THE CONCEPTION, DEVELOPMENT, IMPLEMENTATION OF A NMPC CONTROLLER FOR THE OMNIMORPH

K. (Kevin) Jeulink

MSC ASSIGNMENT

Committee:
dr. ir. C. Salm
prof. dr. ir. A. Franchi
dr. D. Bicego
Y.A.L.A. Aboudorra, MSc

June, 2023

017RaM2023
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Summary

Manipulation and interaction tasks with robots have gained popularity in research. Companies, research institutes and universities in Europe have joined multiple Europian Projects that focus on research in this domain. In the scope of one of those Europian Projects, the Aerial Core project[1], the University of Twente has developed an omnidirectional multi-rotor aerial vehicle with actively tilting propellers, named the OmniMorph. In this thesis, a Non-Linear Model Predictive Control (NMPC) algorithm is designed and implemented. NMPC optimizes the control input by looking into the future, using a prediction horizon and reference trajectory inside a cost/objective function. The propellers can actively tilt by the use of a single servo motor, in order to switch from unidirectional thrust mode to multidirectional thrust (morphing). The controller generates the propeller force derivatives and servo rate as a control input. Also, it takes into account constraints on the input, propeller forces, servo angle and position and orientation. The NMPC can automatically optimize energy consumption by changing the servo angle. Several simulation experiments are given regarding model mismatches, the influence of different cost/objective functions, the length of the prediction horizon and a comparison with an existing reactive controller. Finally, a conclusion and possible directions for future work are given. Potential directions for future work include for example the addition of machine learning for the many tunable parameters of the controller, connecting the controller to a physics simulator, doing experiments with the real platform or simulations/experiments with interaction forces.

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

Acknowledgements

First of all, I would like to thank my daily supervisors Youssef Aboudorra and Davide Bicego for their amazing and incredibly helpful support. During my thesis, they have helped me in every possible way to learn and understand the topics regarding control, robotics and research in general. During meetings and small chats, they pushed me in the right direction when needed and gave me a lot of motivation. Feedback was always helpful and made me think critically. Their contribution to this work is incredibly valuable.

I would also like to thank Antonio Franchi, who was the chair of my thesis project. He gave me the amazing opportunity to do research on the super interesting topics of Non-Linear Model Predictive Control and multi-rotor aerial vehicles. Furthermore, I would like to thank him for the super interesting feedback sessions, where he motivated me a lot and gave me fascinating ideas about the topic and the direction I could go.

Finally, I would like to thank my girlfriend Katharina, my parents, Andre and Jacquelien and the rest of my family for their amazing support during my thesis. They gave me the space and support to work hard and detach when needed. I have enjoyed every minute of the project and I am looking forward to starting my career in the field of NMPC and robotics.

Kevin Jeulink
Sunday 11th June, 2023

Contents

List of Figures

List of Tables

List of Acronyms

UAV Unmanned Aerial Vehicle

MRAV Multi Rotor Aerial Vehicle

VTOL Vertical Takeoff and Landing

UA Under Actuated

FA Fully Actuated

OA Over Actuated

NMPC Non-Linear Model Predictive Control

PID Proportional Integral Derivative

CoM Center of Mass

RPY Roll Pitch Yaw

ESC Electronic Speed Controller

DC Direct Current

ABAG Adaptive Bias and Adaptive Gain

UDT Uni Directional Thrust

MDT Multi Directional Thrust

RMSE Root Mean Square Error

OCP Optimal Control Problem

AD Automatic Differentiation

QP Quadratic Program

NLP Non-Linear programming problem

SQP Sequantial Quadratic Programming

RTI Real-Time Iteration

1 Introduction

1.1 Context

In the last decades, unmanned aerial vehicles (UAVs) have gained popularity in research but also commercially. Applications include mapping and inspection in the mining industry [10], inspection and monitoring of power lines [11], assisting in search and rescue missions [12, 13], traffic monitoring [14] or logistics [15]. Most of these applications mentioned are based on visual sensing or sensing without physical interaction with the environment. However, recently UAV research has been shifted towards aerial manipulators [16]. Applications for such platforms can be for example load transportation [17] or direct contact measurements [18, 19, 20].

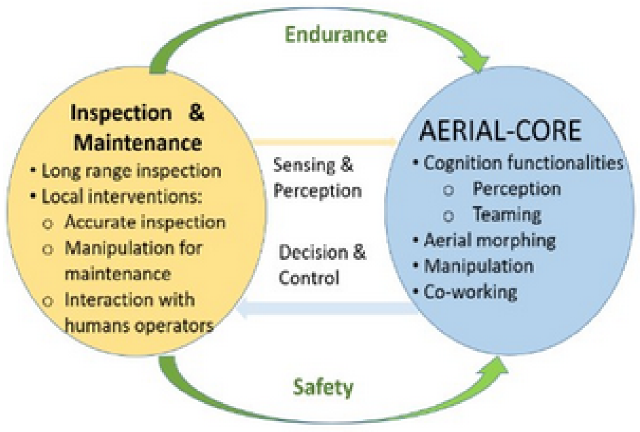


Figure 1.1: A graphical summary of the goals of Aerial-Core[1]

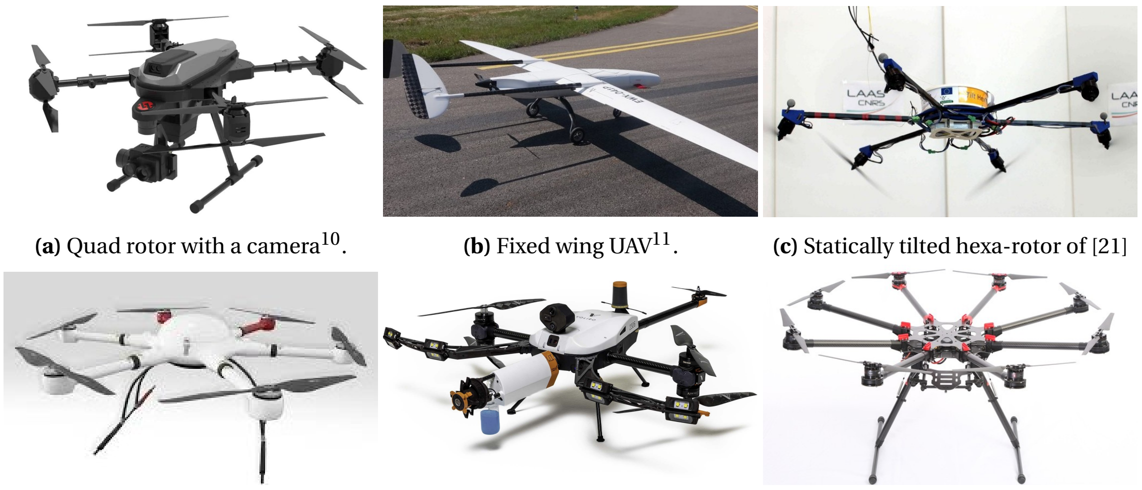
In the scope of physical interaction with UAVs, the University of Twente is part of the European Aerial Core project[1]. This project conducts research in the area of physical interaction with multi-rotors, with a focus on assisting in the construction and maintenance of long linear structures e.g., power lines, see Fig. 1.1. The technical objectives of the project are to have aerial robots with (1) cognitive capabilities such as perception, reasoning, planning, context awareness or learning and adaption, (2) morphing capabilities such as actively tilting propellers, (3) cognitive aerial manipulation, (4) cognitive safe aerial co-working, and (5) integrated aerial robotic systems. Two examples are given in Fig. 1.2. In the last years, there have been more such projects in the EU, where many companies and institutes collaborated in gathering innovative research regarding a related goal. Some of the projects are, Arcas[2], Aeroarms[3], Aeroworks[4], Airobots[5], Aerobi[6], Hyfliers[7], Spectors[8] and Pro-act[9]. First, a closer look is given at the existing literature before defining the goals of this thesis.

[1]https://aerial-core.eu/
[2]https://arcas-project.org/
[3]https://aeroarms-project.eu/
[4]http://www.aeroworks2020.eu/
[5]http://airobots.dei.unibo.it/
[6]https://www.aerobi.eu/
[7]https://www.oulu.fi/hyfliers/
[8]https://spectors.eu/
[9]https://www.h2020-pro-act.eu

(a) A drone that has a load which interacts with a power line.

(b) A drone that holds a load and interacts with a human. (Co-working)

Figure 1.2: Two examples of Aerial core drone applications.



(a) Quad rotor with a camera[10].

(b) Fixed wing UAV[11].

(c) Statically tilted hexa-rotor of [21]

(d) The M600WP Hexa-rotor drone[12].

(e) Voliro-T[13].

(f) Octo-rotor[14]

Figure 1.3: Different examples of UAVs build for consumers or research.

Various kinds of UAVs exist, such as fixed-wing UAVs or multi-rotor UAVs (MRAVs), examples are shown in Fig. 1.3. For this thesis, the focus will be on particular vertical take-off and landing (VTOL) MRAVs. In [8], authors give a relevant characterization of VTOL MRAVs, where reviews are given on the abilities and properties of such platforms based on their input allocation.

One of the most commonly built and used multi-rotors is the quad-rotor, which has four aligned in-planar actuators and is able to generate a unidirectional thrust w.r.t. its body fixed frame. The reason for their popularity is the low production cost, mechanical simplicity and widely investigated control methods. Quad-rotors are used for a variety of tasks, e.g., in surveillance [22] or package delivery [23]. The translation and orientation of such quad-rotors are coupled, due to their unidirectional thrust, which means they are underactuated (UA) and can only change lateral position by changing their orientation. This makes them sensitive to lateral disturbances and hard to use for physical interaction.

A control method for such quad-rotor with coupled translation and orientation is for example given in [24], where they use Integral Backstepping control. This control method divides the system into several sub-systems. Control laws are designed for each subsystem. Other control methods for quad-rotors include Proportional-integral-derivative (PID) control or Linear

[10]https://www.unmannedsystemstechnology.com/
[11]https://www.uavos.com/
[12]https://www.embention.com/
[13]https://voliro.com/
[14]https://www.dji.com/nl

Quadratic control (LQC), which is compared in for instance [25]. PID control is based on a feedback loop, where the error in tracking is used to compute control inputs for the system, whereas LQC is an optimization-based control method that aims to minimize a quadratic cost function.

To overcome the under-actuation limitations of quad rotors, static tilted multi-rotors have been developed, for example in [26] and [27]. In both papers, a tilted hexa-rotor has been modelled, designed and controlled. Actuators are mounted at specific tilt angles. Having at least six propellers in such a configuration enables the platform to independently apply a force and/or torque in any direction, by changing the propeller velocities of the individual rotors. This makes the UAV fully actuated (FA). Due to the allowance of multi-directional thrust, such a FA platform is capable of better handling lateral disturbances and performing certain interaction tasks like, push and slide operations, as shown in [18]. In [28] an extensive literature review of FA platforms is given.

Control for FA UAVs can be done in several ways. One commonly used method is full-pose geometric control, which is done in, e.g., [29], where also limits in lateral forces are taken into account. The controller accepts a full pose reference and computes the control inputs in a cascaded structure. Other control methods can be a combination of feedback linearization and a linear controller (e.g. PID) [18, 26].

A main limit of hexa-rotors with static tilted propellers is a waste of energy due to internal forces. Such energy waste can be fixed by actively tilting the propellers using one or more servo motors (morphing). For instance, [7] developed a platform where one servo is used to switch between UA and FA. Control is performed by full-pose geometric control with prioritized position tracking. One of the limits of their control is the manual control of the tilt angle by an operator or high-level control algorithm. In other words, the controller does not manage this additional degree of freedom. Other papers that introduce UAVs that have actively tilting propellers are for example [30] with radial tilt direction and [31] with tangential tilt direction.

Next to the previously mentioned platforms, there exists a subset of FA UAVs which are omnidirectional platforms. Omnidirectional platforms can sustain their weight in any orientation. Omnidirectional platforms have a recent interest in research, due to better tracking of attitude and position and better interaction capabilities. One of the first UAVs with omnidirectional capabilities was developed by [1], shown in Fig. 1.5. In this work, an eight-rotor configuration is designed that has optimized agility in any direction. The rotors can not tilt but do have bidirectional thrust, such that they can exert a force in positive and negative directions. The control consists of two separate control loops, one for position and one for attitude. The control outputs a desired body wrench. This wrench is converted to individual propeller thrusts using the pseudo-inverse of the allocation matrix or an optimization algorithm in the second version of the paper [2]. One of the limitations of this platform is the internal forces caused by the statically tilted propellers, which wastes energy. This could be fixed by making the platform able to morph between unidirectional thrust and omnidirectional configuration. This will enable the platform to be energy efficient for position tracking and have decoupled orientation and translation when physical interaction tasks are required. Related literature that uses omnidirectional platforms is listed in table 1.1.

Most controllers mentioned before are reactive controllers, which means they compute control inputs based on the current and past state by, for example, the difference between the actual and desired trajectory. A more recently investigated control algorithm is Nonlinear Model Predictive Control (NMPC), which is, in contrast with reactive controllers, an optimization-based control method. NMPC takes into account the future trajectory of a system, for example, a multi-rotor. In [9] authors apply NMPC to the Tilthex, a FA hexa-rotor, and perform real experiments with this platform. Furthermore, the method is applied to the Fasthex, a morphing FA hexa-rotor, but only in simulations results are presented. Other papers using NMPC are listed

Table 1.1: Related literature that uses omnidirectional platforms.

Paper	Morphing?	Control method	Control allocation	Bi-directional propeller?
[3]	no	PID	Inverse of the allocation matrix	yes
[1]	no	Cascaded control	Pseudo inverse	yes
[2]	no	Cascaded control	Two-Norm optimization	yes
[4]	no	PID pose control	Infinity-Norm optimization	yes
[32]	yes	PID (position), quaternion error + rate controller (attitude)	variable transformation + pseudo inverse	no
[33]	yes	Geometric control	Pseudo inverse	no
[5]	yes	LQRI	Weighted differential allocation	no
[6]	yes	MPC (generate optimal body wrench)	pseudo inverse	no

Table 1.2: Table of the literature that uses NMPC for control, where most papers have the wrench map inside the optimal control problem (OCP).

Paper	Morphing?	Control allocation
[9]	In simulation	Wrench map inside OCP
[34]	no	Wrench map inside OCP
[35]	no	Wrench map inside OCP
[36]	no	Wrench map inside OCP
[37]	no	Wrench map inside OCP
[20]	yes	Wrench map inside OCP
[6]	yes	Pseudo inverse

in table 1.2. Most papers have the allocation inside the optimal control problem of the NMPC, meaning the NMPC computes directly the individual propeller forces or propeller speeds.

1.2 Problem description

As part of the previously mentioned Aerial core project, the Robotics and Mechatronics group of the University of Twente has developed the OmniMorph, shown in Fig. 1.4. This platform is a cubic-like octo-rotor platform, that has one servo motor to change the orientation of the propellers. The OmniMorph is an omnidirectional multi-rotor. Hence, it might be more suitable than non-omnidirectional platforms to perform the defined tasks of the Aerial-Core project. These tasks include safe interaction with human operators, manipulation tasks and contact measurements.

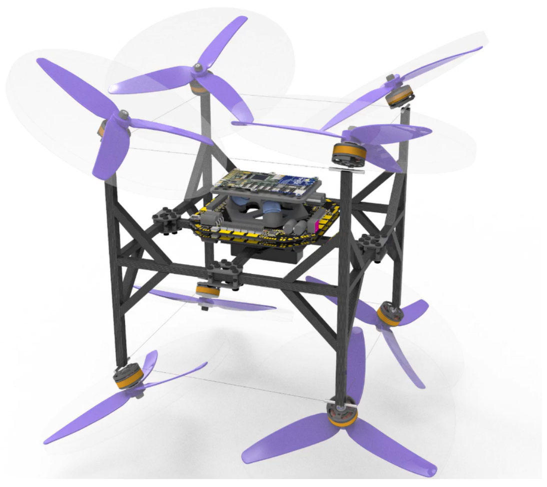


Figure 1.4: A virtual model of the OmniMorph, where the propellers are tilted by a certain angle.

For this master thesis, a control algorithm needs to be designed for the OmniMorph. The platform is over-actuated, i.e., it has more inputs than controllable outputs. Furthermore, it has limitations such as limited propeller forces, limited derivatives of the propeller forces, limited servo angles and limited energy consumption. It is crucial to take those limitations into account because it can prevent unstable behaviour, which can be dangerous with human coworking. Furthermore, considering the limitations can improve performance with flying and interaction tasks. Finally, knowing the energy limitations can help to optimize energy consumption and improve flight time. To control the platform and take into account such limitations, we wish to implement an NMPC algorithm for the OmniMorph. NMPC is an algorithm that is able to take the limitations of the states and inputs into account. Further explanation about NMPC will be given in the theoretical background.

Using NMPC gives rise to several challenges. First, NMPC is highly dependent on the mathematical model of the platform, as it uses the model to predict the control inputs over a prediction horizon. Secondly, the algorithm is computationally expensive compared to reactive controllers. Third, the controller needs to know about the ability to morph, such that it is energy efficient. In other words, the platform should be only FA and therefore have tilted propellers, when needed. Based on those challenges five research questions are defined below. The focus is on (1) the model, (2) the objective function (a core part of NMPC), (3) the energy consumption, (4) the prediction horizon, (5) the computational time and (6) the comparison with a reactive controller. For the computational time preliminary tests, assessing the computational effort, were carried out on a laptop. Tests on an embedded computer are planned after the thesis. The research questions are defined as follows:

1. What is the "best" model to use in an optimization-based control for the OmniMorph? The model includes the dynamic equations of motion and the input constraints (Tradeoff extensiveness/complexity)
2. What is the "best" objective function to use in order to encompass both tracking ability and energy high-fidelity efficiency coming from the actual energy consumption model and measurements?
3. What influence does the prediction horizon have on the performance of the controller?
4. Will the NMPC be able to control the real platform in real-time while being implemented onboard with the computational capabilities available onboard?
5. Will the NMPC outperform an existing Lyapunov-based reactive controller?

1.3 Related literature

To the best of the writer's knowledge, there exist five papers that relate closely to the goals of this thesis. The first platform is of [1] and [2], shown in Fig. 1.5, and shortly mentioned before. The platform consists of eight bi-directional propellers positioned in an optimized way, to allow omnidirectional capabilities. While this platform allows for omnidirectional capabilities, the rotors are static and cannot be tilted, wasting energy while hovering. In contrast, the proposed design of the OmniMorph is able to orientate its propellers in an efficient way, allowing it to be more energy efficient.

The control of [1, 2] is done by two reactive control loops that control the position and attitude separately. The wrench computed by those control loops is fed into an allocation method. In the first version of the paper, the authors use the pseudo-inverse of the allocation matrix. Because the platform has more inputs than degrees of freedom, there exists no unique map. Therefore the pseudo-inverse could produce infeasible propeller forces, making the platform unstable. In the second version of their paper, the allocation is improved and done by a two-norm optimization algorithm that minimizes the thrust norm and rate of thrust. Furthermore, a hysteresis set is used to prevent the bi-directional propellers from switching too often.

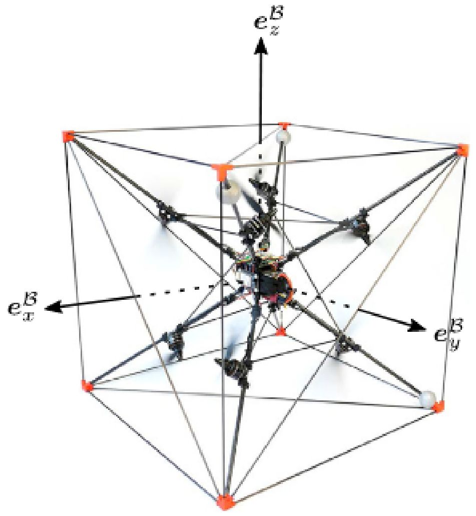
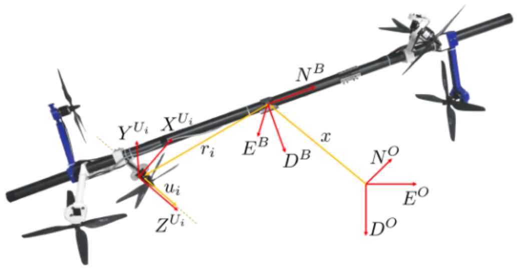
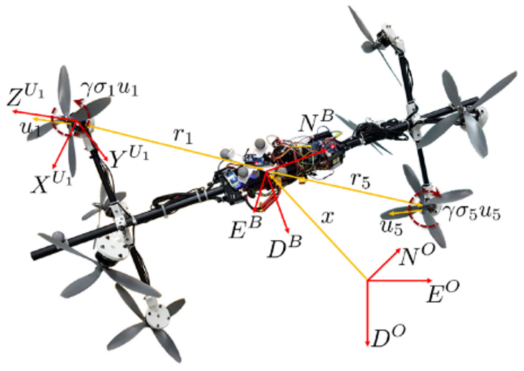


Figure 1.5: Statically tilted octo-rotor platform with bi-directional propellers of [1, 2].

Next, a six- and eight-propeller omnidirectional platform was built by [3] and [4] respectively. The eight-propeller version is an improved design of the six-propeller platform. Both versions of the platform are shown in Fig. 1.6. The platform design is the result of an optimization framework that maximizes the minimum guaranteed control force/torque for any attitude. The control goals are tracking pose trajectories on SE(3), hybrid pose/wrench control and bilateral teleoperation. For pose tracking a PID algorithm is used. The allocation is done by an infinity norm optimization algorithm to address the actuation redundancy, minimizing the thrust norm and rate of thrust. The platform is non-morphing, which again introduces energy waste when for example in hovering mode.



(a) The omnidirectional hexa-rotor of [3]



(b) The omnidirectional octo-rotor of [4]

Figure 1.6: The two versions of a omnidirectional platform of [3, 4]

The third design is given by [5], which is a hexa-rotor with two propellers per arm, giving more thrust force. The platform is shown in Fig. 1.7. Each arm is individually tiltable by a servo motor. The control is done by a linear quadratic regulator with integral action, which is an optimization-based reactive controller. The linearized error dynamics used in the controller are at jerk level, such that the dynamics of the rotating arms are accessible in the allocation. The mapping of the body jerk to the individual rotor commands is done by a differential actuator allocation and task prioritization in the null space. A limit of this method is the reactive nature of the controller, therefore not optimizing the inputs using a prediction along a specified horizon. Furthermore, the integration of constraints in actuator allocation was left for future work.

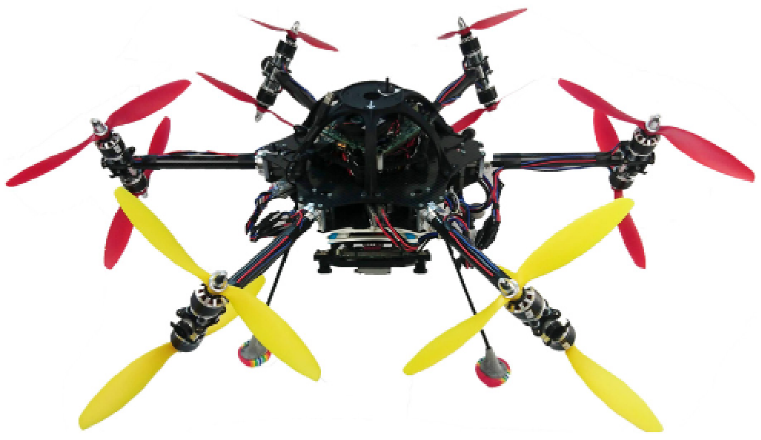


Figure 1.7: Actively tilting hexa-rotor omnidirectional platform of [5, 6].

