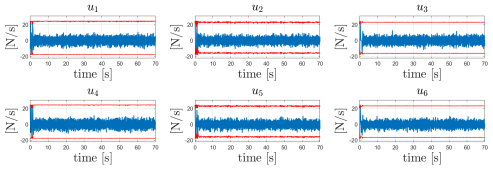


Figure 4.166: Propellers thrusts derivatives with state-dependent force

This first simulation for the rather simple case of the hovering case validates the 2 main goals of the current part of the thesis, position tracking and force regulation. In this specific case, no notable propellers thrusts saturation is introduced, this is why the orientation is not disturbed, containing only the noise components.

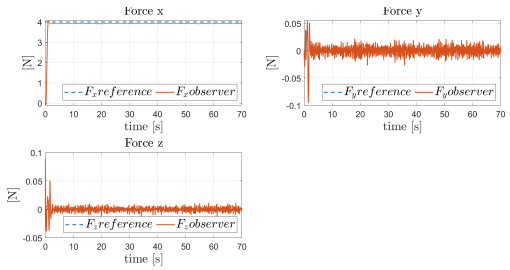


Figure 4.167: Contact force tracking with state-dependent force

Furthermore the property which is almost always also taken into consideration, i.e. stability, is again fulfilled despite actuator saturation.

4.5.2 Square trajectory tracking and Force regulation

Encouraged by the previous case, the final scenario was imposed, the square on the y-z coordinates, while keeping the same desired force on the x coordinate.
The position is regulated, with a certain tracking error on both y and z dimensions. The x dimension has the same steady-state error, as in the previous simulation (4.169 and (4.170)).
The orientation is undisturbed, containing only the components of the noise (4.172 and 4.173). Similar to previous simulation, influences of the dynamic couplings (4.175) reflect into errors in the orientation tracking in the first part of the simulation. However, different than previous case, although they are present for longer period of time, their effect on the orientation is not seen during the whole duration of this period.
Stability is again maintained despite the saturation of the force derivatives at the beginning of the motion (4.176).
The reference force is also regulated again (4.177).

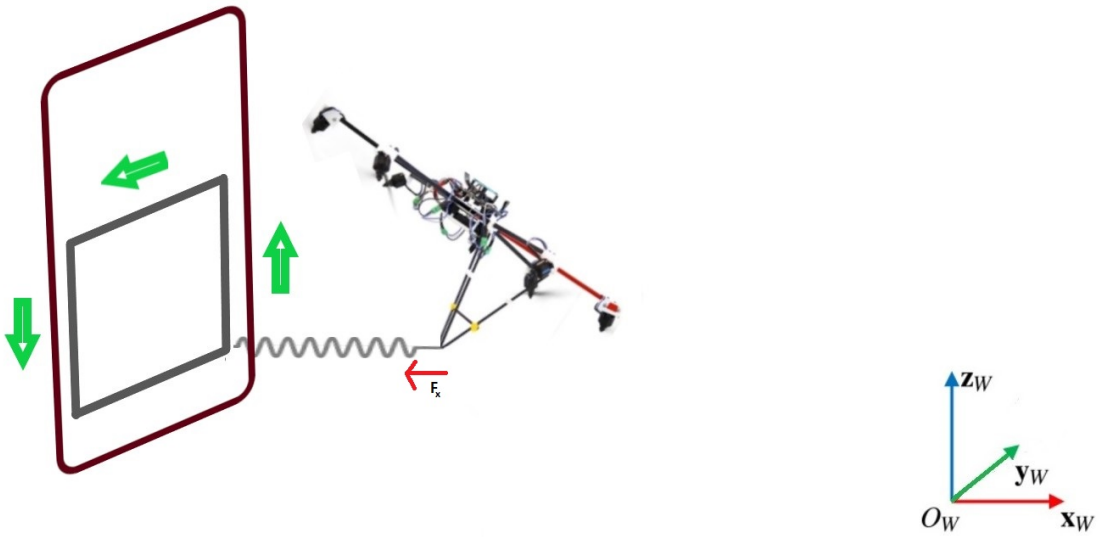


Figure 4.168: Tilt hex force and position tracking, square trajectory

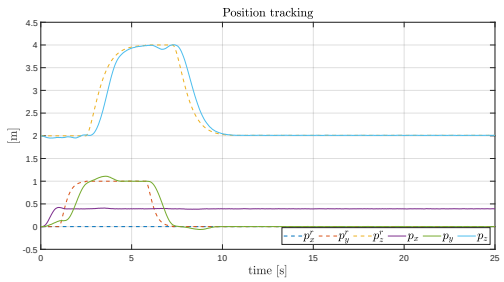


Figure 4.169: Position tracking with state-dependent force, square trajectory

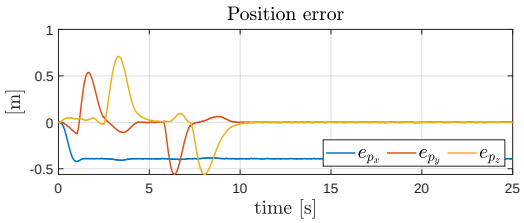


Figure 4.170: Position error with state-dependent force, square trajectory

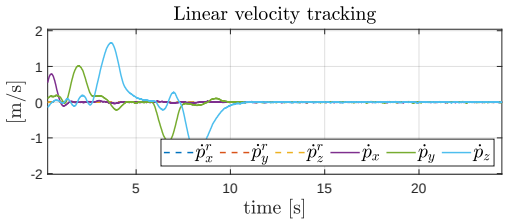


Figure 4.171: Linear velocity tracking with state-dependent force, square trajectory

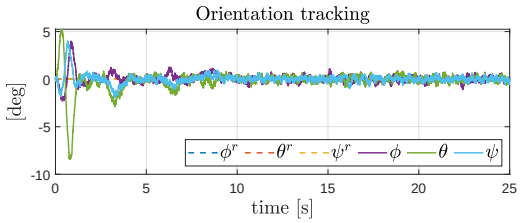


Figure 4.172: Attitude tracking with state-dependent force, square trajectory

In this simulation a position trajectory different than the hovering one was given as reference. Quite notable here is the fact that even if the propellers are saturated for a certain period of time at the beginning of the simulation, their effect is not seen in the orientation tracking for the same period of time. The 2 main goals of the current

