Utilizing Model Predictive Control to Haptically Assist Users in Piloting a Quadcopter

MSc. Thesis: System & Control
Mechanical Engineering

Isabelle van Osnabrugge

TU Delft

Utilizing Model Predictive Control to Haptically Assist Users in Piloting a Quadcopter

by

Isabelle van Osnabrugge

Student Number: 4345517
Instructor: L. Marchal-Crespo
Teaching Assistant: R. Ferrari
Project Duration: August 25th, 2023 - March 4th, 2025
Faculty: Faculty of Mechanical Engineering, Delft

TU Delft

Preface

The process of researching and writing this thesis has been a challenging experience. Not only in terms of academic pursuits but also in terms of personal growth. When I first started began working on my thesis topic, Laura helped me a lot by accommodating for both my interests, but also by helping find a topic suitable for a student working on finishing a double degree.

This journey has allowed me to explore fields I was not overly familiar with before, such as motor learning, haptic feedback, and model predictive control. One of the most rewarding experiences during the project was working directly with a haptic robot—a piece of technology I have never had the opportunity to work with before. This hands-on experience, coupled with the task of creating a virtual game in Unity, had me try a variety of different things throughout my thesis, which was something that I have thoroughly enjoyed.

My thesis was submitted in accordance with the requirements for my master degrees in Systems and Control, and Mechanical Engineering (Track Biorobotics) at the Delft University of Technology. My supervisors throughout the project were Dr. Ing. Laura Marchal-Crespo and Dr. Riccardo Ferrari at the Department of Cognitive Robotics within the Faculty of Mechanical, Maritime, and Materials Engineering. Early in the project, I was also supported by phd students Stefano Dalla Gasperina and Jean Gonzalez Silva.

Isabelle van Osnabrugge
Delft, February 2025

Acknowledgements

I would like to express my thanks to my two main supervisors, Laura Marchal-Crespo and Riccardo, for their support throughout the course of my thesis. Their insightful feedback, interesting ideas and the critical eye on my work helped shape my work. But most of all, I am grateful for their unwavering encouragement and understanding during the challenging moments, both in my personal life and throughout the course of the project. Additionally, I would also like to acknowledge the support of my two daily supervisors, Stefano Dalla Gasperina and Jean Gonzalez Silva, who guided me during the initial phase of my project. Although they have since moved on to pursue their own careers, their support was much appreciated throughout my literature research and the start of my thesis project.

On a personal note, I want to thank my friends and family. Their emotional and physical support has meant the world to me during, what were probably some of, the most demanding moments of this journey.

Lastly, I would like to express my deepest appreciation to everyone who has been a part of this thesis journey——whether directly or indirectly——without whose help and support this work likely would not have been possible.

Thank you so much everyone.

Table of Content

Table of Content

Utilizing Model Predictive Control to Haptically Assist Users in Piloting a Quadcopter

Isabelle van Osnabrugge, MSc Student, University of Technology Delft
Departments of Systems and Control, Biomechanical Design, and Cognitive Robotics

***Abstract*—Haptic technology focuses on the recreation of haptic information, i.e., a type of sensory input that uses tactile cues, forces, vibrations, or pressure to provide users with the sensation of touch, enabling users to interact physically with virtual or remote environments. One promising application of this technology lies in haptic training, where the possibility of using haptic feedback to facilitate or promote motor learning is studied. The focus of this paper lies on performance-enhancing haptic training methods, with a focus on designing a dynamic motor task. The objective of this paper is, therefore, to establish a preliminary framework that can be used to provide minimal haptic feedback while flying a quadcopter through a set of gates. We focused on creating a preliminary framework that provides haptic feedback on the altitudinal axis of the quadcopter to the pilot using the control method Model Predictive Control (MPC). The haptic feedback is provided on the z-axis of a haptic Sigma.7 robot, which is also used as a remote controller to fly the quadcopter. The MPC implements the dynamical models of the quadcopter, and a haptic Sigma.7 robot, to determine the minimal force required to steer the Sigma.7 robot towards motor task completion. The system should provide minimal haptic force feedback within the proposed design requirements to prevent reliance on the assistance. We evaluated the effectiveness of our framework by evaluating its ability to control the quadcopter to the desired altitude setpoint under autonomous conditions using a haptic Sigma robot. Additionally, the design and performance of each of the individual building blocks of this framework, i.e. the quadcopter model, the haptic interface, and the MPC, were evaluated separately. The quadcopter, with the implementation of the onboard PID controllers, eliminating the steady-state errors and meeting the required settling times. The Sigma.7 model was sufficient within the established time horizon and range of operation, although shows limitations due to unmodelled frictional forces. The completed framework is capable of providing the Sigma.7 with the necessary input command to autonomously guide the quadcopter to its desired references in real-time, therefore completing its primary objective. Future work should explore improving the model components and integrating human elements into the predictive model.**

***Index Terms*—Haptic Technology, Kinesthetic Haptic Feedback, Motor Learning, Model Predictive Control, MPC, Dynamic Motor Tasks, Haptic Training, Quadcopter, Sigma.7, Unity.**

I. Introduction

HAPTIC technology focuses on recreating haptic information—a type of sensory input that uses tactile cues, forces, vibrations, or pressures to provide users with the sensation of touch [1]. It paves the way for people to physically interact with remote or virtual environments. A robotic arm may, for example, enable a person to feel the impact of a tennis racket in a virtual environment as depicted in Figure 1. The combination of haptic technology with virtual environments provides the possibility of facilitating personalized, replicable, and diverse virtual training environments, which may otherwise be too complex, intense, risky, or even expensive to implement in the real world. This has some interesting implications for motor learning, as it raises the question whether the addition of haptic technology is capable of facilitating or even enhancing the process, leading to the development of haptic training.

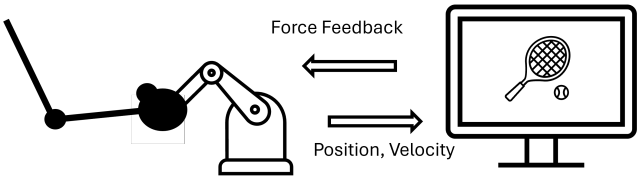

Fig. 1. Haptic technology enables the sensation of playing a virtual tennis game, where the player, depicted on the left, can feel the impact of the tennis ball using a haptic interface, e.g. the robotic arm.

Haptic training is an instructional method that leverages the provision of haptic information using haptic technology to facilitate motor learning. Within haptic training methods, one category is performance-enhancing methods, wherein a learner may be guided toward an optimal trajectory teaching the necessary features of the motor task in the process [2, 3]. Numerous motor learning studies have examined the efficacy of performance-enhancing haptic training methods. However, these studies have yielded widely varying results, making for very little consistent evidence that such guidance is beneficial for human motor learning [3, 4, 5]. It is even theorized that this may be due to the very nature of such guidance. In the field of motor learning, one prevalent concept is the *Guidance Hypothesis* [6]. The Guidance Hypothesis is a theory in motor learning that proposes over-reliance and excessive physical or cognitive assistance during skill acquisition can hamper the motor learning process [7]. Another theory in the world of motor learning known as the *Challenge Point framework* conceptualizes the idea that there is an optimal level of functional task difficulty or challenge that can facilitate skill acquisition and retention in a learner. It suggests that the level of task difficulty should be tailored to the current skill level of the learner, and that learning may be obstructed in the presence of either excessive or insufficient information. Studies have shown tentative results that haptic guidance as

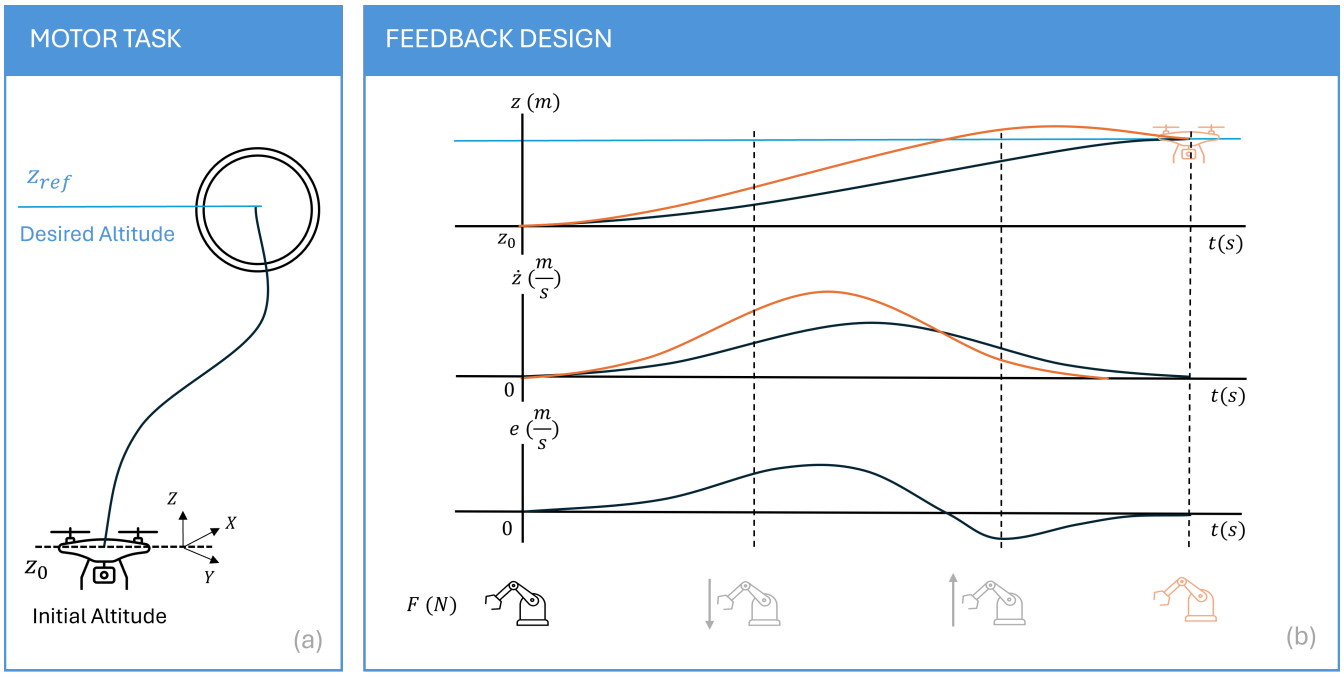

Fig. 2. (a) In this example, one may observe a quadcopter being flown to a desired altitude z_{ref} from an initial altitude z_0. (b) To reach the desired altitude, we may compare the ideal behaviour with that of a learning pilot. In this figure, the quadcopter altitude z, the altitude rate $\dot{z}$, and the altitude rate error $e = \dot{z} - \dot{z}_{\text{ideal}}$ are depicted in the form of graphs over time. In the first two graphs, the black line in the image represents the ideal path, whereas a learning pilot may fly the trajectory depicted in orange. We may then observe that in the altitude rate graph, the learning pilot has slightly increased the drone's velocity beyond what would be the ideal case. This excessive behaviour is represented as an error in the third plot. Whenever the learning pilot deviates from the ideal scenario the error will increase, it is this error that is used to determine the desired force-feedback F. The haptic interface provides the pilot with this force-feedback towards the correct action in an attempt to guide them towards the better action. The magnitude of the force feedback is, in turn, determined by a Model Predictive Controller.

a training method is significantly more effective when the individual is initially less skilled at the task than when they are already somewhat familiar with it [2, 8]. Together, these theorems imply that haptic feedback is ideally only provided when necessary, such that the motor task remains at adequate complexity during execution and preventing overreliance on the haptic assistance.

While the field of motor learning has investigated the effectiveness of robot-mediated haptic training methods in learning simple motor tasks, there is still a lack of systematic research on their effectiveness in learning dynamic motor tasks, i.e., tasks with dynamic complexity [3, 4]. These tasks play a crucial role in our daily lives, from routine tasks such as commuting by bicycle or car, to sports such as tennis or golf. These tasks require a person to manipulate objects with complex dynamics, necessitating experience, and understanding of how to handle these objects to apply them in context effectively. There is no guarantee that the findings regarding simple motor tasks may be generalized to dynamic tasks. Research even suggests that methods known to enhance learning for simple tasks should be reevaluated for more complex and dynamic tasks [3, 4].

One objective of this paper is to establish a framework for training motor tasks with dynamic complexity that may potentially be used to evaluate motor learning. The first requirement is to choose a relevant motor task of dynamic complexity, such as piloting a quadcopter [9, 10]. Quadcopters are increasingly used in professional occupations such as cinematography, logistics, surveillance, and agriculture, as well as for recreational purposes such as First-Person-View (FPV) racing and photography [11]. The main objective of the task takes inspiration from FPV racing, namely to effectively fly a quadcopter through a set of rings [10]. The remote controller conventionally used to fly the quadcopter is replaced with a sigma.7 haptic robot, where haptic feedback is provided to the user solely on the z-axis, to achieve a desired altitude during the motor task.

But how do we define the optimal force feedback in this scenario? The ideal situation is to provide feedback only when necessary—in other words, minimal force feedback should be provided to the pilot when the quadcopter is moving in the right direction. Therefore, force feedback is determined based on the error in velocity while aiming to reach a desired position, as illustrated in Figure 2. To evaluate whether the quadcopter's current behaviour is optimal, an ideal velocity trajectory must be determined, one that results in it reaching the objective position. To further explore the idea of providing minimal haptic feedback whilst achieving this objective position, the control method Model Predictive Control (MPC) was chosen. MPC has recently gained attention in the field of haptic training for its potential to achieve situational-dependent

minimal force feedback [3, 12, 13, 14]. MPC is an optimal control method that can determine the optimal trajectory in real-time, allowing for minimized force feedback based on the current behaviour of an individual and a known model of the system [15, 16]. The MPC can determine, within the system's constraints in its current state, the ideal velocity trajectory the quadcopter should follow to approach the desired altitude over a given time horizon while also minimizing the provided force feedback.

This paper strives to use the implementation of the MPC to create a preliminary framework to provide minimal force feedback during task execution, with the overall objective of motor task completion. To this end, the framework is tested on its ability to complete the motor task under autonomous circumstances. The MPC provides the Sigma.7 haptic robot with relevant force commands such that the quadcopter may be flown towards a reference altitude. In this manner, the system viability may be verified and its performance in task completion evaluated.

In this paper, we first present a general overview of the framework in subsection II-A, defining the necessary building blocks of the project, including their desired functionalities and the available devices. Then a more detailed explanation of the individual building blocks is provided in subsection II-B and subsection II-C, elaborating on their design processes and implementation. Lastly, we evaluate and discuss the framework in section III and section IV, noting any shortcomings and potential improvements for future work.

II. Methods

A. System Overview

To create a framework that implements MPC to promote motor learning of a Quadcopter Pilot, four main building blocks are required:

- A dynamical model or physical system of an Unmanned Aerial Vehicle (UAV) quadcopter. This paper discusses a simplified theoretical model of a quadcopter in subsection II-B.
- The haptic interface serves both as the remote control input for the quadcopter, providing user commands to the drone, and as a means of providing haptic feedback to the user. The available device is the Force Dimension Sigma.7 end-effector.
- The motor task, and visualization thereof. This is accomplished in Unity.
- An MPC that will determine the magnitude of the force-feedback provided back to the user, which will be discussed in more detail in subsection II-C.

In Figure 3 the interconnections between each building block are depicted and how haptic feedback is provided during motor task execution.

Each of the system components operates across different software platforms: Unity for task visualization, MATLAB and Simulink for simulating the system dynamics and control, and Microsoft Visual Studio to compile the software required for communication with the Sigma.7 haptic interface. The intercommunication across various platforms is accomplished using User Datagram Protocols (UDP).

Quadcopter

Unmanned Aerial Vehicles (UAV), also commonly referred to collectively as drones, are aerial vehicles capable of flying either autonomously using onboard equipment, or through various remote controllers such as joysticks, smartphone applications, and occasionally voice, or gesture-driven communicative methods.

The *quadcopter* is a specific type of multirotor drone, lending its name to the often symmetrical, four rotors used for propulsion (an example is depicted in Figure 4). It is the most common drone flown and commercially available, likely lending its success to its relatively mechanical simplicity [9, 17]. Due to its commercial success, mechanical simplicity, and general availability to the public, it is a viable choice to implement in this work [9].

Fig. 4. An example of a quadcopter UAV, the E58 mini drone.

In the world of First-Person View (FPV) racing and hobbyist drone flying, there are several ways in which an UAV may be flown manually. The most popular mode is that of *acrobatic mode*. This mode uses the inputs from the Radio Controller (RC) sticks to control the angular velocities of the UAV frame on each axis. It is considered to be the most intuitive manner by which to fly, however, it is also one of the most difficult [9] [10]. Manually flying the drone in this mode implies no stabilization around any axis during flight, the pilot is continuously prompted to correct the drone's altitude, roll, pitch, and yaw angles. Due to this difficulty, there is another mode considered to be more beginner-friendly and generally easier to learn. This mode is known as *stabilized mode*, also sometimes referred to as *angle mode*. This mode is the same as acrobatic mode but controls the pitch and roll angles instead of their rates. This results in a self-levelling effect at zero input, making self-correction much easier in practice. To increase the feasibility of dynamic motor task completion, this is the flight mode implemented in this study. The model itself is

implemented in MATLAB/Simulink, the process of which will be discussed in more detail in subsection II-B.

Haptic Interface

To provide haptic feedback to the quadcopter pilot, a Force Dimension Sigma.7 haptic device is used (see Figure 5). The Sigma.7 is an end-effector haptic device with 7 degrees of freedom capable of providing feedback for both forces and torques as depicted in Table I. It has three translational axes, three rotational axes (axis 0-2), and a gripping mechanism (axis 3) at the end effector.

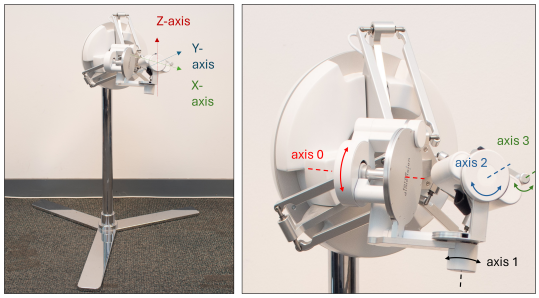

Fig. 5. The Force Dimension Sigma.7. Image courtesy and property of the Human-Robot Interaction Group (HRI) of Delft University of Technology.

TABLE I
THE TECHNICAL SPECIFICATIONS OF SIGMA.7 DEVICE [18]

	Axis	Value
Workspace	Translation	⌀ 190 × 130 mm
	Rotation	235 × 140 × 200 deg
	Gripper	25 mm
Forces	Translation	20.0 N
	Rotation	400 mNm
	Gripper	8.0 N

For the current setup, the Sigma.7 device communicates via USB with a Windows computer, reading and processing information using the executable C/C++ code compiled in Microsoft Visual Studios 2022 IDE. The code implements the provided Force Dimension SDK tools to facilitate the communication with the Sigma.7. The bandwidth of the kinaesthetic sensing system of the human body has been estimated at 20 to 30 or even 50 Hz [19, 20]. This is considered the absolute lower bound to the sampling frequency to keep the experience smooth for the human operator. As such, data are read at a sampling period of 0.01 s [20] keeping in mind potential delayed packages due to the UDP communication.

To operate the quadcopter, four of the device axes are used as references to the internal drone controllers, the z-axis, and the three rotational axes. These axes directly correspond to their associated quadcopter behaviour for flight in stabilized mode; the z-axis to the desired altitude rate ($\dot{z}_{\text{ref}}$), axis 0 to the desired roll angle (ϕ_{ref}), axis 1 to the desired pitch angle (θ_{ref}), and axis 2 to the desired yaw rate ($\dot{\psi}_{\text{ref}}$).

To provide further support to the pilot, additional constraints are applied. The x-y plane of the haptic device is constrained to zero such that the pilot only needs to focus on the prior mentioned four axes. The constraints are implemented using Proportional-Derivative (PD) controllers, depicted in Equation 1 and Equation 2, to provide a spring-damper-like behaviour encouraging the pilot to remain centred during task completion.

$$F_{x,\text{s}} = -Kx_{\text{s}} - B\dot{x}_{\text{s}} \tag{1}$$
$$F_{y,\text{s}} = -Ky_{\text{s}} - B\dot{y}_{\text{s}}, \tag{2}$$

where $K = 300$ N/m and $B = 6$ N/(m/s) denote the gains of the PD controller in the form of spring and damper coefficients. x_{s} and y_{s} represent the Cartesian X, and Y coordinates of the Sigma.7 device. The gains were determined through trial and error, with the objective of reasonable pushback, the response should not be entirely rigid to allow for comfortable interaction by the pilot yet provide some pushback towards the zero reference. Additionally, it was required that the end-effector returns to the zero reference with a reasonably small error when it is let go of by the pilot.