Concurrent work, given by [6], uses the same model as [5] but implements NMPC for this platform. The NMPC generates the optimal body wrench rate. The wrench rate is used to impose costs and constraints on it. The allocation is performed by the pseudo inverse, which could produce infeasible inputs making the platform unstable. Using the derivative of the body wrench as inputs makes it possible to set limits on those inputs, but not the more realistic inputs, namely the propeller speeds. Having the allocation inside the NMPC would allow setting more realistic hardware limitations on the rotor speeds, which in theory, could give better performance.

The last design that relates to the topic of morphing is given by [7], where the first concept was given in [38]. In this work, a morphing hexa-rotor is designed, controlled and experimentally validated. The UAV is shown in Fig. 1.8. The platform can switch between under-actuated and fully actuated (not omnidirectional) by the use of one tilting motor. Control is done by a full-pose geometric control algorithm, where position tracking is prioritized. The controller consists of a high-level controller for morphing, a position controller, an attitude controller and a wrench mapper, using the pseudo-inverse of the static allocation matrix.

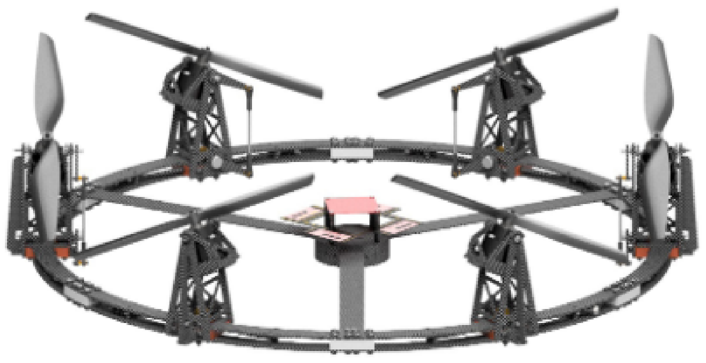


Figure 1.8: Actively tilting hexa-rotor platform of [7].

1.4 Thesis contributions

Based on the limitations of the previous works, the contribution of this thesis is to develop a control algorithm for a platform, that combines the characteristics of an omnidirectional morphing multi-rotor with NMPC and a realistic actuator model. In other words, NMPC will be used to control an omnidirectional platform, that can switch between UA and OA, taking into account energy efficiency automatically by the use of a cost function. Furthermore, the controller takes into account the realistic limits on the rotor velocities, rotor acceleration, tilting angle of the propellers and its tilting rate. This can be achieved by having the allocation inside the optimal control problem (OCP) of the NMPC.

1.5 Goal

The goal of the thesis is to extend the prior work of [9], which exploits the Matlab model predictive control (MATMPC) framework of [39]. The framework should control generically tilted MRAVs with actuator constraints, deal with the case of morphing platforms and, more specifically, control the OmniMorph. Furthermore, simulations need to be conducted to test the performance and influence of the controller.

1.6 Document outline

The outline of the rest of the report is the following. In the next chapter, the theoretical background is discussed. Then, the NMPC design for the OmniMorph is covered. In the fourth chapter, the results are given by the use of Matlab simulations. Chapter five gives a small comparison between the NMPC and the pseudo-inverse controller of [1]. Finally, a conclusion is given in chapter six.

2 Theoretical background

In the following chapter, a theoretical background is given on the modelling of the OmniMorph, possible orientation representations and their error definitions, the actuation of the UAV and finally NMPC.

Notation. Throughout the report, the following notation is used:

- Bold text represents vectors and matrices, with smaller and upper cases respectively.
- Upper case subscript indicates the frame the variable holds information of, whereas upper case superscript indicates the frame it is expressed in.
- Lower case subscript indicates a description, for example, the ith propeller force: f_i
- A rotation matrix $\mathbf{R}_{\bullet}^{\star}$ indicates a rotation from frame $\bullet$ to $\star$
- $[\cdot]_{\times}$ is the skew-symmetric matrix representation of a vector.
- $\otimes$ is the Hamilton product of two quaternions.
- $\times$ is the cross product of two vectors.
- $\cdot^{-1}$ indicates the inverse of a variable, vector or matrix.
- $(\cdot)^{V}$ is the vee map, mapping a rotation matrix in SO(3) to a vector in $\mathbb{R}^3$
- $\mathbf{I}_n$ is a $n \times n$ identity matrix, whereas $\mathbf{O}_n$ is a $n \times n$ zero matrix.
- The transpose of a vector or matrix is given by $[\cdot]^T$.
- SO(3) refers to the group that represents rotations, the special orthogonal group.
- $\mathbf{e}_3$ represents the unit vector $[0,0,1]^T \in \mathbb{R}^3$

2.1 Modelling

Because the NMPC depends highly on the model, it should be defined properly such that the control algorithm can predict close to the real world. A realistic graphical model of the OmniMorph is given in Fig. 2.1 indicating the important frames. In Fig. 2.2, an abstract schematic of the OmniMorph is given, showing the axes and numbers of the propellers. The red, green and blue axes represent the x,y and z-axis respectively. In table 2.1 the main symbols used throughout the paper are given. To model a multi-rotor, multiple frames should be defined. The first frame is the inertial world frame $\mathscr{F}_I = \{O_I, x_I, y_I, z_I\}$. The second frame is the body fixed frame, centred at the CoM of the platform, $\mathscr{F}_B = \{O_B, x_B, y_B, z_B\}$. The other frames are the ones coinciding with the CoM of the propellers and are defined as $\mathscr{F}_{Ai} = \{O_{Ai}, x_{Ai}, y_{Ai}, z_{Ai}\}$. The orientation of the world frame $\mathscr{F}_I$ is defined such that the z-axis is facing up against gravity, $\mathbf{g} = [0,0,-g]^T$. Furthermore, the orientation of the body frame $\mathscr{F}_B$ is such that the z-axis is facing towards the top of the platform. The orientation of the actuator frame $\mathscr{F}_{Ai}$ is such that the z-axis coincides with the direction of the thrust of the individual actuator when the platform is hovering with an orientation angle of zero degrees. The position, velocity and acceleration of the multi-rotor with respect to $\mathscr{F}_I$ in $\mathscr{F}_I$ are expressed as $\mathbf{p}_B^I, \dot{\mathbf{p}}_B^I, \ddot{\mathbf{p}}_B^I,$. The orientation of the UAV with respect to $\mathscr{F}_I$ can be written as a rotation matrix $\mathbf{R}_B^I \in SO(3)$, euler angles $\boldsymbol{\eta}_B^I \in \mathbb{R}^3$ or a quaternion $\mathbf{q} \in Q$. The angular velocity and acceleration of the body frame expressed in $\mathscr{F}_B$ are defined as $\boldsymbol{\omega}$, $\dot{\boldsymbol{\omega}}$ respectively. The position of an individual actuator w.r.t. $\mathscr{F}_B$ in $\mathscr{F}_B$ is defined as $\mathbf{p}_{Ai}^B$. Finally, the orientation of the propellers relative to the body frame is given by $\mathbf{R}(\alpha)_{A_i}^B$. The

Table 2.1: Overview of the main symbols used in this paper.

Definition	**Symbol**	**Set**
Inertial World Frame with origin O_I and orthogonal axes $\{x_I, y_I, z_I\}$	$\mathscr{F}_I$	
Multi-rotor Body Frame with origin O_B and orthogonal axes $\{x_B, y_B, z_B\}$	$\mathscr{F}_B$	
Frame of i-th actuator with origin O_{A_i} and orthogonal axes $\{x_{A_i}, y_{A_i}, z_{A_i}\}$	$\mathscr{F}_{A_I}$	
Position, velocity and acceleration of $\mathscr{F}_B$ w.r.t $\mathscr{F}_I$ expressed in $\mathscr{F}_I$	$\mathbf{p}_B^I, \dot{\mathbf{p}}_B^I, \ddot{\mathbf{p}}_B^I$	$\in \mathbb{R}^3$
Position of the ith actuator expressed in $\mathscr{F}_B$	$\mathbf{p}_{A_i}^B$	$\in \mathbb{R}^3$
Rotation from frame $\mathscr{F}_B$ to frame $\mathscr{F}_I$	$\mathbf{R}_B^I$	$\in \mathbb{R}^{3\times3}$
Rotational rate of frame $\mathscr{F}_B$ w.r.t frame $\mathscr{F}_I$	$\dot{\mathbf{R}}_B^I$	$\in \mathbb{R}^{3\times3}$
Quaternion $[q_x, q_y, q_z, q_w]$	$\mathbf{q}$	$\in \mathbb{Q}^4$
Quaternion rate	$\dot{\mathbf{q}}$	$\in \mathbb{Q}^4$
Rotation from frame $\mathscr{F}_{A_i}$ to frame $\mathscr{F}_B$	$\mathbf{R}_{A_i}^B$	$\in \mathbb{R}^{3\times3}$
Translational body wrench in body frame	$\mathbf{f}_B^B$	$\in \mathbb{R}^3$
Rotational body wrench in body frame (torque)	$\boldsymbol{\tau}_B^B$	$\in \mathbb{R}^3$
Inertia matrix of the OmniMorph in body frame	$\mathbf{J}$	$\in \mathbb{R}^{3\times3}$
Minimal rotation representation, Euler angles, of $\mathscr{F}_B$ relative to $\mathscr{F}_I$	$\boldsymbol{\eta}_B^I$	$\in \mathbb{R}^3$
Angular rate of frame $\mathscr{F}_B$ w.r.t frame $\mathscr{F}_I$	$\dot{\boldsymbol{\eta}}_B^I$	$\in \mathbb{R}^3$
Angular velocity of the platform in frame $\mathscr{F}_B$	$\boldsymbol{\omega}_B^B$	$\in \mathbb{R}^3$
Angular acceleration of the platform in frame $\mathscr{F}_B$	$\dot{\boldsymbol{\omega}}_B^B$	$\in \mathbb{R}^3$
The servo angle, tilting the propellers	α	$\in \mathbb{R}$
The static allocation matrix, dependent on α	$\mathbf{G}(\alpha)$	$\in \mathbb{R}^{6\times8}$
Propeller thrust force vector	$\boldsymbol{\lambda}$	$\in \mathbb{R}^8$
The rotor velocity of a propeller	ω	$\in \mathbb{R}$
The gravitational force	g	$\in \mathbb{R}$

servo can impose a change in the orientation of the individual propellers, where the rotation angle α is in the positive direction around the x-axis of the propeller frame.

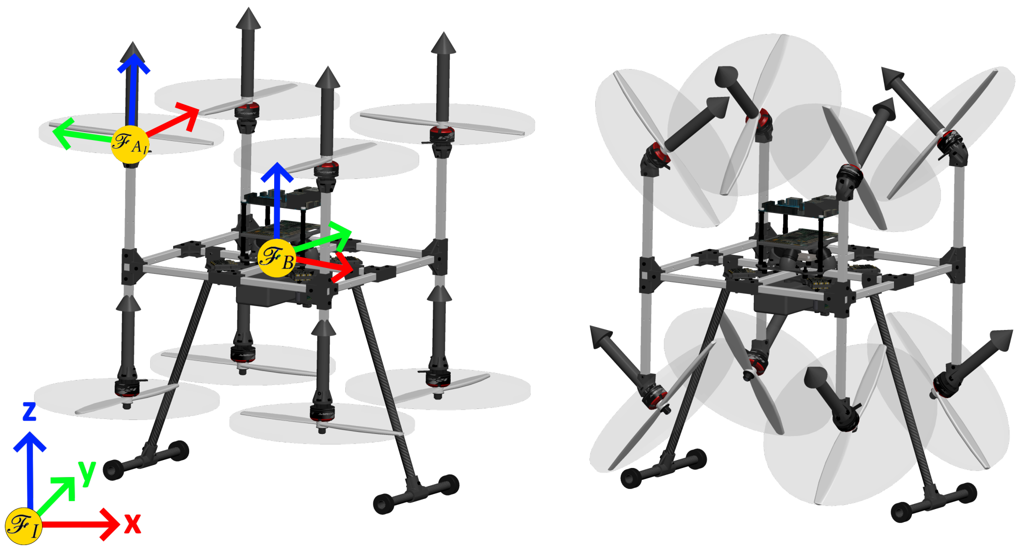


Figure 2.1: Schematic of the OmniMorph indicating the frames with its orientations, on the left in uni-directional thrust (UDT) mode. On the right in multidirectional thrust (MDT) mode.

The dynamics of the OmniMorph are shown in equation 2.1 using the Newton-Euler formalism. The translational dynamics are in the inertial frame $\mathscr{F}_W$, whereas the rotational dynamics are expressed in body frame $\mathscr{F}_B$. The mass of the platform is defined as m. $\mathbf{I}_3$ and $\mathbf{O}_3$ are a 3 by

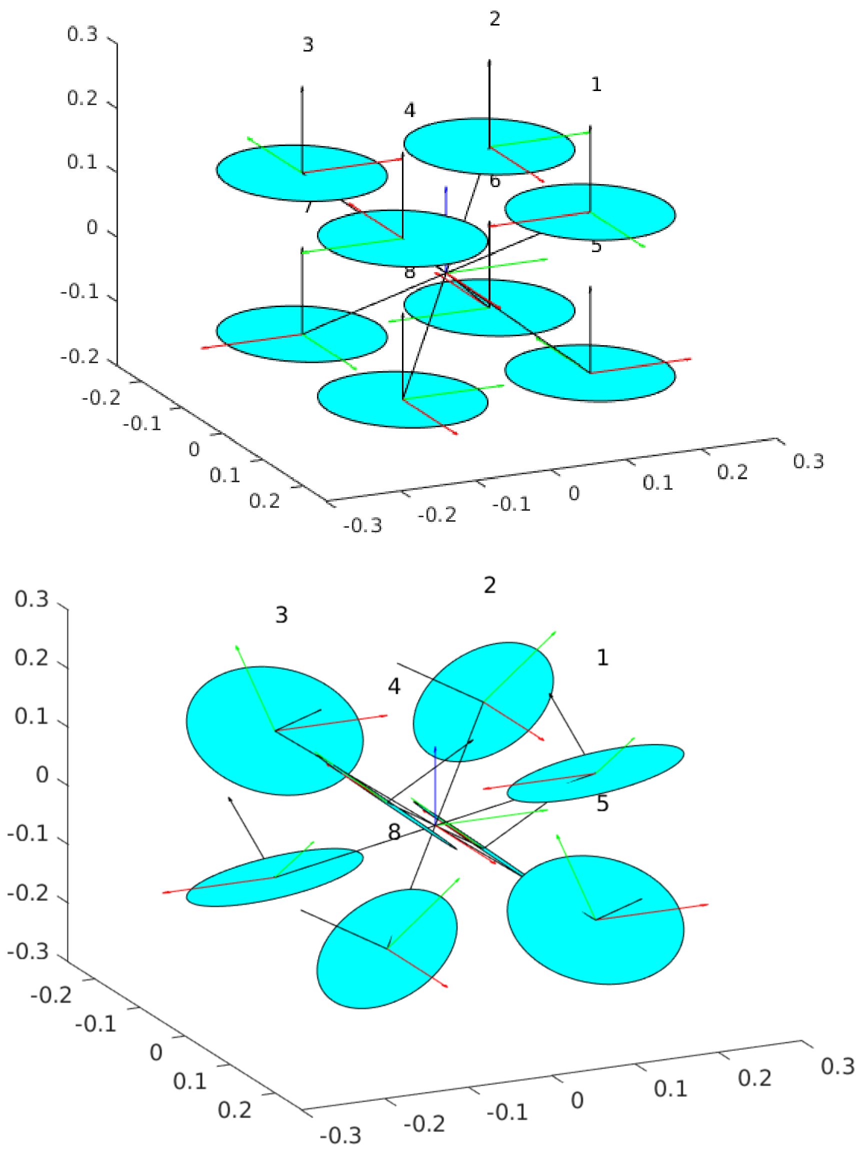


Figure 2.2: Abstract schematic of the OmniMorph indicating body frame and propeller frames. On the top, the servo angle is zero, whereas on the bottom the servo angle is set to 50 degrees.

3 identity and zero matrix respectively. $\mathbf{J}$ is the rotational inertia matrix of the platform. $\mathbf{e}_3$ is the unit vector pointing in the positive z direction. Finally, $\mathbf{f}_B^B$ and $\boldsymbol{\tau}_B^B$ are the translational and rotational part of the body wrench in $\mathscr{F}_B$.

$$\begin{bmatrix} \ddot{\mathbf{p}}_B^I \\ \dot{\boldsymbol{\omega}}_B^B \end{bmatrix} = \begin{bmatrix} m\mathbf{I}_3 & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{J} \end{bmatrix}^{-1} \left(\begin{bmatrix} -mg\mathbf{e}_3 \\ -\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} \end{bmatrix} \right) + \begin{bmatrix} \mathbf{R}_B^I & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \mathbf{f}_B^B \\ \boldsymbol{\tau}_B^B \end{bmatrix} \tag{2.1}$$

2.2 Orientation representations

The orientation kinematics of the platform can be defined by different representations. One possibility is a minimal angle representation, for example, Euler angles. Other representations can be quaternions or rotation matrices. In this section, the different representations will be covered.

2.2.1 Minimal representation angle

A minimal angle representation can be for example Euler angles, defined as $\boldsymbol{\eta} = [\phi, \theta, \psi]$. The relation of the derivative of $\boldsymbol{\eta}$ and the angular velocity ω is shown in equation 2.2. For Euler angles the order of rotation is important and can be for example the well-known roll-pitch-yaw (RPY) sequence. The matrix **T** for this specific order is shown in equation 2.3. Because the rotation is order dependent, Euler angles have a singularity called the Gimbal Lock, i.e., in specific orientations, two rotation axis are aligned and one degree of freedom is lost. With RPY this would happen when we have a pitch angle of 90 degrees. Therefore, this orientation representation is not sufficient for the OmniMorph, when omnidirectional capabilities are wanted.

$$\boldsymbol{\omega}_B^I = \mathbf{T} \cdot \dot{\boldsymbol{\eta}} \tag{2.2}$$

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & -sin(\theta) \\ 0 & cos(\phi) & sin(\phi)cos(\theta) \\ 0 & -sin(\phi) & cos(\phi)cos(\theta) \end{bmatrix} \tag{2.3}$$

When Euler angles are used in reference tracking, the error definition is simply defined by the difference between the real orientation and the reference, shown in equation 2.4. It needs to be noted that, this error definition is not geometrically consistent but sufficient for small rotations.

$$\mathbf{e}_\eta = \boldsymbol{\eta}_m - \boldsymbol{\eta}_r \tag{2.4}$$

2.2.2 Rotation matrices

Another way to represent orientation is rotation matrices. A rotation of the platform w.r.t. the world frame would be defined as $\mathbf{R}_B^I \in \mathbb{R}^{3\times3}$. The relation to the angular velocity is given in equation 2.5, where $[\cdot]_\times$. is the skew-symmetric matrix representation. The error between the real orientation $\mathbf{R}_B$ and the reference $\mathbf{R}_r$ can not simply be the difference, because the rotation matrices are not at the same place on the manifold SO(3). One possible definition of the error is shown in equation 2.6, where $(\cdot)^V$ is the vee map from SO(3) to $\mathbb{R}^3$. The advantage of rotation matrices is the absence of singularities. However, a disadvantage could be the computational load in an OCP, due to the use of 9 variables for a rotation.

$$\dot{\mathbf{R}}_B^I = \mathbf{R}_B^I [\boldsymbol{\omega}_B^I]_\times \tag{2.5}$$

$$\mathbf{e}_R = \frac{1}{2}(\mathbf{R}_B^T \mathbf{R}_r - \mathbf{R}_r^T \mathbf{R}_B)^V \tag{2.6}$$

2.2.3 Quaternions

The last representation that is covered is quaternions. The definition of a unit quaternion is given in 2.7, where q_w is the scalar part and $q_{x,y,z}$ is the vector part. Its relation with the rotational velocity in body frame ω_B is shown in 2.8, where $\otimes$ is the Hamilton product of two quaternions.

$$\mathbf{q} = [q_x, q_y, q_z, q_w]^T \tag{2.7}$$

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

For the error, it is not possible to use the difference between two quaternions. In equations 2.9 and 2.10 two possible error definitions are given. The first uses the Hamilton product of the model quaternion and the reference inverted. The product results in another quaternion. The second is using the dot product of two quaternions and resulting in a scalar error. The advantage of the Hamilton product is, that the error can be weighted differently for the three rotation axis, giving an order of priority to them in e.g., a cost function. For the dot product method, no distinction in error can be made between the rotation axis, because the result is a single value.

$$\mathbf{q}_{err} = \mathbf{q}_m \otimes \mathbf{q}_r^{-1} \tag{2.9}$$

$$e_q = 1 - |\mathbf{q}_m \cdot \mathbf{q}_r| \tag{2.10}$$

The general advantage of using quaternions is the fact of having no singularity. Furthermore, it only uses 4 variables to represent an orientation, making it computationally less intensive for OCPs, compared to rotation matrices. When needed, each representation can be converted from the other.

2.3 Actuation

The body wrench of the OmniMorph defined in the Euler Newton equations is the result of the individual propeller thrusts f_i. The total body thrust $\mathbf{f}_B^B$ is the sum of all propeller thrusts individually rotated to the body frame, shown in equation 2.11 where $\mathbf{e}_3 = [0,0,1]^T$ indicates the direction of the propeller thrust in the propeller frame. $\mathbf{R}(\alpha)_{A_i}^B$ is the rotation matrix dependent on the tilt angle α mapping the thrust forces to the body frame. A schematic of how the tilting works for a propeller is given in Fig. 2.3, where the black disk represents the base of the platform and the smaller black cylinder a servo motor, that tilts radially.

$$\mathbf{f}_B^B = \sum_{i=1}^{8} \mathbf{f}(\alpha)_i^B = \sum_{i=1}^{8} \mathbf{R}(\alpha)_{A_i}^B \mathbf{e}_3 f_i \tag{2.11}$$

Next, the body torque $\boldsymbol{\tau}_B^B$ is the result of drag torques τ_{d_i} caused by the propellers and the moment of the propeller thrusts τ_{f_i} on the centre of mass. The definition is shown in equation 2.12, where $\times$ is the cross product of two vectors. c_i is -1 or 1 depending on the direction of induced drag of the rotor, which can be clockwise or anti-clockwise. c_f^{τ} is the ratio between the thrust and drag produced by the propeller. This constant is dependent on the characteristics of the rotor.