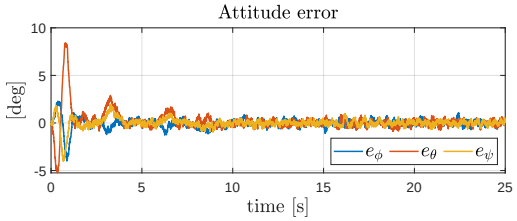


Figure 4.173: Attitude error with state-dependent force, square trajectory

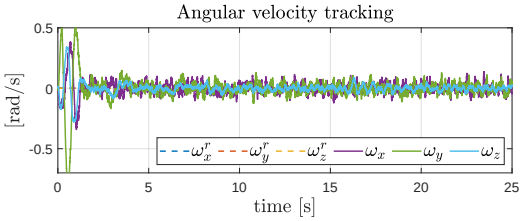


Figure 4.174: Angular speed with state-dependent force, square trajectory

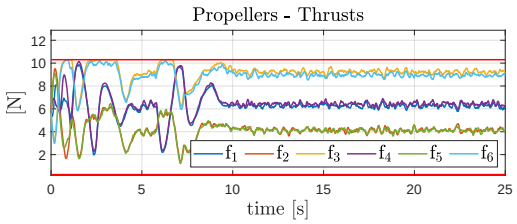


Figure 4.175: Propellers thrusts with state-dependent force, square trajectory

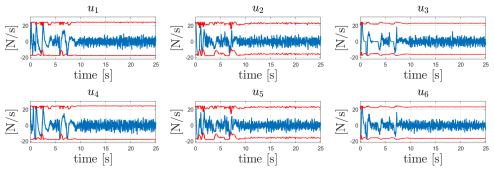


Figure 4.176: Propellers thrusts derivatives with state-dependent force, square trajectory

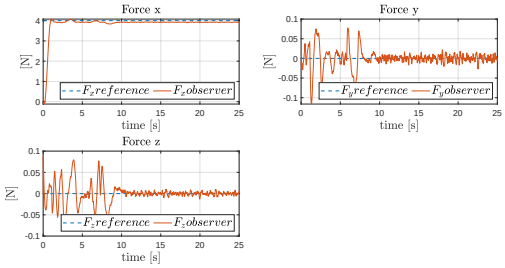


Figure 4.177: Contact force tracking with state-dependent force, square trajectory

part of the thesis, position tracking and force regulation, are shown to be fulfilled. Stability is also again fulfilled despite actuator saturation.

4.5.3 Effect on force error due to ratio between weight force and weight position

The last analysis was to determine the relationship that exists between the ratio weight force vs. weight position and the determined error on the force tracking. For this purpose, the weight on the position was maintained to 100 for all simulations and the weights on force were varied in the interval [50, 70, 90, 110, 130 and 150]. The results are grouped in 2 figures, 4.178 and 4.179.
The behaviors are the same for all figures, with the difference in maximum error on x dimension. It is seen that starting with the value of 70 for the force weight, the maximum error of the force on x (dimension on which the force is pushing) is approx. 5N and it remains the same until its weight is 3 times the weight of the position. In the RMS error dependency the dependency is almost the same, with a constant behavior followed however by a slight increase for the last ratio.
As for the other dimensions, these have a maximum error in force of 0.11-0.15N and a maximum RMS error of 0.1-0.2N, the biggest ones being seen however for the last ratio.
To sum up, the maximum error on the x dimension is 5N with the exception when the force weight is 3 times the weight on position, where it seems to be 0. On the other hand, the RMS error on the same dimension is below 0.5N for all weight ratios.