Similarly, on the z-axis, we included two PD controllers at both ends of the workspace. These measures are mainly to pre-

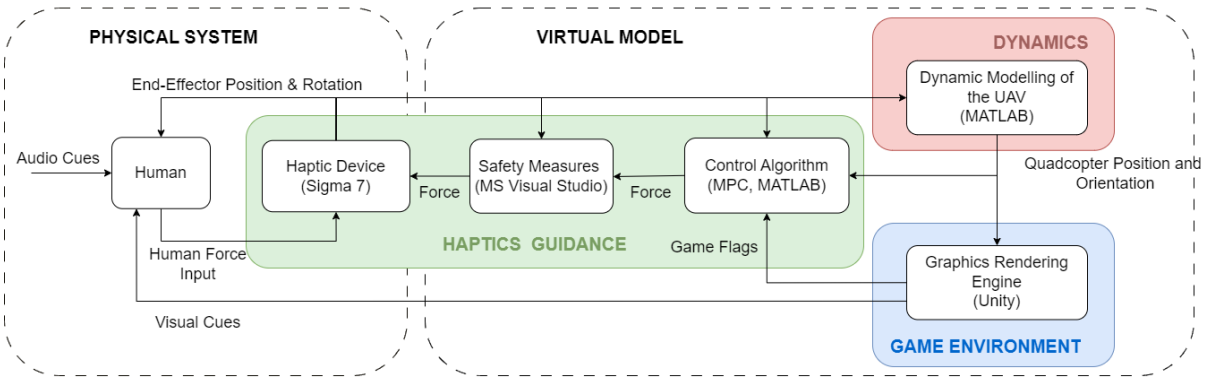

Fig. 3. A general overview of the framework: A real-world end-effector device (Sigma.7) is used to provide haptic feedback to the user and to measure the end-effector's position and rotational information as input references to the virtual quadcopter model. In Simulink/MATLAB, the quadcopter model dynamics are simulated, and the updated states are visualized in Unity. Using the measured inputs, and the quadcopter output measurements, a Model Predictive Controller (MPC) determines the minimal haptic feedback required and sends this information back to the Sigma.7 as the user executes their motor task.

vent mechanical damage to the device due to the end-effector colliding with the physical boundaries of its workspace.

$$F_{z,s,ub} = -K_{\text{safety}}(z_s - 0.1 \text{ m}) - B_{\text{safety}}\dot{z}_s \quad (3)$$

$$F_{z,s,lb} = -K_{\text{safety}}(z_s + 0.08 \text{ m}) - B_{\text{safety}}\dot{z}_s, \quad (4)$$

where $K_{\text{safety}} = 600\,\text{N/m}$ and $B_{\text{safety}} = 6\,\text{N/(m/s)}$. Similarly to before, z_s represent the Cartesian Z coordinate of the Sigma.7 device

Motor Task and Visualization

In the FPV racing circuit, one very common manner to practice flying and controlling the drone in 3D space is to use race gates. Gates are a target through which to fly the drone. They are often in the shape of ringlike structures made from flexible material. The gates may be positioned near the ground or higher up in the air, forming a parkour course within which to practice. Taking inspiration from this, we designed a similar motor task that requires flying through a set of rings positioned at varying heights.

The quadcopter and motor task were created and implemented in a virtual environment using Unity (2021.3.17f1), in the form of a simple game environment, represented in Figure 6. The goal of the operator is to fly the quadcopter through the successive gates. The quadcopter is visualized using a quadcopter Unity asset acquired from the Unity Asset store with a Standard Unity Asset Store EULA license agreement. Redistribution is not permitted, for access please refer to original source at https://assetstore.unity.com/packages/3d/vehicles/air/realistic-drone-66698.

A variety of torus rings are then used to represent the gates. The center y-coordinate of the gates represent the reference altitudes (z_{ref}) depicted in Figure 2 and provided to the MPC. A standard Unity box-collider is then applied to check whether the quadcopter is successfully flown through the torus ring during task execution. The red cube in Figure 6 designates the starting location.

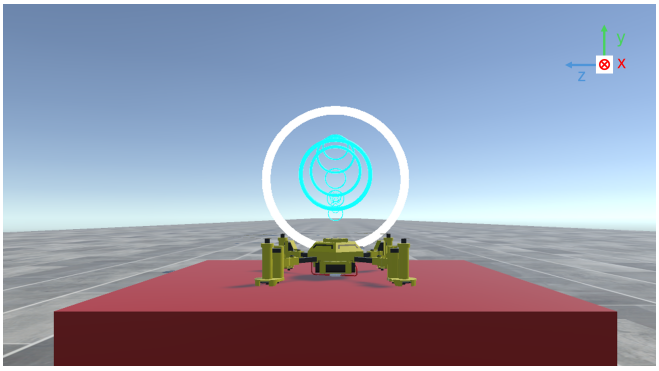

Fig. 6. The quadcopter model in Unity at its starting position. In the background, the gates are visible.

Model Predictive Controller

Model Predictive Control is known as an optimal control policy capable of dealing with multivariate problems and incorporating constraints. It uses a model of the system to make predictions as to the future states of the system and uses these predictions to, in turn, solve a constrained optimization problem. This optimization can be applied online, allowing for a flexible use of trajectories and force feedback in the context of haptic feedback. In this framework, the MPC is implemented in MATLAB/Simulink, the design of which will be discussed in more detail in subsection II-C.

Network Protocol

The communication between different software environments is done using the User Datagram Protocol (UDP). It is a suitable transmission protocol for real-time systems, where waiting for packages and retransmission are a larger burden on the system than dropping packages during runtime. The UDP is used to provide communication lines between different software environment on the same computer. One IP-adress is used with several ports to create two UDP lines, sharing information back and forth:

1) UDP Link 1

   The first UDP provides a communication line between the executable code compiled in the Microsoft Visual Studio (C++) environment, where the Force Dimension Sigma.7 haptic interface data is being read and processed, and the MATLAB Simulink environment. The main purpose of this link is to transmit the desired inputs to the quadcopter model, and the required force feedback to the Sigma.7 device.

2) UDP Link 2

   The second UDP communication line between Unity (C#) and MATLAB/Simulink [21]. It is primarily used to communicate the information necessary to visualize and interact with the virtual environment of the motor task. The states of the quadcopter are transmitted to Unity, and in turn, the completion flags and altitude references (z_{ref}) of the motor task are sent back to MATLAB/Simulink.

Both UDP Links communicate at a sampling period of 0.01 seconds. An overview of the complete framework may be seen in Figure 7.

B. Quadcopter UAV Model

The quadcopter is a Multi-Input Multi-Output (MIMO) system with six degrees of freedom—three rotational and three positional—, actuated by four independent rotors. The dynamics of a quadcopter are inherently nonlinear, underactuated, and influenced by numerous aerodynamic uncertainties. Due to this complexity, various assumptions are typically made to simplify the mathematical models required to work with these systems. The most common assumptions in literature often include the following and are considered reasonable for the current model as well [22]:

- The quadcopter frame is considered to be rigid, and of a symmetrical structure, resulting in a corresponding diagonal inertia matrix.

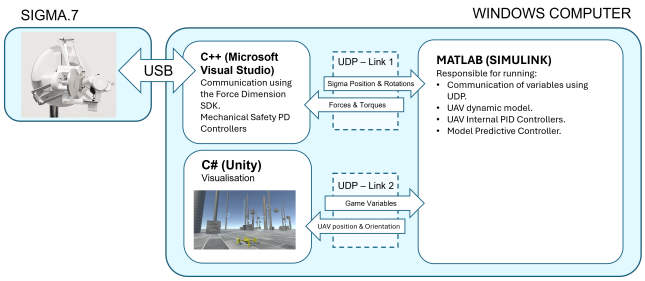

Fig. 7. This figure represents a more in-depth view of the general framework. The Sigma.7 haptic interface communicates via USB with a (Windows) computer. The positional and rotational information of the end-effector is read, processed, and stored using the Force Dimension SDK in the Microsoft Visual Studio 2022 executable. Additionally, the mechanical safety PD controllers are also run here. In MATLAB, the quadcopter dynamics and its onboard controllers are implemented. Lastly, the visualization of the task is done in Unity. Two UDP links are used to communicate between the different software environments on the computer.

- The Center of Gravity (CoG) is positioned at the centre of the quadcopter frame.
- The system is considered to be time-invariant.
- Specific aerodynamical effects, such as blade-flapping due to rotor design, are not considered. Similarly, the ground effect, which describes the interaction effect of the quadcopter with the ground when it is close, is also neglected [23]. This effect generally results in an increase in lift and a decrease in aerodynamic drag.
- All the motors positioned on the quadcopter are considered identical.
- The thrust and drag constants are proportional to the square of the motor speed.
- The thrust generated by each of the motors is proportional to the square of the motor speed.

Nonlinear Dynamics

The quadcopter drone model dynamics are derived from the papers by Musa [22], and Zulu and Samuel [24], with additional simplified aerodynamic friction. More detail on the derivations may be found in the Appendix in section I. Together with the assumptions, the nonlinear equations of motion (EqoM) of the quadcopter are formulated as

$$\ddot{x} = -\frac{U_1}{m}(\mathrm{c}_\phi \mathrm{s}_\theta \mathrm{c}_\psi + \mathrm{s}_\phi \mathrm{s}_\psi) - \frac{K_t}{m}\dot{x} \tag{5}$$

$$\ddot{y} = -\frac{U_1}{m}(\mathrm{c}_\phi \mathrm{s}_\theta \mathrm{s}_\psi - \mathrm{s}_\phi \mathrm{c}_\psi) - \frac{K_t}{m}\dot{y} \tag{6}$$

$$\ddot{z} = -g + \frac{U_1}{m}\mathrm{c}_\phi \mathrm{c}_\theta - \frac{K_t}{m}\dot{z} \tag{7}$$

$$\ddot{\phi} = \left(\frac{I_{yy} - I_{zz}}{I_{xx}}\right)\dot{\theta}\dot{\psi} - \frac{K_r}{I_{xx}}\dot{\phi} + \frac{l}{I_{xx}}U_2 \tag{8}$$

$$\ddot{\theta} = \left(\frac{I_{zz} - I_{xx}}{I_{yy}}\right)\dot{\phi}\dot{\psi} - \frac{K_r}{I_{yy}}\dot{\theta} + \frac{l}{I_{yy}}U_3 \tag{9}$$

$$\ddot{\psi} = \left(\frac{I_{xx} - I_{yy}}{I_{zz}}\right)\dot{\phi}\dot{\theta} - \frac{K_r}{I_{zz}}\dot{\psi} + \frac{l}{I_{zz}}U_4 \tag{10}$$

where $\mathbf{u} = [U_1,\ U_2,\ U_3,\ U_4]^T$ represent the inputs to the system, elaborated upon in Table II. Similarly, the states of the quadcopter are represented by the vector $\mathbf{q} = [x,\ y,\ z,\ \phi,\ \theta,\ \psi,\ \dot{x},\ \dot{y},\ \dot{z},\ \dot{\phi},\ \dot{\theta},\ \dot{\psi}]^T$ and are also elaborated upon in Table II. $\mathbf{I}$ = diag(I_{xx}, I_{yy}, I_{zz}) is the diagonal inertia matrix, and K_t and K_r are the translational and rotational friction associated with aerodynamics. Lastly, m depicts the mass of the quadcopter and l the length of the quadcopter arm, i.e. the distance from the rotational axis of the motors to the centre of the quadcopter frame.

The definition of the control inputs for a quadcopter can vary depending on how its dynamics are represented; they might represent thrust, angular velocities, or even voltages. In the context of this paper, these control inputs, depicted by vector $\mathbf{u}$, represent the thrust and torques applied by the rotor combinations such that the relevant movements, vertical lift, roll, pitch, and/or yaw, are generated. The general convention regarding the definition of these control inputs and the relevant movement will be adhered to as shown in Table II.

TABLE II
DEFINITION OF THE QUADCOPTER CONTROL INPUTS AND STATES

Control Input	Description	Units
U_1	The total motor thrust, also related to the altitude rate.	N
U_2	The motor thrust related to the roll movement.	N
U_3	The motor thrust related to the pitch movement.	N
U_4	The motor thrust related to the yaw movement.	N
State	**Description**	**Units**
x	X Position in Inertial Frame $\mathcal{I}$	m
y	Y Position in Inertial Frame $\mathcal{I}$	m
z	Z Position in Inertial Frame $\mathcal{I}$	m
ϕ	Roll Attitude in Inertial Frame $\mathcal{I}$	rad
θ	Pitch Attitude in Inertial Frame $\mathcal{I}$	rad
ψ	Yaw Attitude in Inertial Frame $\mathcal{I}$	rad

Linearized System Dynamics

To use linear MPC, the nonlinear dynamics of the Quadcopter are linearized around hovering conditions. Linearizing the system around hovering conditions results in a linear time-invariant system around the operating point $(\mathbf{q}_e, \mathbf{u}_e) = (\mathbf{0}^5, \psi, \mathbf{0}^9, mg)$ results in the following system.

$$\begin{aligned}\dot{\mathbf{q}} &= \frac{\partial f(\mathbf{q},\mathbf{u})}{\partial \mathbf{q}}\Big|_{\substack{\mathbf{q}=\mathbf{q}_e \\ \mathbf{u}=\mathbf{u}_e}}(\mathbf{q}-\mathbf{q}_e) + \frac{\partial f(\mathbf{q},\mathbf{u})}{\partial \mathbf{u}}\Big|_{\substack{\mathbf{q}=\mathbf{q}_e \\ \mathbf{u}=\mathbf{u}_e}}(\mathbf{u}-\mathbf{u}_e) \\ &= \mathbf{A}_c\Delta\mathbf{q} + \mathbf{B}_c\Delta\mathbf{u} \qquad (11)\\ \mathbf{y} &= \mathbf{C}_c\Delta\mathbf{q} \qquad (12)\end{aligned}$$

Here $\mathbf{A}_c$ and $\mathbf{B}_c$ are given by

$$\mathbf{A}_c = \begin{bmatrix}
0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&-\frac{U_1}{m}\mathrm{s}_\psi&-\frac{U_1}{m}\mathrm{c}_\psi&0&-\frac{K_t}{m}&0&0&0&0&0\\
0&0&0&\frac{U_1}{m}\mathrm{c}_\psi&-\frac{U_1}{m}\mathrm{s}_\psi&0&0&-\frac{K_t}{m}&0&0&0&0\\
0&0&0&0&0&0&0&0&-\frac{K_t}{m}&0&0&0\\
0&0&0&0&0&0&0&0&0&-\frac{K_r}{I_{xx}}&0&0\\
0&0&0&0&0&0&0&0&0&0&-\frac{K_r}{I_{yy}}&0\\
0&0&0&0&0&0&0&0&0&0&0&-\frac{K_r}{I_{zz}}
\end{bmatrix} \tag{13}$$

$$\mathbf{B}_c = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & \frac{l}{I_{xx}} & 0 & 0 \\ 0 & 0 & \frac{l}{I_{yy}} & 0 \\ 0 & 0 & 0 & \frac{l}{I_{zz}} \end{bmatrix} \tag{14}$$

where c_ψ and s_ψ refer to the functions cosine and sine with their respective operators in shorthand notation. The output of the system is $\mathbf{y} = [\dot{z},\ \phi,\ \theta,\ \dot{\psi}]^T$ and state-space matrix $\mathbf{C}_c$ reflects this.

One of the states, yaw (ψ) is left as a variable to update every simulation time step, resulting in a varying state-space system. This is necessary as the operator's perspective is that of a First Person View (FPV) through a camera mounted underneath the quadcopter. As a result, it is desirable to ensure that the direction of flight has the pitch aligning with forward movement and the roll aligning with lateral movement. To adjust the operator's perspective so that it aligns with the quadcopter's body frame, the linearized system must be regularly updated around the yaw-axis.

The system should represent a simplified but realistic drone, to this end the parameters of the dynamics are inspired by [25].

TABLE III
QUADCOPTER PARAMETERS

Parameters	Value
Mass, m	0.468 [kg]
Length rotor arm (from COM), l	0.225 [m]
Inertia, I_{xx} = I_{yy}	4.856e-3 [kgm^2]
Inertia, I_{zz}	8.801e-3 [kgm^2]
Linear Drag Coefficient, K_t	0.5
Angular Drag Coefficient, K_r	0.2340

Quadcopter Control

The quadcopter will be flown in *Stabilized Mode*, as mentioned in subsection II-A. To realize this, an additional control structure is required, depicted in Figure 8.

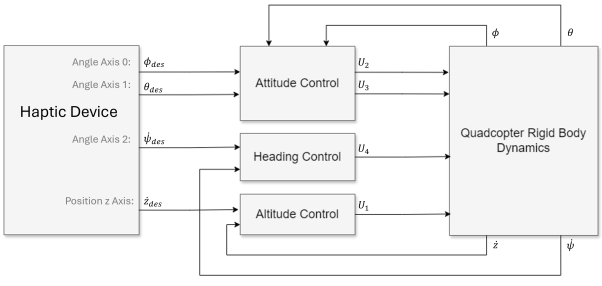

Fig. 8. The Quadrotor Control Block Diagram of the flight mode *Stabilized Mode*. The measured outputs from the z, 0, 1, and 2 axes of the Sigma.7 are used as reference inputs to the control infrastructure. The attitude, heading, and altitude controllers then determine the relevant inputs required to allow the flight of the quadcopter in *Stabilized Mode*

The control infrastructure consists of four PID controllers regulating the roll and pitch angles, and the yaw and altitude rates. These control structures can be categorized into three different types, described below.

1) Altitude Control

The altitude control structure regulates the altitude rate of the quadcopter. It maintains the desired vertical velocity using a PI control structure.

$$U_1(t) = K_{1,P}\ e_1(t) + K_{1,I} \int^t e_1(t)dt \tag{15}$$

$$e_1(t) = \dot{z}_{\text{ref}}(t) - \dot{z}(t) \tag{16}$$

2) Attitude Control

The attitude control structure consists of two identical PD controllers responsible for maintaining the desired orientation of the quadcopter. It regulates the roll and pitch angles.

$$U_{2,3}(t) = K_{2,P}\ e_{2,3}(t) + K_{2,D}\ \frac{d}{dt}\ e_{2,3}(t) \tag{17}$$

$$e_2(t) = \phi_{\text{ref}}(t) - \phi(t) \tag{18}$$

$$e_3(t) = \theta_{\text{ref}}(t) - \theta(t) \tag{19}$$

3) Heading Control

The heading control structure manages the heading, or the yaw rate, of the quadcopter. It maintains the yaw rate of the quadcopter using a PI control structure.

$$U_4(t) = K_{4,P}\ e_4(t) + K_{4,I} \int^t e_4(t)dt \tag{20}$$

$$e_4(t) = \dot{\psi}_{\text{ref}}(t) - \dot{\psi}(t) \tag{21}$$

The main objectives of these control structures are to remove any steady state errors and allow the quadcopter to reach a reference input with reasonable settling and rise times. For the pitch and roll angles, reasonable settling times are assumed within the range of $0.5 \leq t_{s,\phi,\theta} \leq 1.5\ s$ [26]. Reasonable values for the vertical acceleration of a drone seem to lie in the range of $0.5 \leq \ddot{z} \leq 5\ m/s^2$ [27] [28], thus an educated guess is made $0.2 \leq t_{s,\dot{z}} \leq 2\ s$. Lastly, the heading rate is designed with the objective to eliminate the steady state error, its uncontrolled response is already quite fast. The control parameters, depicted in Table IV, were then tuned using these objectives as guidelines to acquire reasonable quadcopter behaviour.

TABLE IV
THE QUADCOPTER PID CONTROL GAINS.

Control Gains	Value	Units	Description
$K_{1,P}$	6	kg/s	Altitude Control Proportional Gain.
$K_{2,P}$	3	kgm/s^2	Attitude Control Proportional Gain.
$K_{4,P}$	2	kgm/s	Heading Control Proportional Gain.
$K_{1,I}$	2	kg/s^2	Altitude Control Integral Gain.
$K_{4,I}$	5	kgm/s^2	Heading Control Integral Gain.
$K_{2,D}$	1	kgm/s	Attitude Control Derivative Gain.