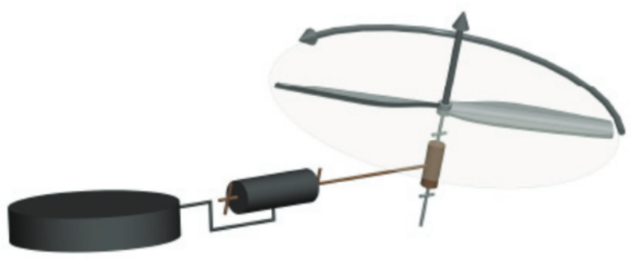


Figure 2.3: A schematic representation of a propeller that can tilt radially. [8]

$$\begin{aligned}\boldsymbol{\tau}_B^B &= \sum_{i=1}^{8} \tau_f(\alpha)_i^B + \tau_d(\alpha)_i^B = \sum_{i=1}^{8} \mathbf{p}_{A_i}^B \times \mathbf{f}(\alpha)_i^B + c_i c_f^\tau \mathbf{f}(\alpha)_i^B \\ &= \sum_{i=1}^{8} ([\mathbf{p}_{A_i}^B]_\times + c_i c_f^\tau \mathbf{I}_3)\mathbf{R}(\alpha)_{A_i}^B \mathbf{e}_3 f_i\end{aligned} \tag{2.12}$$

Combining equation 2.11 and 2.12 gives the map from individual propeller forces to the body wrench. This map is called the static allocation matrix $\mathbf{G}(\alpha)$ and is shown in equations 2.13, where $\boldsymbol{\lambda} = [f_1...f_8]^T$ are the individual propeller forces. For the complete static allocation matrix the reader is invited to appendix A.

$$\begin{bmatrix}\mathbf{f}_B^B \\ \boldsymbol{\tau}_B^B\end{bmatrix} = \mathbf{G}(\alpha)\boldsymbol{\lambda} = \begin{bmatrix}\mathbf{R}(\alpha)_{A_1}^B \mathbf{e}_3 & \dots & \mathbf{R}(\alpha)_{A_8}^B \mathbf{e}_3 \\ ([\mathbf{p}_{A_1}^B]_\times + c_1 c_f^\tau \mathbf{I}_3)\mathbf{R}(\alpha)_{A_1}^B \mathbf{e}_3 & \dots & ([\mathbf{p}_{A_8}^B]_\times + c_8 c_f^\tau \mathbf{I}_3)\mathbf{R}(\alpha)_{A_8}^B \mathbf{e}_3\end{bmatrix}\boldsymbol{\lambda} \tag{2.13}$$

For the control inputs, it is assumed that there is a low-level control of the actuators performed by the Electronic Speed controllers (ESCs) of the brushless DC motors. This low-level control is done by an adaptive bias and adaptive gain (ABAG) algorithm, introduced by [40]. Therefore, the real input of the propeller actuators is the rotor velocity. So, a map should be used that maps the rotor velocity to the thrust forces. In [30] an experimentally validated definition is given, which is shown in equation 2.14, where ω_i is the ith-rotor velocity in radians per second. c_f is the propeller force constant, depending on the properties of the individual propellers.

$$f_i = c_f \omega_i^2 \tag{2.14}$$

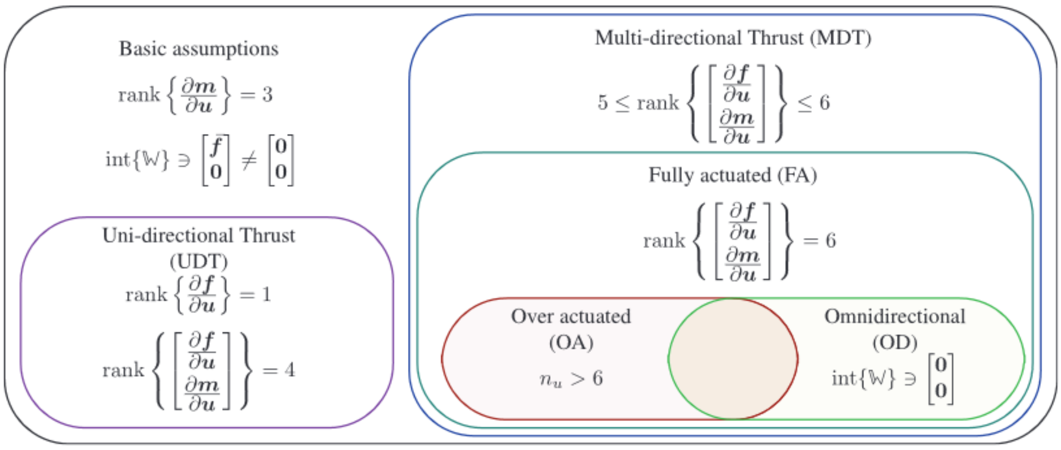
2.4 Admissible wrenches

The OmniMorph is able to switch between uni-directional thrust (UDT), see Fig. 2.1 on the left, and multi-directional thrust (MDT), see Fig. 2.1 on the right. When being in the UDT state, the platform is only able to move laterally by changing its orientation, which means it is under-actuated (UA). When being in an MDT state, the platform can change its position without changing its orientation, which corresponds to fully actuated (FA).

To show this mathematically, we can take a closer look at the full allocation matrix of the platform, which is for example done in [8] and will be repeated in short here. The static and full allocation matrices differ. The static allocation matrix relates the propeller thrust forces to the total body wrench, having a constant servo angle. The full allocation matrix relates all inputs to the body wrench, which is in our case, the propeller thrusts derivatives and the servo angle rates to the change in the body wrench. A shorthand notation of the full allocation matrix $\mathbf{F}(\mathbf{u})$ is given in equation 2.15. In this equation, $\mathbf{w}$ is the total body wrench, and $\mathbf{u}$ is the total input vector including the propeller thrusts and servo angle. Next, $\mathbf{f}$ and $\boldsymbol{\tau}$ are the force and moment part of the body wrench respectively. Then, $\mathbf{u}_\lambda$ are the control inputs related to the propeller thrusts and $\mathbf{u}_V$ is the control input related to the servo angle. Finally, $\mathbf{F}_1(\mathbf{u})$ and $\mathbf{F}_2(\mathbf{u})$ are the force and moment allocation matrices respectively.

$$\mathbf{F}(\mathbf{u}) = \frac{\delta \mathbf{w}}{\delta \mathbf{u}} = \begin{bmatrix}\frac{\delta \mathbf{f}}{\delta \mathbf{u}_\lambda}(\mathbf{u}_V) & \frac{\delta \mathbf{f}}{\delta \mathbf{u}_V}(\mathbf{u}) \\ \frac{\delta \boldsymbol{\tau}}{\delta \mathbf{u}_\lambda}(\mathbf{u}_V) & \frac{\delta \boldsymbol{\tau}}{\delta \mathbf{u}_V}(\mathbf{u})\end{bmatrix} = \begin{bmatrix}\mathbf{F}_1(\mathbf{u}) \\ \mathbf{F}_2(\mathbf{u})\end{bmatrix} \tag{2.15}$$

To analyse the controllability and understand when the platform is under-actuated or fully actuated, we can look at the rank of the full allocation matrix. When the servo angle α is zero and constant, the rank is 4 and the platform is UA. When the servo angle is non-zero, the rank is 6 and the platform is FA. Furthermore, when the rank of the OmniMorph is 6, and the platform can produce thrusts in all directions greater than its mass times the gravity, it is omnidirectional. Omnidirectional is a subset of fully actuated platforms. A simple overview of the thrust-related properties regarding the full allocation matrix and its rank is shown in Fig. 2.4, obtained from [8]. Note that in this overview **m** represents the body total moment, which is equal to our notation $\boldsymbol{\tau}$.



Basic assumptions
$\text{rank}\left\{\frac{\partial m}{\partial u}\right\} = 3$
$\text{int}\{\mathbb{W}\} \ni \begin{bmatrix} f \\ 0 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
Uni-directional Thrust (UDT)
$\text{rank}\left\{\frac{\partial f}{\partial u}\right\} = 1$
$\text{rank}\left\{\begin{bmatrix} \frac{\partial f}{\partial u} \\ \frac{\partial m}{\partial u} \end{bmatrix}\right\} = 4$
Multi-directional Thrust (MDT)
$5 \leq \text{rank}\left\{\begin{bmatrix} \frac{\partial f}{\partial u} \\ \frac{\partial m}{\partial u} \end{bmatrix}\right\} \leq 6$
Fully actuated (FA)
$\text{rank}\left\{\begin{bmatrix} \frac{\partial f}{\partial u} \\ \frac{\partial m}{\partial u} \end{bmatrix}\right\} = 6$
Over actuated (OA)
$n_u > 6$
Omnidirectional (OD)
$\text{int}\{\mathbb{W}\} \ni \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Figure 2.4: Overview of thrust-related properties regarding the static allocation matrix. [8]

2.5 Non-Linear Model Predictive Control

The control method used in this thesis is Non-Linear Model Predictive Control (NMPC). NMPC is a predictive algorithm based on a mathematically defined model of the UAV, which was defined in the previous sections. The main advantage of NMPC is that it takes into account constraints and optimizes the control input at the current time step, by keeping future time steps into account. In Fig. 2.5 a plot is given that demonstrates this. The controller is given a reference trajectory that needs to be followed. Using the prediction model of the UAV, predicted control inputs are calculated that follow the reference trajectory as close as possible in the prediction horizon, starting from time k until time k+N, where N is the number of time steps of the horizon with a specified resolution. Then, the first predicted control input is applied at time k to the real UAV. Before the same procedure is done at time step k+1, the predicted state at k+1 is updated to the new measured current state. Now again predicted control inputs are computed obtaining the best possible predicted outputs and only the first control input is again fed to the platform etc. To obtain the inputs that give the best tracking results, an objective function is used. This objective function will be given in the next chapter.

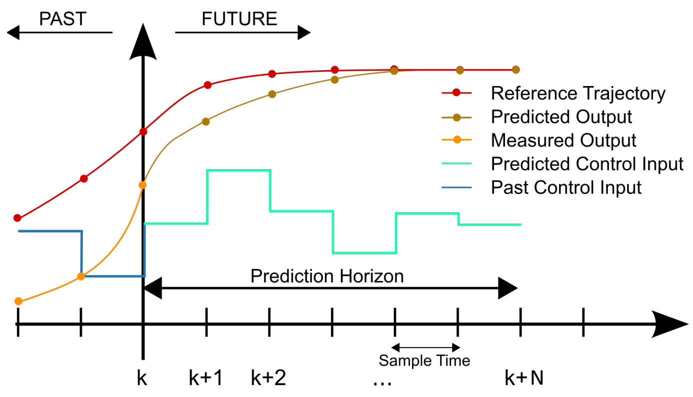


Figure 2.5: Graphical plot[15] of the NMPC prediction horizon, reference trajectory, outputs and inputs.

[15]https://en.wikipedia.org/wiki/Model_predictive_control

3 NMPC design for the OmniMorph

Based on the theoretical background the NMPC controller can be designed. In the following chapter, the design is explained. The design consists of the objective function, reference vector, output vector, states vector, weights and finally the constraints. Fig. 3.1 gives an abstract overview of the designed system.

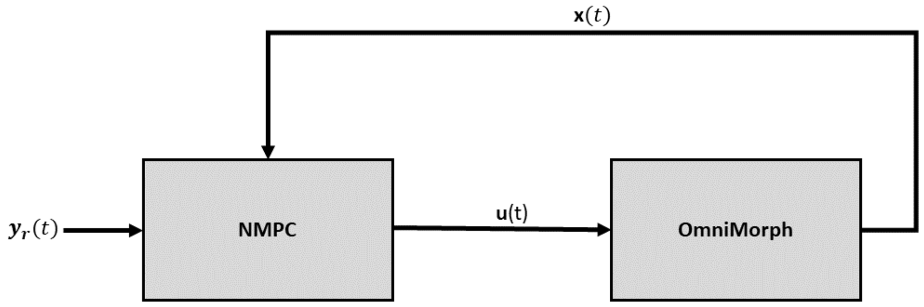


Figure 3.1: Block diagram of the NMPC, including the simulation model.

3.1 Objective function

To obtain the predicted control inputs, such that the platform follows the reference trajectory as closely as possible in the future prediction horizon, an optimization algorithm is used. This algorithm tries to find the arguments that minimize the cost of a cost function, see equation 3.1. $\mathbf{y}_h$ are the first parts of the cost terms of the model, also called the output vector which comes from the prediction model and state vector $\mathbf{x}$. Furthermore, $\mathbf{y}_r$ is the reference vector that the platform should follow. Next, is the input vector $\mathbf{u}$ that consists of the realistic inputs of the platform, namely the propeller thrusts and servo angle. Then, $\mathbf{Q}$ is the matrix consisting of the relative weights of the terms in the cost function, that enables us to prioritize certain terms over others. Finally, $\mathbf{M}$ is defined in order to select only the actuator forces from the state vector $\mathbf{x}$. The optimization algorithm also keeps into account defined constraints on the input u and output x of the system. Which are the control inputs and states of the UAV. All vectors mentioned above will be explained in the following sections.

$$
\begin{aligned}
\min_{\mathbf{x}_0\ldots\mathbf{x}_N,\mathbf{u}_0\ldots\mathbf{u}_{N-1}} \sum_{h=0}^{N} \{||\mathbf{y}_h-\mathbf{y}_{r,k+h}||^2_{\mathbf{Q}_h}+||\mathbf{u}_h||^2_{\mathbf{R}_h}\}+||\mathbf{y}_N-\mathbf{y}_{r,k+N}||^2_{\mathbf{Q}_N} \\
\text{s.t. } \mathbf{x}_0=\mathbf{x}_k \\
\mathbf{x}_{h+1}=\boldsymbol{\phi}(\mathbf{x}_h,\mathbf{u}_h),\ h=0,1,...,N-1 \\
\mathbf{y}_h=\mathbf{h}(\mathbf{x}_h,\mathbf{u}_h),\ h=0,1,...,N \\
\underline{\lambda}\le\mathbf{M}\mathbf{x}_h\le\overline{\lambda},\ h=0,1,...,N \\
\underline{\dot{\lambda}}_{k+h}\le\mathbf{u}_h\le\overline{\dot{\lambda}}_{k+h},\ h=0,1,...,N
\end{aligned}
\tag{3.1}
$$

3.2 Control input

The control input that is generated by the NMPC can be in several forms. Previous literature has used for example the total body wrench as generated control signals. However, this introduces fictitious inputs, i.e., those inputs are not the realistic inputs of the platform. A more realistic input of the system is the input to the individual propellers, as mentioned in the theoretical background, the rotor velocity. An ABAG algorithm within the ESCs is used for the low-level control of the rotational velocity. Therefore, it could give better stability and behaviour if the rotor velocity, or propeller thrust (using equation 2.14), would be the generated control signals.

However, to have the prediction model even closer to reality, we want to have constraints on the acceleration of the rotors, or the propeller thrust derivatives, using the relation between rotor velocity and thrust. Therefore, we choose to have the output of the NMPC generate the derivative of the propeller thrusts, which enables the ability to set constraints on them and take those into account in the control algorithm. Those generated control signals are then integrated and fed into the ABAG algorithm of the ESCs. A schematic overview of this implementation is given in Fig. 3.2. Having constraints on those specific inputs possibly gives better stability and behaviour.

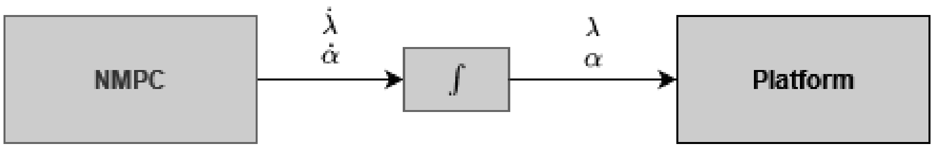


Figure 3.2: Overview of the generated inputs by the NMPC and the connection to the simulated platform, using integration.

3.3 Reference vector

Next, let us define the reference vector $\mathbf{y}_r$. The platform should follow trajectories that consist of linear and angular motion and at the same time keep the platform stable. We assume that the reference is twice continuously differentiable. Furthermore, we want to have the derivatives of the states follow the defined references. Next, because the platform is able to switch from UDT mode, being energy efficient, and MDT mode, but more energy-consuming, we also want the platform to stay energy efficient whenever possible. Therefore, we introduce an energy reference, which is defined as the total thrust that is needed to hover, equally divided over the individual propellers, then squared and multiplied by 8. This equation is equal to the dot product of the propeller forces when they equally contribute against gravity. The reference vector is shown in equation 3.2, where the energy term is shown in 3.3. The energy reference could also be set to zero. However, the platform always needs to counteract its own weight. Another possibility could be to use the real voltages used during flight. However, for our design, the voltage is not part of the state and is not possible to use. If voltage should be in the state, the prediction model should have more detail regarding the motor models, but this introduces more computational load. However, more detailed implementations could be an interesting topic for future work.

The reference vector consists of the position, velocity, acceleration, quaternions, angular velocity, angular acceleration and energy. All variables except for the energy are changing over time. Because they are varying over time, the variable should also be properly sampled in the prediction horizon. In other words, over the prediction horizon, the reference signals are also changing like they would in real-time. This enables better tracking compared to having the reference signals constant in the prediction horizon because the optimization is done by the correct reference samples. For example, if agile movements are part of the reference, such as a quick change in movement from left to right, the controller would be performing better when it knows in advance the switching in direction.

$$\mathbf{y}_r(t) = \begin{bmatrix} \mathbf{p}_r(t) \\ \dot{\mathbf{p}}_r(t) \\ \ddot{\mathbf{p}}_r(t) \\ \mathbf{q}_r(t) \\ \boldsymbol{\omega}_r(t) \\ \dot{\boldsymbol{\omega}}_r(t) \\ E_r \end{bmatrix} \tag{3.2}$$

$$E_r = (\frac{mg}{8})^2 \cdot 8 \tag{3.3}$$

3.4 State vector

Secondly is the state vector. The current states are obtained from the platform, whereas the prediction states are gathered from the prediction model. The state vector is shown in equation 3.4. The vector consists of the position, velocity, orientation, angular velocity and the real inputs of the system, namely the servo angle and propeller thrusts.

$$\mathbf{x} = [\mathbf{p}^T, \dot{\mathbf{p}}^T, \mathbf{q}^T, \boldsymbol{\omega}^T, \alpha, \boldsymbol{\lambda}^T]^T \tag{3.4}$$

3.5 Output vector

Next is the output vector of the system. The output vector together with the reference is used to compute the error in tracking. This means the output vector should consist of the same elements as the reference vector. The definition is shown in equation 3.5. The elements in the vector are calculated by the prediction model, using the inputs and states coming from this model. The position, velocity, orientation and angular velocity are obtained from the state vector. The acceleration and angular acceleration are calculated using the inputs and states of the model. Finally, the energy term is obtained by the dot product of the propeller thrusts. Those propeller thrusts are also obtained from the state vector.

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t)) = \begin{bmatrix} \mathbf{p}(t) \\ \dot{\mathbf{p}}(t) \\ \ddot{\mathbf{p}}(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{q}(t) \\ \boldsymbol{\omega}(t) \\ \dot{\boldsymbol{\omega}}(\mathbf{x}(t), \mathbf{u}(t)) \\ \boldsymbol{\lambda}(t)\boldsymbol{\lambda}^T(t) \end{bmatrix} \tag{3.5}$$

3.6 Weights

Within the cost function, the individual error terms can be weighted relatively. For example, the position error can be weighted more, relative to the attitude error. This will affect the behaviour of the controller accordingly. The determination of the weights depends highly on the wanted behaviour in specific scenarios, a starting point is given in table 3.1. For the translational and rotational terms, the weights are given in the order of x, y and z. The weights were obtained by some tuning and were used throughout the results unless specified differently.

3.7 Constraints

The big advantage of NMPC is the ability to take constraints into account. Constraints can be set on the states and inputs to the system coming from the NMPC.

In table 3.2 the constraints on the state and servo rate are given. It needs to be noted that those constraints are tunable, and the values were given at the specification of the thesis. The con-

Table 3.1: Relative weights of the error terms in the cost function. The weights are obtained after some tuning. The values are used in all simulations unless specified differently.

Weighted error term	**Value**
Position	$[30, 30, 30]$
Attitude	$[10, 10, 10]$
Translational velocity	$[1, 1, 1]$
Rotational velocity	$[1, 1, 1]$
Translational acceleration	$[1e^{-4}, 1e^{-4}, 1e^{-4}]$
Rotational acceleration	$[1e^{-4}, 1e^{-4}, 1e^{-4}]$
Energy	$1e^{-4}$
Thrust force derivative	$5e^{-5}$
Servo rate	$5e^{-2}$

straints on the position in x y and z are now set such that the platform stays within a cube. When travelling over large distances is needed, those constraints can also be removed. Or when moving through a very small hole is wanted, the constraints can be set accordingly. The constraints can also depend on time or be made dependent on other parameters if wanted. Constraints on the propeller forces, servo angle and servo rate are set because in reality there are limits on them as well. This can give better performance in tracking and stability.

Table 3.2: State constraints used during the experiments.

State	**Min value**	**Max value**
x	-5 m	5 m
y	-5 m	5 m
z	-5 m	5 m
f_{prop}	-6 N	6 N
α	0 deg	50 deg
$\dot{\alpha}$	-10 deg/s	10 deg/s

Finally, constraints are set on the derivative of the thrust forces, to be closer to reality. The constraints on the derivative of the thrust forces can be made dependent on the propeller speeds. This was, for example, done in [9], where they experimentally verified the acceleration constraints of the propellers. The derivative of the thrust forces is directly related to the rotor accelerations, shown in equation 3.6. This relation gives therefore also the relation between the constraints. In table 3.3, an example of the rotor acceleration constraints are given, with the values of [9]. However the acceleration limits of the propellers of the Omnimorph were not determined yet, therefore the acceleration limits are set to -300 Hz/s and 300 Hz/s for now. In order to improve the results, the experiments to determine the constraints should be done with the propellers of the OmniMorph, because they are normally not constant over the rotor velocity range.

$$\dot{f} = \frac{\partial f}{\partial \omega}\frac{\partial \omega}{\partial t} = 2c_f\omega\dot{\omega} \tag{3.6}$$

Table 3.3: Rotor acceleration constraints, given by [9].

ω [Hz]	30	40	50	60	70	80	90
$\underline{\dot{\omega}}$ [Hz/s]	-120	-160	-200	-140	-160	-160	-140
$\overline{\dot{\omega}}$ [Hz/s]	200	200	200	160	180	180	180

4 Matlab simulations

In the following chapter, the proposed NMPC design is implemented with Matlab and Simulink. Implementing the design first in such a tool is an important preliminary step for testing before deployment on the real robot, for example, running onboard with a lower-level implementation.

First, the simulation setup is given, including the hardware specifications used. Secondly, the implementation is covered, using a framework called MATMPC created by [39]. Then a comparison in performance is made between orientation representations, namely Euler angles, and quaternions. Next, robustness against model mismatches is examined. Afterward, the effect of energy consumption on the cost function is analyzed. Finally, the influence of the prediction horizon is investigated.

4.1 Simulation setup

For the implementation of the NMPC the goal is to extend the prior work of [9], which uses the MATMPC framework of [39], to be able to control the OmniMorph. The current implementation does not use quaternions, and therefore this needs to be changed within the framework, using the theory of the previous chapter. An overview of the implementation model is given in Fig. 4.1.