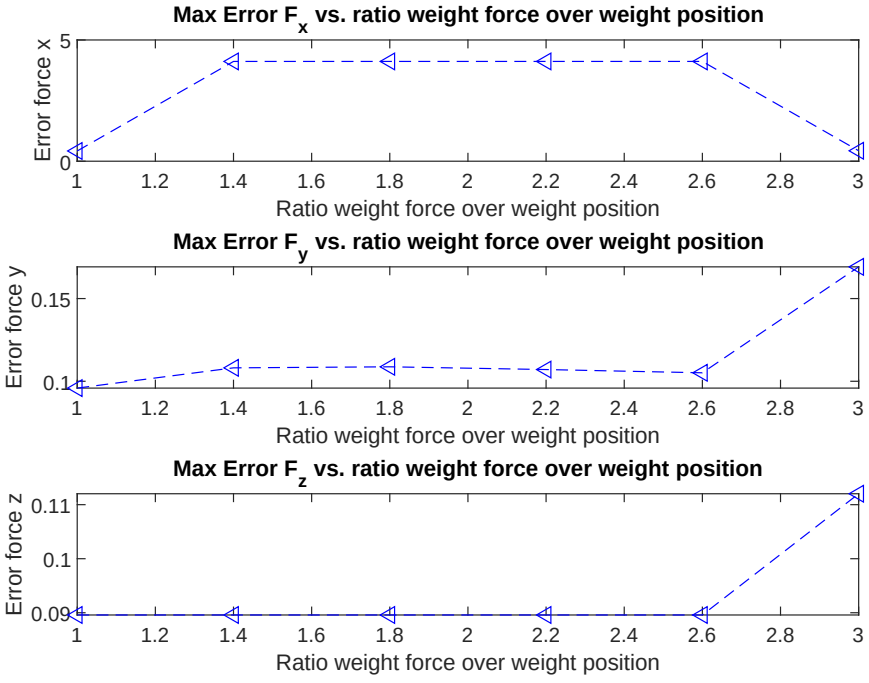


Figure 4.178: Max error dependency

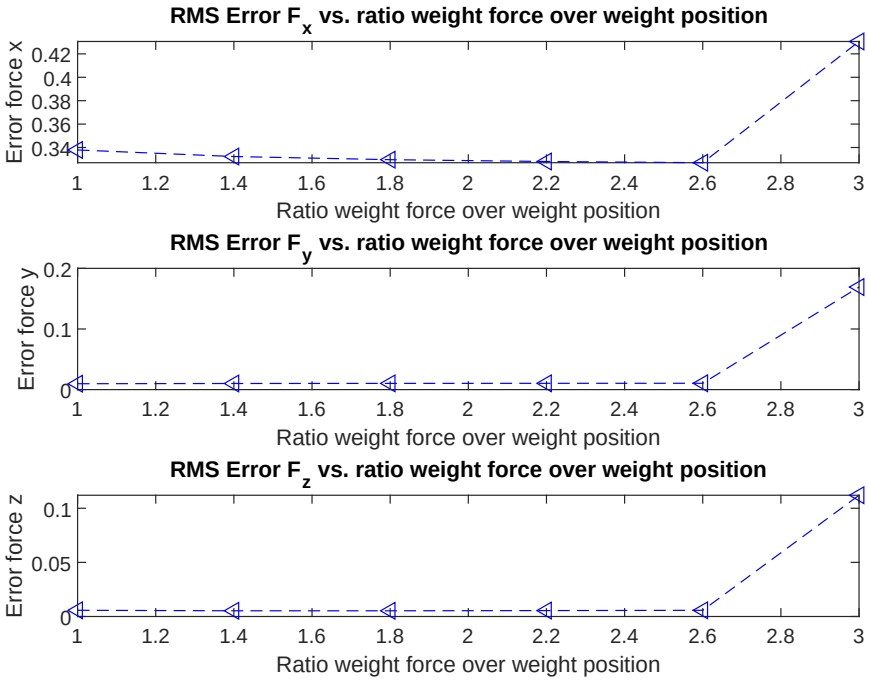


Figure 4.179: RMS error dependency

Chapter 5

Conclusions and future work

5.1 Conclusions

5.1.1 Trajectory following simulations

The trajectory following numerical simulations have shown and validated the influence that the weight matrix has in the performance of the output variables. The references were set on the position and/or orientation angles. Even though the references should be also set on the velocities and accelerations, the purpose was to verify the capabilty of the NMPC controller to generate also a feasible trajectory with only these references.
Position tracking and the orientation angles are directly influenced by how big their corresponding weights are set in the controller. This has been seen in the numerical simulations, as big weights on the position made its tracking error get smaller. The orientation angles also proved to have less tracking error as their corresponding weights got bigger.
The same results have been observed for the linear speed and angular velocity. As their weights were almost the same in all numerical simulations, their evolutions were similar for almost all numerical simulations as well. In the case of the angular speed, it was influenced in some simulations by the whole chosen weights, so it was not fully independent from the other variables used in the NMPC criteria.
The thrusts were saturated due to the trajectory being infeasible, the dynamic effects being seen in the orientation tracking, which was not fully followed, visible especially in the sinus-reference case.Furthermore, the stability was maintained despite the propellers being saturated, a good characteristic of the NMPC control technique.
The gains proposed in these simulations were just to test/validate the capability of the controller to handle different situations. They were used as trial and error, other techniques being needed for the identification of the optimal ones (like neural

networks).
The final set of gains used in the following parts was the best one from this set, with 100 for position and 500 for orientation.

5.1.2 External wrench disturbance. Validation of the observer and tracking performance of an agnostic NMPC

The main objective was to test the capability of the observer to estimate correctly the simulated forces and torques. Throughout all simulations it was seen that they are estimated correctly, also in the case when the contact force and torque were both simulated in the same time.
It was seen that a contact force on one dimension from the x-y plane causes a position error (the biggest one being on the corresponding axis) and possibly an angle with a significant error on the opposite axis.
A contact torque on one dimension from the x-y plane causes the biggest error on the angle from the same axis and possibly a significant one on the position's opposite axis.
If dynamic couplings appear due to saturation of the propellers, bigger disturbances will be seen on the opposite axis of the position and angle of the same dimension in case of the disturbance force (y and roll in our case) and similarly for the disturbance torque (x and pitch in our case).
Again, the actuators reach their upper limits again, stability being maintained.
Noise in the states introduce an offset in the force estimations on all dimensions.

5.1.3 External wrench disturbance: tracking performance of an aware NMPC

In this part, the external wrench has been included as part of the NMPC state in order to see the improvement on the position and orientation tracking.
The first conclusion is that, at least for external wrench consisting only of constant force, even though the propellers get saturated, when the force is included in the state vector, the position will be regulated. This behavior does not happen for an external torque disturbance. In addition, also the inclusion of the force in the state doesn't seem to disturb more the Euler angles, explained by the fact that the platform is fully-actuated.
The second conclusion applies to the case when there are present both external force and torque disturbances in the system. The inclusion of force in the NMPC state will improve the evolution of the Euler angle corresponding to the same dimension and the same conclusion is drawn at the inclusion of the disturbance

torque in the state. It will improve the evolution of the position of the same dimension on which is applied/simulated.
The 3rd conclusion is that even when a quasi-static disturbance wrench is simulated and if this is considered as constant in the state of the NMPC, the controller can handle this situation. However, it was seen through many combinations of the frequencies that growing them too much will affect the observers' precision. So, a trade-off should be considered between the desired performance in terms of position and orientation and maximum achievable frequencies that can assure this performance.

5.1.4 External wrench disturbance: tracking and regulation performance of an aware NMPC, state-dependent contact force

Finally, the external/disturbance force was also included in the objective function in order to be able to track a reference, state-dependent force, which now could be safely called a contact force. It was shown that position and force could not be tracked simultaneously on the same dimension, so the desired final behavior was to track a desired trajectory on one plane, while pushing with a certain force on the contact surface for the 3rd component of the Euclidean space.
It was first imposed a reference force+position for the hovering scenario and then it was extended to the square trajectory. The objectives were fulfilled, the reference force being regulated and the desired trajectory tracked.

5.2 Future work

First, as already mentioned, each main part of the thesis can be used independently in terms of the chosen application. If it is not desired to track accurately the 6D trajectory, then the form of the controller as in the second part can be chosen, with the deduced implications for the pose. Then, if also the regulation of the trajectory is desired, the form of the controller as in the 3rd part should be chosen. Finally, if the regulation of the external/contact force is desired, the final form of the controller is to be chosen.
Then, in order to provide more realistic results, more general external force should be tried out (especially with the final version of the controller) and possibly more accurate models for the interaction with the environment. In addition, these simulations should also consider more accurate models of the MRAV and the constraints imposed by the interaction with the environment (a Gazebo simulator could be used as Hardware-in-the-loop component to provide more realistic conditions for this pur-

pose). Furthermore, in order to cope with the non-linearity of the system and of the contact force, gain scheduling for the spring constant and weight scheduling for the controller should be considered.
Finally, the extension to human aerial-physical interaction would be the following logic step. For this, usage of vision should be considered in order to provide another layer for the control algorithm, necessary to understand the human actions and respond accordingly. Consequently, the usage of intelligent control would be necessary in order to determine the right action of the flying robot to a wide range of human demands.

References

[1] Y. Chen, M. Bruschetta, E. Picotti, and A. Beghi, "Matmpc - a matlab based toolbox for real-time nonlinear model predictive control," 18^{th} *European Control Conference,Napoli,Italy*, 2019.

[2] A. Franchi and A. Mallet, "Adaptive closed-loop speed control of bldc motors with applications to multi-rotor aerial vehicles." *IEEE Int. Conf. on Robotics and Automation*, 2017.

[3] T. Tomic and S. Haddadin, "A unified framework for external wrench estimation, interaction control and collision reflexes for flying robots," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 4197–4204, 2014.

[4] G. Antonelli, E. Cataldi, G. Muscio, M. Trujillo, Y. Rodriguez, F. Pierri, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero, "Impedance control of an aerial-manipulator: Preliminary results," *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. Daejeon, South Korea*, p. 3848–3853, 2016.

[5] G. Gioioso, M. Ryll, D. Prattichizzo, H. H. Bülthoff, and A. Franchi, "Turning a near-hovering controlled quadrotor into a 3d force effector," *2014 IEEE Int. Conf. on Robotics and Automation, Hong-Kong*, p. 6278–6284, 2014.

[6] B. Yüksel, C. Secchi, H. H. Bülthoff, and A. Franchi, "A nonlinear force observer for quadrotors and application to physical interactive tasks," *2014 IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics, Besançon, France.*, p. 433–440, July 2014.

[7] T. Tomic, C. Ott, and S. Haddadin, "External wrench estimation collision detection, and reflex reaction for flying robots," *IEEE Transactions on Robotics*, p. 99, 2017.

[8] S. Rajappa, H. Bülthoff, and P. Stegagno, "Design and implementation of a novel architecture for physical human-uav interaction," *The International Journal of Robotics Research*, 2017.

[9] F. Augugliaro and R. D'Andrea, "Admittance control for physical human-quadrocopter interaction," *12th European Control Conference, Zurich, Switzerland*, 2013.