C. Model Predictive Control design

In subsection II-B, the quadcopter design is discussed. With the availability of a quadcopter model, a virtual environment, and a motor task, we have effectively created a virtual Quadcopter simulator. In subsection II-C the force feedback design, and consequently the implementation of a Model Predictive Controller will be discussed, with further details on the design available in section IV.

As mentioned in the introduction, the ideal situation is to provide feedback only when necessary—in other words, minimal force feedback should be provided when the quadcopter is moving in the right direction. However, the velocity error and the provided force feedback by the Sigma.7 are variables naturally in opposition to one another. To decrease the velocity error, the force feedback must be increased to incite movement, but excessive increase of force feedback is undesirable. As a result, minimizing both requires a compromise or, in this case, an optimization process. Which is where the MPC comes into play. The main objective of the MPC is to drive the states of the system to their desired set points as the control input is minimized. It does this by solving consecutive optimizations over a time horizon using a modelled linear approximation of the real system.

In the current circumstances, the complete system consists of several subsystems contributing to the complexity of the motor task execution; The quadcopter, the haptic interface, and the human operator. Together, each of these subsystems combine to form the control infrastructure implemented to provide the learner with force feedback during task execution. An overview of the complete system may be seen Figure 9.

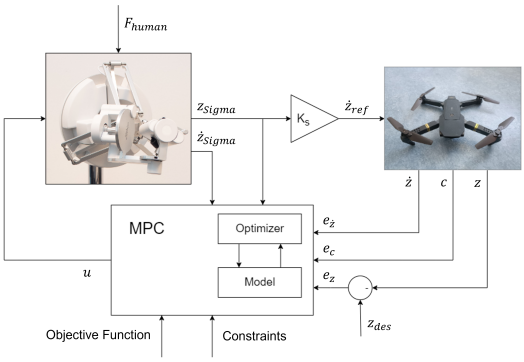

Fig. 9. Overview of the complete System structure. The MPC has an internal model of the Sigma.7, quadcopter and the human arm dynamics

K_s functions as a gain that bridges the workspace differences between the Sigma.7 and the quadcopter. Its value determines the maximum $\dot{z}_{\text{ref}}$ that can be provided to the quadcopter and as such limits its attainable velocity.

The Quadcopter Model

In subsection II-B the full quadcopter dynamics are established. Force feedback will only be provided on one axis, namely the z-axis, related to the altitude rate of the drone. The linearization and separate control structures of the plant depicted in Equation 11 decouples the axes from one another. As a result, the controlled plant dynamics of the z-axis may be separated from the rest as an approximation. The quadcopter dynamics are represented by Equation 22, further details may be found in section III.

$$\begin{bmatrix} \dot{z} \\ \ddot{z} \\ \dot{c} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{K_t+K_{1,P}}{m} & \frac{K_{1,I}}{m} \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} z \\ \dot{z} \\ c \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_{1,P}}{m} \\ 1 \end{bmatrix} \dot{z}_{\text{ref}} \tag{22}$$

$$y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} z \\ \dot{z} \\ c \end{bmatrix} \tag{23}$$

where $c = \int e_1 dt$ denotes the controller state or the altitude controller. Further information on its derivation may be found in section III.

The Sigma.7 Model

The second subsystem is the Force Dimension haptic interface, the Sigma.7. The dynamic model of this device is not publicly available, necessitating a system identification process to identify a workable model of the device. In comparison to the quadcopter dynamics, the responsiveness of the Sigma.7 device is relatively slow, resulting in a delay between the forces provided to the user and the forces perceived at the end-effector. This discrepancy between the quadcopter and the Sigma.7 forms a significant bottleneck in the responsiveness of the complete system, and is relevant information to the MPC design process. This is especially true for low-force inputs, where aspects such as friction play a considerable role.

As an initial assumption, the Sigma.7 device is modelled as a mass-spring-damper system.

$$M\ddot{z}_s + C\dot{z}_s + Kz_s = u \tag{24}$$

where

$$M : \text{End-effector mass (kg).} \tag{25}$$
$$C : \text{Viscous damping coefficient (Ns/m).} \tag{26}$$
$$K : \text{Linear spring coefficient (N/m).} \tag{27}$$
$$u : \text{Forces applied on the system by the motors (N).}$$
$$z_s : \text{The z-coordinate of the sigma.7 device (m)} \tag{28}$$

The system may be rewritten to a state-space representation, as depicted in Equation 29.

$$\begin{bmatrix} \dot{z}_s \\ \ddot{z}_s \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{K}{M} & -\frac{C}{m} \end{bmatrix} \begin{bmatrix} z_s \\ \dot{z}_s \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \end{bmatrix} u \tag{29}$$

$$y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z \\ \dot{z} \end{bmatrix} \tag{30}$$

For design, safety, and comfort purposes, the operation range for this project consists of approximately -2.5 N $\leq F \leq$ 2.5 N. Within this region of interest, friction forms one of the biggest issues in acquiring a theoretical model of the Sigma.7. Even as the system already implements internal gravity and friction compensation, in the approximate zone of -0.3 N $\leq F \leq 0.7$ N the system is not incited to move at all and just beyond there is a region with viscous

friction, where some movement is incited but not enough to be consistent. Within the regions $F \leq -0.5$ N and $F \geq 1.1$ N is consistent in behaviour which may be observed in section III in Figure 12. As such, there is a considerable range where friction is significant enough to impact the expected behaviour of the device. As the MPC attempts to minimize the force exerted by the Sigma.7 this is a problematic component to neglect for long-term implementation. The Sigma.7 is likely to be a nonlinear system or one that requires a multitude of linear models to identify an accurate model over all ranges of operation. However, to effectively implement a model with the MPC, it only needs to be accurate enough within the time horizon of operation. As such, to prevent the project scope from growing out of bounds, the linear approach is considered a valid if not suboptimal choice, allowing for linear MPC to be implemented.

The linear second-order model was identified using a dataset obtained from the Sigma.7 wherein a pseudorandom binary signal was used as a force input to the z-axis and the position and velocity data were measured on the same axis. As the xy plane is constrained using PD controllers, the identification process is focused on attaining a representative model of the vertical behaviour of the system under these specific conditions. The data were sampled at a sampling rate of 0.001 s. Using the MATLAB system identification toolbox, the *idgrey* function was used to determine the best-fitting model using Equation 29. The workspace of the system is limited and there is a set of PD controllers at the edges, as such it is desirable to remove any of the data associated with this to prevent the PD control dynamics from influencing the system dynamics. The data was pre-processed to remove these unreliable data points and trends. More information about the system identification process is provided in section III. Three parameters were identified using this process, depicted in Table V, providing a complete picture of the Sigma.7 subsystem together with Equation 24.

TABLE V
THE THREE IDENTIFIED PARAMETERS FOR THE SIGMA.7 SUBSYSTEM

Parameter	Value	Unit
M	2.2227	kg
C	19.4007	Ns/m
K	7.6397	N/m

The Human Operator

The last subsystem is the human operator. This subsystem is often modelled by looking at a simplified model of the human arm. It has been observed in experiments regarding haptic feedback that the human arm often tends to act as a passive element, not dissimilar to a low-pass filter. As a result, it is often modelled as a mass-spring-damper system [29, 30, 31]. However, due to the lack of a force sensor within the current setup, it is extremely difficult to determine the composition of forces caused by the human operator on the haptic interface. Without an accurate model of either system, and no force sensor, it is extremely difficult to validate both subsystems. As such, the performance of the preliminary framework evaluated is based on its ability to stabilize itself and the influence of the human operator on the system is thus neglected for now.

The Complete System Dynamics

The final representation of the system dynamics combining the quadcopter and Sigma.7 subsystems may be denoted by Equation 31-35. The full derivation may be found in section III.

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}u(k) \tag{31}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x}(k) \tag{32}$$

where,

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & -\frac{K_p+K_t}{m} & \frac{K_I}{m} & \frac{K_pK_s}{m} & 0 \\ 0 & -1 & 0 & K_s & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{K}{M} & -\frac{C}{M} \end{bmatrix} \tag{33}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{M} \end{bmatrix} \tag{34}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{35}$$

Here, the state vector $\mathbf{x}(k) = \begin{bmatrix} z & \dot{z} & c & z_\mathrm{s} & \dot{z}_\mathrm{s} \end{bmatrix}^T$ is a concatenation of quadcopter and Sigma.7 systems as defined earlier, and input $u(k)$ represents the force input provided to the sigma.7.

The Model Predictive Controller

The objective of the MPC is to drive the states to their desirable set points as the control input is minimized. The linear time-invariant system denoted by Equation 31-35 is discretized using $T_s = 0.05$ s and the zero-order hold method in MATLAB using the *c2d* command. This results in the discrete LTI system described in the following form

$$\mathbf{x}(k+1) = \mathbf{A}_d\mathbf{x}(k) + \mathbf{B}_d u(k) \tag{36}$$

$$\mathbf{y}(k) = \mathbf{C}_d\mathbf{x}(k), \tag{37}$$

where $\mathbf{A}_d \in \mathbb{R}^{n\times n}$, $\mathbf{B}_d \in \mathbb{R}^n$, and $\mathbf{C}_d \in \mathbb{R}^{p\times n}$ for which n is defined to be the number of states, and p the number of outputs. Similarly, $\mathbf{x} \in \mathbb{R}^n$, $u \in \mathbb{R}$, and $\mathbf{y} \in \mathbb{R}^p$.

The system comprises a cascade of several systems, the Force Dimension Sigma.7 haptic interface, the quadcopter dynamics with its internal controller, and a gain to compensate for the differences in workspaces. As a result, the system is subject to several constraints due to mechanical limitations and desired constraints. These constraint sets are given as follows

$$\mathbb{U} := \{\Delta u(k) \in \mathbb{R}, \quad |\Delta u(k)| \leq 2\} \tag{38}$$

$$\mathbb{X} := \left\{ \mathbf{x}(k) \in \mathbb{R}^n, \quad \begin{bmatrix} -0.115 \\ -2 \end{bmatrix} \leq \begin{bmatrix} z_\mathrm{s}(k+1) \\ x_u(k+1) \end{bmatrix} \leq \begin{bmatrix} 0.132 \\ 2 \end{bmatrix} \right\}. \tag{39}$$

To incorporate the slew constraint— the constraints on the rate of change of the input(s), the basic state-space representation is augmented with an additional state $x_u(k) = u(k-1)$ and rewritten using $\Delta u(k) = u(k) - u(k-1)$ to the following form.

$$\begin{aligned}\begin{bmatrix}\mathbf{x}(k+1)\\ x_u(k+1)\end{bmatrix} &= \begin{bmatrix}\mathbf{A}_d & \mathbf{B}_d\\ \mathbf{0} & 1\end{bmatrix}\begin{bmatrix}\mathbf{x}(k)\\ x_u(k)\end{bmatrix} + \begin{bmatrix}\mathbf{B}_d\\ 0\end{bmatrix}\Delta u(k) \qquad (40)\\ &= \mathbf{A}_a\mathbf{x}_a(k) + \mathbf{B}_a\Delta u(k)\\ \mathbf{y}(k) &= \begin{bmatrix}\mathbf{C}_d & 0\end{bmatrix}\begin{bmatrix}\mathbf{x}(k)\\ x_u(k)\end{bmatrix} \qquad (41)\\ &= \mathbf{C}_a\mathbf{x}_a(k)\end{aligned}$$

where $\mathbf{A}_a$, $\mathbf{B}_a$, and $\mathbf{C}_a$ represent the augmented state-space matrices, $\mathbf{x}_a$ the augmented state vector, and the subscript a references the augmented system.

The objective function to be minimized in the MPC optimization is a quadratic function as follows:

$$J(\mathbf{x}(0)) = \sum_{j=0}^{N-1}\{\ell(\mathbf{x}(j), \Delta u(k)(j))\} + V_f(\mathbf{x}(\mathbf{N})) \qquad (42)$$

with

$$\ell(\mathbf{x}(k), \Delta u(k)) = ||\mathbf{x}(k)||^2_{\mathbf{Q}} + ||\Delta u(k)||^2_R \qquad (43)$$

$$V_f(\mathbf{x}(k)) = ||\mathbf{x}(k)||^2_{\mathbf{Q}_f} \qquad (44)$$

where $\ell(\cdot)$ depicts the stage cost and $V_f(\cdot)$ the terminal cost of the cost function. The MPC time horizon $N = 10$, corresponding to a future window of 0.5 s. Weight matrices $\mathbf{Q} = \begin{bmatrix}10 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1\end{bmatrix}$ and $\mathbf{Q}_f = \begin{bmatrix}100 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1\end{bmatrix}$ correspond to the states $\mathbf{x}(k)$, where subscript f refers to the terminal cost. Similarly, $R = 0.01$ is the weight corresponding to the control input $\Delta u(k)$.

The MPC problem is then formulated as in Equation 45 optimal control problem.

$$\mathbb{P}_N(\mathbf{x}_a(0)) : \begin{cases}\min\limits_{\Delta u_0,\ldots,\Delta u_{N-1}} & J(\mathbf{x}_a(0), \Delta\mathbf{u}_{a,N})\\ s.t & \mathbf{x}_a(k+1) = \mathbf{A}_a\mathbf{x}_a(k) + \mathbf{B}_a\Delta u(k)\\ & \mathbf{x}_a(0) \in \mathbb{X}\\ & \mathbf{x}_a(k+1) \in \mathbb{X}, \quad k = 0,\ldots,N\\ & \Delta u(k) \in \mathbb{U}, \quad k = 0,\ldots,N-1\end{cases} \qquad (45)$$

The optimal control problem formulated in Equation 45 effectively minimizes the cost function, by minimizing the states of the system. As a result, it assumes the desired references to be zero. The objective of the system is to guide the user towards a desired altitude (z_{ref}) to complete the task. To incorporate reference tracking into the optimal problem defined in Equation 45, the problem is reformulated slightly.

The new objective function $J_{\text{R}}(\cdot)$ is adjusted slightly in comparison to $J(\cdot)$ for the reference tracking problem. The stage cost $\ell(\cdot)$ and the terminal cost $V_f(\cdot)$ of the objective function are now defined relative to the reference.

$$\ell(\mathbf{x}(k), \Delta u(k)) = ||\mathbf{x}(k) - \mathbf{x}_{ref}(k)||^2_{\mathbf{Q}} + \ldots \qquad (46)$$

$$||\Delta u(k) - \Delta u_{ref}(k)||^2_R$$

$$V_f(\mathbf{x}(k)) = ||\mathbf{x}(k) - \mathbf{x}_{ref}(k)||^2_{\mathbf{Q}_f} \qquad (47)$$

The reference is usually defined using outputs measurements $y(k) - y_{\text{ref}}(k)$ in the objective function. Assuming that the output matrix C is identity, and we have full information state feedback, $\mathbf{y}_{\text{ref}}(k) = \mathbf{x}_{\text{ref}}(k)$ the translation between both notations is one-to-one. Under the current circumstances, the altitude reference is the only potentially non-zero state of interest, the other state references default to zero. The altitude reference is considered constant throughout the horizon length and is modelled as a series of step responses with considerable time between jumps with respect to the MPC time horizon. Additionally, due to the objective of minimizing the force-feedback to the individual, the desirable $\Delta u_{\text{ref}}(k) = 0$ is also known a-priori.

The optimal control problem is then reformulated to Equation 48.

$$\mathbb{P}_N(\mathbf{x}_a(0)) :$$

$$\begin{cases}\min\limits_{\Delta u_0,\ldots,\Delta u_{N-1}} & J_{\text{R}}(\mathbf{x}_a(0), \Delta\mathbf{u}_{a,N})\\ s.t & \mathbf{x}_a(k+1) = \mathbf{A}_a\mathbf{x}_a(k) + \mathbf{B}_a\Delta u(k)\\ & \mathbf{x}_a(0) = \mathbf{x}_a(0) \in \mathbb{X} - x_r,\\ & \mathbf{x}_a(k+1) \in \mathbb{X} - x_r,\\ & \qquad k = 0,\ldots,N\\ & \Delta u(k) \in \mathbb{U} - u_r,\\ & \qquad k = 0,\ldots,N-1\end{cases} \qquad (48)$$

D. Pilot Experiment

The completed framework will be evaluated experimentally on whether it is capable of achieving its main objectives autonomously. The physical system, as described in Figure 9, is provided with an altitude reference consisting of a sequence of step responses and the MPC should provide the Sigma.7 with the correct inputs such that it is capable of steering the quadcopter to its desired reference. Ideally, the human operator would have been included into the design, but as mentioned in subsection II-C, it has not. However, we are still interested in the impact of a human participant on the system as whole. As such, an additional pilot experiment is conducted to observe the effects of having the human-in-the-loop. The experiment is conducted by having a human participant hold onto the Sigma.7 end-effector as we repeat the experiment. They will do so twice, but in slightly different circumstances:

- T1 In the first trial, the participant is asked to feel the forces provided by the Sigma.7 and to move along with them. In this trial they will not be provided with visual information on what the altitude reference signal looks like, as such they are blindly following the forces they are provided with. They will be participating passively to the best of their ability.
- T2 In the second trial, the participant is provided visual information on the altitude reference. For this trial, they are asked to actively follow the reference as accurately

as they can manage. They may actively ignore or move along with the forces provided to them by the Sigma.7 in their attempt at completing this task.

III. Results

With the framework design completed, it is time to evaluate and validate each subsystem within it, focusing on the performance evaluation of individual components and the overall system.

A. The Quadcopter Model

The uncontrolled quadcopter dynamics are first evaluated with a step response of 1 N for each of the four motor input configurations. In this manner, we may first observe the characteristics of the system, such as the settling times, rise times, overshoot, and steady-state errors. This information may then be used for the evaluation and validation of the quadcopter design in *Stabilized Mode* by, similarly, applying a step response as depicted in Figure 10.

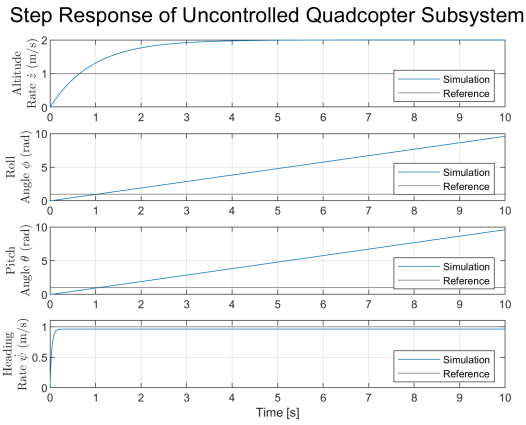

Fig. 10. The Step Response of the Quadrotor Dynamics.

TABLE VI
Transfer function Step Response

Step Response	Alt. Rate	Roll	Pitch	Yaw Rate	Units
Settling Time	2.8040	NA	NA	0.1127	s
Rise Time	2.0566	NA	NA	0.0826	s
Overshoot	0	NA	NA	0	%
Settling Max	2.0000				(m/s)
		NA	NA		(rad)
				0.9615	(rad/s)

The step response of the uncontrolled quadcopter dynamics allows us to determine the system characteristics, depicted in Table VI. The quadcopter behaves are expected, a step input provided to the linearized system is the equivalent of the motors exerting an external force on the quadcopter frame. As a result, the altitude and heading rates increase velocity until they reach terminal velocity, reaching an equilibrium between the forces exerted by the motor and drag. Both responses show steady-state errors with respect to the design requirements. The roll and pitch angles also increase as a constant input is provided to the system, resulting in a tumbling motion of the quadcopter about the roll and pitch axes.

To have the quadcopter behave according to the desired flight mode—*Stabilized* mode—, four control structures are applied to the dynamics. With the control structures implemented, depicted in Figure 11, the steady-state errors on each of the axes were removed. Each of the axes is able to reach the desired set point of 1 within the time durations established in the design requirements in subsection II-B. The settling times were obtained with a 5% threshold.

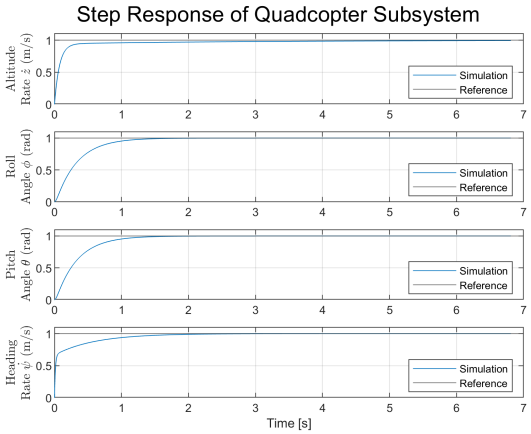

Fig. 11. The step response of the quadcopter system and the controllers implemented, for the final control parameters.