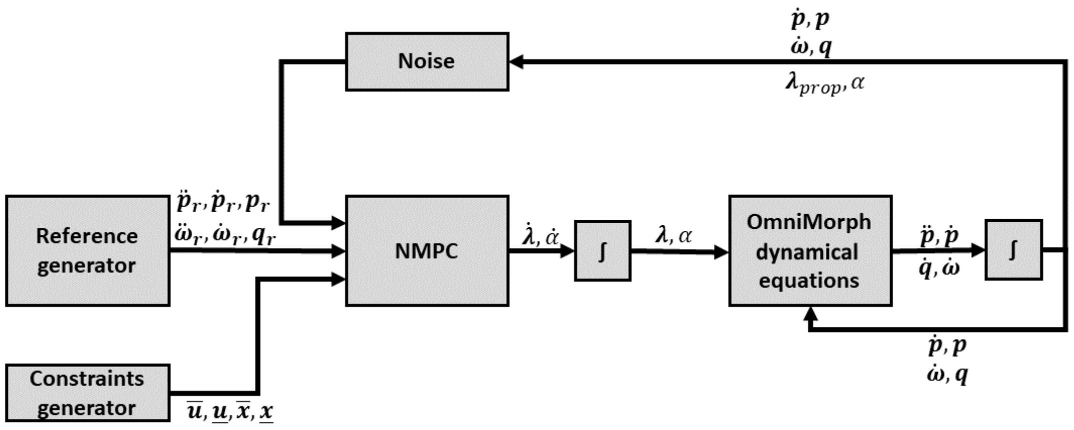


Figure 4.1: Block diagram of the NMPC simulation setup in Matlab. All signals represented in the diagram are dependent on time.

First, a reference trajectory is generated in the reference generator, where the reference consists of the position and orientation tracking, up to the second derivative. The reference is sampled properly along the prediction horizon, meaning it changes along the prediction. Next, the constraints are generated, consisting of the upper and lower bounds of the input and states. Then the NMPC block is shown, generating the control inputs, using the reference, constraints and states. The states coming from the simulation model are manipulated by adding noise, which is Gaussian white noise with zero mean. It needs to be noted, that the servo angle and propeller forces do not have noise added at the moment, however, this is possible to add, if wanted. The noise bounds are given in table 4.2, these values correspond to three times the standard deviation of the generated noise. All important model and controller parameters are shown in table 4.1.

Table 4.1: Model and controller parameters used in the thesis.

Symbol	Value	Definition
m	1 Kg	Total mass of the platform
I_x	0.042 kgm^2	Rotational inertia around the x-axis
I_y	0.042 kgm^2	Rotational inertia around the y-axis
I_z	0.083 kgm^2	Rotational inertia around the z-axis
l	0.2165 m	Distance of the propeller frame from the platform's centre of mass
c_f	$1.24e^{-4}$	Propeller's thrust coefficient.
c_t	$2.2e^{-6}$	Propeller's thrust coefficient.
N	15	Prediction horizon steps
Ts	$0.004s$	Simulation step size
Ts_{st}	$0.1s$	NMPC shooting step size

Table 4.2: Noise bounds of the Gaussian white noise with zero mean, used to add to the measured states.

Variable	Noise Bound
Position	0.005 m
Linear velocity	0.02 m/s
Orientation	2 deg
Angular velocity x and y	10 deg/s
Angular velocity z	5 deg/s

All simulations will be done on a laptop consisting of an intel core i7-10750H CPU with a clock speed of 2.60GHz and 12 threads, 16Gb of RAM and an Nvidia Quadro T1000 graphics card. The operating system is Linux Ubuntu 18.04, running Matlab R2021b.

4.1.1 MATMPC framework

The MATMPC framework of [39] consists of multiple algorithmic modules in order to solve a defined OCP, such as automatic differentiation (AD), direct multiple shooting, condensing, linear quadratic program (QP) solvers and globalization. The modules are compiled into MEX functions in order to obtain faster runtime performance than traditional Matlab packages.

First, the model is defined, including dynamical equations, inputs, outputs, constraints and the objective function. Together this forms the OCP. Next, this information is used to generate C code of the model's analytic functions and derivatives. Those derivatives are calculated using AD in CasADi[16].

Next, the OCP is discretized into a Nonlinear programming problem (NLP), using direct multiple shooting. For the approximation of the continuous trajectories of the system, efficient numerical integrators are used, such as explicit and implicit Runge-Kutta integrators.

In order to solve the NLP, sequential quadratic programming (SQP) methods are implemented. The sparse QP is converted into a (partial) dense QP using stable condensing algorithms. The QP is then solved by a solver such as qpOASES[17] [41]. This solution is then used to update the solution of the NLP. An alternative way is to terminate the SQP iteration before it converged. For many applications, it is sufficient to use only one iteration with a full Newton step. This is called Real-Time Iteration (RTI) given by e.g., [42, 43].

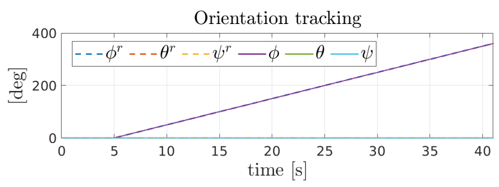
[16]https://web.casadi.org/
[17]https://github.com/coin-or/qpOASES

4.2 Orientation representations and its singularities

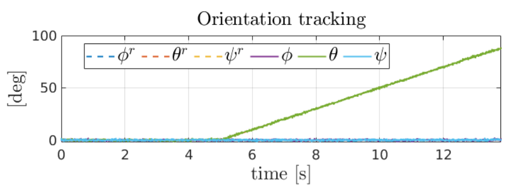
In this section, the first results are given. A comparison is made between two orientation representations inside the NMPC controller. This comparison is important to understand what orientation model should be used and is in line with research question 1, about the model. Because the OmniMorph is capable of full rotations around its body frame axis, a minimal representation of angles would not be sufficient. When, for example, using the RPY Euler angles, the controller will fail when a reference trajectory is given, of 90 degrees or more in pitch angle.

4.2.1 Euler

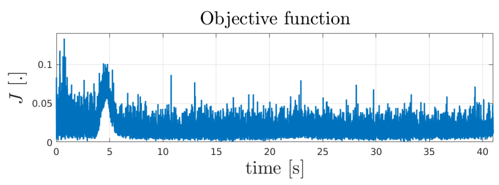
To show the singularity problem, let us define first a reference signal of 360 degrees around the x-axis (roll), this trajectory will be possible to follow. Next, a reference path will be given of 360 degrees around the y-axis (pitch), which introduces the problem. The reference is a simple ramp function in orientation angle with a slope of 10 degrees per second. The angular velocity is a constant of 10 degrees per second. The reference is not continuous, but also not necessary for showing the singularity issue. The results are shown in Fig. 4.2. On the left a successful roll is shown including the objective function plot, and on the right, a pitch scenario is plotted, that fails just before 90 degrees. This issue is caused due to the singularity at 90 degrees pitch for the specific RPY convention used in the proposed design. This singularity point is also called the Gimbal lock.



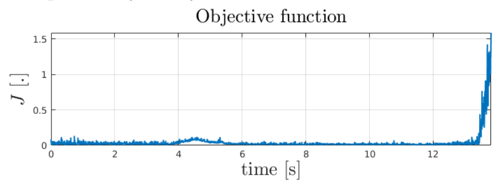
(a) A plot of the orientation tracking of the Omnimorph. With a roll trajectory.



(b) A plot of the orientation tracking of the Omnimorph. With a pitch trajectory.



(c) The cost of the objective function during the roll trajectory.



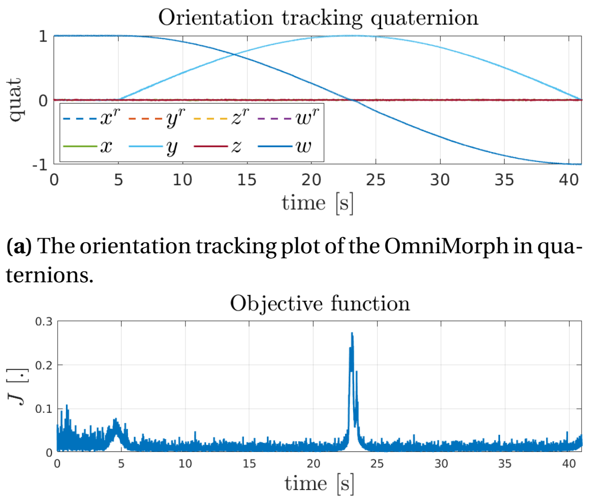
(d) The cost of the objective function during the pitch trajectory.

Figure 4.2: A comparison between a roll and pitch reference tracking with the OmniMorph. On the left, a roll turn is made. On the right, a pitch turn is performed, where the controller crashes.

4.2.2 Quaternion

Let us now track the same trajectory but using the quaternion representation inside the controller and simulation model. This will solve the Gimbal lock and enables the platform to smoothly rotate in any direction without crossing singularities that are due to a too-restricting attitude representation. Solving the Gimbal lock is important for an omnidirectional platform to keep its omnidirectional capabilities. The reference signal is again the same ramp function but converted to quaternions as shown in equation 4.1. The results are shown in Fig. 4.3, where the orientation plot shows the quaternion values. The plot can not be converted to, the more understandable, Euler angles because this gives an unclear interpretation of the trajectory due to the singularity.

$$\theta_r = 10t$$
$$\mathbf{q}_r = \begin{bmatrix} 0 & sin(\frac{\theta_r}{2}) & 0 & cos(\frac{\theta_r}{2}) \end{bmatrix}^T \tag{4.1}$$



(a) The orientation tracking plot of the OmniMorph in quaternions.

(b) The cost of the objective function during a pitch turn, using quaternions.

Figure 4.3: A successful pitch turn around the y-axis performed by the OmniMorph, using quaternions.

4.2.3 Discussion

As can be seen from the results, Euler angles fail to control the platform for a complete rotation around a certain axis. The objective function increases at the moment close to the 90 degrees pitch, and results in a crash of the controller. For the quaternion representation, the controller is able to control the OmniMorph and make a full rotation around the y-axis. Therefore, if the platform should be capable of full rotations, quaternions should be used instead of Euler angles. However, for trajectories with angles less than 90 degrees, Euler angles would be sufficient.

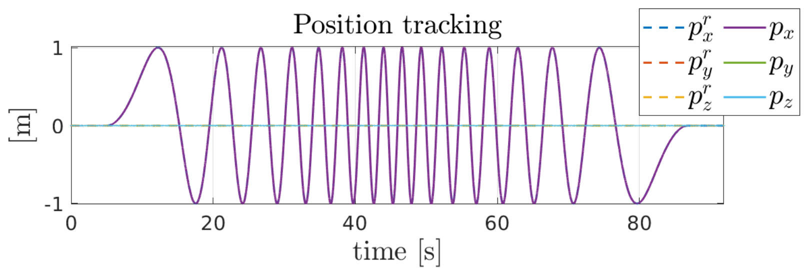
4.3 Robustness against model mismatches

The performance of NMPC is highly dependent on the model, therefore an analysis is given regarding mismatches in model parameters. This analysis is also in line with research question one, concerning the model. In order to understand the effect of the mismatch, the following parameters will be examined individually: (1) mass, (2) inertia, (3) the thrust coefficient, and finally (4) the drag coefficient. The mismatch is introduced in the prediction model inside the controller by adding or removing a certain percentage from the original value. The tracking error is then plotted in RMSE against the percentage of mismatch.

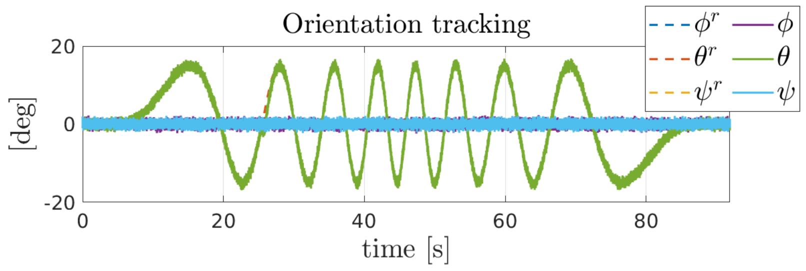
Let us define a reference trajectory of a chirp in x translation and a chirp in y rotation. For the orientation quaternions are used. A chirp is used, to show the capabilities and stability of the platform with fast-changing movements, this can be necessary with, for example, operator interactions. The trajectory is shown in Fig. 4.4, where the orientation is plotted in Euler angles, by transforming the quaternions after simulation. Transforming the quaternions is possible in this situation because the angles stay below 90 degrees. The amplitudes are one meter and 15 degrees for the translation and rotation respectively. The maximum velocity for the translation is 2.46 m/s and for the rotation 27.29 deg/s. The maximum acceleration for the translation is 5.85 m/s^2 and for the rotation 21.25 deg/s^2. The equations for the references are given in equation 4.2 and 4.3. The equations only represent the reference from time 5 seconds till 45.9331 seconds, which is the increasing chirp part. After this time, the trajectory is the same but decreases in chirp frequency, so mirrored in time.

$$\begin{aligned} p_{x_r} &= sin(0.03t^2) \\ \dot{p}_{x_r} &= 0.03 \cdot 2t \cdot cos(0.03t^2) \\ \ddot{p}_{x_r} &= 0.03 \cdot 2 \cdot cos(0.03t^2) - 4 \cdot (0.03t)^2 \cdot sin(0.03t^2) \end{aligned} \tag{4.2}$$

$$\begin{aligned} \theta_{y_r} &= 15 \cdot sin(0.015t^2) \\ \omega_{y_r} &= 0.015 \cdot 2 \cdot 15t \cdot cos(0.015t^2) \\ \dot{\omega}_{y_r} &= 0.015 \cdot 2 \cdot 15 \cdot cos(0.015t^2) - 4 \cdot 15 \cdot (0.015t)^2 \cdot sin(0.015t^2) \end{aligned} \tag{4.3}$$



(a) The position tracking plot of a chirp reference, done by the OmniMorph.



(b) The orientation tracking plot of a chirp reference, done by the OmniMorph.

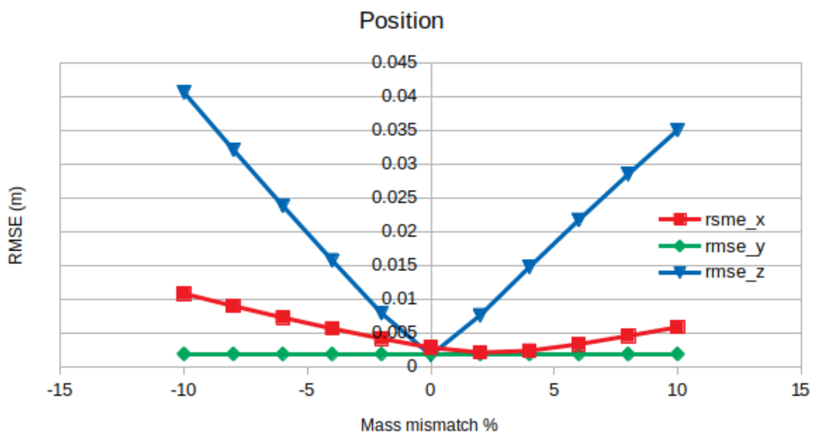
Figure 4.4: A chirp trajectory in position and orientation, performed by the OmniMorph, without any mismatch in parameters.

4.3.1 Mass mismatch

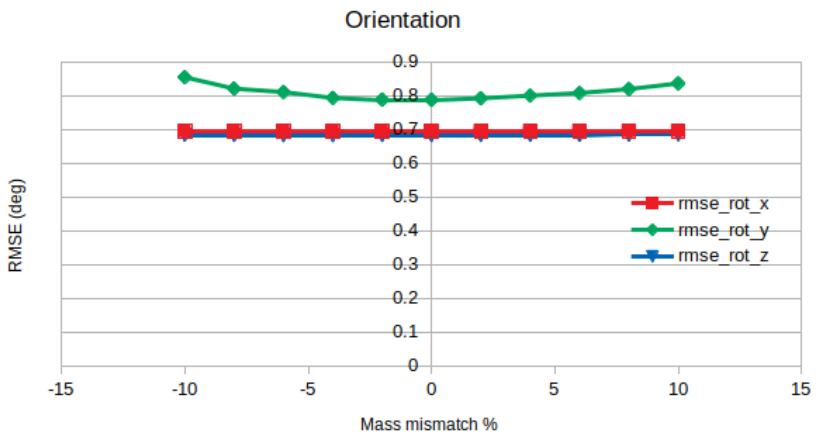
The first mismatch is done in the mass, which is originally set to 1.0 kg. The results for the position and rotational RMS errors are given in Fig. 4.5. The mismatch inside the controller ranges from -10 per cent to 10 per cent, meaning from 0.9 kg to 1.1 kg.

From the results, it can be seen that the mass mismatch mostly influences the RMSE of the z position. Which has a linear shape relation. In Fig. 4.6, the result in position and rotation error are given for a mismatch of 10 per cent. In those plots, it is visible that the mismatch in mass gives an offset in the z-axis error. Such a constant error could be removed by introducing dynamic model variables in the controller, that automatically adapt by the use of, for example, a learning or optimization algorithm.

Next to the error in z, also errors are introduced in the x position and y rotation. This comes from the fact that the reference trajectory in the x position and y rotation changes over time, whereas the other axes are kept constant. To give a more practical direction the results could give an indication, of to what degree a mismatch in the mass is reasonable, such that it stays within safe operating ranges. For example, if a safe operating error margin allows a maximum of 1 centimetre in position for this specific scenario, the mismatch should not be more than approximately 2.5 per cent.

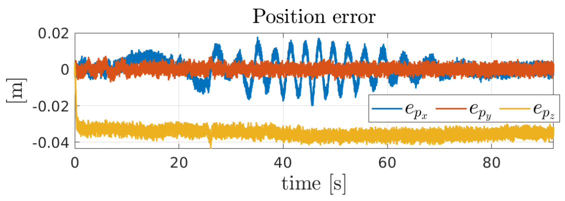


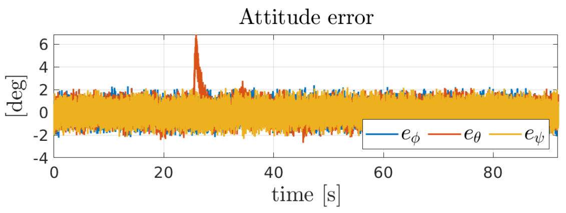
(a) Plot of the root mean square error of the position tracking against a mismatch in the mass.



(b) Plot of the root mean square error of the orientation tracking against a mismatch in the mass.

Figure 4.5: The root mean square error of the tracking of the OmniMorph with different mismatch values in the mass.



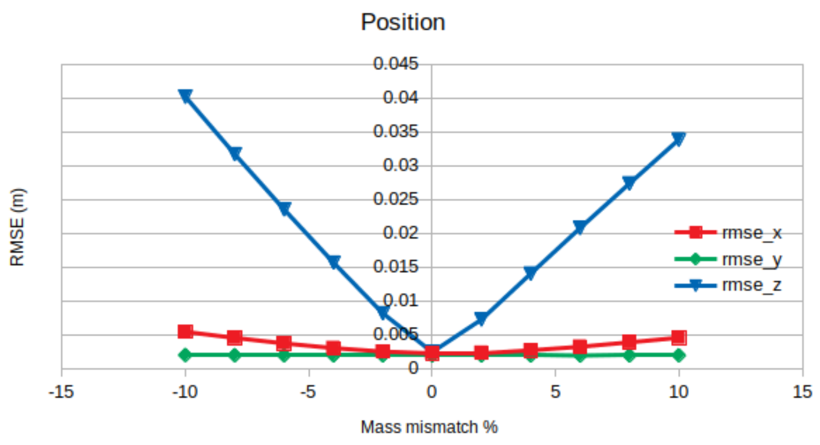


(a) Plot of the error in position, of the OmniMorph, while performing a chirp in translation. The mismatch in mass is 10 per cent.

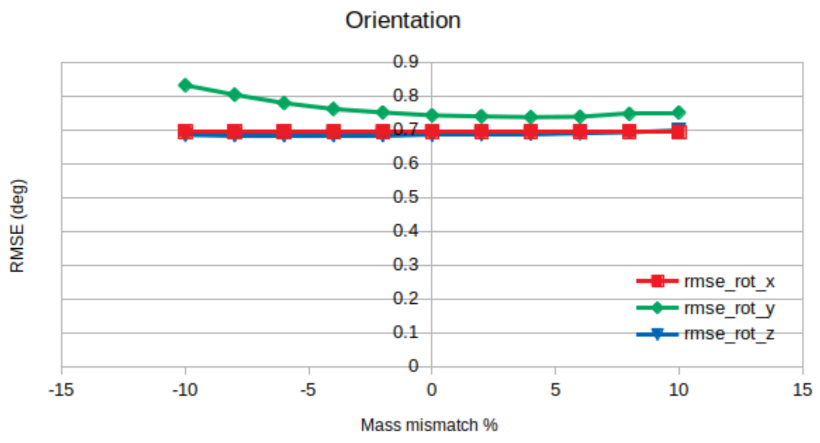
(b) Plot of the error in orientation, of the OmniMorph, while performing a chirp in orientation. The mismatch in mass is 10 per cent.

Figure 4.6: Error plots of the OmniMorph, with a mismatch of 10 per cent in the mass. The scenario is a chirp in translation and orientation.

To further analyse the offset in z another scenario is given, where the reference is a chirp in z translation, instead of x translation, and y rotation. The reference is now in the same direction as the offset of the previous results. The mismatch performances are shown in Fig. 4.7. The tracking with a 10 per cent mass mismatch is shown in Fig. 4.8. From the error plots with a 10 per cent error mismatch, it can be seen that the mean offset is approximately the same as before, however, an oscillation is added coming from the fact that the trajectory is now an oscillation in this z-axis. For the overall RMSE in Fig. 4.7, the errors are very similar, however, the error in x is now approximately 50 per cent less. Tracking in x improved because the chirp is not in x anymore.

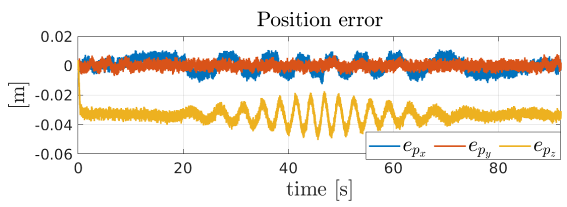


(a) RMSE performance in the position.

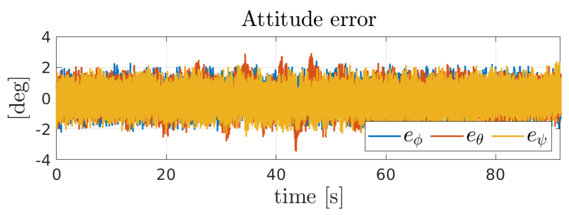


(b) RMSE performance in the rotation.

Figure 4.7: Performance plots of a chirp in z translation and y rotation, where the mismatch ranges from -10 to 10 per cent, which corresponds to 0.9 kg and 1.1 kg.



(a) Plot of the position error.