[10] C. D. McKinnon and A. P. Schoellig, "Unscented external force and torque estimation for quadrotors," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE.*, pp. 5651–5657, 2016.

[11] H. Nguyen and D. Lee., "Hybrid force/motion control and internal dynamics of quadrotors for tool operation," *2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

[12] G. Gioioso, A. Franchi, G. Salvietti, S. Scheggi, and D. Prattichizzo, "Hybrid force/motion control and internal dynamics of quadrotors for tool operation," *2014 IEEE Int. Conf. on Robotics and Automation, Hong Kong, China*, pp. 4335–4341, May 2014.

[13] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M. Hehn, M. W. Mueller, J. S. Willmann, F. Gramazio, M. Kohler, and R. D'Andrea, "The flight assembled architecture installation: Cooperative construction with flying machines," *IEEE Control Systems Magazine 34.4*, p. 46–64, 2014.

[14] B. Yüksel, C. Secchi, H. H. Bülthoff, and A. Franchi, "Reshaping the physical properties of a quadrotor through ida-pbc and its application to aerial physical interaction," *2014 IEEE Int. Conf. on Robotics and Automation. Hong Kong, China*, p. 6258–6265, May 2014.

[15] N. Staub, M. Mohammadi, D. Bicego, D. Prattichizzo, and A. Franchi, "Towards robotic magmas: Multiple aerial-ground manipulator systems," *2017 IEEE Int. Conf. on Robotics and Automation. Singapore*, May 2017.

[16] K. Sreenath and V. Kumar, "Dynamics, control and planning for cooperative manipulation of payloads suspended by cables from multiple quadrotor robots," *Robotics: Science and Systems. Berlin, Germany*, June 2013.

[17] A. Tagliabue, M. Kamel, S. Verling, R. Siegwart, and J. Nieto, "Collaborative transportation using mavs via passive force control," *IEEE Int. Conf. on Robotics and Automation. Singapore*, p. 5766–5773, 2016.

[18] M. Fumagalli, R. Naldi, A. Macchelli, R. Carloni, S. Stramigioli, and L. Marconi, "Modeling and control of a flying robot for contact inspection," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 3532–3537, 2012.

[19] S. Kim, S. Choi, and H. J. Kim, “Aerial manipulation using a quadrotor with a two dof robotic arm,” *IEEE/RSJ International Conference on Intelligent Robots and Systems. Nov. 2013*, p. 4990–4995, 2013.

[20] K. Kondak, F. Hubert, M. Schwarzbach, M. Laiacker, D. Sommer, M. Bejar, and A. Ollero, “Aerial manipulation robot composed of an autonomous helicopter and a 7 degrees of freedom industrial manipulator,” *2014 IEEE Int. Conf. on Robotics and Automation. Hong Kong, China,*, p. 2108–2112, 2014.

[21] A. Suarez, G. Heredia, and A. Ollero, “Lightweight compliant arm for aerial manipulation,” *2015 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. Hamburg, Germany*, p. 1627–1632, Sept. 2015.

[22] K. Baizid, G. Giglio, F. Pierri, M. Trujillo, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero, “Behavioral control of unmanned aerial vehicle manipulator systems,” *Autonomous Robots 35.8*, pp. 1–18, 2016.

[23] G. Muscio, M. A. Pierri, F.and Trujillo, E. Cataldi, G. Giglio, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero, “Experiments on coordinated motion of aerial robotic manipulators,” *IEEE Int. Conf. on Robotics and Automation. Stockholm, Sweden*, p. 1224–1229, May 2016.

[24] G. Muscio, F. Pierri, M. A. Trujillo, E. Cataldi, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero, “Coordinated control of aerial robotic manipulators: Theory and experiments,” *IEEE Transactions on Control Systems Technology PP.99 (2017)*, pp. 1–8, 2017.

[25] M. Tognon, B. Yüksel, G. Buondonno, and A. Franchi, “Dynamic decentralized control for protocentric aerial manipulators,” *2017 IEEE Int. Conf. on Robotics and Automation. Singapore,*, p. 6375–6380, May 2017.

[26] D. Bicego, G. 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*, 2020.

[27] 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 38.9 (2019)*, p. 1045–1062, 2019.

[28] H. Romero, S. Salazar, A. Sanchez, and R. Lozano, “A new uav configuration having eight rotors: dynamical model and real-time control,” *46th IEEE Conf. on Decision and Control. New Orleans, LA*, p. 6418–6423, Dec. 2007.

[29] S. Salazar, H. Romero, R. Lozano, and P. Castillo, “Modeling and real-time stabilization of an aircraft having eight rotors,” *Journal of Intelligent and Robotic Systems 54.1*, p. 6418–6423, Dec. 2007.

[30] B. Crowther, A. Lanzon, M. Maya-Gonzalez, and D. Langkamp, “Kinematic analysis and control design for a nonplanar multirotor vehicle,” *AIAA Journal of Guidance, Control, and Dynamics 34.4 (2011)*, p. 1157–1171, 2011.

[31] D. Langkamp, G. Roberts, A. Scillitoe, I. Lunnon, A. LlopisPascual, J. Zamecnik, S. Proctor, M. Rodriguez-Frias, M. Turner, A. Lanzon, and W. Crowther, “An engineering development of a novel hexarotor vehicle for 3d applications,” *Proceedings of the International Micro Air Vehicle conference and competitions 2011 (2011)*, 2011.

[32] M. Tognon and A. Franchi, “Omnidirectional aerial vehicles with unidirectional thrusters: Theory, optimal design, and control,” *IEEE Robotics and Automation Letters 3.3*, p. 2277–2282, 2018.

[33] D. Bicego, “Design and control of multi-directional thrust multi-rotor aerial vehicles with applications to aerial physical interaction tasks,” Ph.D. dissertation, University of Toulouse, 2019.

[34] K. Alexis, C. Papachristos, R. Siegwart, and A. Tzes, “Robust explicit model predictive flight control of unmanned rotorcrafts: Design and experimental evaluation,” *European Control Conference*, p. 498–503, 2014.

[35] N. Geisert, M.and Mansard, “Trajectory generation for quadrotor based systems using numerical optimal control,” *Robotics and Automation (ICRA), 2016 IEEE International Conference on IEEE*, p. 2958–2964, 2016.

[36] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart, “Fast nonlinear model predictive control for multicopter attitude tracking on so (3),” *2015 IEEE Conference on Control Applications (CCA),IEEE*, p. 1160–1166, 2015.

[37] Y. Chen, D. Cuccato, M. Bruschetta, and A. Beghi, “A fast nonlinear model predictive control strategy for real-time motion control of mechanical systems,” *IEEE International Conference on IEEE Advanced Intelligent Mechatronics (AIM)*, pp. 1780–1785, 2017.

[38] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, “Pampc: Perception aware model predictive control for quadrotors.” *Intelligent Robots and Systems IROS 2018 IEE RSJ International Conference on IEEE RSJ (2018)*, 2018.

[39] 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,” *Robotics: Science and Systems (RSS) 2020*, 2020.

[40] 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,” *IEEE International Conference on Robotics and Automation 2021*, 2021.