TABLE VII
Controlled System Step Response

Step Response	Alt. Rate	Roll	Pitch	Yaw Rate	Units
Settling Time	0.3748	0.9810	0.9810	1.1314	s
Rise Time	0.2041	0.7048	0.7048	0.7169	s
Overshoot	0	0	0	0	%
Settling Max	0.9933				m/s
		0.9999	0.9999		rad
				1.0000	rad/s

Overall, the system characteristics of the controlled system, as depicted in Table VII, are within the design objectives established in subsection II-B. For the altitude rate, reasonable settling times were assumed within the range of $0.2 \leq t_{s,\dot{z}} \leq 2\ s$, which has been achieved. Similarly, for the attitude of the quadcopter settling times were assumed within the range of $0.5 \leq t_{s,\phi,\theta} \leq 1.5\ s$, was also achieved.

B. The Sigma.7 Model

The Sigma.7 device implements internal gravity and friction compensation during operation. However, within some regions of force commands, the system is still unable to overcome friction. By applying a set of step force commands to the Sigma.7, its response can be observed, by which we may determine the regions within which friction is of the most influence. As seen in Figure 12, in the approximate zone of $-0.3\ \text{N} \leq F \leq 0.7\ \text{N}$ the system experiences considerable Coulomb friction and is not incited to move at all. In the range just beyond, there is a region with viscous friction, where some movement is incited but not enough to be consistent. Within the regions $F \leq -0.5$ N and $F \geq 1.1$ N, the system is consistent and repeatable in behaviour.

The identified Sigma.7 subsystem model in subsection II-C does not adequately describe the dynamics of the system over its entire range of operation; however, as mentioned before, the model should perform adequately over the chosen horizon time of the MPC. As such, the performance is experimentally evaluated with respect to a horizon time of 0.5 s. For this experiment, both the identified model and the Sigma.7 haptic interface receive the same input data; however, every 0.5 s, the identified linear model receives updated initial conditions using the measurements of the Sigma.7. As a result, the identified model does not perform perfectly but is capable of staying within an adequate range of the actual measurements, as seen in Figure 13. For a window of 0.5 s the Normalized Root Mean Square Error (NRMSE) for the positional data is 68.9 %, and for the velocity data it is 40.9 %.

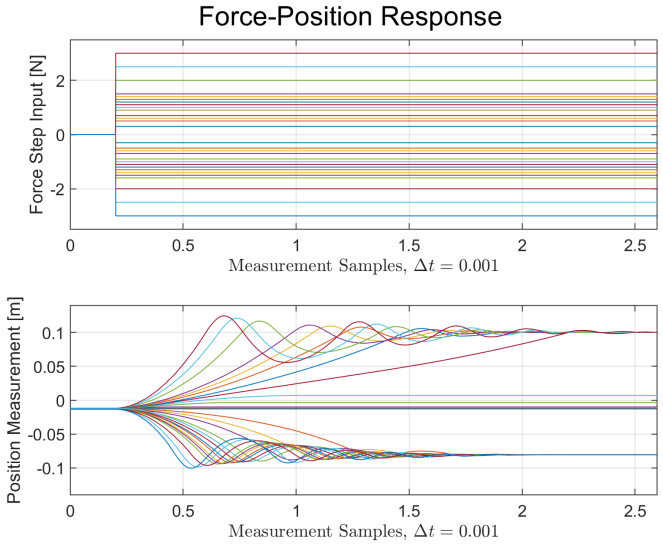

Fig. 12. Step responses of the Sigma7 in position for a range of force inputs. Please note that the overshoot fluctuations that may be observed at the edge of the device workspace coincide with the safety PD controller implemented to protect the system from damage and do not originate from the system dynamics.

At very specific input sequences, however, such as the more extreme swings observed between 2 and 7 seconds, the errors briefly increase to relatively high proportions, as depicted in Figure 14. These larger error spikes generally correspond to the system dynamics at the end of the time horizon. The current model cannot capture the dynamics governing the behaviour at very small inputs and large sudden changes in input. Additionally, when large sudden changes in input occur, it is also more likely for communication delays to occur between MATLAB and the C++ executable. In such moments, the Sigma.7 measurements remain constant for a brief period of time. These errors are less prominent, nor quite as large, in comparison but add to the overall error.

C. The Final Framework

The final framework is capable of controlling the system such that the main objectives are achieved. The quadcopter altitude

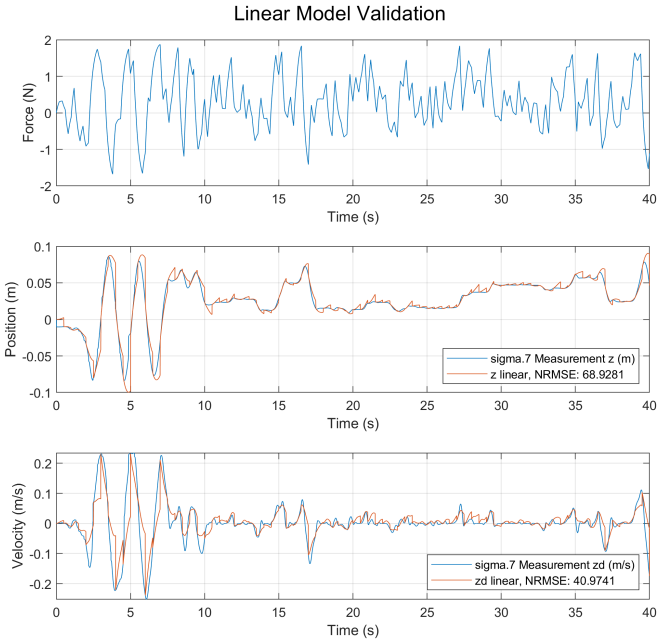

Fig. 13. The experimental results of the Sigma.7 measurements and the identified linear model. The first plot includes the input data provided to both systems. The second plot shows the position output measurement and simulation results. The third plot provides the velocity output measurements as provided by the Sigma.7, and the simulation results. Every 0.5 seconds, the initial conditions for the linear model are reapplied based on the measured position and velocity data of the Sigma.7. Additionally, the NMRSE errors, based on the mean of the measured data, have been noted in the figure legend.

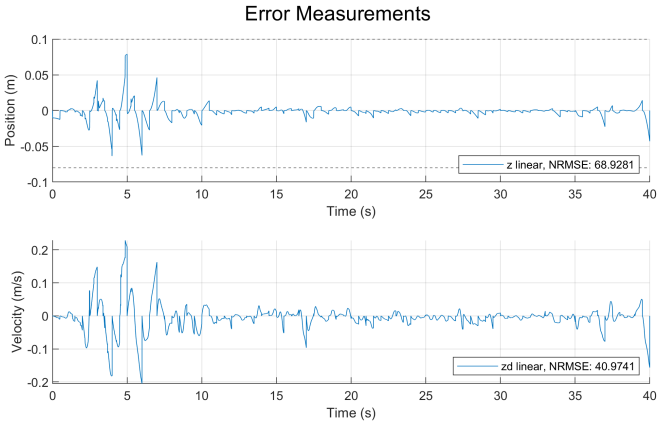

Fig. 14. This figure provides the error curve for both the position output (the top window), defined as $e_z = z_s - z_{s,\text{model}}$, and the velocity output (the bottom window), defined as $e_{\dot{z}} = \dot{z}_s - \dot{z}_{s,\text{model}}$. These results correspond with Figure 13.

is driven to its desired altitude reference, while each of the other states are driven to zero, as depicted in Figure 15. The quadcopter altitude, however, consistently suffers from a steady state error at about 0.02 m. As a result of the steady state error in the quadcopter altitude, the control input, depicted in Figure 16, remains non-zero in an attempt to compensate. The force command provided to compensate the steady-state error is, at -0.23 N, not quite enough to overcome the internal

friction of the Sigma.7 as discussed in section II. Which may be indicative of additional unmodeled sources of friction.

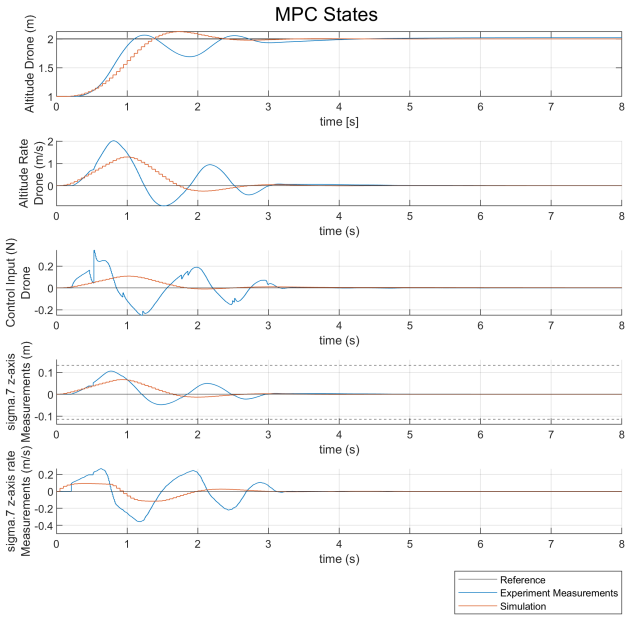

Fig. 15. A single step reference provided as an altitude reference to the MPC (depicted by the black line). The blue line and red lines present the experimental and simulation results, respectively, for comparison.

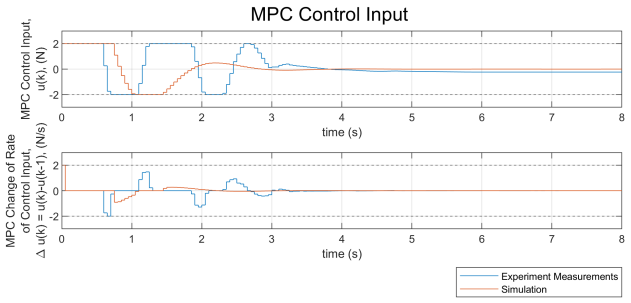

Fig. 16. A companion figure to Figure 15 depicting the control input determined by the MPC. This input is provided to the physical (blue) or simulated (red) system.

The simulated system behaves quite different to that of the experimental data, however, it does show the same overall trend in behaviour. The system characteristics, depicted in Table VIII, are similar for the step response. However, as a whole, the simulated system responds in a fashion much less oscillatory than the experimental results. This discrepancy in behaviour is likely indicative of a suboptimal model of the system dynamics, causing a mismatch between the theoretical model and the physical system. This aggressive behaviour of the physical system is also apparent in the MPC control input, depicted in Figure 16, which attempts to correct the error much more aggressively than the simulated system, resulting in a faster but also more oscillatory response.

The final test is to apply the complete system to the game designed in Unity, requiring a range of viable successive step responses of varying magnitudes over the duration of the game, shown in Figure 17-18. Similar to Figure 16 the steady

TABLE VIII
FRAMEWORK STEP RESPONSE - ALTITUDE

Step Response	Experimental	Simulated	Units
Settling Time	2.2146*	1.9124*	s
Rise Time	0.9464	1.1423	s
Overshoot	3.3038	6.2508	%
Settling Min	1.6909	1.8090	m
Settling Max	2.0661	2.1250	m

TABLE IX
*WITH A SETTLING TIME THRESHOLD OF 5%

state error remains. Although the steady state errors, even at larger references, as may be observed in Figure 17, it remains relatively small. The largest observed steady state error was between 30-40 s at -0.15 m. Overall, the system exhibits quite a bit of oscillatory behaviour as compared to its theoretical counterpart, however, it manages to complete the set tasks reaching the state objectives.

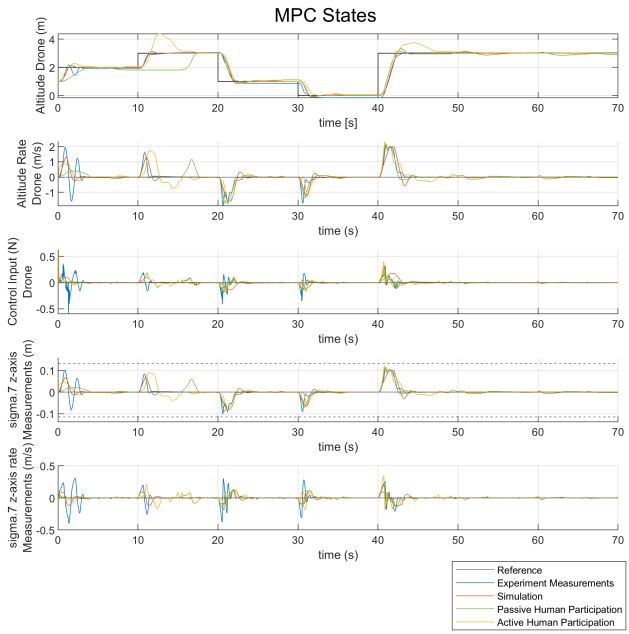

Fig. 17. A varying range of viable step references provided as an altitude reference to the MPC (depicted by the black line). The blue line and red lines present the experimental and simulation results, respectively, for comparison. Additionally, a pilot experiment was conducted, depicted by the green and yellow lines. These results showcase the changes when a human participant is added into the loop. Passive participation is depicted by the green line, whereas active participation is depicted by yellow.

A pilot experiment was also conducted in which a human participant held the Sigma.7 end-effector during both passive (T1) and active (T2) participation. In T1, the participant accurately followed the reference despite the lack of visual feedback. The addition of human dynamics into the loops seems to stabilize the oscillations exhibited by the autonomous system, particularly of the drone altitude rate, and of the Sigma.7 z-axis position and rate measurements. In T2, active participation also showcases some considerable overshoot.

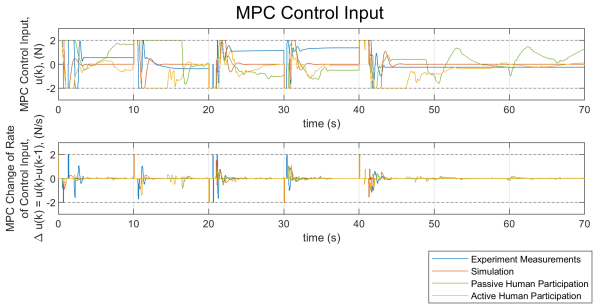

Fig. 18. A companion figure to Figure 17 depicting the control input determined by the MPC. This input is provided to the physical (blue) or simulated (red) system.

D. Friction

Despite the incorporation of viscous friction into the model, the impact of additional unconsidered friction sources can be observed during the experiment. When the machine moves along the negative z-axis, corresponding to the direction of gravity, there is a considerable steady state error that is not compensated. This behaviour is quite consistent throughout each of the measurements and is most prominent when the system must compensate for movement along the negative z-axis, as may be observed in Figure 19. As the error grows, the control inputs provided to the Sigma.7, as depicted in Figure 20, also grows to compensate. After some time, the two balance one another and the device is no longer incited to move. As mentioned in section II, one type of friction that may explain this behaviour is Coulomb friction, otherwise known as stick friction. Which may be preventing the device from moving at very small force inputs.

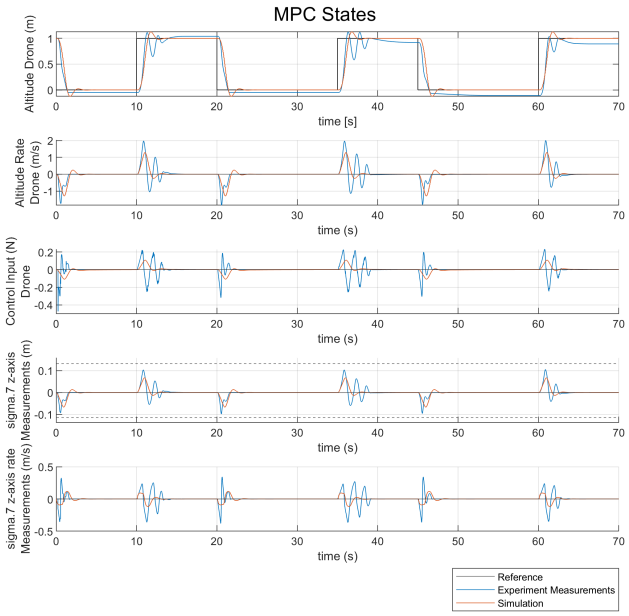

Fig. 19. A set of repeated step references provided as an altitude reference to the MPC (depicted by the black line) to create an overview of the effects of friction on the system. The blue line and red lines present the experimental and simulation results, respectively, for comparison.

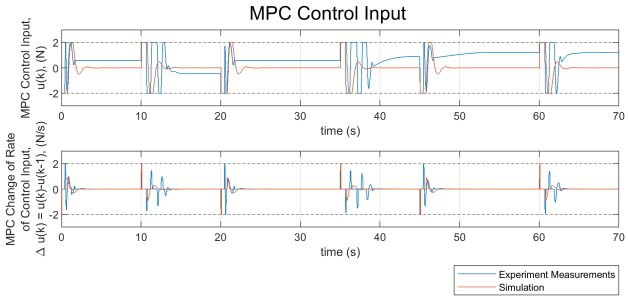

Fig. 20. A companion figure to Figure 19 depicting the control input determined by the MPC. This input is provided to the physical (blue) or simulated (red) system.

IV. Discussion

In the previous section the results were presented, highlighting the performance of the quadcopter and Sigma.7 as individual subsystems before transitioning into the performance of the completed system. The quadcopter demonstrated successful stabilization after the incorporation of the control infrastructures, eliminating the steady-state errors and meeting the required settling times. The Sigma.7 model was effective within the established time horizon and range of operation, although shows limitations due to unmodelled frictional forces. The complete system has demonstrated discrepancies between the theoretical simulated model and the physical system; however, it still successfully achieves its main objectives. The physical system is able to provide the Sigma.7 with the necessary inputs to autonomously guide the quadcopter to its desired references. However, despite successfully reaching the quadcopter's altitude reference, persistent small steady-state errors were observed, with a maximum deviation of -0.15 m during the experiments. These errors are likely due to unmodeled friction or other unaccounted-for disturbances that affect system performance.

In this discussion, we will analyse these results, addressing their implications with respect to current literature, and discuss the limitations of the current model. Particularly the potential improvements to the different subsystems, and their implementation as a whole. Lastly, we will also explore the implications of these findings for future work and its applications.

A. Model Accuracy and Computational Performance

The current Sigma.7 model implementation exhibits significant limitations in accuracy, as shown in subsection III-B. The linear approximation of the system is not capable of fully capturing all the relevant dynamics of the Sigma.7 that encompass its behaviour. In subsection III-C, it is shown that the MPC input never returns to a steady zero input. One potential explanation for this is the presence of unmodeled friction in the Sigma.7 model, such as Coulomb friction at low force inputs. Additionally, there may also be other additional dynamics that are not considered in the current model. However, incorporating friction is likely to introduce nonlinearities into the model, which may necessitate a nonlinear MPC approach, requiring careful design so as not to

potentially compromise the real-time capabilities of the system [32]. Notably, this study demonstrates that even with shorter horizons, effective control can be achieved. With the current model, the MPC is capable of controlling the quadcopter to the desired altitude reference whilst minimizing the other states and the input, effectively allowing for the completion of the design objectives.

B. The Influence of Human-in-the-Loop

The addition of human dynamics into the loop, as depicted in Figure 17, seems to stabilize the oscillations exhibited by the autonomous system, particularly of the drone altitude rate, and of the Sigma.7 z-axis position and rate measurements. This was observed both during passive, and active participation, and is likely due to the additional damping effects provided by the human arm at the end-effector [30]. In T1, the participant was able to successfully complete the motor task by only responding to the forces provided by the Sigma.7. They were able to do so without visual feedback, and without active participation. A lack of active participation during task execution is something strongly thought to inhibit the motor learning process [33] [34]. Additionally, the *Challenge Point Framework*, states that motor learning can be optimized by matching the difficulty of the to-be-learned task to the learner's skill level. The ease by which the task was completed blindly suggests that the force feedback provided by the current design is still stronger than desired. As this was a single-participant experiment, the results glimpsed cannot be generalized, but it may be beneficial to revise the current design and implement human dynamics into the MPC design. Additionally, during T1, we observed that sometimes the participant was unable to respond to the forces because they were not always aware of them. This is clearly observable in Figure 17 between 10-20 s where the correction of the passive response towards the reference happens quite late within that time window. Despite the fact that the Sigma.7 provided a full 2N in force to the participant, as depicted in Figure 18, it went unnoticed for more than 5 seconds. The participant shared that it was only once they moved the slightest bit, causing a force differential, that they became aware of the force exerted by the end-effector.