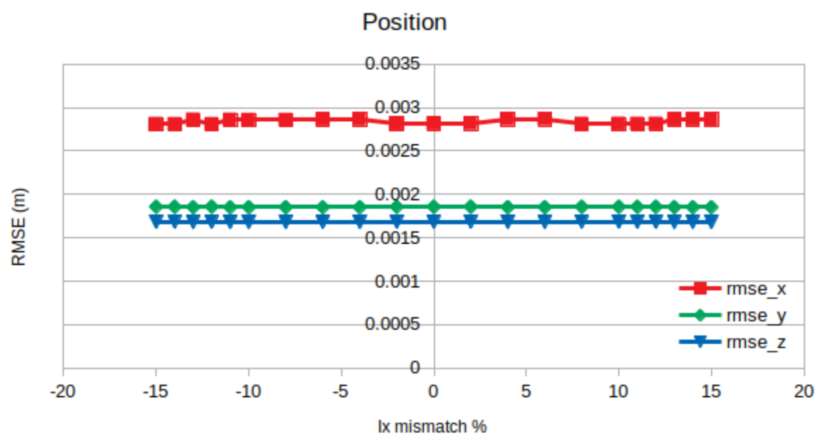
(b) Plot of the orientation error.

Figure 4.8: Performance plots of a chirp in the z translation and y rotation. The mass mismatch is set to 10 per cent.

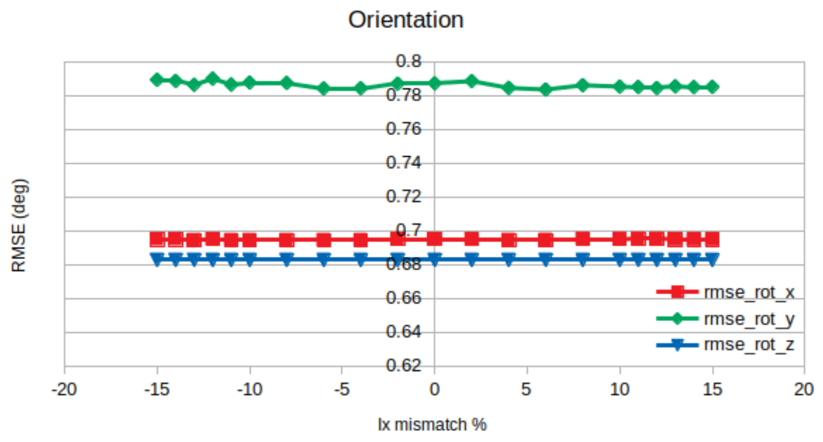
4.3.2 Inertia mismatch

The next mismatches are introduced in the inertia values, where the results are shown separately for the mismatch in I_x, I_y, and I_z. The plots are given in figures 4.9, 4.10 and 4.11. The mismatch ranges from -15 per cent to 15 per cent, resulting in a range of 0.0357 kgm^2 till 0.0483 kgm^2 for I_x and I_y. For I_z the mismatch ranges from 0.07055 kgm^2 till 0.09545 kgm^2.

Looking at the results, the errors in position are not affected by the mismatch in inertia. The offset of the RMSE in x comes from the fact that the reference trajectory is in x, resulting in a relatively higher error than the other position axes. For the rotation, the error is only affected by the mismatch in y, because the rotational reference trajectory is around the y-axis, meaning the tracking error gets worse when the mismatch in I_y gets worse. If, during a practical scenario, save operations require an error in the rotation of fewer than, for example, 0.8 degrees, the mismatch in the axis that corresponds to the reference axis, should stay for this situation below 15 per cent.



(a) RMSE in position with a mismatch in I_x.



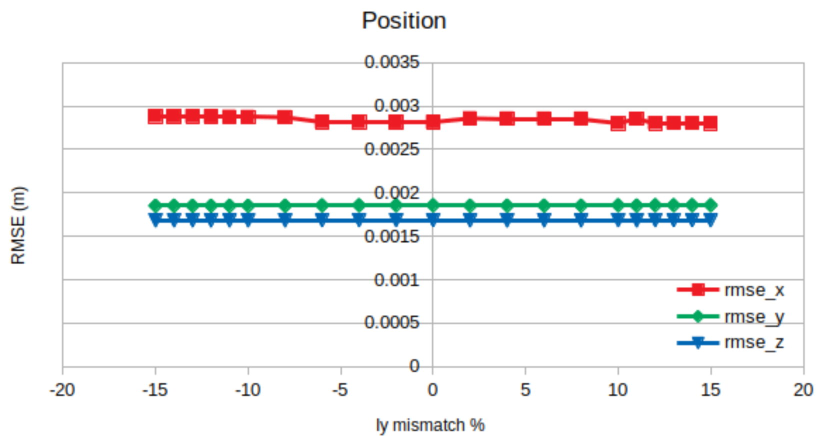
(b) RMSE in orientation with a mismatch in I_x.

Figure 4.9: Performance in tracking with a mismatch in I_x, ranging from -15 till 15 percent, which corresponds to 0.0357 kgm^2 till 0.0483 kgm^2

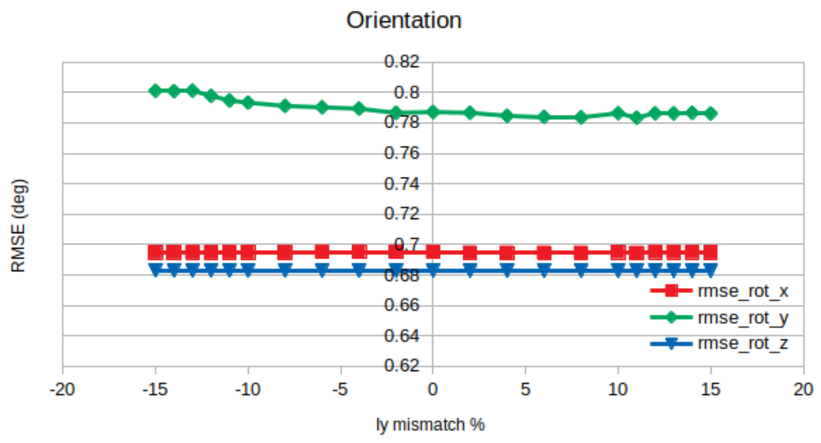
4.3.3 Propeller coefficient mismatch

The final mismatches are given in the thrust and drag coefficients of the propellers. Both are treated separately. The range of the mismatch is from -15 till 15 per cent, meaning for c_f from $1.054 \cdot 10^{-4}$ till $1.426 \cdot 10^{-4}$ and for c_t from $1.87 \cdot 10^{-6}$ till $2.53 \cdot 10^{-6}$. The results are given in 4.12 and 4.13.

Looking at the results, the position errors are not influenced by the model mismatches in c_f and c_t, whereas with orientation, the RMSE becomes worse with an increasing mismatch. This



(a) RMSE in position with a mismatch in I_y.



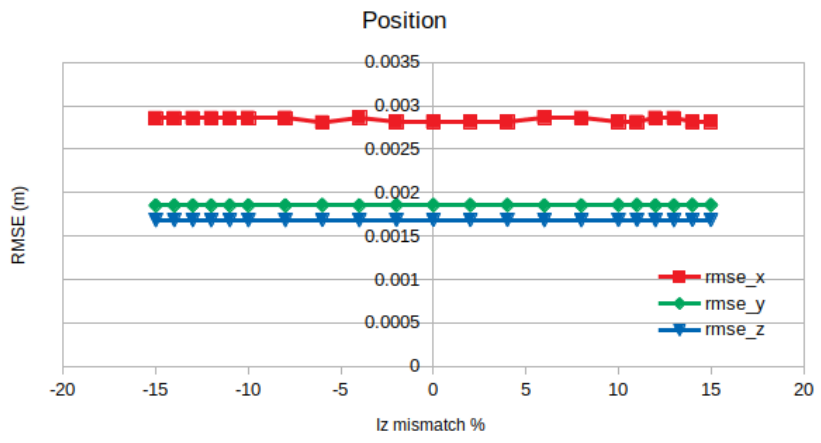
(b) RMSE in position with a mismatch in I_y.

Figure 4.10: Performance in tracking with a mismatch in I_y, ranging from -15 till 15 percent, which corresponds to 0.0357 kgm^2 till 0.0483 kgm^2.

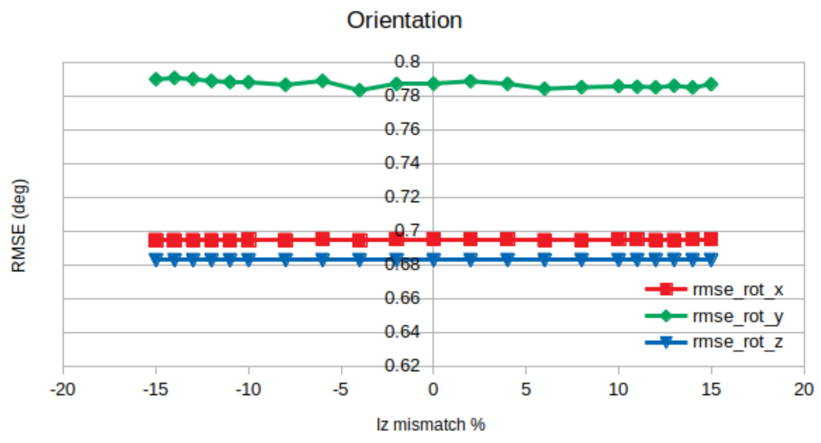
shows that mismatch in the propeller coefficients influences the error in all three rotation axis. Furthermore, as expected the propeller speed changes linearly with the mismatch in c_f. This can be confirmed by the equation 2.14 because the propeller speeds need to be higher to obtain the same needed thrust. Again, thinking about a practical scenario it can be stated that, when a safe operation requirement is, to stay below a rotation error of, for example, 0.9 degrees, the mismatch in c_f and c_t should stay below 15 per cent.

4.3.4 Discussion

From the model mismatches it can be seen that, for this specific chirp scenario, the mass mismatch influences the position error the most, by introducing an offset in the error of the z direction. Furthermore, for this scenario, the c_f and c_t mismatch introduce the most significant errors in rotation tracking. The given plots can indicate the maximum mismatch allowed to safely operate inside certain specified error boundaries. The results give preliminary indications of the behaviour of the controller, regarding mismatch. However, those measurements are done with Matlab simulations, which are not close enough to reality. In reality, mismatches could have more influence because there could be different noise sources, multiple mismatches at the same time, unknown disturbances, etc. Also, in future work, more trajectories/scenarios should be analysed to better understand the behaviour and performance of the controller.



(a) RMSE in position with a mismatch in I_z.



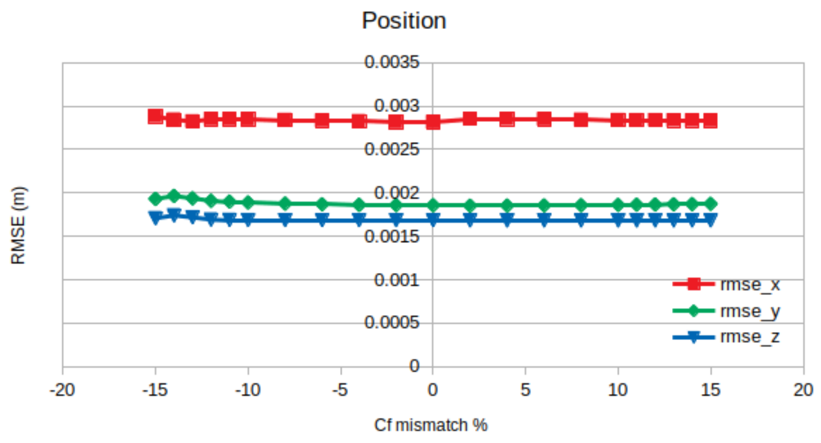
(b) RMSE in position with a mismatch in I_z.

Figure 4.11: Performance in tracking with a mismatch in I_z, ranging from -15 till 15 percent, which corresponds to 0.07055 kgm^2 till 0.09545 kgm^2.

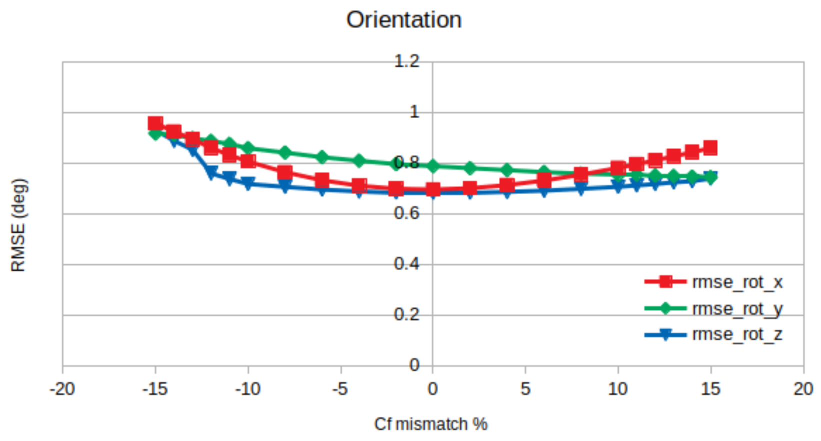
4.4 Energy usage and the objective function

The OmniMorph has the ability to change the tilting angle of the propellers, which enables the platform to switch between UA and FA, making the platform energy efficient and energy inefficient respectively. The control of energy consumption can be implemented by adding the energy cost term to the objective function, shown in the theoretical background. The following paragraphs will cover several scenarios, showing different behaviours of the controller, depending on the energy term, noise and relative weights in the controller. This analysis is in line with research question two, regarding the objective function. All simulations in section 4.4 are done with quaternions. Orientation plots are given in Euler angles by transforming quaternions after simulation.

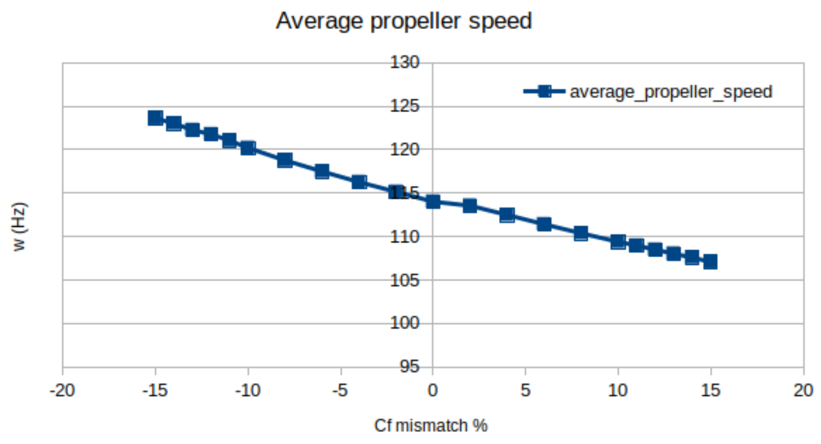
The first scenario is hovering, without the energy cost and the initial servo angle is 50 degrees. The noise is turned off. The result is shown in Fig. 4.14. Then, the energy cost is added, to see its influence. The weights of the energy and servo rate are $1 \cdot 10^{-3}$. The result is shown in Fig. 4.15. As expected, the servo angle converges only to zero when the energy term is inside the cost function, trying to minimize this term by changing the servo angle. The servo converges slower around the second 5, due to the fact that the energy cost and servo rate cost are at this time in the same magnitude.



(a) RMSE in position while having a mismatch in c_f.



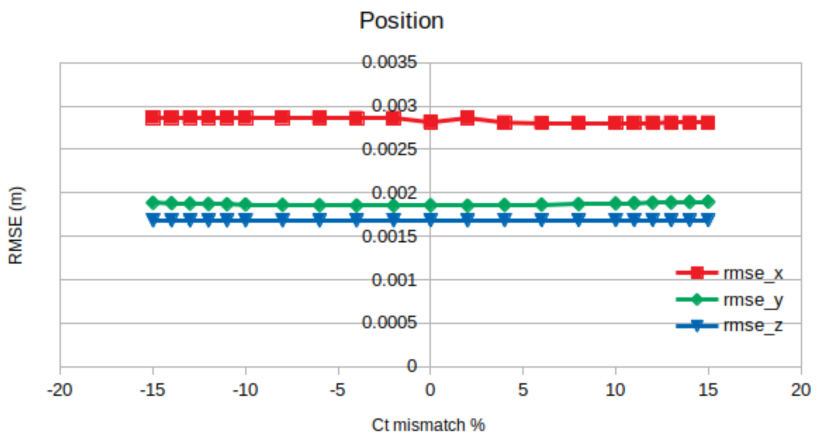
(b) RMSE in orientation while having a mismatch in c_f.



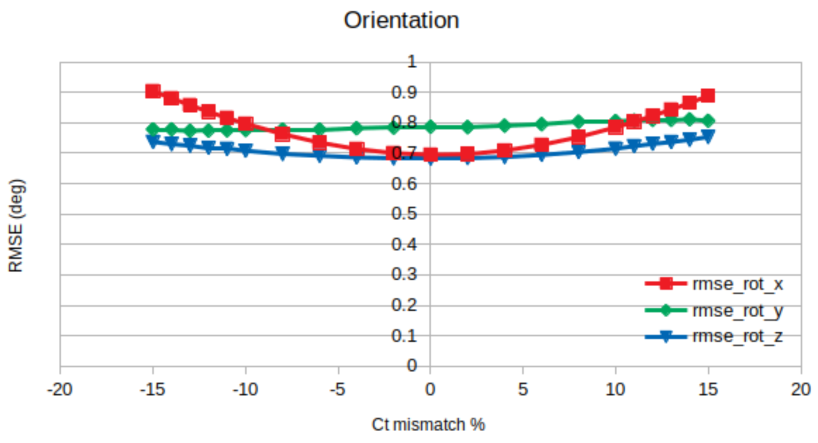
(c) Plot of the propeller speed against the mismatch in c_f.

Figure 4.12: Performance plots of the tracking while having a mismatch in c_f, ranging from -15 till 15 percent, which corresponds to the values $1.054 \cdot 10^{-4}$ till $1.426 \cdot 10^{-4}$.

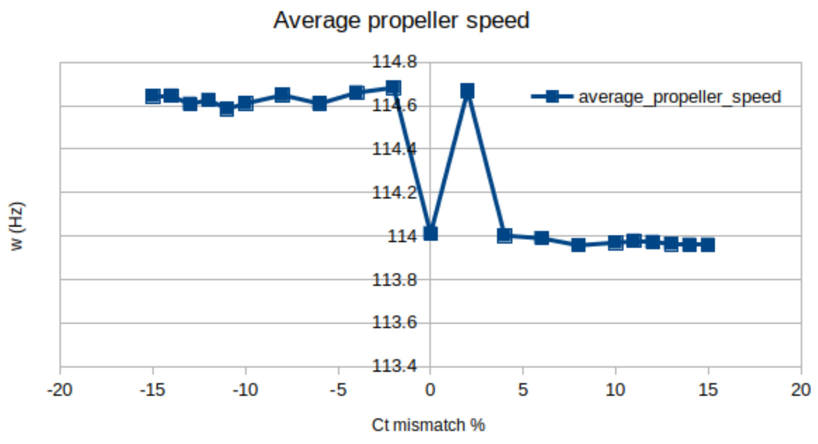
The next scenario is a 180 degrees roll around the x-axis and then hover. The motivation for such a roll scenario is to show an omnidirectional movement, which can be needed for e.g. measurements. First, there is again no noise and no energy cost, the initial servo angle is now 0



(a) RMSE in position while having a mismatch in c_t.



(b) RMSE in orientation while having a mismatch in c_t.



(c) Plot of the propeller speed against the mismatch in c_t.

Figure 4.13: Performance plots of the tracking while having a mismatch in c_t, ranging from -15 till 15 percent, which corresponds to the values $1.87 \cdot 10^{-6}$ till $2.53 \cdot 10^{-6}$.

degrees. The reference trajectory is sinusoidal, shown in equation 4.4 and the results are shown in Fig. 4.16.

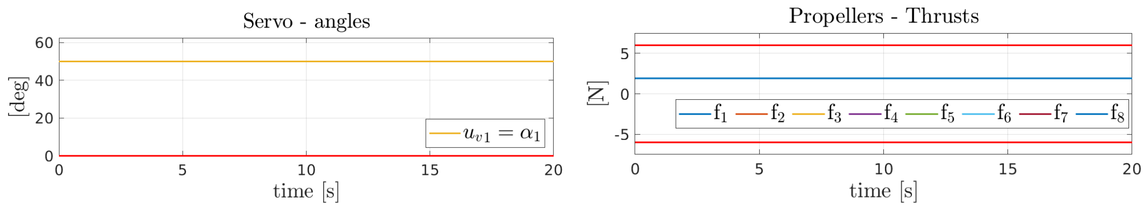


Figure 4.14: Behaviour of the OmniMorph when hovering and no energy cost is apparent. The initial servo angle is 50 degrees.

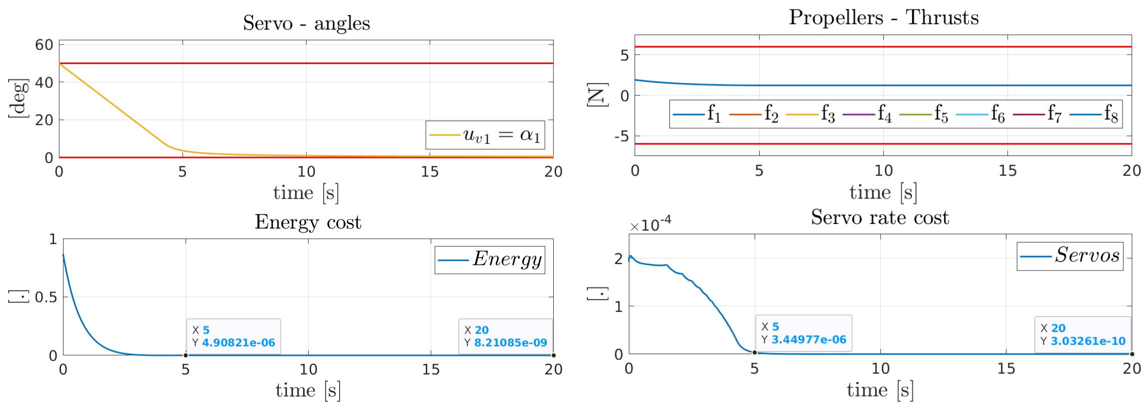


Figure 4.15: Behaviour of the OmniMorph when hovering with an initial servo angle of 50 degrees and the energy cost term included. The weights of the energy and servo rate are both $1 \cdot 10^{-3}$.

$$
\begin{aligned}
\theta_{x_r} &= -90 \cdot cos(2\pi \cdot \frac{1}{60} \cdot (t-5)) + 90 \\
\omega_{x_r} &= 90 \cdot sin(2\pi \cdot \frac{1}{60} \cdot (t-5)) \cdot 2\pi \cdot \frac{1}{60} \\
\dot{\omega}_{x_r} &= 90 \cdot cos(2\pi \cdot \frac{1}{60} \cdot (t-5)) \cdot 4\pi^2 \cdot (\frac{1}{60})^2
\end{aligned}
\tag{4.4}
$$

Looking at the propeller thrusts something interesting happens. Only six propellers are used to hover after the roll movement, whereas the other two, namely 4 and 6, have the same direction of thrust force as gravity, making this a very inefficient state. Furthermore, the servo angle converges to zero, however, there is no energy cost. This specific behaviour could be caused by the following. At the start of the roll, a fully actuated configuration is needed, which means a wider spread of thrust forces. Therefore, in the beginning, the thrust forces spread out by increasing the servo angle. From time 20 seconds, the speed of the roll movement slows down, making the body roll acceleration go from positive to negative, which means that the body wrench needs to change. To change the body wrench, the propeller thrusts or servo angle should change. In this specific situation, the servo rate cost is less than the thrust rate cost, so changing the servo angle is cheaper, which is confirmed in the cost plots. The servo rate cost increases from the second 20. So, changing the body wrench results in convergence of the servo angle, whereas the propeller thrusts stay spread and end in an inefficient state. In the end state, propellers 4 and 6 produce force in the wrong direction. The weights for the thrust rate and servo rate are $5 \cdot 10^{-5}$ and $1 \cdot 10^{-3}$ respectively.