[41] M. Faessler, A. Franchi, and D. Scaramuzza, “Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories,” *IEEE Robotics and Automation letters*, April 2018.

[42] H. J. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, “qpoases: a parametric active-set algorithm for quadratic programming,” *Math. Prog. Comp.*, March 2012.

[43] R. Mattone, “Sensorless robot collision detection and hybrid force/motion control,” *2005 IEEE Int. Conf. on Robotics and Automation. Barcelona, Spain*, p. 999–1004, 2005.

[44] Y. Chen, “Matmpc user manual,” December 2020.

Appendix A

MATMPC tool [1]

A.1 Overview

MATMPC is an open source software built in MATLAB for nonlinear model predictive control (NMPC).
It facilitates modelling, controller design and simulation for a wide class of NMPC applications.
MATMPC has a number of algorithmic modules, including automatic differentiation, direct multiple shooting, condensing,linear quadratic program (QP) solver and globalization.
It also supports a unique Curvature-like Measure of Nonlinearity (CMoN) MPC algorithm.
It allows making the prototyping easy, also with limited programming knowledge, each module having the possibility to be written in MATLAB API for C.

A.2 NMPC software packages

Existing NMPC software packages can be categorized into two main classes. The first one is characterized by software written in MATLAB (or any other high level interface). These allow development of algorithms, tuning, and offline simulation, as MATLAB functions are flexible to edit and easy to understand. Examples may be GPOPS; ICLOCS2 and CasADi. Furthermore, they are powerful for algorithm prototyping and debugging, but the computational efficiency is not comparable to those designed for embedded application.
The second class of NMPC software target embedded hardware and fast deployment. They are also split into 2 categories, one based on automatic code generation and the other one employing a modular structure. The one based on code

generation generates a tailored piece of code for a specific application: GMRES , ACADO2 , VIATOC , GRAMPC and Forces Pro. The code generated is compact, self-contained and is very efficient and hardware compatible. However, it lacks flexibility and maintainability. It is not suitable for algorithm prototyping and debugging. In the one based on the modular structure independent algorithmic modules are implemented : ACADOS, CT. Such software requires a decent knowledge of low-level programming languages like C/C++, which can be considered a disadvantage, which adds to the problems caused by the different operating systems, hardware and high level interfaces on which they are implemented.

A.3 MATMPC features

MATMPC combines the properties from both packages. First, MATMPC is mainly written in MATLAB language, making Simulink integration and NMPC simulation easy and flexible. Second, it has a modular structure and its time critical modules are written in MATLAB API for C. They are compiled into MEX functions. It doesn't need to be compiled at a given operating system before its usage, it relies only on MATLAB without external library dependencies at compilation time.
It also gives the possibility for each module to be replaced by any other from MATMPC or other user-defined module.
It exploits direct multiple shooting to discretize the optimal control problem (OCP) into Nonlinear Programming problem (NLP) based on dynamic models. Integrators such as explicit and implicit Runge-Kutta integrators, are implemented to approximate continuous trajectory of systems. It uses sequential quadratic programming (SQP) methods to solve the NLP. Moreover, also stable condensing algorithms can be used to convert the sparse quadratic program (QP) into (partial) dense QP. A number of QP solvers are embedded. A line search globalization algorithm is provided for searching local minimum of NLP.
Moreover, a Curvature-like Measure of Nonlinearity (CMoN) SQP algorithm is implemented, which facilitates updating of only part of sensitivities of system dynamics between two consecutive iterations and sampling instants. These are decided in terms of the value of the CMoN.

A.4 Algorithm basics

The NLP is formulated as the result after applying direct multiple shooting to an OCP over the prediction horizon T = [t_0, t_f], which is divided into N shooting intervals $[t_0, t_1, ..., t_N]$:

$$\begin{aligned} min_{x_k,u_k} \Sigma_{k=0}^{N-1} \frac{1}{2}\|h_k(x_k,u_k)\|_W^2 + \frac{1}{2}\|h_N(x_N)\|_{W_N}^2 \\ s.t. \quad x_0 = \hat{x_0} \\ x_{k+1} = \Phi_k(x_k,u_k), k = 1...N-1 \\ \underline{r_k} <= r_k(x_k,u_k) <= \overline{r_k}, k = 1...N-1 \\ \underline{r_N} <= r_N(x_N) <= \overline{r_N}, \end{aligned} \tag{A.1}$$

where $\hat{x_0}$ is the measurement from the current state. System states are defined at the discrete time point t_k for the whole prediction horizon and the control inputs u_k are considered to be piece-wise constant for the same prediction horizon. The second constraint is the "continuity constraint", where ϕ is the numerical integration operator that solves the following problem and return the solution at the next sampling time :

$$0 = f(x(t), \dot{x(t)}, u(t), t), x(0) = x_k \tag{A.2}$$

A.5 Sequential Quadratic Programming

With the state and input vectors defined as :

$$x = [x_0^T, x_1^T, ..., x_N^T]^T u = [u_0^T, u_1^T, ..., u_{N-1}^T]^T \tag{A.3}$$

, problem at A.1 is solved using SQP at each iteration i, by reformulating it as :

$$\begin{aligned} min_{\Delta x,\Delta u} \Sigma_{k=0}^{N-1} \frac{1}{2} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix}^T * H_k^i * \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + {g_k^i}^T * \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + \frac{1}{2} * \Delta x_N^T * H_N^i * \Delta x_N + {g_n^i}^T * \Delta x_N \\ s.t. \quad \Delta x_0 = \hat{x_0} - x_0 \\ \Delta x_{k+1} = A_k^i * \Delta x_k + B_k^i * \Delta u_k + d_k^i \\ \underline{c_k^i} <= C_k^i * \Delta x_k + D_k^i * \Delta u_k <= \overline{c_k^i}, \end{aligned} \tag{A.4}$$

where $\Delta x = x - x^i; \Delta u = u - u^i$. The Hessian matrix :

$$H_k^i = \frac{\partial h_k^i}{\partial (x_k, u_k)} * \frac{\partial h_k^i}{\partial (x_k, u_k)} \tag{A.5}$$

is always positive-definite.

The QP problem A.4 can be solved either by structure exploiting or by sparse solvers,

such as HPIPM, OSQP or Ipopt.
In addition, it can be condensed and the following dense QP problem will be obtained:

$$\begin{aligned} min_{\Delta u} & \frac{1}{2} * \Delta u^T * H_c * \Delta u + g_c^T * \Delta u \\ s.t. \quad & \underline{c_c} <= C_c * \Delta u <= \overline{c_c}, \end{aligned} \tag{A.6}$$

which can be solved by dense QP solvers, like qp_OASES. It is also possible to partially condense and obtain a smaller but still sparse QP problem.
Finally, the solution of the QP optimization problem is used to adapt/update the solution of the NLP problem, according to:

$$x^{i+1} = x^i + \alpha^i * \Delta x^i, u^{i+1} = u^i + \alpha^i * \Delta u^i \tag{A.7}$$

. The tuning parameter α^i is the step length determined by globalization strategies. In MATMPC, this parameter can be determined in 2 ways. First, through a practical line search SQP algorithm employing l_1 merit function. Second, also related to practical situations, the optimal solution becomes a sub-optimal one, the SQP iteration is terminated before convergence is achieved, specifically after one iteration with a full Newton step $\alpha = 1$ (Real-Time Iteration method).