The active participation in T2 showcases some considerable overshoot, reaching almost 119% at 12.7 s. This suggests the participant may have underestimated the feedback strength and struggled to correct the additional error.

C. Limitations

The current use of multiple simulation environments introduces unnecessary redundancy. For efficiency, the Simulink environment may be integrated into the Visual Studio environment for a more efficient and compact implementation. However, maintaining a separate visualization framework is recommended so as not to impact the computational power needed for the quadcopter model and the MPC. By keeping the visualization framework separate, it may also be run on a different computer altogether.

V. Conclusion

During this thesis, a framework was established that utilizes a Model Predictive Controller to haptically assist a quadcopter pilot during motor task completion. With that objective in mind, this framework integrates a quadcopter simulator, a dynamic motor task visualized in Unity, a model of the Sigma.7 haptic interface, and an MPC capable of generating real-time control commands, providing haptic feedback during task execution.

The system was able to complete its main design objectives. The quadcopter demonstrated successful stabilization in its desired flight mode — *Stabilizing Mode* — after the incorporation of the onboard PID controllers. The Sigma.7 model performed sufficient within the established time horizon and range of operation, although shows limitations due to unmodelled friction. The MPC framework is capable of providing the Sigma.7 with the necessary input command to autonomously guide the quadcopter to its desired references in real-time, therefore completing its primary objective. Nevertheless, the system is not without its limitations. The system still shows steady state errors, likely resulting from unmodelled friction, which the current MPC design cannot compensate for. Additionally, a pilot test revealed that the force feedback provided to the participant might have been too large, as active participation was not necessary for task completion.

The field of motor learning, particularly in the context of robot-mediated haptic feedback for dynamic motor tasks, remains an area with many unanswered questions. In this study, we create a framework that is capable of utilizing MPC to haptically assist quadcopter pilots during task execution. The system is capable of providing haptic assistance to the quadcopter pilot with respect to the altitude. We hope that with further improvements, this framework may be used to study the effects of using MPC and haptic feedback on the motor learning of a quadcopter pilot.

A. Future Work

The long-term objective of this project is that we may evaluate whether the application of haptic feedback for a dynamic task can facilitate or promote motor learning. The framework was designed with this future objective in mind, of studying the impact of utilizing MPC for providing haptic feedback on the motor learning of a quadcopter pilot. Based on this premise, and the content of this paper future works may focus on improving the models of the individual subsystems presented in this paper, or on the inclusion of the human operator into the predictive model of the MPC, with the hopes of expanding this framework.

Real-World Validation of Virtual Dynamics
The virtual drone dynamics currently incorporates several simplifications and assumptions. Practical validation is necessary to ensure that the virtual model mirrors its real-world counterpart sufficiently for viable transfer learning. This may require the addition of some common real-world problems—such as wind, atmospheric pressures, and interactive obstacles.

In a similar trend, the Sigma.7 model should also be improved upon. It is recommended that a more thorough analysis of the device behaviour should allow for a better representation of the device dynamics. Specifically, whether the unmodeled friction is really the main limiting factor or if there are other unconsidered relevant factors.

Human Interaction & Motor Learning Goals

To demonstrate that the stabilization of the system is possible, this paper shows the autonomous response of the Sigma.7, stabilizing itself and the quadcopter at the desired reference height. However, the current system design does not take into account the role of the human learner will have in this process. With the addition of the human dynamics into the MPC design, the force feedback provided to the individual may also take into account the damping effects of the human arm, its delay, and depending on the model even the average response time of the human participant. Adjusting this model to the individual beforehand may provide even more personalized force feedback. To this end, it is also suggested to add a force sensor to the sigma.7 setup. In this way, the interaction forces between the human operator and the haptic interface may also be quantified.

References

[1] Adilzhan Adilkhanov, Matteo Rubagotti, and Zhanat Kappassov. "Haptic Devices: Wearability-Based Taxonomy and Literature Review". In: *IEEE Access* 10 (2022), pp. 91923–91947. DOI: 10.1109/ACCESS.2022.3202986.

[2] Camille K. Williams and Heather Carnahan. "Motor learning perspectives on haptic training for the upper extremities". In: *IEEE Transactions on Haptics* 7 (2 2014), pp. 240–250. DOI: 10.1109/TOH.2013.2297102.

[3] Ekin Basalp, Peter Wolf, and Laura Marchal-Crespo. "Haptic Training: Which Types Facilitate (re)Learning of Which Motor Task and for Whom? Answers by a Review". In: *IEEE Transactions on Haptics* 14 (4 2021), pp. 722–739. DOI: 10.1109/TOH.2021.3104518.

[4] Roland Sigrist et al. "Augmented visual, auditory, haptic, and multimodal feedback in motor learning: A review". In: *Psychonomic Bulletin and Review* 20 (1 2013), pp. 21–53. DOI: 10.3758/s13423-012-0333-8.

[5] Dane Powell and Marcia K. O'Malley. "The task-dependent efficacy of shared-control haptic guidance paradigms". In: *IEEE Transactions on Haptics* 5 (3 2012), pp. 208–219. DOI: 10.1109/TOH.2012.40.

[6] Richard A. Schmidt et al. *Motor Control and Learning: A Behavioral Emphasis, 6th Edition*. 6th, pp. 1–730.

[7] Laura Marchal-Crespo, Jan Furumasu, and David J Reinkensmeyer. *A robotic wheelchair trainer: design overview and a feasibility study*. 2010. DOI: 10.1186/1743-0003-7-40.

[8] Mark Guadagnoli and Timothy Lee. "Challenge Point: A Framework for Conceptualizing the Effects of Various Practice Conditions in Motor Learning". In: *Journal of motor behavior* 36 (July 2004), pp. 212–24. DOI: 10.3200/JMBR.36.2.212-224.

[9] Dante Tezza and Marvin Andujar. "The State-of-the-Art of Human–Drone Interaction: A Survey". In: *IEEE Access* 7 (2019), pp. 167438–167454. DOI: 10.1109/ACCESS.2019.2953900.

[10] Dante Tezza, Denis Laesker, and Marvin Andujar. "The Learning Experience of Becoming a FPV Drone Pilot". In: *HRI '21 Companion* (Mar. 2021), pp. 1–3. URL: https://dl.acm.org/doi/abs/10.1145/3434074.3447167.

[11] Viviane Herdel, Lee J. Yamin, and Jessica R. Cauchard. "Above and Beyond: A Scoping Review of Domains and Applications for Human-Drone Interaction". In: Association for Computing Machinery, Apr. 2022. DOI: 10.1145/3491102.3501881.

[12] Simonas Draukšas et al. *Model Predictive Control-based haptic steering assistance to enhance motor learning of a bicycling task: A pilot study*. 2022. URL: https://engrxiv.org/preprint/view/2811/5233.

[13] Ali Safavi et al. "A novel MPC approach to optimize force feedback for human-robot shared control". In: vol. 2015-December. Institute of Electrical and Electronics Engineers Inc., Dec. 2015, pp. 3018–3023. ISBN: 9781479999941. DOI: 10.1109/IROS.2015.7353793.

[14] Ali Safavi and Mehrdad H. Zadeh. "Model-Based Haptic Guidance in Surgical Skill Improvement". In: Institute of Electrical and Electronics Engineers Inc., Jan. 2016, pp. 1104–1109. ISBN: 9781479986965. DOI: 10.1109/SMC.2015.198.

[15] Max Schwenzer et al. "Review on model predictive control: an engineering perspective". In: *International Journal of Advanced Manufacturing Technology* 117 (5-6 Nov. 2021), pp. 1327–1349. DOI: 10.1007/s00170-021-07682-3.

[16] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017. DOI: 10.1017/9781139061759.

[17] Mithra Sivakumar and Naga Tyj. "A Literature Survey of Unmanned Aerial Vehicle Usage for Civil Applications". In: *Journal of Aerospace Technology and Management* 13 (Nov. 2021). DOI: 10.1590/jatm.v13.1233.

[18] *Force Dimension: sigma.7*. URL: https://www.forcedimension.com/products/sigma. (accessed: 17-06-2024).

[19] Marcia O'Malley and Abhishek Gupta. "Haptic Interfaces". In: Dec. 2008, pp. 25–73. ISBN: 9780123740175. DOI: 10.1016/B978-0-12-374017-5.00002-X.

[20] Ma and Payandeh. "Analysis and Experimental Study of a 4-DOF Haptic Device". In: *Advances in Haptics*. Ed. by Mehrdad Hosseini Zadeh. Rijeka: IntechOpen, 2010. Chap. 9. DOI: 10.5772/8687.

[21] Cihad Dogan. *UDP Communication between Unity and Matlab/Simulink*. 2019. URL: https://github.com/CihadDogan/UDPExample.

[22] Sumaila Musa. "Techniques for Quadcopter Modelling and Design: A Review". In: 5 (May 2018).

[23] Samir Bouabdallah. “Design and Control of quadrotors with application to autonomous flying”. In: (Jan. 2007). DOI: 10.5075/epfl-thesis-3727.

[24] Andrew Zulu and Samuel John. “A Review of Control Algorithms for Autonomous Quadrotors”. In: *Open Journal of Applied Sciences* 04 (Jan. 2014), pp. 547–556. DOI: 10.4236/ojapps.2014.414053.

[25] Abdelhamid Tayebi and S. McGilvray. “Attitude stabilization of a four-rotor aerial robot”. In: vol. 2. Jan. 2005, 1216–1221 Vol.2. DOI: 10.1109/CDC.2004.1430207.

[26] Alfredo Bici et al. “Development and Validation of a LQR-Based Quadcopter Control Dynamics Simulation Model”. In: *International Journal of Aerospace Engineering* 34 (Nov. 2021). DOI: 10.1061/(ASCE)AS.1943-5525.0001336.

[27] Du Ho Duc et al. “Vertical Modeling of a Quadcopter for Mass Estimation and Diagnosis Purposes”. In: Oct. 2017. DOI: 10.1109/RED-UAS.2017.8101665.

[28] Mathias Bos et al. “Modeling and Identification of Multirotor Drone Dynamics for Onboard MPC Motion Planning”. In: 2021. URL: https://www.imavs.org/papers/2021/3.pdf.

[29] Michael J. Fu and M. Cenk Cavusoglu. “Human-Arm-and-Hand-Dynamic Model With Variability Analyses for a Stylus-Based Haptic Interface”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42.6 (2012), pp. 1633–1644. DOI: 10.1109/TSMCB.2012.2197387.

[30] Steven Cutlip, Jim Freudenberg, and R. Brent Gillespie. “Respecting the Coupled Dynamics: Haptic Feedback Carries both Power and Information”. In: *2020 IEEE Haptics Symposium (HAPTICS)*. 2020, pp. 718–723. DOI: 10.1109/HAPTICS45997.2020.ras.HAP20.10.e37f63b1.

[31] T. Murakami, F. Yu, and K. Ohnishi. “Torque sensorless control in multidegree-of-freedom manipulator”. In: *IEEE Transactions on Industrial Electronics* 40.2 (1993), pp. 259–265. DOI: 10.1109/41.222648.

[32] Mark Cannon. “Efficient nonlinear model predictive control algorithms”. In: *Annual Reviews in Control* 28 (2 2004), pp. 229–237. ISSN: 13675788. DOI: 10.1016/j.arcontrol.2004.05.001.

[33] Martin Lotze et al. “Motor learning elicited by voluntary drive”. In: *Brain* 126 (4 Apr. 2003), pp. 866–872. DOI: 10.1093/brain/awg079.

[34] Özhan Özen, Karin A. Buetler, and Laura Marchal-Crespo. “Promoting Motor Variability During Robotic Assistance Enhances Motor Learning of Dynamic Tasks”. In: *Frontiers in Neuroscience* 14 (Feb. 2021). DOI: 10.3389/fnins.2020.600059.

[35] D. Mellinger, M. Shomin, and V. Kumar. “Control of Quadrotors for Robust Perching and Landing”. In: (2010).

[36] Xiaodong Zhang et al. “A Survey of Modelling and Identification of Quadrotor Robot”. In: *Abstract and Applied Analysis* 2014.1 (2014), p. 320526. DOI: https://doi.org/10.1155/2014/320526.

[37] K. Salisbury et al. *Haptic Rendering: Programming Touch Interaction with Virtual Objects*. 1995. DOI: 10.1145/199404.199426.

[38] Özhan Özen, Karin A. Buetler, and Laura Marchal-Crespo. “Towards functional robotic training: motor learning of dynamic tasks is enhanced by haptic rendering but hampered by arm weight support”. In: *Journal of NeuroEngineering and Rehabilitation* 19 (2022). DOI: 10.1186/s12984-022-00993-w.

[39] Ekin Basalp et al. “Rowing Simulator Modulates Water Density to Foster Motor Learning”. In: *Frontiers in Robotics and AI* 6 (Aug. 2019). DOI: 10.3389/frobt.2019.00074.

[40] Ekin Basalp et al. “Configurable 3D rowing model renders realistic forces on a simulator for indoor training”. In: *Applied Sciences (Switzerland)* 10 (3 Feb. 2020). DOI: 10.3390/app10030734.

I. Appendix I - Quadcopter Dynamics

A. Coordinate System

To determine the nonlinear dynamics of the quadcopter using the Newton-Euler method, it is common to define two coordinate systems within which the states of the quadcopter are defined [35] [36]. The first coordinate system is the earth-fixed

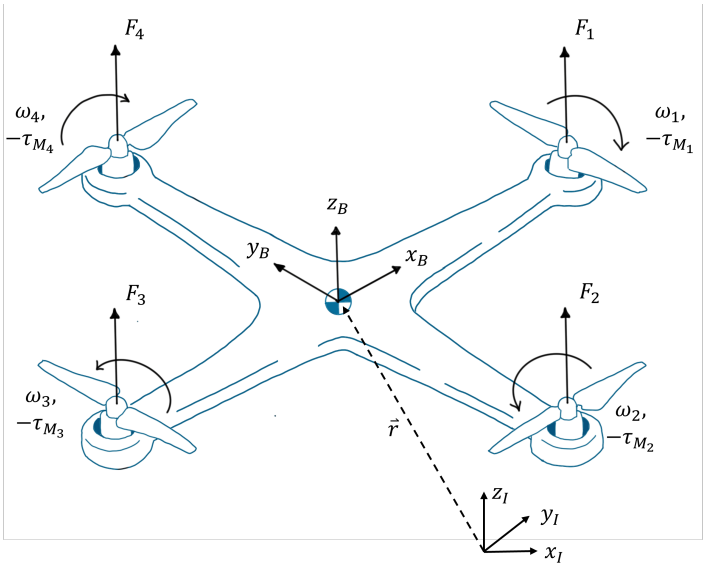

Fig. 21. The defined coordinate systems and directions.

inertial frame ($\mathcal{I} = (x_\mathcal{I},\ y_\mathcal{I},\ z_\mathcal{I})$), also known as the world Frame. The second coordinate system is the body frame ($\mathcal{B} = (x_\mathcal{B},\ y_\mathcal{B},\ z_\mathcal{B})$), fixed to the Center of Mass (CoM) of the quadcopter. Each of the motors rotates with an angular velocity of ω_i producing a force F_i along the rotor axis (assumed to be $z_\mathcal{B}$). The motor also generates a moment M_i around its rotor axis, opposite to its direction of rotation.

To translate movement from one coordinate system to the other, a rotational matrix $\mathbf{R}(\phi, \theta, \psi)$ is defined. This is where the roll (ϕ), pitch (θ), and yaw (ψ) angles are introduced. They generally refer to the canned rotations required to move from the inertial frame to the body frame, and back.

$$\mathbf{q}_\mathcal{I} = \mathbf{R}_x(\psi)\mathbf{R}_y(\theta)\mathbf{R}_z(\phi)\mathbf{q}_\mathcal{B} = \mathbf{R}(\phi, \theta, \psi)\mathbf{q}_\mathcal{B}$$

$$\mathbf{R}(\phi, \theta, \psi) = \begin{bmatrix} \mathrm{c}_\psi \mathrm{c}_\theta & \mathrm{s}_\phi \mathrm{c}_\psi \mathrm{s}_\theta - \mathrm{c}_\phi \mathrm{s}_\psi & \mathrm{c}_\psi \mathrm{s}_\theta \mathrm{c}_\phi + \mathrm{s}_\phi \mathrm{s}_\psi \\ \mathrm{c}_\theta \mathrm{s}_\psi & \mathrm{s}_\psi \mathrm{s}_\phi \mathrm{s}_\theta + \mathrm{c}_\phi \mathrm{c}_\psi & \mathrm{s}_\psi \mathrm{s}_\theta \mathrm{c}_\phi - \mathrm{c}_\psi \mathrm{s}_\phi \\ -\mathrm{s}_\theta & \mathrm{c}_\theta \mathrm{s}_\phi & \mathrm{c}_\phi \mathrm{c}_\theta \end{bmatrix} \tag{49}$$

The representation of $\mathbf{R}$ may vary depending on the definitions of the rotations and coordinate systems. It is an orthogonal matrix, as such $\mathbf{R}^{-1} = \mathbf{R}^T$.

B. Equations of Motion

Due to the complexity of working in multiple frames of reference, 12 states are required to express the nonlinear dynamics of a quadcopter, as shown in Table X.

TABLE X
DEFINITION OF THE NONLINEAR QUADCOPTER STATES

State	Description	Units
x	X Position in Inertial Frame $\mathcal{I}$	m
y	Y Position in Inertial Frame $\mathcal{I}$	m
z	Z Position in Inertial Frame $\mathcal{I}$	m
ϕ	Roll Attitude in Inertial Frame $\mathcal{I}$	rad
θ	Pitch Attitude in Inertial Frame $\mathcal{I}$	rad
ψ	Yaw Attitude in Inertial Frame $\mathcal{I}$	rad
u	X Translational Velocity in Body Frame $\mathcal{B}$	m/s
v	Y Translational Velocity in Body Frame $\mathcal{B}$	m/s
w	Z Translational Velocity in Body Frame $\mathcal{B}$	m/s
p	Roll Rotational Velocity in Body Frame $\mathcal{B}$	rad/s
q	Pitch Rotational Velocity in Body Frame $\mathcal{B}$	rad/s
r	Yaw Rotational Velocity in Body Frame $\mathcal{B}$	rad/s

Deriving the equations of motion of the quadcopter is somewhat easier in the body frame. According to [23] the dynamics of a rigid body under external forces that are applied to the center of mass, and are expressed in the body fixed frame can be expressed in Newton-Euler formalism as

$$\begin{bmatrix} m\mathbf{I}_{3\times 3} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{V}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\omega} \times m\mathbf{V} \\ \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \boldsymbol{\tau} \end{bmatrix} \tag{50}$$

where $\mathbf{I}_{3\times 3}$ is an identity matrix, and $\mathbf{I} = \text{diag}(I_{xx},\ I_{yy},\ I_{zz})$ the diagonal inertia matrix of the quadcopter frame. The following vectors are then defined in the body frame; the linear velocity $\mathbf{V} = [u,\ v,\ w]^T$, the angular velocity $\boldsymbol{\omega} = [p,\ q,\ r]^T$, the external forces $\mathbf{F} = [f_x,\ f_y,\ f_z]^T$, and the external moments $\boldsymbol{\tau} = [\tau_\phi,\ \tau_\theta,\ \tau_\psi]^T$. Resulting in:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}, \tag{51}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \mathbf{I}^{-1} \begin{bmatrix} qr(I_{yy} - I_{zz}) \\ pr(I_{zz} - I_{xx}) \\ pq(I_{xx} - I_{yy}) \end{bmatrix} + \mathbf{I}^{-1} \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \tag{52}$$

The external forces and moments acting on the quadrotor are then modeled as a summation of various physical phenomena affecting its flight.

$$\mathbf{F} = \mathbf{R}^T \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix} + \mathbf{f}_f \tag{53}$$

The first term incorporates the effect of gravity on the quadcopter frame, the second the total thrust generated by the motors, and the last term depicts the friction.

$$\boldsymbol{\tau} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ l(F_1 - F_2 + F_3 - F_4) \end{bmatrix} + \boldsymbol{\tau}_f - \mathbf{g}_a \tag{54}$$

For the moments acting on the quadcopter the first term incorporates the effects of the generated torques by the motors, the second the friction, and the last term depicts the gyroscopic effects of the propeller.

$$\mathbf{g}_a = \begin{bmatrix} -J_r \Omega \dot{\theta} \\ -J_r \Omega \dot{\phi} \\ 0 \end{bmatrix} \tag{55}$$

where J_r is the inertia of the rotors and $\Omega = \omega_1 + \omega_2 - \omega_3 - \omega_4$. $\mathbf{g}_a$ is very often neglected due to the small values of J_r as mentioned in [35], [36], which will also be done for this model for control purposes.