In order to confirm the conclusion made previously, a sensitivity analysis is done with the same scenario. In Fig. 4.17, the end steady states of the propeller forces is plotted against different servo rate weights. The weight of the derivative of the propeller forces is kept $5 \cdot 10^{-5}$. From this plot, it can be seen that by increasing the weight of the servo rate cost, the spread of the final thrust forces decreases. At a servo rate weight of approximately $2.8 \cdot 10^{-3}$ or higher the propellers are all in the correct direction.

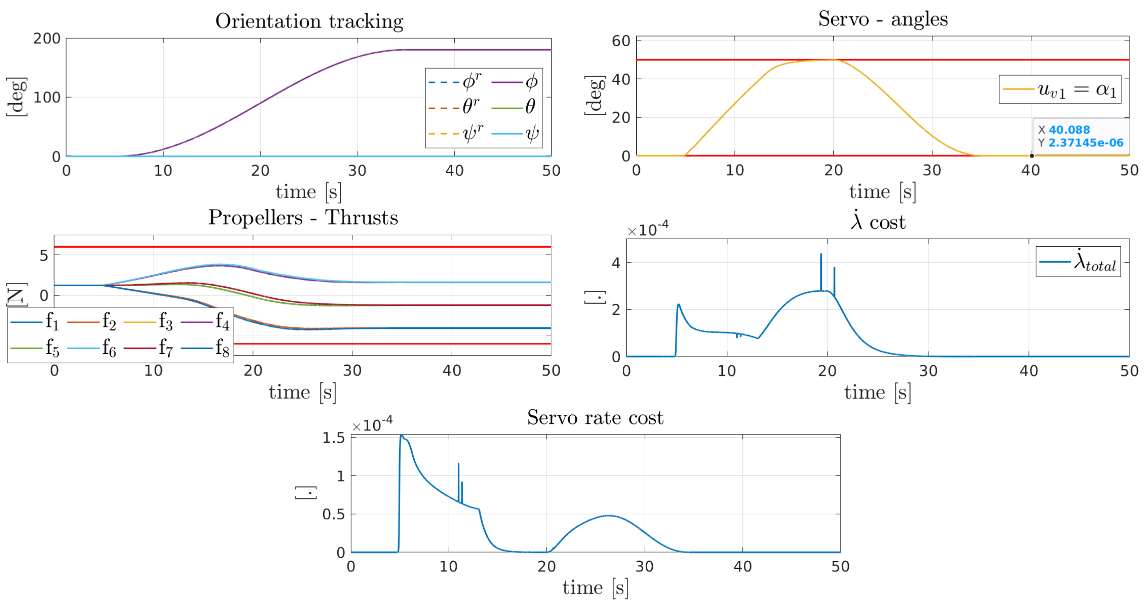


Figure 4.16: Behaviour of the OmniMorph when performing a 180 degrees roll and then hovers. The weights for the thrust rate and servo rate are $5 \cdot 10^{-5}$ and $1 \cdot 10^{-3}$ respectively. The energy cost term is not in the objective function.

Next, in Fig. 4.18, the end steady states of the propeller forces are plotted against different lambda dot weights, and now the servo rate weight is kept $1 \cdot 10^{-3}$. From this plot, it can be seen that by increasing the weights of lambda dot (the derivative of propeller forces), the final spread of propeller thrusts increases. At approximately a weight of $2 \cdot 10^{-5}$ or higher for the lambda rate, propellers 4 and 6 are in the wrong direction, namely the same as the gravity.

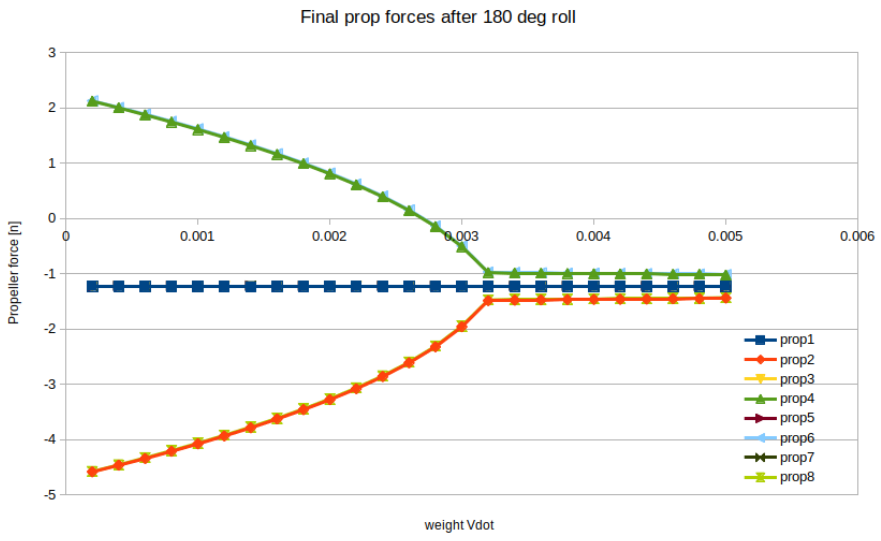


Figure 4.17: Plots of the final steady states propeller forces against different weights for the servo rate. The scenario is a 180 degrees roll and then hover.

To conclude, for this specific trajectory and controller configuration, the final thrust forces are influenced by the ratios of the input weights. It confirms that with this movement, the controller chooses between both inputs to change the body wrench, and can introduce an end configuration that might not be wanted. However, when propellers 4 and 6 are in the same

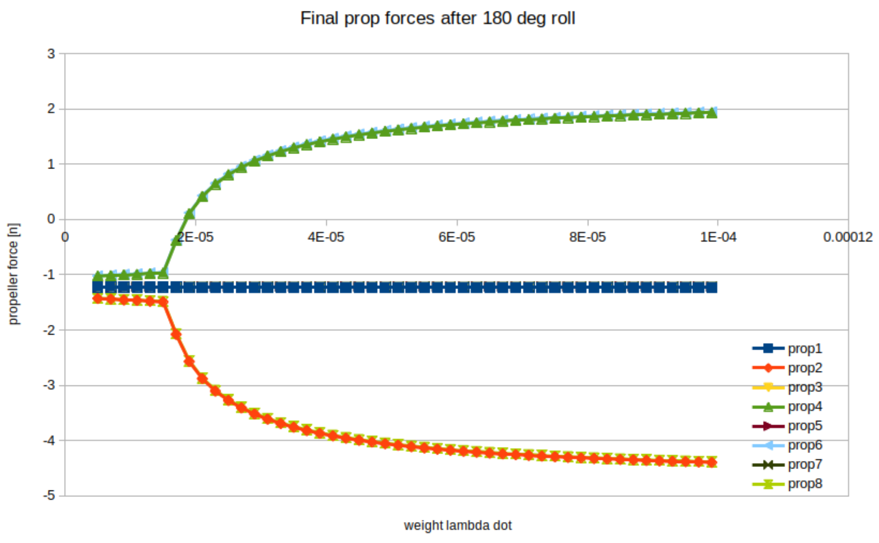


Figure 4.18: Plots of the final steady states propeller forces against different weights for the propeller force rate. The scenario is a 180 degrees roll and then hover.

direction of gravity, the platform might be better at handling disturbances, nevertheless, this should be investigated more.

Next, we take again the input weights that caused propellers 4 and 6 to be in the direction of gravity at the end configuration. Namely for the thrust rate and servo rate $5 \cdot 10^{-5}$ and $1 \cdot 10^{-3}$ respectively. However, now we introduce the energy cost term. The results are shown in Fig. 4.19. It can be seen that, now the energy cost forces the propeller thrusts to converge.

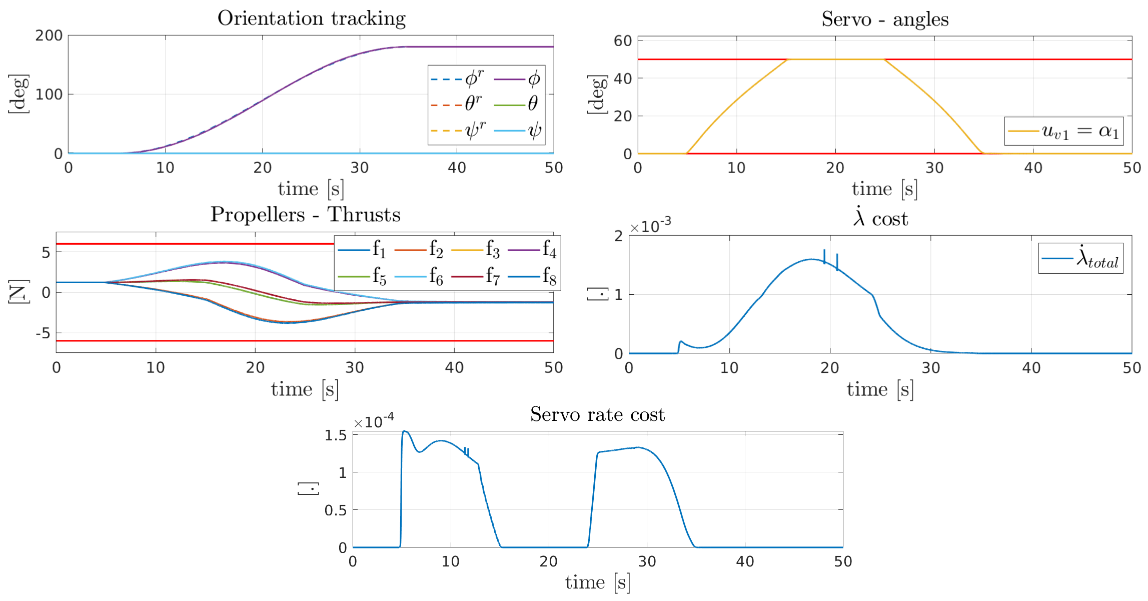


Figure 4.19: Behaviour of the OmniMorph when performing a 180 degrees roll and then hovers. The weights for the thrust rate and servo rate are $5 \cdot 10^{-5}$ and $1 \cdot 10^{-3}$ respectively. The energy weight is $1 \cdot 10^{-4}$

If we remove from the previous results the energy term and add noise instead, another interesting result is obtained, which is shown in Fig. 4.20. The noise causes the thrusts to converge slowly. This could be due to the fact that the noise imposes the propeller forces to change in order to counteract the fast disturbance noise. Stabilizing against noise is done by the propeller forces and not by the servo angle because the maximum rate of the servo angle is too slow. Having to change propeller forces constantly with small values to counteract the noise, prevents the possibility to have the equilibrium phenomenon of before, and forces the servo to go back to zero, in order to hover stable. In order to completely understand and conclude the behaviour of Fig. 4.20, more analysis should be done.

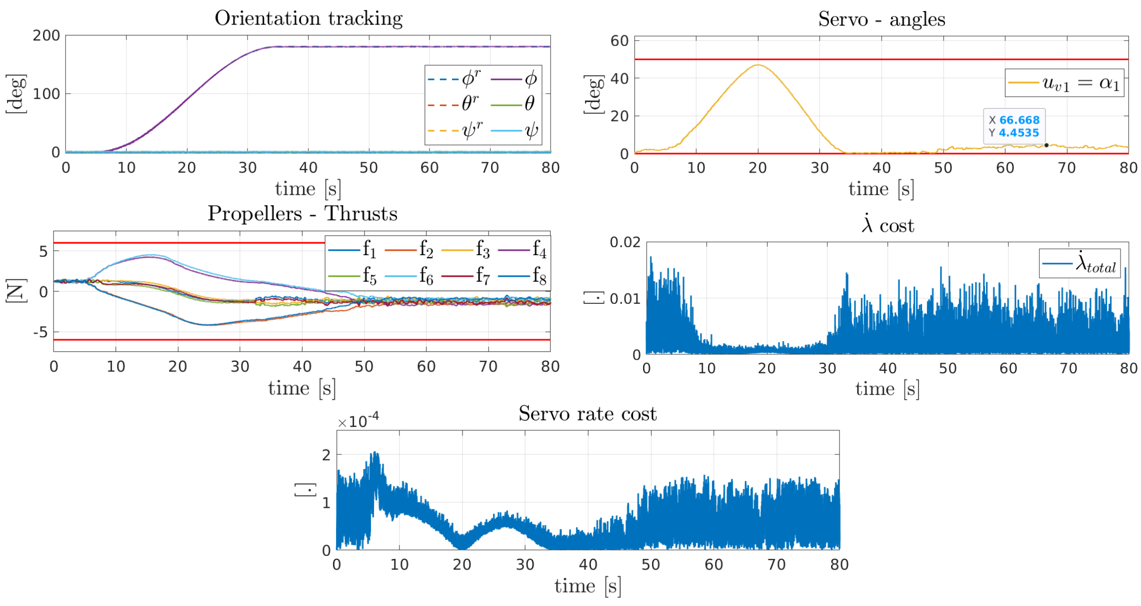


Figure 4.20: Behaviour of the OmniMorph when performing a 180 degrees roll and then hovers. The noise is turned on. The weights for the thrust rate and servo rate are $5 \cdot 10^{-5}$ and $1 \cdot 10^{-3}$ respectively. The energy cost term is not in the objective function.

The next scenario introduces a higher weight to the servo rate cost while having noise but no energy cost. This scenario is in line with the previous sensitivity analysis. The results are shown in Fig. 4.21, where the weight for the thrust rate and servo rate are $5 \cdot 10^{-5}$ and $5 \cdot 10^{-2}$ respectively. From the plots, it can be seen that the servo angle does not converge to zero anymore, because the cost of changing the servo is relatively higher than the change in propeller thrusts. The servo angle also does not reach its limit of 50 degrees, to be omnidirectional. In this specific situation, it is not needed to reach the servo limit, because the propeller thrusts give at their limits enough force to perform the roll turn.

Finally, a complete setup is given, where noise and the energy term are included. The results are shown in Fig. 4.22. The energy weight is $1 \cdot 10^{-4}$. The maximum energy consumption is almost divided by 3 compared to the scenario with noise and without the energy cost. In other words, the energy term makes the platform more energy efficient. The servo angle diverges slower to zero at this result compared to, for example, Fig. 4.20 due to a higher weight for the servo rate. To let the servo angle converge faster the servo rate cost could be decreased, however, this could affect other aspects of the performance.

4.4.1 Discussion

From all the results presented in this section, it can be concluded that the behaviour highly depends on the terms in the cost function and its relative weights. Dependent on the wanted behaviour in a specific scenario, the weights should be adjusted. When energy usage is important, the term should be included, however, it will affect its overall performance and behaviour. If energy is not a problem, better tracking can be obtained, but behaviour depends still on the other weights, for example, the thrust rates and servo rates (inputs), possibly resulting in a servo angle that does not converge to zero or does converge. Interesting research could be in

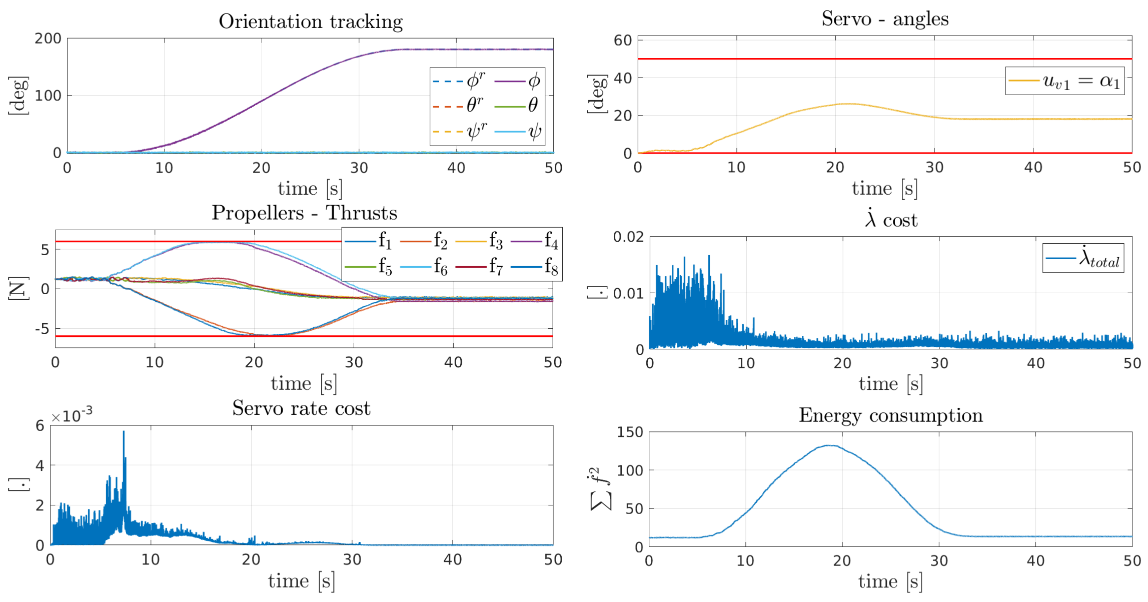


Figure 4.21: Behaviour of the OmniMorph when performing a 180 degrees roll and then hovers. The noise is turned on. The weights for the thrust rate and servo rate are $5 \cdot 10^{-5}$ and $5 \cdot 10^{-2}$ respectively. The energy cost term is not in the objective function.

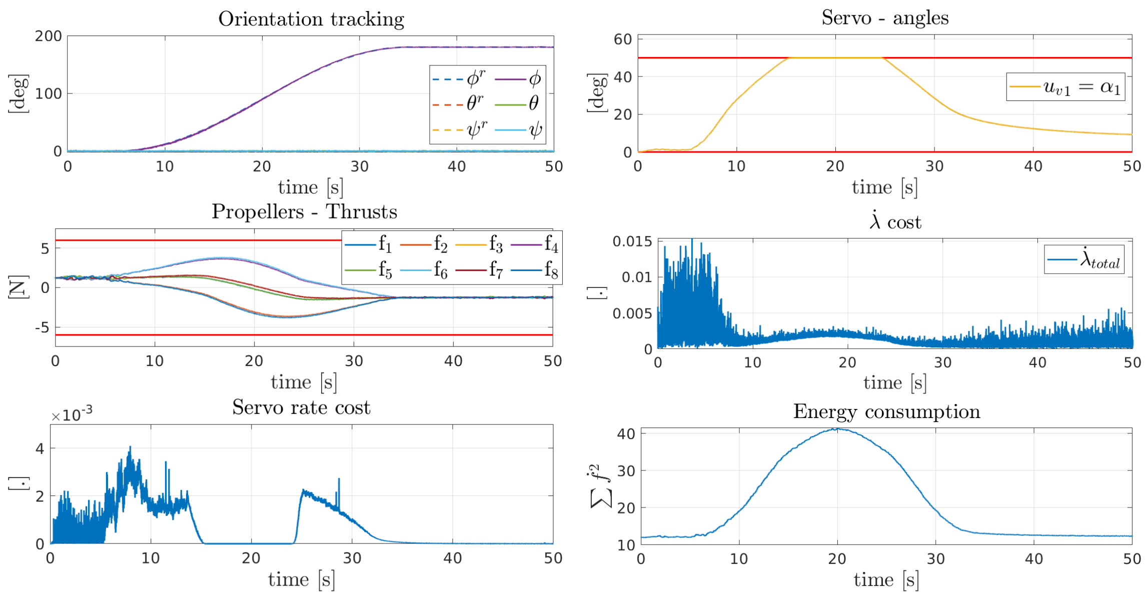


Figure 4.22: Behaviour of the OmniMorph when performing a 180 degrees roll and then hovers. The noise is turned on. The weights for the thrust rate and servo rate are $5 \cdot 10^{-5}$ and $5 \cdot 10^{-2}$ respectively. The energy cost term has a weight of $1 \cdot 10^{-4}$.

the direction of using machine learning for the cost function weights or analysing the effect of the cost terms and weights with other scenarios.

4.5 Prediction horizon and its influence

Next, the influence of the prediction horizon will be examined, which is in line with research questions three and four. The prediction horizon can have different lengths and resolutions, influencing performance and computational load. In this section, the prediction horizon length will be changed, whereas the resolution stays the same, namely 0.1 seconds per shooting interval. Let us introduce again the chirp scenario from Fig. 4.4, with the same parameters. The noise is on and the energy term in the cost function is removed, to only look at tracking performance for the moment. First, the scenario and measurements were done with an initial servo angle of 50 degrees to start already FA. Next, the trajectory is given with an initial angle of zero, starting UA. The RMSE and computational time results are shown in figures 4.23 and 4.24.

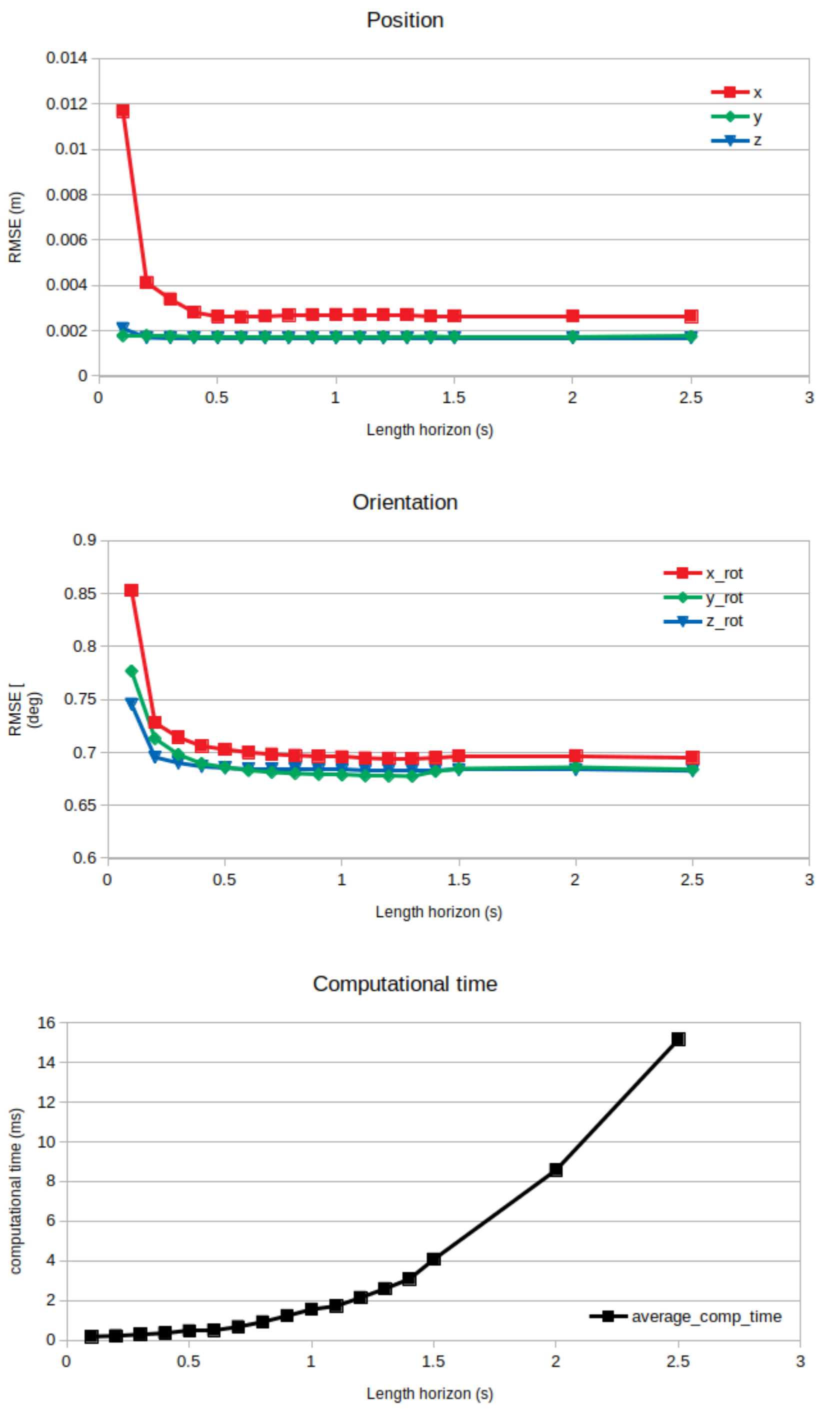


Figure 4.23: Tracking performance and computational time of the OmniMorph and NMPC during a chirp scenario, while having an initial servo angle of 50 degrees.