A.6 Curvature-like measure of nonlinearity SQP

In MATMPC, it is possible to use the CMoN-SQP algorithm to adaptively update system sensitivities on-line, according to the following update rule:

$$\nabla\Phi_k^i = \begin{cases} \nabla\Phi_k^{i-1}, if \kappa_k^i <= \eta_{pri}^i \quad \tilde{\kappa}^i <= \eta_{dual}^i \\ eval(\nabla\Phi_k^i) otherwise. \end{cases} \tag{A.8}$$

$\Phi_k = [A_k B_k]$, k=0...N-1 is the sensitivity matrix and the pair (η_{pri}^i and η_{dual}^i) represent the CMoN threshold. CMoN-SQP only requires two user defined parameters ($\epsilon^{abs}, \epsilon^{rel}$) off-line. These are the absolute and relative tolerances of the accuracy of the solutions of the NLP optimization problem. The threshold values (η_{pri}^i and η_{dual}^i) are defined in terms of them. Then, the CMoN thresholds κ_k^i and $\tilde{\kappa}^i$ (which have the role of local sensitivities) are defined on-line, at each iteration step. In the end, only a number of sensitivities will be updated on-line, according to the rule in A.8, decreasing the computation time and resources overall.

A.7 Modules of MATMPC

MATMPC consists of two main functions, namely the $model\_generation$ and the $simulation$.

The model generation function takes user-defined dynamic models and generates C codes of model analytic functions and their derivatives, using also automatic differentiation (AD) from the CasADi tool (B). MATMPC only generates codes from model dynamics. The model and optimization parameters are taken as parameters that can be modified on-line. Therefore, model generation function doesn't need to be repeatedly run in MATMPC.

The simulation function is used for running closed-loop NMPC simulations in MATLAB. Available options in MATMPC are given in Table A.1.

The NMPC controller in MATMPC is a MATLAB function that calls a number of modules: qp generation for multiple shooting, condensing for performing (partial) condensing routines, qp solve for calling QP solvers, and line search for performing globalization. These modules don't need to allocate any memory on-line, they use the same ones created at initialiation. In addition, it is possible to pause simulation and inspect intermediate data, just like debugging standard MATLAB functions.

In MATMPC, there are two sources of external dependencies. The first is CasADi for performing AD and generating C codes of model functions and derivatives. The second source is from QP solvers. These two external dependencies do not require additional compiling or installing processes. More details can be found in [1] and [44].

The options used for the simulations in the current thesis are summarized in table A.2.

Hessian Approximation	Newton-Euler
Integrator	Explicit Runge Kutta 4; Implicit Runge Kutta
Condensing	non; full; partial
QP solver	qpOASES; Matlab quadprog; Ipopt OSQP; HPIPM
Globalization	l_1 merit function line search; Real-Time Iteration
Additional features	CMoN-SQP; input MB

Table A.1: Available options in MATMPC

hotstart	no
Integrator	(Explicit Runge Kutta 4) ERK4
Condensing	full
QP solver	qpOASES
Globalization	RTi (Real-Time Iteration)

Table A.2: Options used in current simulations

Appendix B

CasADi software package

B.1 Overview

CasADi started out as a tool for algorithmic differentiation (AD). While it continued the development, it concentrated towards optimization. CasADi provides a set of general-purpose building blocks needed for numerical optimal control, without sacrificing efficiency. Thus, it is suitable for teaching optimal control to graduate-level students and allows researchers and industrial practitioners to write codes for a particular application or problem structure.
The core of CasADi consists of a symbolic framework that allows users to construct expressions and use these to define automatically differentiable functions (best represented in MATLAB's Symbolic Toolbox or Python's SymPy package). Once created, they can either be used to obtain new expressions for derivatives using AD or be evaluated in CasADi's virtual machines or by using CasADi to generate self-contained C code.

B.2 Syntax and usage

CasADi uses a MATLAB inspired "everything-is-a-matrix" type syntax.Scalars are treated as 1-by-1 matrices and vectors as n-by-1 matrices. Furthermore, all matrices are sparse. Working with a single sparse data type makes the tool easier to learn and maintain.

B.3 Graph representation – Scalar expression type

Scalar in this context does not refer to the type itself – SX is a general sparse matrix type – but the fact that each nonzero element is defined by a sequence of scalar-valued operations. CasADi uses the same compressed column storage format to store matrices, (the same used to represent sparse matrices in MATLAB). The difference is that in CasADi entries are allowed to be structurally nonzero but numerically zero.

$$\begin{aligned} C &= SX.eye(2); \\ C(1,1) &= SX.sym('x'); \\ C(2,1) &= 0; \\ disp(C); \end{aligned} \tag{B.1}$$

The output will be : [[x,00],[0,1]].
The structural zero is 00, while the numerical one is 0. Symbolic matrices are always sparse in CasADi, while in Matlab's Symbolic Math Toolbox expressions are always dense.
In the SX case, expressions are stored as a directed acyclic graph (DAG) where each node (/atomic operation) is: – A symbolic primitive, created with SX.sym as above – A constant – A unary operation, e.g. sin – A binary operation, e.g. , + This facilitates numerical evaluation with little overhead, either in a virtual machine or in generated C code.
The second expression type in CasADi is the matrix expression type – MX, for which each operation is a matrix operation. In the most general case, an MX operation can have multiple matrix-valued inputs and return multiple matrix-valued outputs.

$$\begin{aligned} x &= MX.sym('x', 2); \\ A &= MX.sym('A', 2, 2); \\ e &= A * sin(x); \\ disp(e); \end{aligned} \tag{B.2}$$

The output will be : mac(A,sin(x),zeros(2x1)). In the resulting expression there are two matrix valued symbolic primitives (A and x), 2 x 1 all-zero constant, a unary operation (sin) and a matrix multiply-accumulate operation, mac(X1,X2,X3) := X3 +X1*X2.
The choice of atomic operations for the MX type was made so that derivatives calculated either using the forward or reverse mode of algorithmic differentiation can be efficiently expressed using the same set of operations.

A special type of atomic operation is a function call node, which represents a call to a function object created at runtime. For certain expressions there can be multiple calls to the same function object, which results in maintaining the size of the expression graphs small.