To rewrite these equations of motion in the Inertial frame, we can use the Rotational Matrix $\mathbf{R}(\phi, \theta, \psi)$ for the linear transformations and Rotational Matrix $\mathbf{T}(\phi, \theta, \psi)$ for the angular transformation [36].

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \mathbf{R}^T \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \tag{56}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \mathrm{s}_\phi \mathrm{t}_\theta & \mathrm{c}_\phi \mathrm{t}_\theta \\ 0 & \mathrm{c}_\phi & -\mathrm{s}_\phi \\ 0 & \mathrm{s}_\phi / \mathrm{c}_\theta & \mathrm{c}_\phi / \mathrm{c}_\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \mathbf{T} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{57}$$

Now by applying Newton's law the equations of motion may be rewritten to a form that is quite useful for its application in control as in [35], [36].

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \mathbf{R} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{58}$$

$$\begin{aligned} &= \begin{bmatrix} -F_t(\mathrm{c}_\phi \mathrm{s}_\theta \mathrm{c}_\psi + \mathrm{s}_\phi \mathrm{s}_\psi) \\ -F_t(\mathrm{c}_\phi \mathrm{s}_\theta \mathrm{s}_\psi - \mathrm{s}_\phi \mathrm{c}_\psi) \\ -mg + F_t \mathrm{c}_\phi \mathrm{c}_\theta \end{bmatrix} + \mathbf{R}\mathbf{f}_f \\ &= \begin{bmatrix} -F_t(\mathrm{c}_\phi \mathrm{s}_\theta \mathrm{c}_\psi + \mathrm{s}_\phi \mathrm{s}_\psi) \\ -F_t(\mathrm{c}_\phi \mathrm{s}_\theta \mathrm{s}_\psi - \mathrm{s}_\phi \mathrm{c}_\psi) \\ -mg + F_t \mathrm{c}_\phi \mathrm{c}_\theta \end{bmatrix} - \mathbf{A} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \end{aligned} \tag{59}$$

where $F_t = F_1 + F_2 + F_3 + F_4$ depicts the thrust exerted on the quadcopter by all four rotors in the body frame. Diagonal matrix $\mathbf{A} = \mathrm{diag}(A_x,\ A_y,\ A_z)$ refers to an approximation of aerodynamic friction in the inertial frame, enabling the quadcopter to reach a terminal velocity or decelerate in the xy-plane without input.

For small angle approximations of the roll and pitch angles, $[p\ q\ r]^T \approx [\dot{\phi}\ \dot{\theta}\ \dot{\psi}]^T$, and $\mathbf{T}(\phi, \theta, \psi)$ approaching identity, resulting in Equation 60.

$$\mathbf{I}\begin{bmatrix}\ddot{\phi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix} \approx \begin{bmatrix}(Iyy - I_{zz})\dot{\theta}\dot{\psi}\\ (Izz - I_{xx})\dot{\phi}\dot{\psi}\\ (Ixx - I_{yy})\dot{\phi}\dot{\theta}\end{bmatrix} + \boldsymbol{\tau} \tag{60}$$

C. Quadcopter Inputs

Choosing the quadcopter system inputs to follow the following convention:

$$U_1 = F_t = F_1 + F_2 + F_3 + F_4 \tag{61}$$
$$U_2 = F_2 - F_4 \tag{62}$$
$$U_2 = F_3 - F_1 \tag{63}$$
$$U_2 = F_1 - F_2 + F_3 - F_4 \tag{64}$$

Allows for Equation 59 and Equation 60 to be rewritten to Equation 5-10.

As mentioned in the quadcopter assumptions, the motors are considered to abide by the following assumptions.

- The thrust and drag constants are proportional to the square of the motor speed.
- The thrust generated by each of the motors is proportional to the square of the motor speed.

As a result, the vertical forces and moments produced by the rotors can be, if so desired, described by Equation 65

$$F_i = k_F \omega_i^2 \tag{65}$$
$$M_i = k_M \omega_i^2 \tag{66}$$

where k_F and k_M are motor constants relating the forces and moments to the motor rotational velocity.

II. Appendix II - Quadcopter Control

A. Linearized System

The linearized system is considered around hovering conditions, resulting in a linear time-invariant system around the equilibrium operating point $(\mathbf{q}_e, \mathbf{u}_e) = (\mathbf{0}^5, \psi, \mathbf{0}^9, mg)$.

$$\begin{aligned}\dot{\mathbf{q}} &= \frac{\partial f(\mathbf{q},\mathbf{u})}{\partial \mathbf{q}}\Big|_{\substack{\mathbf{q}=\mathbf{q}_e \\ \mathbf{u}=\mathbf{u}_e}} (\mathbf{q}-\mathbf{q}_e) + \frac{\partial f(\mathbf{q},\mathbf{u})}{\partial \mathbf{u}}\Big|_{\substack{\mathbf{q}=\mathbf{q_e} \\ \mathbf{u}=\mathbf{u_e}}} (\mathbf{u}-\mathbf{u}_e) \\ &= \mathbf{A}_c\Delta\mathbf{q} + \mathbf{B}_c\Delta\mathbf{u} \qquad (67)\\ \mathbf{y} &= \mathbf{C}_c\Delta\mathbf{q} \qquad (68)\end{aligned}$$

where $\mathbf{A}_c$, $\mathbf{B}_c$ and $\mathbf{C}_c$ are the continuous-time state-space matrices given by

$$\mathbf{A}_c = \begin{bmatrix} 0&0&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&1\\ 0&0&0&-\frac{U_1}{m}\mathrm{s}_\psi&-\frac{U_1}{m}\mathrm{c}_\psi&0&-\frac{K_t}{m}&0&0&0&0&0\\ 0&0&0&\frac{U_1}{m}\mathrm{c}_\psi&-\frac{U_1}{m}\mathrm{s}_\psi&0&0&-\frac{K_t}{m}&0&0&0&0\\ 0&0&0&0&0&0&0&0&-\frac{K_t}{m}&0&0&0\\ 0&0&0&0&0&0&0&0&0&-\frac{K_r}{I_{xx}}&0&0\\ 0&0&0&0&0&0&0&0&0&0&-\frac{K_r}{I_{yy}}&0\\ 0&0&0&0&0&0&0&0&0&0&0&-\frac{K_r}{I_{zz}} \end{bmatrix} \qquad (69)$$

$$\mathbf{B}_c = \begin{bmatrix} 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ \frac{1}{m}&0&0&0\\ 0&\frac{l}{I_{xx}}&0&0\\ 0&0&\frac{l}{I_{yy}}&0\\ 0&0&0&\frac{l}{I_{zz}} \end{bmatrix} \qquad (70)$$

$$\mathbf{C}_c = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&1 \end{bmatrix} \qquad (71)$$

where $\mathbf{I}$ = diag(I_{xx}, I_{yy}, I_{zz}) represents the diagonal inertia matrix of the quadcopter frame. K_t and K_r are the translational and rotational friction associated with aerodynamics. Lastly, m depicts the mass of the quadcopter and l the length of the quadcopter arm, i.e. the distance from the rotational axis of the motors to the center of the quadcopter frame. The output of the system is $\mathbf{y} = [\dot{z},\ \phi,\ \theta,\ \dot{\psi}]^T$ and state-space matrix C_c reflects this.

B. Transfer Functions

To identify the optimal PID control structures, the state-space representation is converted into a transfer function representation. This transformation allows for an easier analysis of the closed-loop dynamics and the subsequent impact of the control methods on the overall stability and performance of the system. The transformation of the state space to a transfer function is accomplished using the Laplace domain.

$$\mathbf{G}(s) = \frac{\mathbf{Y}(s)}{\mathbf{R}(s)} = \mathbf{C}_c(s\mathbf{I} - \mathbf{A}_c)^{-1}\mathbf{B}_c \qquad (72)$$

which results in the four decoupled transfer functions depicted in Equation 73, each row corresponding to their respective output in y.

$$\mathbf{G}(s) = \begin{bmatrix} \frac{1}{ms+K_t} & 0 & 0 & 0 \\ 0 & \frac{l}{I_{xx}s^2+K_r s} & 0 & 0 \\ 0 & 0 & \frac{l}{I_{yy}s^2+K_r s} & 0 \\ 0 & 0 & 0 & \frac{l}{I_{zz}s+K_r} \end{bmatrix} \tag{73}$$

Under these circumstances, the system axes are decoupled, simplifying the required control design.

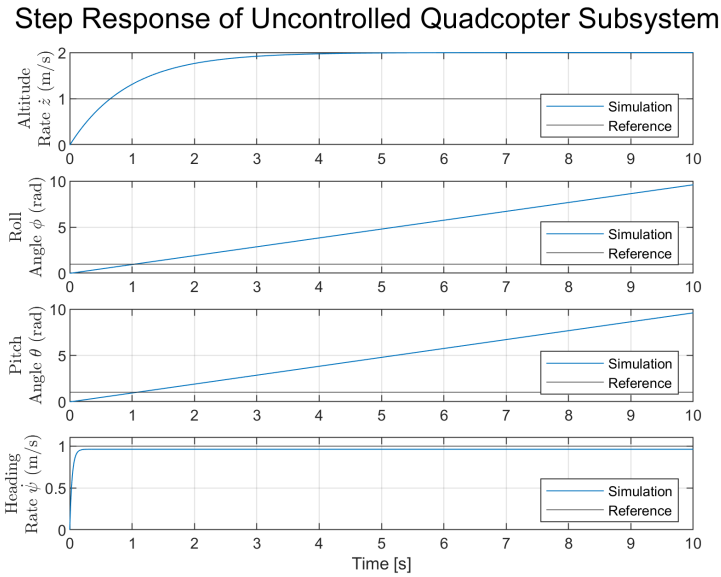

Fig. 22. The Step Response of the Quadrotor Dynamics.

TABLE XI
TRANSFER FUNCTION STEP RESPONSE

Step Response	Alt. Rate $G_1(k)$	Roll $G_2(k)$	Pitch $G_3(k)$	Yaw Rate $G_4(k)$	Units
Settling Time	2.8040	NA	NA	0.1127	s
Rise Time	2.0566	NA	NA	0.0826	s
Overshoot	0	NA	NA	0	%
Settling Max	2.0000				[m/s]
		NA	NA		[rad]
				0.9615	[rad/s]

Now the step response of the system is observed, the altitude rate response is quite slow and shows a steady state error. Similarly, the heading rate response also shows a steady state error, although its response is quite fast. The roll and pitch angles simply increase as an input is provided to the system.

To change this behaviour four general PID controllers are implemented. These four general PID controllers are stacked diagonally in a matrix, each one responsible for one of the four axes of the quadcopter. As seen in Equation 74.

$$\mathbf{H}(s) = \begin{bmatrix} H_1(s) & 0 & 0 & 0 \\ 0 & H_2(s) & 0 & 0 \\ 0 & 0 & H_3(s) & 0 \\ 0 & 0 & 0 & H_4(s) \end{bmatrix} \tag{74}$$

where $H_k(s)$ may be seen in Equation 75 depicting the general formulation of the PID controller in the Laplace domain.

$$H_k(s) = \frac{K_{k,d}s^2 + K_{k,p}s + K_{k,i}}{s} \tag{75}$$

where $k = \{1, 2, 3, 4\}$ each corresponding to the relevant subsystem in Equation 73.

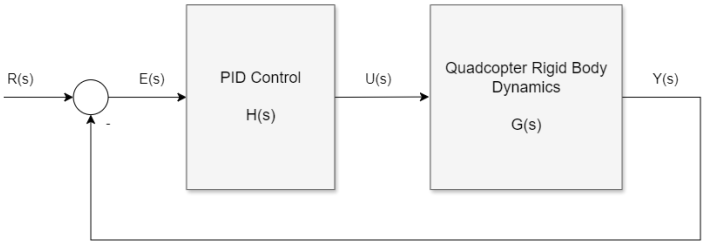

Fig. 23. Quadrotor Control Diagram

C. Control Requirements

To determine the appropriate type of PID control for this system, a few preliminary checks are required.

First, we will determine if the system suffers from steady-state errors in response to a step input. This is achieved by computing the transfer function(s) of the entire system and applying the *Final Value Theorem* to evaluate the steady-state error.

$$e(\infty) = \lim_{s \to 0} sE(s) = \lim_{s \to 0} \frac{sR(s)}{1 + G(s)H(s)} \tag{76}$$

where the transfer function from the Error signal to the Reference signal may be represented by:

$$\frac{\mathbf{E}(s)}{\mathbf{R}(s)} = (\mathbf{I} + \mathbf{G}(s)\mathbf{H}(s))^{-1} \tag{77}$$

$$= \text{diag}(f_1(s), f_2(s), f_3(s), f_4(s)) \tag{78}$$

$$\mathbf{f}(s) = \begin{bmatrix} \frac{s(ms+K_t)}{(m+K_{1,d})s^2+(K_t+K_{1,p})s+K_{1,i}} \\ \frac{s^2(I_{xx}s+K_r)}{I_{xx}s^3+s^2(K_r+lK_{2,d})+lK_{2,p}s+lK_{2,i}} \\ \frac{s^2(I_{yy}s+K_r)}{I_{yy}s^3+s^2(K_r+lK_{3,d})+lK_{3,p}s+lK_{3,i}} \\ \frac{s(I_{zz}s+K_r)}{(I_{zz}+lK_{4,d})s^2+(K_r+lK_{4,p})s+lK_{4,i}} \end{bmatrix} \tag{79}$$

The diagonal entries $f_k(s)$, where $k = 1, 2, 3, 4$, correspond to the elements of the vector $\mathbf{f}(s)$. Applying a step response ($\frac{1}{s}$) to these subsystems then results in the following steady-state errors. As the off-diagonals are zero, the error signals are only influenced directly by their corresponding reference. Thus, it is only the diagonal entries that are of interest.

$$\begin{aligned} \mathbf{e}(\infty) &= \lim_{s\to 0} s\mathbf{R}(s)(\mathbf{I}+\mathbf{G}(s)\mathbf{H}(s))^{-1} \\ &= \lim_{s\to 0} (\mathbf{I}+\mathbf{G}(s)\mathbf{H}(s))^{-1} \\ &= \lim_{s\to 0} \begin{bmatrix} \frac{ms^2+K_ts}{(m+K_{1,d})s^2+(K_t+K_{1,p})s+K_{1,i}} \\ \frac{I_{xx}s^3+K_rs^2}{I_{xx}s^3+s^2(K_r+lK_{2,d})+lK_{2,p}s+lK_{2,i}} \\ \frac{I_{yy}s^3+K_rs^2}{I_{yy}s^3+s^2(K_r+lK_{3,d})+lK_{3,p}s+lK_{3,i}} \\ \frac{I_{zz}s^2+K_rs}{(I_{zz}+lK_{4,d})s^2+(K_r+lK_{4,p})s+lK_{4,i}} \end{bmatrix} \\ &= \lim_{s\to 0} \begin{bmatrix} \frac{ms+K_t}{(m+K_{1,d})s+(K_t+K_{1,p})+\frac{K_{1,i}}{s}} \\ \frac{I_{xx}s+K_r}{I_{xx}s+(K_r+lK_{2,d})+l\frac{K_{2,p}}{s}+l\frac{K_{2,i}}{s^2}} \\ \frac{I_{yy}s+K_r}{I_{yy}s+(K_r+lK_{3,d})+l\frac{K_{3,p}}{s}+l\frac{K_{3,i}}{s^2}} \\ \frac{I_{zz}s+K_r}{(I_{zz}+lK_{4,d})s+(K_r+lK_{4,p})+l\frac{K_{4,i}}{s}} \end{bmatrix} \end{aligned} \tag{80}$$

Based on Equation 80 the four appropriate control structures may be reduced to the following choices.

1) Altitude Controller

   Assuming a simple P-controller results in a steady-state error $\frac{K_t}{Kt+K_{1,P}}$. To ensure that the steady-state error approaches zero over time an integral action, $K_{i,1} \neq 0$, is necessary. As such, the control structure is chosen to be a PI controller.

2) Attitude Controllers

To ensure that the steady-state error approaches zero over time, the proportional action is enough. As such, the control structure is chosen to be a PD controller. Whilst the D-action is not necessary, it can increase the response time of the system. The next subsection will show why this is desirable for the attitude controller.

3) Heading Controller

Assuming a simple P-controller results in a steady-state error $\frac{K_r}{Kr + lK_{4,P}}$. To ensure that the steady-state error approaches zero over time, an integral action, $K_{i,1} \neq 0$, is necessary. As such, the control structure is chosen to be a PI controller.

D. Control Design - Tuning

Given the control structures and subsystems provided in subsection II-B-II-C the desired control gains may now be tuned such that the drone behaves as desired.

The closed loop transfer function of each subsystem may be represented by Equation 81.

$$H_{cl,k}(s) = \frac{G_k(s)H_k(s)}{1 + G_k(s)H_k(s)} \tag{81}$$

where $k = \{1, 2, 3, 4\}$ each corresponding to the relevant subsystem in Equation 73.

To start the tuning process, all of the K_p gains to 1, with the other control gains at 0, this already provides some insight into whether the chosen control structures were correct.

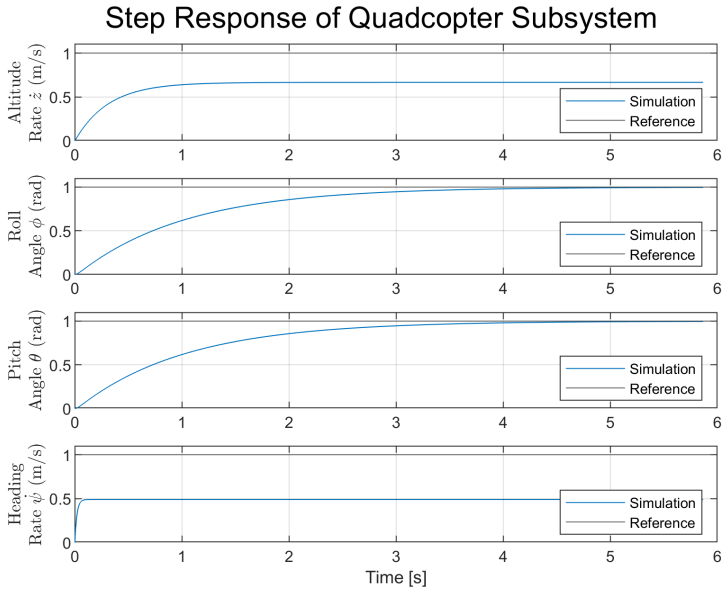

Fig. 24. The step response of the quadcopter system and the controllers implemented, for $K_p = 1$

The subsystem for the altitude rate and the heading rate clearly still show a steady state error, which will be resolved by adding an integrative action. However, the heading rate steady state error is quite large, even as its response is fast, which implies a relatively large integral action. The roll and pitch angles now settle nicely at a step reference of 1, however, they are somewhat slow which will be solved by adding a derivative action.

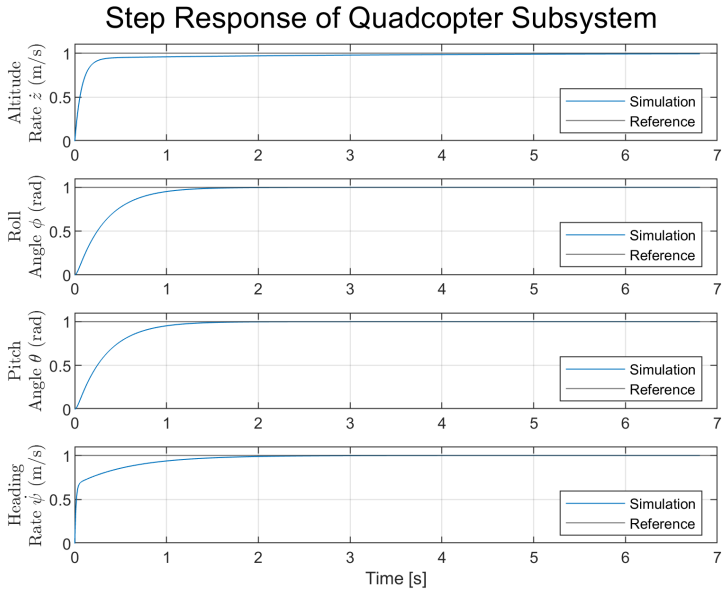

Fig. 25. The step response of the quadcopter system and the controllers implemented, for the final control parameters.