4.5.1 Discussion

For the first situation, where the initial servo angle is 50 degrees, relatively no improvement was obtained with a prediction horizon longer than 0.5 seconds. However, when the initial servo

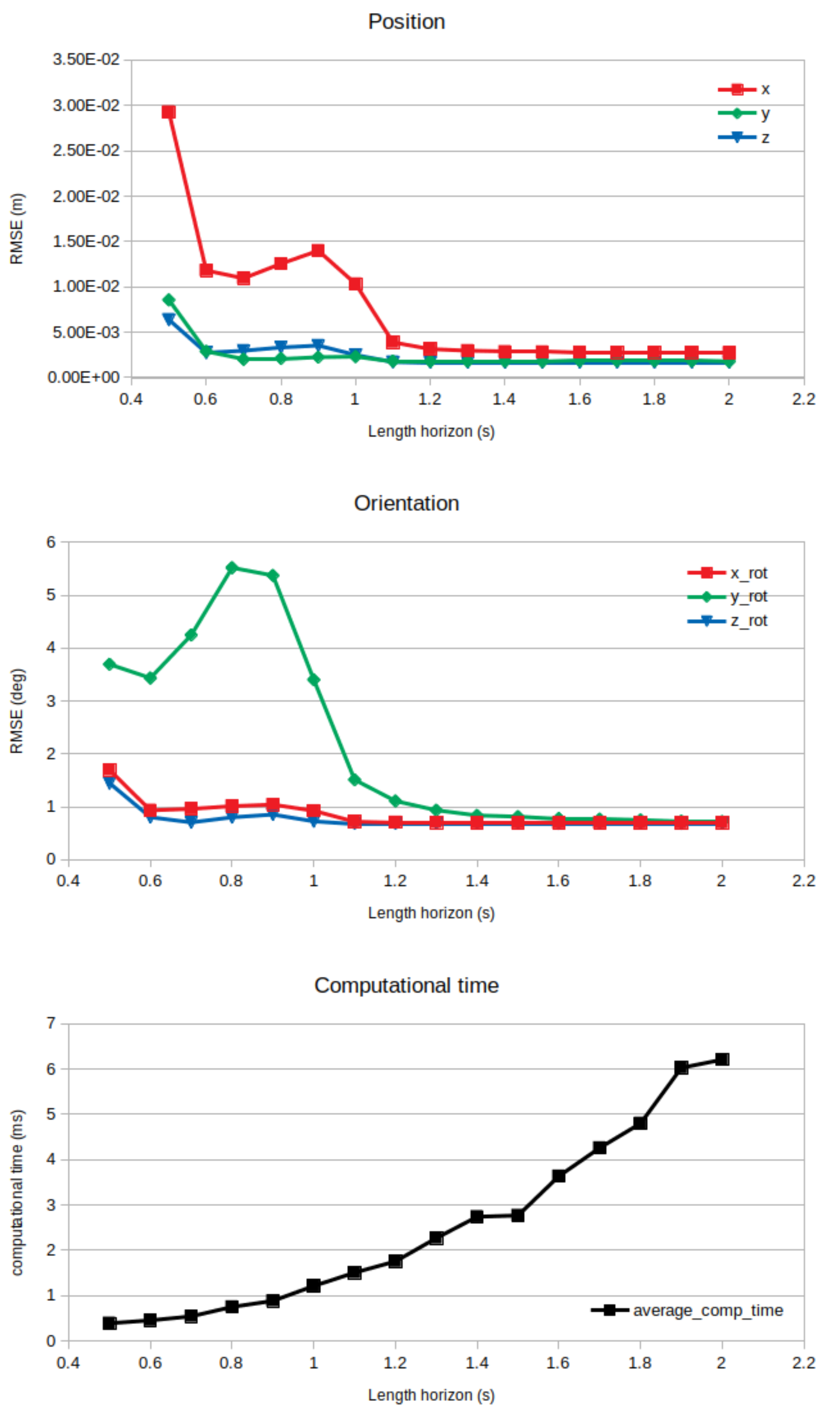


Figure 4.24: Tracking performance and computational time of the OmniMorph and NMPC during a chirp scenario, while having an initial servo angle of 0 degrees.

angle is 0 degrees, the performance increases until a prediction horizon length of 1.5 seconds is reached. Therefore, in this specific scenario, it is better to have a longer prediction horizon, when the platform starts UA. For the given chirp trajectory, the platform needs to be FA to have sufficient tracking performance, therefore it switches at the start from UA to FA. Because the servo angle is a slow-changing input, and therefore needs time to change, the controller can track better when it knows earlier in time, that it needs to change to FA. In other words, a longer prediction horizon gives the controller the opportunity to act earlier if a slow-changing input needs to change.

Regarding the computational time plot, we see a quadratic increase in time, however, for the prediction horizon length of 1.5 seconds the values are still below the shooting time interval. In other words, the controller is on time with a solution before the next control input needs to be determined. Furthermore, the resolution of the prediction should be small enough compared to the frequency of the reference trajectory, otherwise, aliasing could appear. This happens for example, if a chirp trajectory is provided where the period of oscillation is close or smaller than the prediction horizon resolution, which causes the prediction horizon samples to lose detail of the trajectory.

5 NMPC versus Reactive controller

The final results are a comparison of the NMPC controller with a reactive controller. A reactive controller is a controller that responds directly to the current state of the system without considering the past or future states, it computes control inputs based on those current states. The reactive controller used in this comparison is based on the one of [1] and [2]. Furthermore, it was tried to implement the improved allocation of [2], nevertheless, this was not finished due to time. Instead, in this chapter first, their reactive position control and attitude control are shortly explained. Next, an explanation is given about the two-norm optimization allocation method. Finally, a scenario is performed by the NMPC and the reactive controller with the pseudo inverse for allocation, in order to compare the performance of both controllers. For both implementations in this section, the platform has no actively tilting propellers. The angle of orientation is set statically to 50 degrees around the x-axis of the propeller frame because the reactive controller implementation does not allow actively tilting propellers.

5.1 Reactive controller implementation

The reactive control of the statically tilted octo-rotor is done in a cascaded structure. There is a position controller that produces a body force command. In parallel, there is an attitude controller that computes the body's rotational velocity. Then, an angular velocity controller produces the body torque command based on the wanted angular velocity. An overview is given in Fig. 5.1.

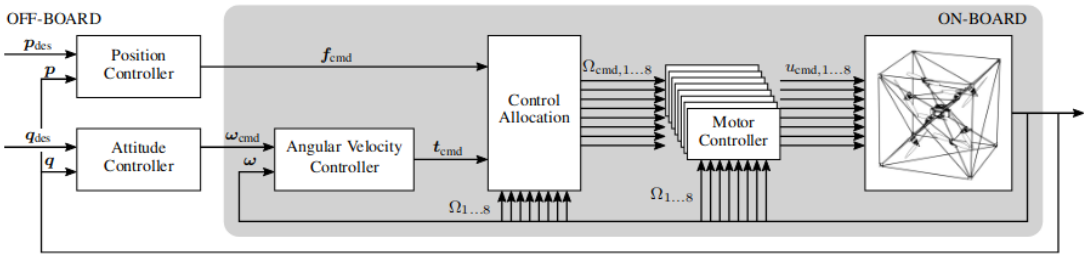


Figure 5.1: Control overview of the cascaded reactive controller of [2].

5.1.1 Position control

The body force commands are calculated, based on the position error, integral of the position error, velocity error and desired acceleration. The integral term is added to counteract disturbances and modelling errors. The errors are defined as $\mathbf{p}_{err} = \mathbf{p}_{des} - \mathbf{p}$. The force command equation is given in 5.1, for the controller gains, the reader is invited to the paper of [2].

$$\mathbf{f}_{cmd} = m\mathbf{R}(\mathbf{q})(k_{pos,p} + k_{pos,i}\int \mathbf{p}_{error}dt + k_{pos,d}\dot{\mathbf{p}}_{error} + \ddot{\mathbf{p}}_{des} + \mathbf{g}) \tag{5.1}$$

5.1.2 Attitude control

The desired body rotational velocity is calculated as shown in equation 5.2. Which is based on the quaternion error shown in equation 5.3. The equations consist of the quaternion error, the integral of the quaternion error and the rotational velocity of the platform.

$$\boldsymbol{\omega}_{cmd} = k_{att,p}sgn(q_{err,w})\mathbf{q}_{err,xyz} + k_{att,i}\int sgn(q_{err,w})\mathbf{q}_{err,xyz}dt + \mathbf{R}(\mathbf{q}_{err})^{-1}\boldsymbol{\omega}_{des}. \tag{5.2}$$

$$\mathbf{q}_{err} = \mathbf{q}_{des} \otimes \mathbf{q}^{-1} \tag{5.3}$$

Next, the angular velocity command is controlled by computing the body torque, which is the inner control loop and is designed as a linear time-invariant first-order system. The control equation is shown in 5.4. For the comparison between NMPC and the reactive controller in this thesis, the function is implemented without the last term, which is the sum of rotor inertias.

$$\mathbf{t}_{cmd} = \frac{1}{\tau_\omega}\mathbf{J}(\boldsymbol{\omega}_{cmd} - \boldsymbol{\omega}) + \boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega} + \sum_{i=1}^{8} J_{rot}\Omega_i \mathbf{n}_i) \tag{5.4}$$

5.2 Allocation

For the allocation, there are two versions. One solution is given in [1], where the pseudo inverse of the static allocation matrix is used, as shown in equation 5.5. Because the platform is OA (8 inputs, 6 outputs), there exists no unique map from the body wrench to the propeller forces. Therefore, by using the pseudo inverse, there is a chance of generating infeasible propeller forces, making the platform unstable.

$$\mathbf{f}_{prop} = \mathbf{G}^{\dagger} \begin{bmatrix} \mathbf{f}_{cmd} \\ \mathbf{t}_{cmd} \end{bmatrix} \tag{5.5}$$

In order to prevent the infeasible propeller forces, the authors have made a new allocation implementation, shown in [2]. This allocation is defined as a two-norm optimization problem. In the long term, the power consumption is minimized and in the short term, the difference between the commanded and current propeller forces is minimized. The cost function for the first four propellers is shown in equation 5.6, where ϵ is the relative weight between the long and short-term objectives. $\boldsymbol{\Phi}_{1,hyst} \subset \boldsymbol{\Phi_1}$ is the hysteresis set of null vector biases. The hysteresis set is a subset of $\boldsymbol{\Phi}_1$ and limits the switching of the bi-directional propellers, by a time span of τ_{hyst}. The set $\boldsymbol{\Phi_1}$ is shown in equation 5.7. Where $\mathbf{P}_1$ selects the first four propellers and $\boldsymbol{\eta}_1$ is the null vector. For the second four propellers, the optimization is done with ϕ_2 and can be written analogously to 5.6 and 5.7.

$$\begin{aligned} \min_{\phi_1} \sum_{i=1}^{4} (1-\epsilon)|f_{rot,cmd,i} + \phi_1|^{3/2} + \epsilon(f_{rot,cmd,i} + \phi_1 - f_{rot,i})^2 \\ \mathbf{f}_{rot,cmd} = \mathbf{G}^{\dagger} \begin{bmatrix} \mathbf{f}_{cmd} \\ \mathbf{t}_{cmd} \end{bmatrix} \\ \text{subject to } \phi_1 \in \boldsymbol{\Phi}_{1,hyst} \end{aligned} \tag{5.6}$$

$$\boldsymbol{\Phi}_1 = \{\phi_1 \in \mathbb{R} | \mathbf{f}_{min} \leq \left|\mathbf{P}_1(\mathbf{G}^{\dagger} \begin{bmatrix} \mathbf{f}_{cmd} \\ \mathbf{t}_{cmd} \end{bmatrix} + \phi_1 \boldsymbol{\eta}_1)\right| \leq \mathbf{f}_{max}\} \tag{5.7}$$

5.3 Handling of infeasible trajectories and input limits

To compare NMPC with the previously explained controller, it was tried to implement the improved allocation method. However, due to time, it was not possible to repair the problems that were encountered. Therefore, a small comparison is made with the more simple pseudo inverse allocation. For this comparison, both implementations have a platform without actively tilting propellers. The tilt angle is set to 50 degrees. There is no noise. The scenario is a chirp in x translation, with a maximum velocity of $4.48 m/s$ and a maximum acceleration of $9.74 m/s^2$. The amplitude is $2m$. First, the trajectories are tracked by both controllers, with a platform, having a maximum and minimum propeller thrust of $6N$ and $-6N$. The results of the NMPC and reactive controller are shown in Fig. 5.2, left and right respectively. Both controllers are able to track the desired trajectory. Nevertheless, the propeller thrusts of both implementations are not similar. The NMPC version has a less symmetrical profile compared to the reactive controller. This comes from the fact that the NMPC has still limits on the rotor accelerations from $-300 rad/s^2$ till $300 rad/s^2$. Furthermore, there is a spike in orientation tracking which could be caused by this. To analyse this further, another simulation is done with the NMPC where the limits of the rotor accelerations are set to $-10^9 rad/s^2$ till $10^9 rad/s^2$. The limits can not be removed or set to infinite due to the implementation of the framework. The results are shown in figure 5.3. The profile of the propeller thrusts is now similar to the reactive controller, however, the orientation tracking is still different. But this will not influence the point that will be made. Also, a comparison between those two controllers is limited due to the fact that they are completely different. For example, NMPC uses an objective function with relative weights and takes the future into account, whereas the reactive controller does not.

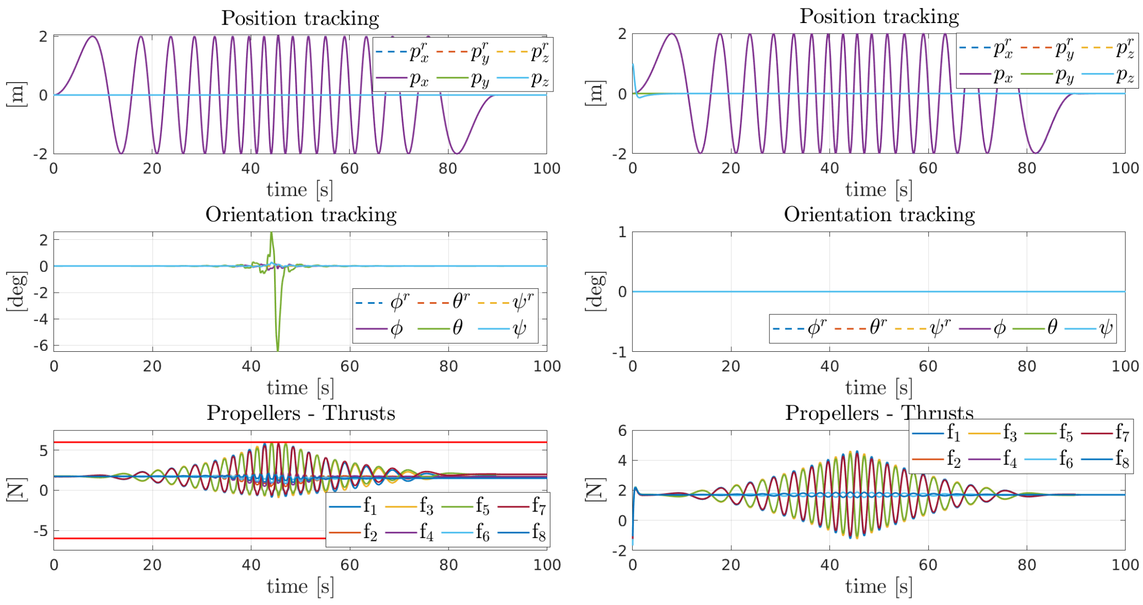


Figure 5.2: Comparison of tracking a chirp trajectory in x translation. The left represents the NMPC controller and on the right the reactive controller. Both are able to track the desired reference.

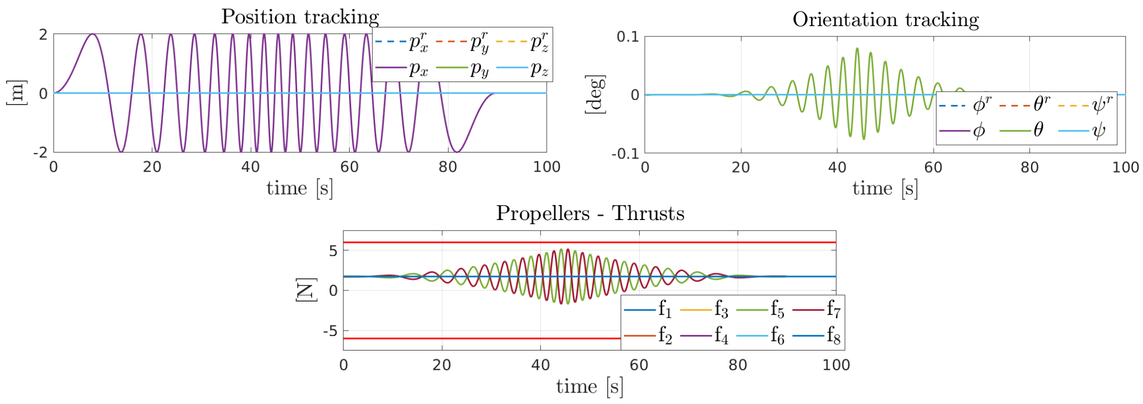


Figure 5.3: A chirp trajectory performed by the OmniMorph with NMPC. The rotor acceleration limits are set to $-10^9 rad/s^2$ till $10^9 rad/s^2$.

Next, both controllers have to control the same trajectory but with limited propeller thrusts of 2 and -2 Newton, where NMPC has again limited rotor accelerations of $-10^9 rad/s^2$ till $10^9 rad/s^2$. The limited thrusts, make the trajectory infeasible. The results are shown in Fig. 5.4, left the NMPC and on the right the reactive controller. The NMPC-controlled platform is not able to track the position completely from time 40 seconds to 50 seconds, therefore it tries to re-orientate to be able to produce stronger thrust in the x-direction. Furthermore, the thrusts are almost the whole manoeuvre at its limits, nevertheless, it does not get unstable. The reactive controller, however, is not able to track the trajectory and gets unstable.

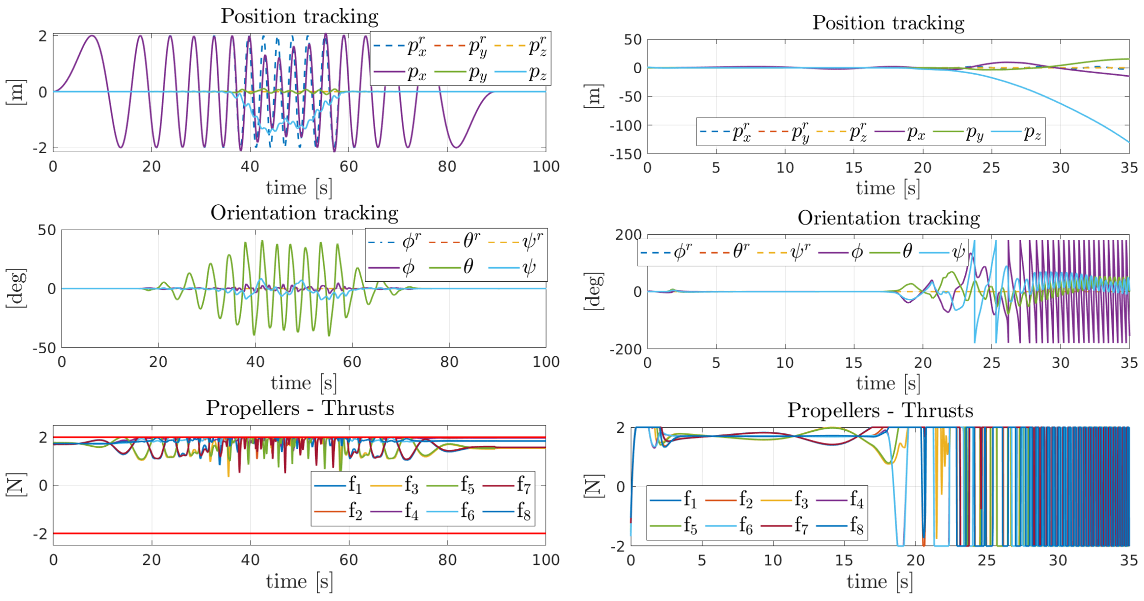


Figure 5.4: Comparison of tracking a chirp trajectory in x translation, which is infeasible due to limits on the propeller thrusts of -2 to 2 Newton. The left represents the NMPC controller and on the right the reactive controller. NMPC is able to track the desired reference, but the reactive controller is not.

5.4 Discussion

From the results, it can be seen that, for this specific infeasible trajectory, the NMPC controller would be a better option. NMPC is able to take into account the limited thrust forces and tries to be as close to the trajectory as possible. For other infeasible trajectories a similar analysis should be done, to understand the behavior. Also, a better comparison should be possible with the improved allocation method. However, even when the allocation problem is improved, the controller is still reactive, meaning it still computes the control inputs based on the current states and not the predicted states. Furthermore, constraints on the propeller force derivatives are also not taken into account, causing possible instability when operating with infeasible propeller accelerations.