B.4 Function objects and virtual machines

The symbolic expressions in CasADi can be used to define function objects, that behave like conventional functions but are created at runtime. They support numerical evaluation, symbolical evaluation, C code generation and derivative calculations. They are created by receiving a name, a list of inputs and a list of outputs:

$$F = Function('F', x, A, e); \tag{B.3}$$

defines a function object with the display name “F” and two inputs (x and A) and one output (e). Function objects can have an arbitrary number of inputs and outputs, but each one is a sparse matrix. If an input has structural zeros, the respective function will not to depend on the corresponding part of the matrix.
When a function object is created in CasADi, the expression graph is topologically sorted, turning the directed acyclic graph (DAG) into an algorithm that can be evaluated. There is no relation between the order in which expressions were created (i.e. a tracing step) and the order in which they appear in the sorted algorithm. CasADi uses a depth-first search to topologically sort the nodes of the DAG.
With the sorted list of operations, CasADi will implement two register based virtual machines (VMs), one for each graph representation. An element (SX VM) or an interval (MX VM) from a work vector are assigned for all inputs and outputs of an operation. To limit the size of the work vector, the live variable range of each operation is analyzed and work vector elements or intervals are then reused in a last-in, first-out manner. Also, if it is possible, operations are performed in-place.
It is possible to define nested function objects. Giving the possibility to embed subexpressions used multiple times into separate function objects allows expression graphs to stay small. Function objects in CasADi, as represented by the Function class, are not mandatory to be defined by symbolic expressions as above.

B.5 Algorithmic differentiation

Algorithmic differentiation (AD)/automatic differentiation is a technique used for calculating derivatives of functions represented as algorithms.
The forward mode of AD for a function y=f(x) gives a method for calculating the Jacobian-times-vector product :

$$\hat{y} = \frac{\partial f}{\partial x} * \hat{x} \tag{B.4}$$

at a computational cost comparable to evaluating the original function f(x).
This extends to the case when the inputs and outputs are matrices, and/or multiple matrices.
The reverse mode gives the method of calculating the Jacobian-transposed-times-vector product:

$$\tilde{x} = \frac{\partial f}{\partial x}^T * \tilde{y} \tag{B.5}$$

again at a computational cost comparable to evaluating the original function f(x), but in general with a larger, often avoidable, memory overhead. Again, it can be applied also to the cases when the inputs and outputs are matrices, and/or multiple matrices.
In any implementation, AD works by breaking down a calculation into a sequence of atomic operations.For example, in the case of matrix-matrix multiplication Y = X1 X2, the forward mode is given by:

$$\hat{Y} = \hat{X_1} * + X_1 * \hat{X_2} \tag{B.6}$$

while the reverse mode is given by:

$$\begin{aligned} \tilde{X_1} &= \tilde{Y} * X_2^T; \\ \tilde{X}2 &= X_1^T * \tilde{Y}; \end{aligned} \tag{B.7}$$

Directional derivatives can be thus calculated using foward and reverse modes. The complete Jacobian, which can be large and sparse, is more difficult to calculate. Higher order derivatives can be treated as special cases, or, can be calculated by applying the AD algorithms recursively.
CasADi implements AD using a source-code-transformation approach, which means that new symbolic expressions, using the same graph representation, are generated whenever derivatives are requested.

B.6 Directional derivatives

For a function object defined by a symbolic expression, CasADi implements the forward and reverse modes of AD by propagating symbolic seeds forward and backward through the algorithm. The result is a new symbolic expression with references to the graph of the non-differentiated function.Whenever a function call node is encountered, a new function object is generated for calculating directional derivatives. The generated function objects for the derivatives are cached in order to limit memory use.
More details related to the way complete Jacobians and Hessians can be implemented, together with implicitly defined differentiable functions, optimization and examples can be found in [44].

Appendix C

Generic MRAV modelling, with focus on the Tilt-Hex MDT-MRAV

Multi-rotor platforms are modeled as rigid bodies having mass m, actuated by $n \epsilon N$ spinning motors coupled with propellers, i.e., n = 6 in the particular hexarotor model used in this thesis.
As reference frames 4 will be defined and used. First, $F_W = O_W, \left\{x_W, y_W, z_W\right\}$ and $F_B = O_B, \left\{x_B, y_B, z_B\right\}$ denote the world inertial frame and the body frame attached to the MRAV. The body frame has as origin the same point as the Center of Mass (CoM) of the aerial platform. Its position w.r.t. the world origin is denoted with $p_B^W \epsilon R^3$ will be used as p in the following. Furthermore, also the orientation of F_B w.r.t. F_W is represented as the rotation matrix $R_W^B \epsilon R^(33)$, used as R in the following mathematical derivations.
Then, the third reference frame employed is $F_{A_i} = O_{A_i}, \left\{x_{A_i}, y_{A_i}, z_{A_i}\right\}$, the reference frame related to the i-th actuator, $i = 1, ..., n$, with O_{A_i} attached to the thrust generation point and z_{A_i} aligned with the thrust direction. In this way it is possible to define the actuator force expressed in its frame as $f_A^i = f_i * e_3$, where e_3 is the usual representation used for the 3rd component of the canonical base in the space R^3. Similarly as for the relationship between the world frame and body frame, the position of one propeller w.r.t. the O_B origin of the body frame is defined as $p_{A_i}^B$ and the orientation of F_{A_i} to F_B is represented with $R_{A_i}^B$. In addition, $p_{A_i}^W$ or simply p_{A_i} is the position of one propeller w.r.t. world-frame and $R_{A_i}^W$ represents the orientation between the actuator and inertial frame.
J represents the vehicle inertia matrix, defined w.r.t. O_B, expressed in F_B. It is a positive-defined matrix. Furthermore, the angular velocity of the vehicle body-frame w.r.t. the world frame, expressed in body-frame is defined as ω_B^B, but is used for simplicity as ω in the following derivations.
The fourth and last reference frame defined is the one attached to the end-effector,

$F_E = O_E, \left\{x_E, y_E, z_E\right\}$, with the origin in the interaction point O_E. As for the other reference frames, its position w.r.t. to body-frame will be defined as p_E^B, its orientation w.r.t. the body-frame as R_E^B, its position w.r.t. the world-frame as p_E^W or simply p_E and the orientation to the world-frame as R_E^W. Finally, its angular velocity w.r.t. the world-frame expressed in its own frame will be defined as ω_E^E.
Previous defined symbols and the additional ones used in the following mathematical derivations are grouped in table C.1.

The orientation kinematics of the body and end-effector are expressed by the equation C.1 :

$$\begin{aligned} \dot{R_R} &= R_R * [\omega_R]_\times \\ \dot{R_E} &= R_E * [\omega_E]_\times \end{aligned} \tag{C.1}$$

where $[]_x$ represents the skew-symmetric matrix associated to the vector (.
The Newton-Euler formalism is used to define the dynamics of the platform as in equation C.2.

$$\begin{bmatrix} m\ddot{p}_R \\ J\dot{\omega}_R^R \end{bmatrix} = - \begin{bmatrix} m * g * e_3 \\ \omega_R^R X J * \omega_R^R \end{bmatrix} + \begin{bmatrix} f \\ \tau^R \end{bmatrix} + \begin{bmatrix} f_R \\ \tau_R^R \end{bmatrix} \tag{C.2}$$

, where the first contribution is due to gravity, the second component are the forces and torques due to propellers forces and the final component is the wrench felt at the body-frame origin due to the force and torque expressed by the environment on the end-effector. The translational dynamics is expressed in world-frame, while the rotational dynamics is expressed in body-frame.
The NCFTP platform is based on a hexarotor structure, with propellers equally-spaced and equidistant from O_R in the x-y-plane of F_R. Full actuation is achieved by tilting every single motor-propeller combination. The orientation between each propeller reference frame (F_{A_i}) to the body-frame is achieved according to the formula C.3.