TABLE XII
CONTROLLED SYSTEM STEP RESPONSE

Step Response	Alt. Rate $G_1(k)$	Roll $G_2(k)$	Pitch $G_3(k)$	Yaw Rate $G_4(k)$	Units
Settling Time	0.3748	0.9810	0.9810	1.1314	s
Rise Time	0.2041	0.7048	0.7048	0.7169	s
Overshoot	0	0	0	0	%
Settling Max	0.9933				m/s
		0.9999	0.9999		rad
				1.0000	rad/s

These values were tuned based on design requirements found in some of the available literature [26] [27] [28]. For the pitch and roll angles, reasonable settling times are often in the range of $0.5 \leq \phi, \theta \leq 1.5\ rad$. Information regarding the altitude and heading rate control, however, is more sparse. As the current flight mode is the equivalent of manual flight, and not autonomous flying, it is difficult to obtain information about reasonable rise times and settling times for a specific model of quadcopter. An estimate is made on a number of papers that have quadcopter parameters in the range of this paper, that provide information on the vertical acceleration of the drone. Reasonable values seem to lie in the range of $0.5 \leq \dot{z} \leq 5\ m/s^2$, thus an educated guess is made. Lastly, the heading rate is simply designed with the objective to eliminate the steady state error, its uncontrolled response is already quite fast. The resulting control gains are depicted in Table IV.

III. Appendix III - Haptic Feedback Subsystems

To determine the performance-enhancing force feedback provided to the learner a model predictive controller (MPC) is designed. For the current framework, haptic feedback is only provided for one axis, namely the z-axis related to the altitude rate of the drone. As a result, the complete system known by the model predictive controller may be simplified such that only a subsystem of the quadcopter dynamics and the dynamics of the haptic interface are incorporated. section III provides more details regarding the derivation and/or identification of these subsystems.

A. Quadcopter Subsystem

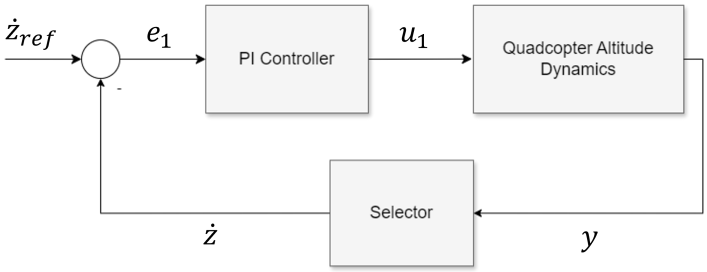

Fig. 26. The control diagram of the quadcopter altitude dynamics and a PI controller.

The altitude dynamics may be described by Equation 82.

$$\begin{bmatrix} \dot{z} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{K_t}{m} \end{bmatrix} \begin{bmatrix} z \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} U_1 = \mathbf{A}_1 \begin{bmatrix} z \\ \dot{z} \end{bmatrix} + \mathbf{B}_1 U_1 \tag{82}$$

$$y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z \\ \dot{z} \end{bmatrix} = \mathbf{C}_1 \begin{bmatrix} z \\ \dot{z} \end{bmatrix} \tag{83}$$

The PI controller is of the following equations given the error definition $e_1 = \dot{z}_{\text{ref}} - \dot{z}$.

$$U_1 = K_{1,P}\ e_1 + K_{1,I} \int e_1 dt \tag{84}$$

We denote the following variable $c = \int e_1 dt$ for ease of notation and combine these equations.

$$\begin{aligned} \begin{bmatrix} \dot{z} \\ \ddot{z} \end{bmatrix} &= \mathbf{A}_1 \begin{bmatrix} z \\ \dot{z} \end{bmatrix} + \mathbf{B}_1 U_1 \\ &= \mathbf{A}_1 \begin{bmatrix} z \\ \dot{z} \end{bmatrix} + \mathbf{B}_1 (K_{1,I}\ c + K_{1,P}\ e_1) \\ &= \mathbf{A}_1 \begin{bmatrix} z \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_{1,I}}{m} \end{bmatrix} c + \begin{bmatrix} 0 \\ \frac{K_{1,P}}{m} \end{bmatrix} e_1 \\ &= \mathbf{A}_1 \begin{bmatrix} z \\ \dot{z} \end{bmatrix} + \mathbf{A}_c c + \mathbf{B}_c (\dot{z}_{\text{ref}} - \mathbf{S}\mathbf{C}_1 \begin{bmatrix} z \\ \dot{z} \end{bmatrix}) \end{aligned} \tag{85}$$

where the selector $\mathbf{S} = \begin{bmatrix} 0 & 1 \end{bmatrix}$. With this formulation, the original system of the drone can be augmented with the PI controller, resulting in a state space representation of the complete quadcopter.

$$\begin{bmatrix} \dot{z} \\ \ddot{z} \\ \dot{c} \end{bmatrix} = \left[\begin{array}{c|c} \mathbf{A}_1 - \mathbf{B}_c\mathbf{S}\mathbf{C}_1 & \mathbf{A}_c \\ \hline -\mathbf{S}\mathbf{C}_1 & 0 \end{array}\right] \begin{bmatrix} z \\ \dot{z} \\ \hline c \end{bmatrix} + \left[\begin{array}{c} \mathbf{B}_c \\ \hline 1 \end{array}\right] \dot{z}_{\text{ref}} \tag{86}$$

$$= \left[\begin{array}{c|c} \dot{z} & \\ \ddot{z} & \\ \hline \dot{c} & \end{array}\right] \left[\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \hline 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c} z \\ \dot{z} \\ \hline c \end{array}\right] \tag{87}$$

B. Sigma.7 subsystem

As an initial assumption the sigma.7 device is modeled as a mass-spring-damper system.

$$M\ddot{z}_s + C\dot{z}_s + Kz_s = u + d \tag{88}$$

where the variables are defined as:

M : End-effector mass [kg].
C : Viscous damping coefficient [Ns/m].
K : Linear spring coefficient [N/m].
u : Forces applied on the system by the motors [N].
d : Other Forces applied on the system, e.g. human inputs, (constant) disturbances, etc. [N]
z_s : The cartesian z-coordinate of the sigma.7 device [m]

Assuming that $d = 0$, the system may also be represented as state-space representation, as depicted in Equation 89.

$$\begin{bmatrix} \dot{z}_s \\ \ddot{z}_s \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{K}{M} & -\frac{C}{m} \end{bmatrix} \begin{bmatrix} z_s \\ \dot{z}_s \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \end{bmatrix} u = \mathbf{A}_s \begin{bmatrix} z_s \\ \dot{z}_s \end{bmatrix} + \mathbf{B}_s u \tag{89}$$

$$y_s = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z_s \\ \dot{z}_s \end{bmatrix} = \mathbf{C}_s \begin{bmatrix} z_s \\ \dot{z}_s \end{bmatrix} \tag{90}$$

The linear second-order model was identified using a dataset obtained from the sigma.7 wherein a pseudorandom binary signal was used as a force input to the z-axis as seen in Figure 27, and the position and velocity data were measured on the same axis.

As the x-y plane is constrained using PD controllers, the identification process is focused on attaining a representative model of the vertical behaviour of the system under these specific conditions. The data were sampled at a sampling rate of 0.001s. Using the MATLAB system identification toolbox, the idgrey function was used to determine the best-fitting parameters using the model defined in Equation 29. The workspace of the system is limited and there is a set of PD controllers at the edges, as such it is desirable to remove any of the data associated with this to prevent the PD

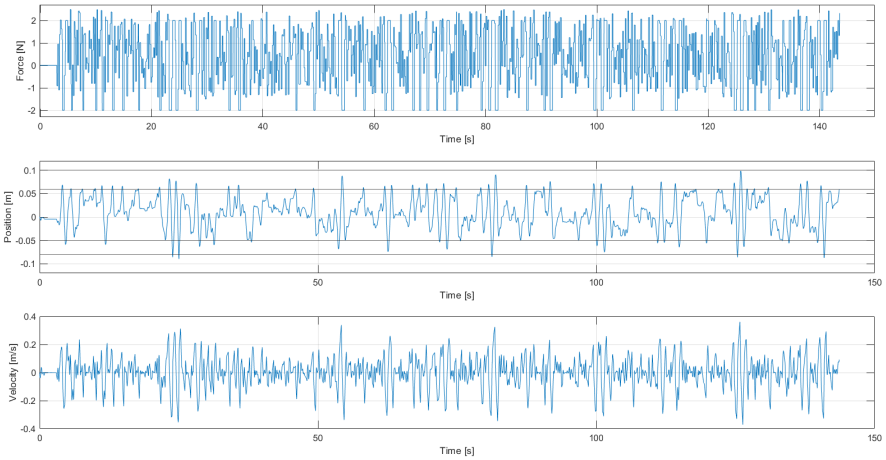

Fig. 27. This figure shows the dataset used for the system identification process of the sigma.7 model. The first dataset is the Psuedo-Random Binary Signal Input. The second plot is the measured z-position of the sigma.7, and the third is the measured velocity.

control dynamics from influencing the system dynamics. The data was pre-processed to remove these unreliable data points. This is done by splitting the dataset into multiple datasets, which are then collected into an iddata structure in Matlab suitable for the identification process. The dataset is split into two, the first half is used for the estimation of the model, the second half is used to validate the model.

As may be observed in Figure 28 the linear identified model performs quite poorly and does not adequately describe the dynamics of the system over its entire range of operation. The normalized root mean square (NRMSE) measure is used to provide a goodness of fit between the output measurement data and that of the simulated model output.

d

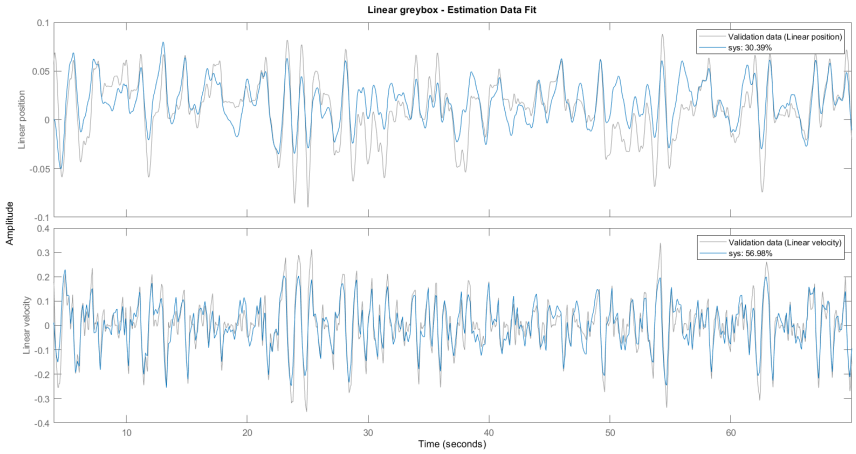

Fig. 28. The identified model output (blue) superimposed over the measured position and velocity data (grey) from the sigma.7. This plot shows the estimation data used to identify the model. The NRMSE values have been included in the figure.

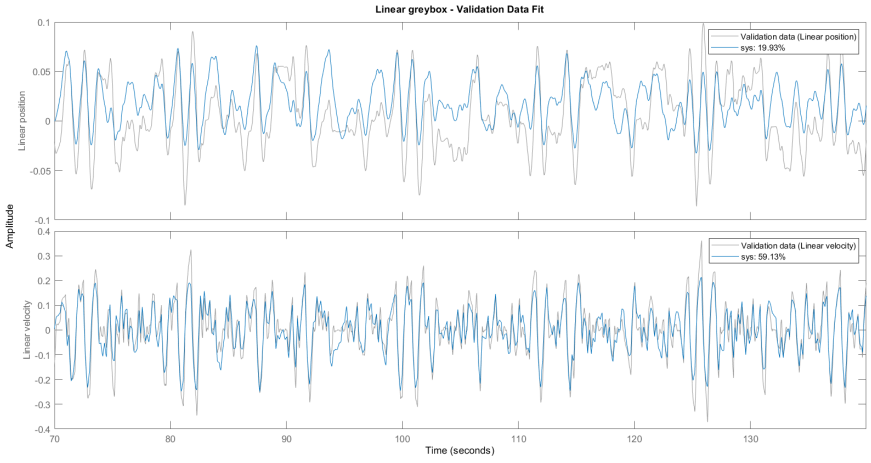

Fig. 29. The identified model output (blue) superimposed over the measured position and velocity data (grey) from the sigma.7. This plot shows the validation data used to validate the model. The NRMSE values have been included in the figure.

However, as mentioned before, the model should perform sufficiently over the chosen horizon time of the MPC. As such, the performance is reevaluated with a small experiment for a horizon time of 0.5 s. Both the identified model and the sigma.7 haptic interface will be provided with the same input data, however, every 0.5 s the identified model is provided with new initial conditions using the measurements of the sigma.7.

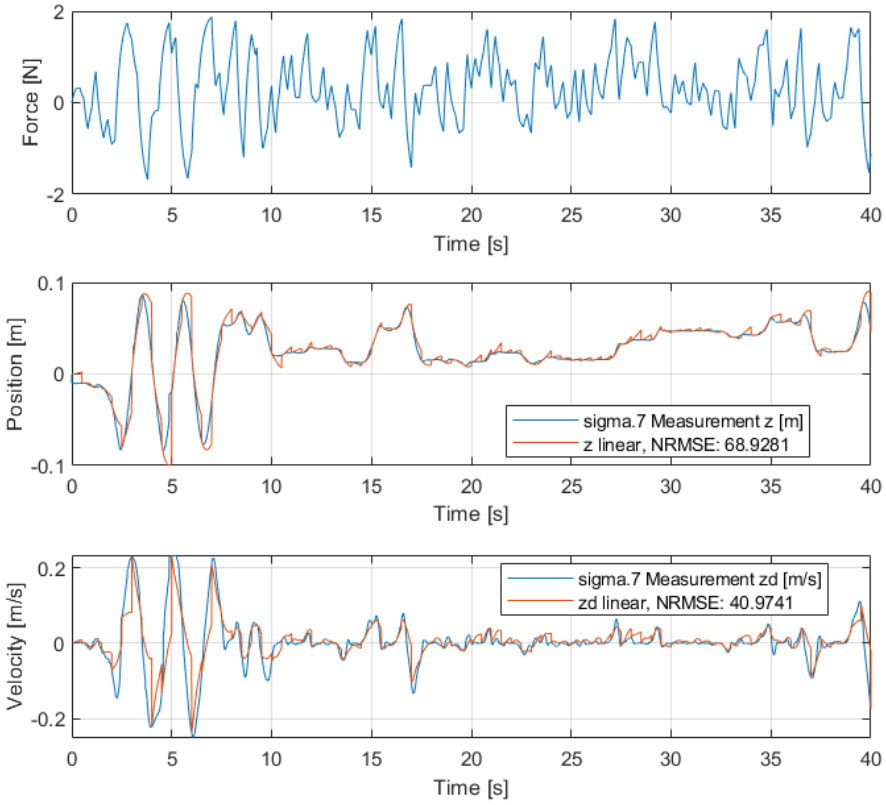

Fig. 30. The identified model compared to a measured data set, for a horizon time of 0.5 seconds. The identified model output (red) is superimposed over the measured position and velocity data (blue) from the sigma.7. The NRMSE values have once more been included in the figure.

As the horizon time is increased, the identified model also performs increasingly worse, as may be observed in Figure 31. This limits the design choices regarding the horizon time for the MPC using this model.

f

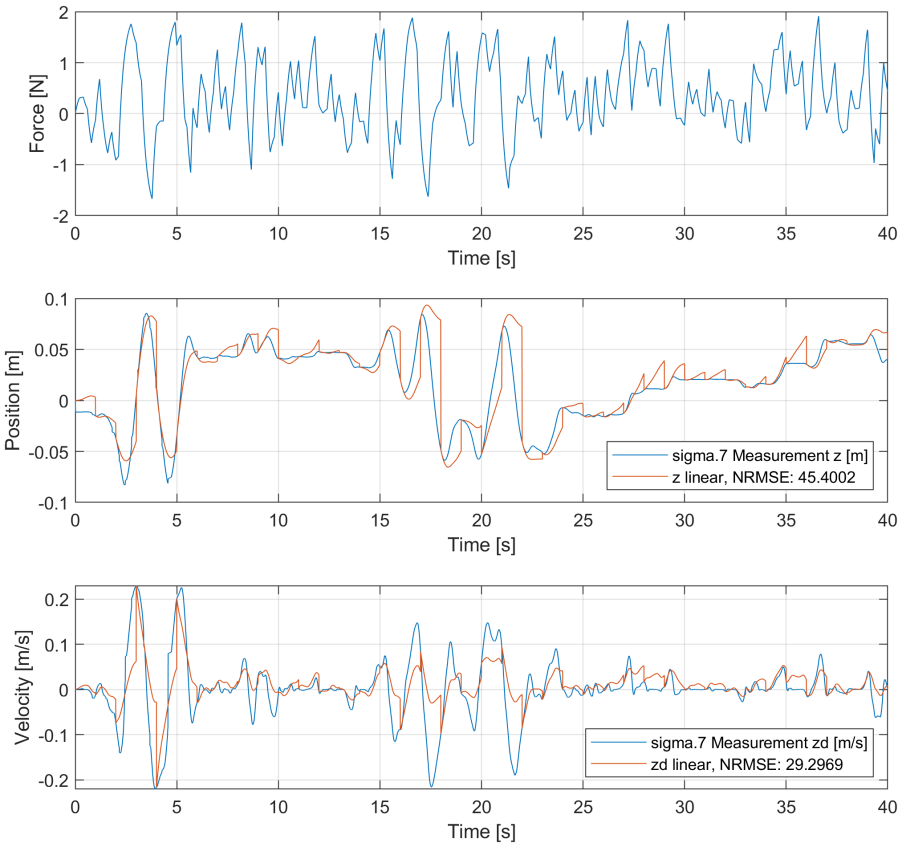

Fig. 31. The identified model compared to a measured data set, for a horizon time of 1.0 seconds. The identified model output (red) is superimposed over the measured position and velocity data (blue) from the sigma.7. The NRMSE values have once more been included in the figure.

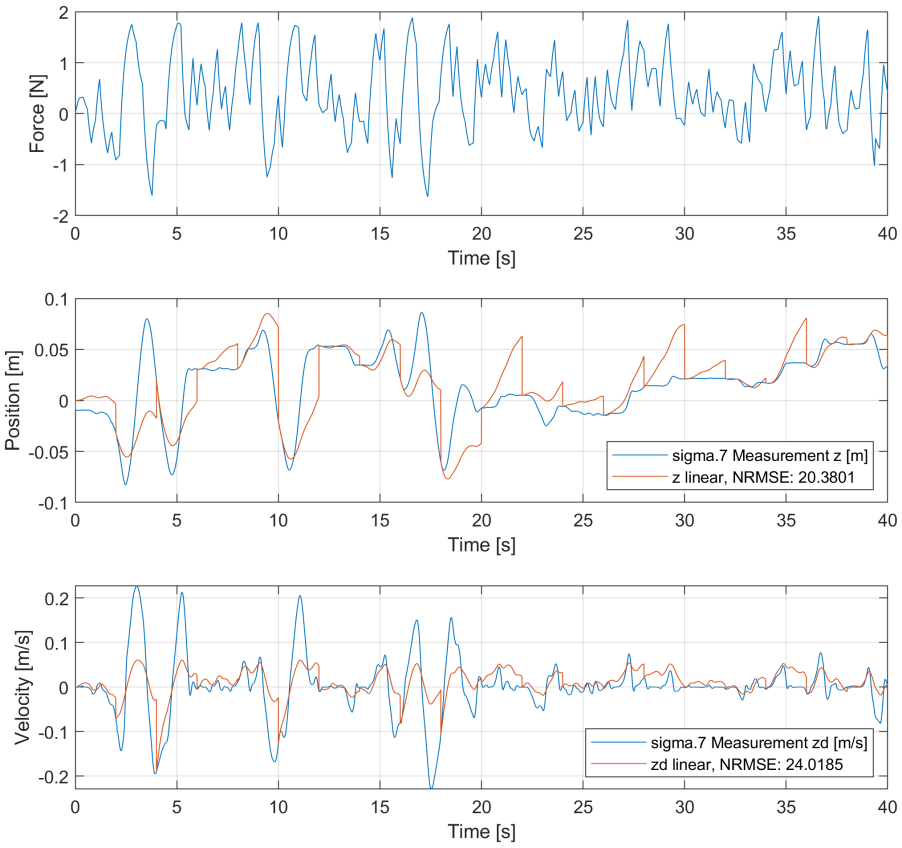

Fig. 32. The identified model compared to a measured data set, for a horizon time of 2.0 seconds. The identified model output (red) is superimposed over the measured position and velocity data (blue) from the sigma.7. The NRMSE values have once more been included in the figure.