A fair comparison between NMPC and a reactive controller is difficult because their implementations are completely different. NMPC uses an objective function with relative weights for all terms inside that objective function. Furthermore, NMPC uses a prediction model and prediction horizon, whereas the reactive controller only uses the tracking error of the current and past states and has no objective function. In order to decide between both implementations, a look should be given to the implementation specifications. If the hardware has limited resources it might be better to use a reactive controller. However, if the computational load is not a problem, NMPC could be used. Moreover, it is important to know what applications the platform will be used for. If, for example, a payload should be carried by the platform, and it is possible to describe this payload in the prediction model of the NMPC, better performance could be reached by using NMPC compared to reactive controllers. To conclude, there is no general best option, it always depends on the scenario.

6 Conclusion

To conclude this thesis assignment, first, the research questions will be answered using the obtained results, and interesting future directions are given. Finally, a small conclusion based on personal experiences is provided.

The first research question was, "What is the best model to use in an optimization-based control for the OmniMorph? The model includes the dynamic equations of motion and the input constraints (Trade-off extensiveness/complexity)". Regarding the orientation representation, we have seen that quaternions would be a better option when rotations larger than approximately 90 degrees are needed. However, when this is not needed, a solution like Euler angles would be sufficient. Next, regarding the mismatches in the model, an error in the mass greatly influences the z-position error. If an inertia mismatch is introduced, it only influences the tracking error in orientation, when the mismatch and direction of rotation reference have corresponding axes. However, in practical cases, there is a high chance that trajectories will be in all directions, and therefore a mismatch in inertia and performance should be checked. A mismatch in the propeller thrust coefficient (c_f) introduces errors in all orientation directions. Furthermore, the propeller speeds are linearly dependent on the error in thrust coefficient, so when the propellers are operating close to their limits, this could affect performance even more. The final mismatch was in the propeller drag coefficient (c_t), which also introduced errors in all orientation directions.

The next research question was, "What is the best objective function to use in order to encompass both tracking ability and energy high-fidelity efficiency coming from an actual energy consumption model and measurements?". In order to decide what terms should exist in the objective function, prior knowledge about the scenario and wanted behaviour should be known. For example, when energy is not important, it might be better to remove this term in order to have the best tracking possible, this could be for example with contact measurements, where precision is required. However, if energy consumption should be minimized, it should be in the cost function, though we saw that it significantly influences the behaviour. In short, there are a lot of interplays between the cost terms and more analysis could be done here. For example, a relation between the tracking error and energy consumption can be examined, which could answer the question, "How much energy does a certain tracking accuracy cost?". Another interesting direction could be the addition of machine learning for the weights in the cost function, or even using machine learning to turn certain terms in the cost function on or off during flight.

The prediction horizon of NMPC is a very important tuning parameter. Therefore the third research question was, "What influence does the prediction horizon have on the performance of the controller?". Again, like the objective function, the length of the prediction horizon depends on the scenario. The length of the prediction horizon is important, for example when the system has a slower changing input, like the servo motor. This makes the controller able to anticipate before the fast trajectory starts. Next to the length, the resolution of the horizon should also be investigated in future work. For example, when a trajectory has a higher frequency relative to the resolution, aliasing could occur. Or when a slower trajectory is desired, it might be better to have a lower resolution, which in turn makes it possible to have longer prediction horizon lengths, because the computational time depends on the number of prediction steps. Therefore, prior knowledge of the scenario is again wanted to optimally set the prediction horizon parameters.

The fourth question was, "Will the NMPC be able to control the real platform in real-time while being implemented onboard with the computational capabilities on board?" Based on simulations on the laptop hardware, plots were given of the average computational time of the optimal control problem (OCP) against the length of the prediction horizon. From those results, it can be concluded that the computational time increases quadratically against the length of the horizon. For example, with a length of 1.5 seconds and resolution of 0.1 seconds, the controller solves the OCP on average in 3 ms. Which is below the shooting interval, even while running Matlab in graphical mode. However, to really answer this question, the next step should be, implementing the controller on the real platform using a dedicated small computer such as an Intel NUC or a microcontroller. This could really answer the question if it is possible to control the OmniMorph with NMPC in real-time. When the computational load is high, choices could be made to make the prediction horizon shorter or change the resolution.

The final goal was, to compare the predictive controller with a reactive controller. Therefore the research question was, "Will the NMPC outperform existing reactive controllers?". In the results, we have seen that the NMPC does not get unstable when a desired trajectory is not feasible, whereas the reactive controller got unstable. For this specific result, the NMPC would be a better option. However, for the general case, it highly depends on the scenario. For example, additions to the reactive controller could be added to handle such infeasible situations. Maybe an option would be to check the maximum acceleration of the desired trajectory beforehand and reformulate the trajectory when it is infeasible. Moreover, both controllers differ highly from each other, making comparison difficult. Also, NMPC is relatively heavier in computational load compared to the reactive controller, therefore the hardware capabilities play an important role in whether to use NMPC or a reactive controller.

6.1 Future work

In this section, some possible directions for future work will be given. This master thesis is the start of a research direction for Omnidirectional morphing platforms, controlled by NMPC and taking into account realistic inputs of the system. Multiple directions were analysed, opening doors for other research.

First, regarding the model, more detail could be given in the prediction model and see how the performance gets influenced. For example, rotor dynamics, motor dynamics, slack of the servo mechanics or the acceleration limits of the servos could be added. Also, regarding model mismatch other scenarios could be tried, or real experiments can be performed.

Concerning the objective function, more scenarios could be analysed and the optimal weights could be determined, such that for more scenarios the optimal performance is achieved. This could be done, for example, by including real-time machine learning. However, machine learning could also be used offline, where extensive simulations can be used, to automatically optimize the weights of the objective function per scenario.

For the prediction horizon, also more scenarios can be examined. Because the platform is designed to handle interactions, an interesting direction could be interaction simulations, for example, a simple lateral push, or more complex procedures.

Another direction could, but also should be, connecting the control with, e.g. Gazebo[18], Mujoco[19] or another physics simulator that would be appropriate, making it closer to reality. In order to connect the controller to e.g., Gazebo, ROS[20] could be used, which is a relatively easy step. Matlab/Simulink have modules that can publish and subscribe to topics. Furthermore, Gazebo also has the option, to publish and subscribe to topics. So, using those topics for the inputs and outputs of the system can connect the controller to the simulation environment. After Gazebo, the control could be implemented on the real platform. Furthermore, a more extensive comparison could be made with the existing reactive controllers. To conclude, many

[18]https://staging.gazebosim.org/home
[19]https://mujoco.org/
[20]https://www.ros.org/

directions are still possible and should be researched, in order to understand the behaviour better and make the system more capable and robust.

6.2 Personal experience

Finally, I would like to give a small conclusion about my personal experience with this thesis. First and most important, I have learned what research is and how to do it. At the start, I had a different view, which made my first approach to the literature review incorrect. However, I understand now that the goal is about doing something that is not done before and opening doors for others. Next, I found that implementing the NMPC with quaternions took more time than expected. It took approximately 8 weeks to get the quaternion implementation working, due to the fact of an implementation error that I could not trace back. After many approaches, I started with a fresh implementation and made it work. So, sometimes it could be better to redo the process because the overview is easily lost after some time of debugging. The final learning point I would like to address, it was easy for me to lose the overall picture (Aerial core project). In other words, when working on implementations, it is easy to go in directions that do not make sense and do not agree with the overall goal of the project. Therefore it is good to constantly interact with the supervisors and ask them for feedback. Furthermore, you can ask yourself, why are we doing certain implementations, simulations or tests?

A Appendix 1

A.1 Static allocation matrix of the OmniMorph

In equation A.1 the complete static allocation matrix is shown of the OmniMorph. This matrix maps the propeller thrusts to the body wrench dependent on the tilting angle α. $s\alpha$ and $c\alpha$ represent $sin(\alpha)$ and $cos(\alpha)$ respectively. L is the distance of the propeller from the centre of mass, kd_i is the direction of rotation of the rotors. c_t and c_f are the propeller torque and thrust constant respectively. Every column corresponds to one propeller and tells what contribution it gives to the total body wrench.

$$\mathbf{G}(\alpha) = \begin{bmatrix} \mathbf{G}_f(\alpha) \\ \mathbf{G}_\tau(\alpha) \end{bmatrix} = \left[\begin{matrix}
-s\alpha & 0 & s\alpha & 0 \\
0 & -s\alpha & 0 & s\alpha \\
c\alpha & c\alpha & c\alpha & c\alpha \\
\frac{\sqrt{3}\cdot L\cdot c\alpha}{3} - \frac{c_t\cdot kd_1\cdot s\alpha}{c_f} & \frac{\sqrt{3}\cdot L\cdot c\alpha}{3} + \frac{\sqrt{3}\cdot L\cdot s\alpha}{3} & \frac{c_t\cdot kd_3\cdot s\alpha}{c_f} - \frac{\sqrt{3}\cdot L\cdot c\alpha}{3} & -\frac{\sqrt{3}\cdot L\cdot c\alpha}{3} - \frac{\sqrt{3}\cdot L\cdot s\alpha}{3} \\
-\frac{\sqrt{3}\cdot L\cdot c\alpha}{3} - \frac{\sqrt{3}\cdot L\cdot s\alpha}{3} & \frac{\sqrt{3}\cdot L\cdot c\alpha}{3} - \frac{c_t\cdot kd_2\cdot s\alpha}{c_f} & \frac{\sqrt{3}\cdot L\cdot c\alpha}{3} + \frac{\sqrt{3}\cdot L\cdot s\alpha}{3} & \frac{c_t\cdot kd_4\cdot s\alpha}{c_f} - \frac{\sqrt{3}\cdot L\cdot c\alpha}{3} \\
\frac{\sqrt{3}\cdot L\cdot s\alpha}{3} + \frac{c_t\cdot kd_1\cdot c\alpha}{c_f} & \frac{\sqrt{3}\cdot L\cdot s\alpha}{3} + \frac{c_t\cdot kd_2\cdot c\alpha}{c_f} & \frac{\sqrt{3}\cdot L\cdot s\alpha}{3} + \frac{c_t\cdot kd_3\cdot c\alpha}{c_f} & \frac{\sqrt{3}\cdot L\cdot s\alpha}{3} + \frac{c_t\cdot kd_4\cdot c\alpha}{c_f}
\end{matrix}\right.$$
$$\left.\begin{matrix}
s\alpha & 0 & -s\alpha & 0 \\
0 & s\alpha & 0 & -s\alpha \\
c\alpha & c\alpha & c\alpha & c\alpha \\
\frac{\sqrt{3}\cdot L\cdot c\alpha}{3} + \frac{c_t\cdot kd_5\cdot s\alpha}{c_f} & \frac{\sqrt{3}\cdot L\cdot c\alpha}{3} + \frac{\sqrt{3}\cdot L\cdot s\alpha}{3} & -\frac{\sqrt{3}\cdot L\cdot c\alpha}{3} - \frac{c_t\cdot kd_7\cdot s\alpha}{c_f} & -\frac{\sqrt{3}\cdot L\cdot c\alpha}{3} - \frac{\sqrt{3}\cdot L\cdot s\alpha}{3} \\
-\frac{\sqrt{3}\cdot L\cdot c\alpha}{3} - \frac{\sqrt{3}\cdot L\cdot s\alpha}{3} & \frac{\sqrt{3}\cdot L\cdot c\alpha}{3} + \frac{c_t\cdot kd_6 s\alpha}{c_f} & \frac{\sqrt{3}\cdot L\cdot c\alpha}{3} + \frac{\sqrt{3}\cdot L\cdot s\alpha}{3} & -\frac{\sqrt{3}\cdot L\cdot c\alpha}{3} - \frac{c_t\cdot kd_8\cdot s\alpha}{c_f} \\
\frac{c_t\cdot kd_5\cdot c\alpha}{c_f} - \frac{\sqrt{3}\cdot L\cdot s\alpha}{3} & \frac{c_t\cdot kd_6\cdot c\alpha}{c_f} - \frac{\sqrt{3}\cdot L\cdot s\alpha}{3} & \frac{c_t\cdot kd_7\cdot c\alpha}{c_f} - \frac{\sqrt{3}\cdot L\cdot s\alpha}{3} & \frac{c_t\cdot kd_8\cdot c\alpha}{c_f} - \frac{\sqrt{3}\cdot L\cdot s\alpha}{3}
\end{matrix}\right] \tag{A.1}$$

Bibliography

[1] D. Brescianini and R. D'Andrea, "Design, modeling and control of an omni-directional aerial vehicle," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3261–3266, May 2016.

[2] D. Brescianini and R. D'Andrea, "An omni-directional multirotor vehicle," *Mechatronics*, vol. 55, pp. 76–93, Nov. 2018.

[3] S. Park, J. Her, J. Kim, and D. Lee, "Design, modeling and control of omni-directional aerial robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1570–1575, Oct. 2016. ISSN: 2153-0866.

[4] S. Park, J. Lee, J. Ahn, M. Kim, J. Her, G.-H. Yang, and D. Lee, "ODAR: Aerial Manipulation Platform Enabling Omnidirectional Wrench Generation," *IEEE/ASME Transactions on Mechatronics*, vol. 23, pp. 1907–1918, Aug. 2018. Conference Name: IEEE/ASME Transactions on Mechatronics.

[5] M. Allenspach, K. Bodie, M. Brunner, L. Rinsoz, Z. Taylor, M. Kamel, R. Siegwart, and J. Nieto, "Design and optimal control of a tiltrotor micro-aerial vehicle for efficient omnidirectional flight," *The International Journal of Robotics Research*, vol. 39, pp. 1305–1325, Sept. 2020. Publisher: SAGE Publications Ltd STM.

[6] M. Brunner, K. Bodie, M. Kamel, M. Pantic, W. Zhang, J. Nieto, and R. Siegwart, "Trajectory Tracking Nonlinear Model Predictive Control for an Overactuated MAV," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5342–5348, May 2020. ISSN: 2577-087X.

[7] M. Ryll, D. Bicego, M. Giurato, M. Lovera, and A. Franchi, "FAST-Hex—A Morphing Hexarotor: Design, Mechanical Implementation, Control and Experimental Validation," *IEEE/ASME Transactions on Mechatronics*, vol. 27, pp. 1244–1255, June 2022. Conference Name: IEEE/ASME Transactions on Mechatronics.

[8] M. Hamandi, F. Usai, Q. Sablé, N. Staub, M. Tognon, and A. Franchi, "Design of multirotor aerial vehicles: A taxonomy based on input allocation," *The International Journal of Robotics Research*, vol. 40, pp. 1015–1044, Aug. 2021. Publisher: SAGE Publications Ltd STM.

[9] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi, "Nonlinear Model Predictive Control with Enhanced Actuator Model for Multi-Rotor Aerial Vehicles with Generic Designs," *Journal of Intelligent & Robotic Systems*, vol. 100, pp. 1213–1247, Dec. 2020.

[10] J. Shahmoradi, E. Talebi, P. Roghanchi, and M. Hassanalian, "A Comprehensive Review of Applications of Drone Technology in the Mining Industry," *Drones*, vol. 4, p. 34, Sept. 2020. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

[11] L. Matikainen, M. Lehtomäki, E. Ahokas, J. Hyyppä, M. Karjalainen, A. Jaakkola, A. Kukko, and T. Heinonen, "Remote sensing methods for power line corridor surveys," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 119, pp. 10–31, Sept. 2016.

[12] B. Mishra, D. Garg, P. Narang, and V. Mishra, "Drone-surveillance for search and rescue in natural disaster," *Computer Communications*, vol. 156, pp. 1–10, Apr. 2020.

[13] D. Câmara, "Cavalry to the rescue: Drones fleet to help rescuers operations over disasters scenarios," in *2014 IEEE Conference on Antenna Measurements & Applications (CAMA)*, pp. 1–4, Nov. 2014.

[14] J. Leitloff, D. Rosenbaum, F. Kurz, O. Meynberg, and P. Reinartz, "An Operational System for Estimating Road Traffic Information from Aerial Images," *Remote Sensing*, vol. 6, pp. 11315–11341, Nov. 2014. Number: 11 Publisher: Multidisciplinary Digital Publishing

Institute.

[15] M. Moshref-Javadi and M. Winkenbach, "Applications and Research avenues for drone-based models in logistics: A classification and review," *Expert Systems with Applications*, vol. 177, p. 114854, Sept. 2021.

[16] A. Ollero, M. Tognon, A. Suarez, D. Lee, and A. Franchi, "Past, Present, and Future of Aerial Robotic Manipulators," *IEEE Transactions on Robotics*, vol. 38, pp. 626–645, Feb. 2022. Conference Name: IEEE Transactions on Robotics.

[17] D. K. D. Villa, A. S. Brandão, and M. Sarcinelli-Filho, "A Survey on Load Transportation Using Multirotor UAVs," *Journal of Intelligent & Robotic Systems*, vol. 98, pp. 267–296, May 2020.

[18] M. Tognon, H. A. T. Chávez, E. Gasparin, Q. Sablé, D. Bicego, A. Mallet, M. Lany, G. Santi, B. Revaz, J. Cortés, and A. Franchi, "A Truly-Redundant Aerial Manipulator System With Application to Push-and-Slide Inspection in Industrial Plants," *IEEE Robotics and Automation Letters*, vol. 4, pp. 1846–1851, Apr. 2019. Conference Name: IEEE Robotics and Automation Letters.

[19] J. R. Kutia, K. A. Stol, and W. Xu, "Aerial Manipulator Interactions With Trees for Canopy Sampling," *IEEE/ASME Transactions on Mechatronics*, vol. 23, pp. 1740–1749, Aug. 2018. Conference Name: IEEE/ASME Transactions on Mechatronics.

[20] L. Peric, M. Brunner, K. Bodie, M. Tognon, and R. Siegwart, "Direct Force and Pose NMPC with Multiple Interaction Modes for Aerial Push-and-Slide Operations," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 131–137, May 2021. ISSN: 2577-087X.

[21] M. Ryll, G. Muscio, F. Pierri, E. Cataldi, G. Antonelli, F. Caccavale, D. Bicego, and A. Franchi, "6D interaction control with aerial robots: The flying end-effector paradigm," *The International Journal of Robotics Research*, vol. 38, pp. 1045–1062, Aug. 2019.

[22] H. C. Yang, R. AbouSleiman, B. Sababha, E. Gjioni, D. Korff, and O. Rawashdeh, "Implementation of an Autonomous Surveillance Quadrotor System," in *AIAA Infotech@Aerospace Conference*, American Institute of Aeronautics and Astronautics, 2009. _eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2009-2047.

[23] F. A. Goodarzi, "Autonomous aerial payload delivery with quadrotor using varying length cable," in *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pp. 394–399, Nov. 2016. ISSN: 2325-0690.

[24] S. Bouabdallah and R. Siegwart, "Full control of a quadrotor," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 153–158, Oct. 2007. ISSN: 2153-0866.

[25] S. Bouabdallah, A. Noth, and R. Siegwart, "PID vs LQ control techniques applied to an indoor micro quadrotor," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2451–2456 vol.3, Sept. 2004.

[26] S. Rajappa, M. Ryll, H. H. Bülthoff, and A. Franchi, "Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4006–4013, May 2015. ISSN: 1050-4729.

[27] M. Ryll, G. Muscio, F. Pierri, E. Cataldi, G. Antonelli, F. Caccavale, and A. Franchi, "6D physical interaction with a fully actuated aerial robot," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5190–5195, May 2017.

[28] R. Rashad, J. Goerres, R. Aarts, J. B. C. Engelen, and S. Stramigioli, "Fully Actuated Multirotor UAVs: A Literature Review," *IEEE Robotics & Automation Magazine*, vol. 27, pp. 97–107, Sept. 2020. Conference Name: IEEE Robotics & Automation Magazine.

[29] A. Franchi, R. Carli, D. Bicego, and M. Ryll, "Full-Pose Tracking Control for Aerial Robotic Systems With Laterally Bounded Input Force," *IEEE Transactions on Robotics*, vol. 34, pp. 534–541, Apr. 2018. Conference Name: IEEE Transactions on Robotics.

[30] M. Ryll, H. H. Bülthoff, and P. R. Giordano, "A Novel Overactuated Quadrotor Unmanned Aerial Vehicle: Modeling, Control, and Experimental Validation," *IEEE Transactions on Control Systems Technology*, vol. 23, pp. 540–556, Mar. 2015. Conference Name: IEEE Transactions on Control Systems Technology.

[31] M. Odelga, P. Stegagno, and H. H. Bülthoff, "A fully actuated quadrotor UAV with a propeller tilting mechanism: Modeling and control," in *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 306–311, July 2016.

[32] M. Kamel, S. Verling, O. Elkhatib, C. Sprecher, P. Wulkop, Z. Taylor, R. Siegwart, and I. Gilitschenski, "The Voliro Omniorientational Hexacopter: An Agile and Maneuverable Tiltable-Rotor Aerial Vehicle," *IEEE Robotics & Automation Magazine*, vol. 25, pp. 34–44, Dec. 2018. Conference Name: IEEE Robotics & Automation Magazine.

[33] K. Bodie, Z. Taylor, M. Kamel, and R. Siegwart, "Towards Efficient Full Pose Omnidirectionality with Overactuated MAVs," in *Proceedings of the 2018 International Symposium on Experimental Robotics* (J. Xiao, T. Kröger, and O. Khatib, eds.), Springer Proceedings in Advanced Robotics, (Cham), pp. 85–95, Springer International Publishing, 2020.

[34] M. Jacquet, G. Corsini, D. Bicego, and A. Franchi, "Perception-constrained and Motor-level Nonlinear MPC for both Underactuated and Tilted-propeller UAVS," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4301–4306, May 2020. ISSN: 2577-087X.

[35] M. Jacquet and A. Franchi, "Motor and Perception Constrained NMPC for Torque-Controlled Generic Aerial Vehicles," *IEEE Robotics and Automation Letters*, vol. 6, pp. 518–525, Apr. 2021. Conference Name: IEEE Robotics and Automation Letters.

[36] A. Alharbat, H. Esmaeeli, D. Bicego, A. Mersha, and A. Franchi, "Three Fundamental Paradigms for Aerial Physical Interaction Using Nonlinear Model Predictive Control," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 39–48, June 2022. ISSN: 2575-7296.

[37] G. Corsini, M. Jacquet, H. Das, A. Afifi, D. Sidobre, and A. Franchi, "Nonlinear Model Predictive Control for Human-Robot Handover with Application to the Aerial Case," in *The 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022)*, (Kyoto, Japan), Oct. 2022.

[38] M. Ryll, D. Bicego, and A. Franchi, "Modeling and control of FAST-Hex: A fully-actuated by synchronized-tilting hexarotor," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1689–1694, Oct. 2016. ISSN: 2153-0866.

[39] Y. Chen, M. Bruschetta, E. Picotti, and A. Beghi, "MATMPC - A MATLAB Based Toolbox for Real-time Nonlinear Model Predictive Control," in *2019 18th European Control Conference (ECC)*, pp. 3365–3370, June 2019.

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

[41] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: a parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, pp. 327–363, Dec. 2014.

[42] M. Diehl, H. Bock, and J. Schloder, "Real-Time Iterations for Nonlinear Optimal Feedback Control," in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 5871–5876, Dec. 2005. ISSN: 0191-2216.

[43] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, “From linear to nonlinear MPC: bridging the gap via the real-time iteration,” *International Journal of Control*, vol. 93, pp. 1–19, Sept. 2016.