$$R_{P_i}^B = R_z((i-1) * \pi/3) * R_x((-1)^{i-1} * \alpha) * R_y(\beta), i = 1...6 \tag{C.3}$$

The inclination of the i-th propeller group w.r.t. the center of mass is defined by the 2 constant parameters α and β. Furthermore, if α has alternating signs for every consecutive propeller assures that the platform is fully actuated.
The position of the i-th propeller group in relation to the body-frame origin can be determined using the equation C.4.

$$p_{P_i}^B = R_z((i-1) * \pi/3) * l + R_x * ((-1)^{i-1} * \alpha) * R_y(\beta) * d, i = 1...6 \tag{C.4}$$

Overview of most symbols used. If constant a valueis presented as well		
Definition	Symbol	Value
Inertial world frame with origin O_W and axes $(x_W; y_W; z_W)$	F_w	
Robot body frame with origin O_R and axes $(x_R; y_R; z_R)$	F_R	
End effector frame with origin O_E and axes $(x_E; y_E; z_E)$	F_E	
Propeller frame with origin O_{A_i} and axes $(x_{A_i}; y_{A_i}; z_{A_i})$	F_{A_i}	
Symbols that can assume the values W;R;A_i or E	*,o	
Position of O^o in F_W	p_o	
Position of O^o in F_*	p_o^*	
Velocity of O^o in F_W	v_o / $\dot{p}_o$	
Velocity of O^o in F_*	v_o^* / $\dot{p}_o^*$	
Acceleration of O^o in F_W	a_o / $\ddot{p}_o$	
Acceleration of O^o in F_*	a_o^* / $\ddot{p}_o^*$	
Rotation matrix expressing the orientation of F_o w.r.t. F_W	R_o	
Rotation matrix expressing the orientation of F_o w.r.t. F_*	R_o^*	
Angular velocity of F_o w.r.t. F_W, expressed in F_o	ω_o	
Tilting angle (around x_{P_i}) of the i^{th} prop. group	α_i	35^o
Tilting angle (around y_{P_i}) of the i^{th} prop. group	β_i	10^o
i^{th} propeller blade spinning frequency about z_{P_i} (in Hz)	$1/c\_f * \sqrt{u_i}$	
Mass of the whole aerial robot	m	2.77 kg
Inertia matrix (w.r.t. the body frame, expressed in body frame)	J	$\begin{bmatrix} 0.083 & 0 & 0 \\ 0 & 0.081 & 0 \\ 0 & 0 & 0.16 \end{bmatrix}$
Gravity acceleration constant	g	9.81 m/s^2

Table C.1: Overview of symbols used

where d is the vector from the center of the tilting rotation to the center of the motor-propeller group and l the one from body frame origin to the center of the tilting rotation (see Figure C.1).

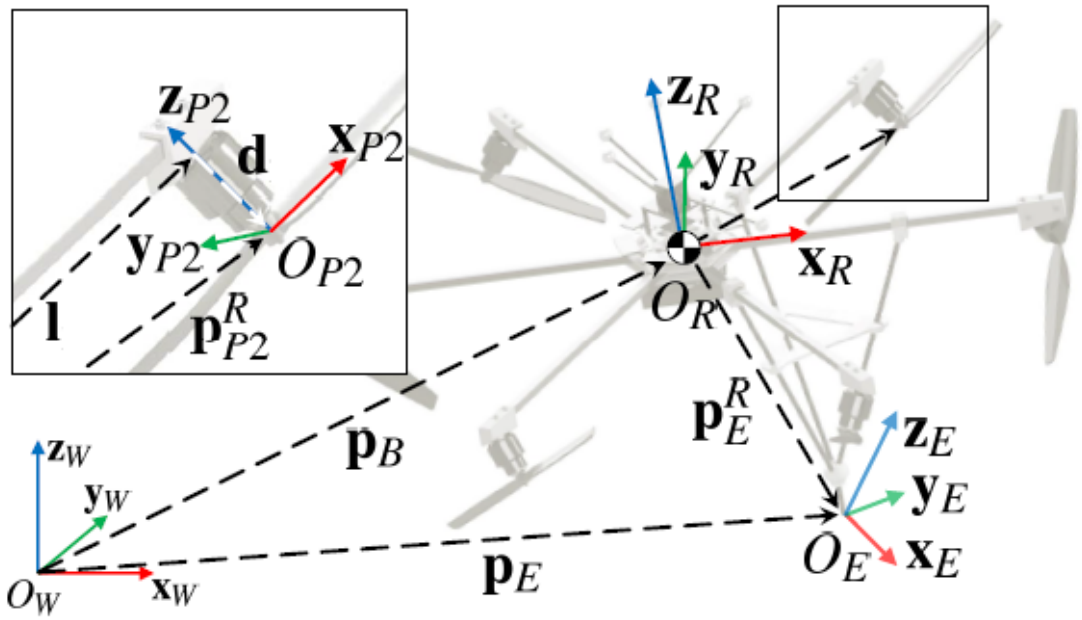


Figure C.1: Tilt-hex with reference frames

The forces generated by the propellers' forces are defined as in C.5.

$$f = \Sigma_{i=1}^{n} f_i = \Sigma_{i=1}^{n} R_{A_i}^{W} * f_i^{A_i} = \Sigma_{i=1}^{n} R_{A_i}^{W} * e_3 * f_i. \tag{C.5}$$

The torques are the result of the torques produces by the propellers forces due to their leverage arms and also the drag moments produced by the counteracting reaction of the air to the propellers' rotation.

$$\tau_B^B = \Sigma_{i=1}^{n} \tau_{f_i}^B + \tau_{d_i}^B = \Sigma_{i=1}^{n} p_{A_i}^B \times f_i^B + c_i * c_f^\tau * f_i^B = \Sigma_{i=1}^{n} ([p_{A_i}^B]_\times + c_i * c_f^\tau * I_3) * R_{A_i}^B * e_3 * f_i \tag{C.6}$$

, where the chosen model of the propeller thrust is the well known one : $f_i = c_f * \omega^2$. The constant parameter $c_f^\tau > 0$ is defined as the intensity ratio between the thrust produced by the propeller rotation and the generated drag torque. Furthermore, c_i is a variable whose value is equal to -1 (respectively, $+1$) in the case the direction of the induced drag torque is opposite (respectively, the same) w.r.t. the generated thrust force, in other words when the propeller spins counter-clockwise or clockwise in terms of its thrust direction. c_f is to be determined experimentally.
The body wrench (forces and torques) generated by the propellers are mapped using the allocation matrix ($G\epsilon R$), which embeds the contributions from equations C.5

and C.6:

$$\begin{bmatrix} f \\ \tau^B \end{bmatrix} = \begin{bmatrix} G1 \\ G2 \end{bmatrix} * \gamma = G * \gamma \tag{C.7}$$

G1 maps the actuator forces to the body forces and G2 maps them to the body moments.