C. Complete System

The complete system that should be known to the Model Predictive Controller may then be visualized in Figure 33. The gain K_s bridges the workspace differences between the sigma.7 and the quadcopter. Its value determines the maximum $\dot{z}_{\text{ref}}$ that can be provided to the quadcopter and as such limits its attainable velocity.

Each of the subsystems as defined in subsection III-A and subsection III-B may then be combined into the final representation of the system dynamics following the schematic provided in Figure 33. The derivation of which can be found below.

$$\begin{bmatrix} \dot{z} \\ \ddot{z} \\ \dot{c} \end{bmatrix} = \left[\begin{array}{c|c} \mathbf{A}_1 - \mathbf{B}_c\mathbf{S}\mathbf{C}_1 & \mathbf{A}_c \\ \hline -\mathbf{S}\mathbf{C}_1 & 0 \end{array}\right] \begin{bmatrix} z \\ \dot{z} \\ \hline c \end{bmatrix} + \left[\begin{array}{c} \mathbf{B}_c \\ \hline 1 \end{array}\right] \dot{z}_{\text{ref}} \tag{91}$$

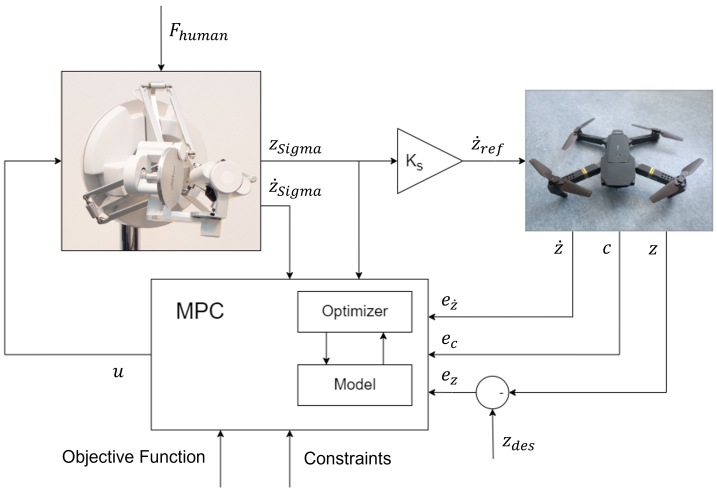

Fig. 33. Overview of System structure

$$\dot{z}_{\text{ref}} = K_s z_s \tag{92}$$

$$\begin{bmatrix} \dot{z}_s \\ \ddot{z}_s \end{bmatrix} = \mathbf{A}_s \begin{bmatrix} z_s \\ \dot{z}_s \end{bmatrix} + \mathbf{B}_s u \tag{93}$$

$$y_s = \mathbf{C}_s \begin{bmatrix} z_s \\ \dot{z}_s \end{bmatrix} \tag{94}$$

Given these subsystems there are two linear system connected sequentially by a gain K_s, they may be augmented into one representation given $\dot{z}_{\text{ref}} = K_s z_s = K_s \begin{bmatrix} 1 & 0 \end{bmatrix} y_s$.

$$\left[\begin{array}{c} \dot{z} \\ \ddot{z} \\ \hdashline \dot{c} \end{array}\right] = \left[\begin{array}{c:c} \mathbf{A}_1 - \mathbf{B}_c\mathbf{S}\mathbf{C}_1 & \mathbf{A}_c \\ \hdashline -\mathbf{S}\mathbf{C}_1 & 0 \end{array}\right] \left[\begin{array}{c} z \\ \dot{z} \\ \hdashline c \end{array}\right] + \left[\begin{array}{c:c} \mathbf{B}_c K_s & \mathbf{0} \\ \hdashline K_s & 0 \end{array}\right] \begin{bmatrix} z_s \\ \dot{z}_s \end{bmatrix} \tag{95}$$

$$\begin{bmatrix} \dot{z}_s \\ \ddot{z}_s \end{bmatrix} = \mathbf{A}_s \begin{bmatrix} z_s \\ \dot{z}_s \end{bmatrix} + \mathbf{B}_s u \tag{96}$$

The states may then be combined by augmenting the state vector to $\mathbf{x}(k) = \begin{bmatrix} z & \dot{z} & c & z_s & \dot{z}_s \end{bmatrix}^T$ resulting in the final representation of the system.

$$\left[\begin{array}{c} \dot{z} \\ \ddot{z} \\ \hdashline \dot{c} \\ \hdashline \dot{z}_s \\ \ddot{z}_s \end{array}\right] = \left[\begin{array}{c:c:c} \mathbf{A}_1 - \mathbf{B}_c\mathbf{S}\mathbf{C}_1 & \mathbf{A}_c & \mathbf{B}_c K_s \quad 0 \\ \hdashline -\mathbf{S}\mathbf{C}_1 & 0 & K_s \quad 0 \\ \hdashline \mathbf{0} & 0 & \mathbf{A}_s \end{array}\right] \left[\begin{array}{c} z \\ \dot{z} \\ \hdashline c \\ \hdashline z_s \\ \dot{z}_s \end{array}\right] + \left[\begin{array}{c} \mathbf{0} \\ \hdashline 0 \\ \hdashline \mathbf{B}_s \end{array}\right] u \tag{97}$$

By filling out the matrices as defined in the earlier sections, the parameters reminiscent of the quadcopter and the sigma.7 return in a state-space representation of the final LTI system.

$$\mathbf{x}(k+1) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & -\frac{K_p+K_t}{m} & \frac{K_I}{m} & \frac{K_pK_s}{m} & 0 \\ 0 & -1 & 0 & K_s & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{K}{M} & -\frac{C}{M} \end{bmatrix} \mathbf{x}(k) \tag{98}$$

$$+ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{M} \end{bmatrix} u_{\text{MPC}}(k) \tag{99}$$

$$\mathbf{y}(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}(k) \tag{100}$$

D. A Nonlinear System

Additional to a linear model of the sigma.7, an attempt at identifying a nonlinear model was also made using nlgrey. The same dataset was used for the identification process of the nonlinear model, but the dynamical equation chosen to represent the system was augmented with an additional friction term, depicted in Equation 101.

$$M\ddot{z}_s + f_v\dot{z}_s + Kz_s + f_c\text{sign}(\dot{z}) + d = u \tag{101}$$

Where the variables are defined as:

M : End-effector mass [kg].
f_v : Viscous damping effects [Ns/m].
f_c : Coulomb damping effects [N].
K : Linear spring coefficient [N/m].
u : Forces applied on the system by the motors [N].
d : Other Forces applied on the system, e.g. human inputs, (constant) disturbances, etc. [N]
z_s : The cartesian z-coordinate of the sigma.7 device [m]

Which may also be written in a nonlinear representation as,

$$f(u, z, \dot{z}) = \begin{bmatrix} \dot{z} \\ \frac{1}{M}u - \frac{G}{M} - \frac{f_v}{M}\dot{z} - \frac{f_c}{M}\text{sign}(\dot{z}) - \frac{d}{M} \end{bmatrix} \tag{102}$$

$$h(z, \dot{z}) = \begin{bmatrix} z \\ \dot{z} \end{bmatrix} \tag{103}$$

where $f(u, z, \dot{z})$ represents the state function, and $h(z, \dot{z})$ the output. By choosing the state vector as $\mathbf{x}(k) = \begin{bmatrix} z & \dot{z} & c & z_s & \dot{z}_s & d \end{bmatrix}$, where $d \neq 0$ and constant. The final system may then be represented, in turn, as

$$\mathbf{f}(\mathbf{x}, u) = \begin{bmatrix} \dot{x}_1(k) \\ \dot{x}_2(k) \\ \dot{x}_3(k) \\ \dot{x}_4(k) \\ \dot{x}_5(k) \\ \dot{x}_6(k) \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{K_p+K_t}{m}x_2 + \frac{K_I}{m}x_3 + \frac{K_pK_s}{m}x_4 \\ -x_2 + K_s x_4 \\ x_5 \\ \frac{1}{M}u - \frac{G}{M} - \frac{f_v}{M}\dot{z} - \frac{f_c}{M}\text{sign}(\dot{z}) - \frac{d}{M} \\ 0 \end{bmatrix} \tag{104}$$

The identified nonlinear system performs quite a bit better than its linear counterpart over its entire range of operation, which may be observed in Figure 34 and Figure 35. The corresponding identified parameters are also shown in Table XIII.

TABLE XIII
THE IDENTIFIED PARAMETERS FOR THE SIGMA.7 SUBSYSTEM

Parameter	Value	Unit
M	1.4402	kg
f_v	1.6496	Ns/m
f_c	0.7074	N
K	5.0636	N/m
d	-0.1838	[N]

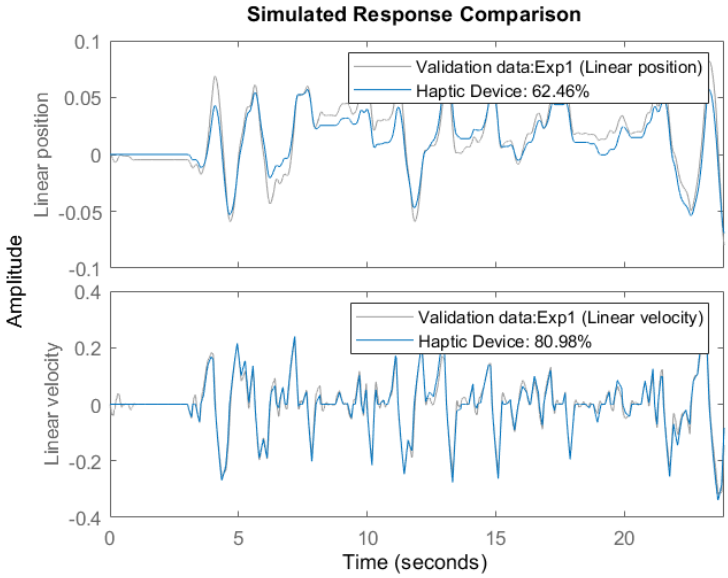

Fig. 34. The identified model output (blue) superimposed over the measured position and velocity data (grey) from the sigma.7. This plot shows the estimation data used to identify the model. The NRMSE values have been included in the figure.

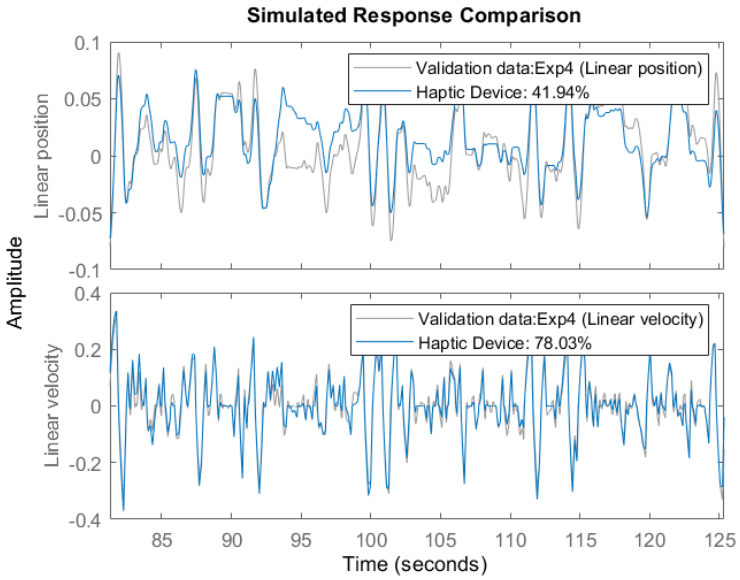

Fig. 35. The identified model output (blue) superimposed over the measured position and velocity data (grey) from the sigma.7. This plot shows the validation data used to validate the model. The NRMSE values have been included in the figure.

The performance of this model over the chosen time horizon of 0.5 s, as depicted in Figure 36, also performs quite a bit better than that of the linear model. However, the most interesting part is that this representation is essentially a piecewise linear system that switches between two states due to the incorporation of the sign function. This behaviour changes with the velocity direction of the sigma.7, which may correspond to the sigma.7 experiencing less friction when moving down into the direction of gravity and more during an upward movement.

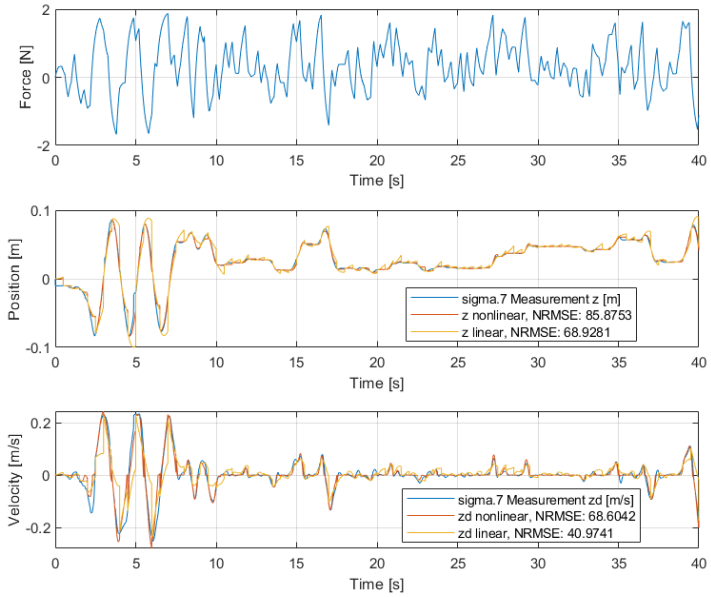

Fig. 36. In this figure, the model output of a nonlinear model and the linear model described in this section have been plotted. Every 0.5 seconds, the initial conditions of the model are reapplied based on the measured position and velocity data of the sigma.7. Additionally, the NMRSE errors, based on the mean of the measured data, have been noted.

Whilst this model does perform significantly better than the linear model, its nonlinear nature made obtaining and implementing this model quite difficult. Due to limitations in time and complexity, the project itself was continued with the linear model.

IV. Appendix IV - Model Predictive Controller Design

Given a discrete LTI system described by the following form.

$$\mathbf{x}(k+1) = \mathbf{A}_d\mathbf{x}(k) + \mathbf{B}_d u(k) \tag{105}$$

$$\mathbf{y}(k) = \mathbf{C}_d\mathbf{x}(k) \tag{106}$$

where $\mathbf{A}_d \in \mathbb{R}^{n\times n}$, $\mathbf{B}_d \in \mathbb{R}^n$, and $\mathbf{C}_d \in \mathbb{R}^{p\times n}$ for which n is defined to be the number of states, and p the number of outputs. Similarly, $\mathbf{x} \in \mathbb{R}^n$, $u \in \mathbb{R}$, and $\mathbf{y} \in \mathbb{R}^p$.

A. Slew constraints

To incorporate slew constraints within the MPC the basic state-space representation is augmented with an additional state $x_u(k) = u(k-1)$ using $\Delta u(k) = u(k) - u(k-1)$ to the following form.

$$\begin{aligned}\begin{bmatrix}\mathbf{x}(k+1)\\ x_u(k+1)\end{bmatrix} &= \begin{bmatrix}\mathbf{A}_d & \mathbf{B}_d\\ \mathbf{0} & 1\end{bmatrix}\begin{bmatrix}\mathbf{x}(k)\\ x_u(k)\end{bmatrix} + \begin{bmatrix}\mathbf{B}_d\\ 0\end{bmatrix}\Delta u(k) \\ &= \mathbf{A}_a\begin{bmatrix}\mathbf{x}(k)\\ x_u(k)\end{bmatrix} + \mathbf{B}_a\Delta u(k)\end{aligned} \tag{107}$$

$$\begin{aligned}\mathbf{y}(k) &= \begin{bmatrix}\mathbf{C}_d & 0\end{bmatrix}\begin{bmatrix}\mathbf{x}(k)\\ x_u(k)\end{bmatrix} \\ &= \mathbf{C}_a\begin{bmatrix}\mathbf{x}(k)\\ x_u(k)\end{bmatrix}\end{aligned} \tag{108}$$

where $\mathbf{A}_a$, $\mathbf{B}_a$, and $\mathbf{C}_a$ represent the augmented state-space matrices, the subscript a referencing to the augmented system.

B. State Solution

By expressing the states in terms of the inputs and the initial condition, the number of variables involved in the optimization may be reduced. This is done by recursively substituting the augmented state vector $\mathbf{x}_a(k)$ into the dynamics of $\mathbf{x}_a(k+1)$, resulting in the following formula.

$$\mathbf{x}_a(k) = \mathbf{A}_a^k\mathbf{x}_a(0) + \sum_{j=0}^{k-1}\mathbf{A}_a^{k-1-j}\mathbf{B}_a\Delta u(j) \tag{109}$$

By placing $\mathbf{x}_a(1)$ until $\mathbf{x}_a(N)$ in a vector $\mathbf{x}_{a,N} \in \mathbb{R}^{nN}$, the state solution over the entirety of the MPC horizon N can be obtained. The same may be done for the inputs indicated by $\Delta\mathbf{u}_{a,N} \in \mathbb{R}^{mN}$.

$$\mathbf{x}_{a,N} = \mathbf{S}\mathbf{x}_a(0) + \mathbf{T}\Delta\mathbf{u}_{a,N} \tag{110}$$

where $\mathbf{S} \in \mathbb{R}^{(n+1)N\times(n+1)}$, $\mathbf{T} \in \mathbb{R}^{(n+1)N\times N}$, and initial condition $\mathbf{x}_a(0) \in \mathbb{R}^{n+1}$ These matrices only depend on $\mathbf{A}_a$ and $\mathbf{B}_a$, allowing them to be computed once.

C. Constraints

The plant consists of a cascade of several systems, the force dimension sigma.7 haptic interface, the quadcopter dynamics with its internal controller, and a gain to compensate for the differences in workspaces. As a result, the system is subject to both constraints related to mechanical limitations and desired constraints. The constraint sets are given as follows.

$$\mathbb{U} := \{\Delta u(k) \in \mathbb{R}, \quad |\Delta u(k)| \leq 2\} \tag{111}$$

$$\mathbb{X} := \left\{ \mathbf{x}_a(k) \in \mathbb{R}^{n+1}, \quad \begin{bmatrix} -0.115 \\ -2 \end{bmatrix} \leq \begin{bmatrix} z_s(k+1) \\ x_u(k+1) \end{bmatrix} \leq \begin{bmatrix} 0.132 \\ 2 \end{bmatrix} \right\} . \tag{112}$$

Similar to subsection IV-B the state constraints may also be rewritten in terms of the initial condition and the input. This is done by combining the state solution in Equation 110, with the augmented system. As a result, they are depicted as inequality constraints in a way that is accepted by most optimization solvers.

$$\begin{bmatrix} -\mathbf{T} \\ \mathbf{T} \end{bmatrix} \Delta \mathbf{u}_{a,N} \leq \begin{bmatrix} -(\mathbf{x}_{min} - \mathbf{S}\mathbf{x}_a(0)) \\ \mathbf{x}_{max} - \mathbf{S}\mathbf{x}_a(0) \end{bmatrix} \tag{113}$$

Due to the dependency on the initial state at the start of each time step, this constraint must be implemented online and recomputed at every time step.

D. Optimal Control Problem

The MPC problem with no reference tracking is slightly different.

$$\mathbb{P}_N(\mathbf{x}_a(0)) : \begin{cases} \min\limits_{\Delta u_0, \ldots, \Delta u_{N-1}} & J(\mathbf{x}_a(0), \Delta \mathbf{u}_{a,N}) \\ s.t & \mathbf{x}_a(k+1) = \mathbf{A}_a \mathbf{x}_a(k) + \mathbf{B}_a \Delta u(k) \\ & \mathbf{x}_a(0) \in \mathbb{X} \\ & \mathbf{x}_a(k+1) \in \mathbb{X}, \quad k = 0, \ldots, N \\ & \Delta u(k) \in \mathbb{U}, \quad k = 0, \ldots, N-1 \end{cases} \tag{114}$$

The objective function $J(\cdot)$ to be minimized in the MPC optimization is a quadratic function as follows:

$$J(\mathbf{x}_a(0), \Delta \mathbf{u}_{a,N}) = \sum_{j=0}^{N-1} \{\ell(\mathbf{x}(j), \Delta u(j))\} + V_f(\mathbf{x}(N))) \tag{115}$$

where $\ell(\cdot)$ depicting the stage cost and $V_f(\cdot)$ the terminal cost of the objective function.

$$\ell(\mathbf{x}(k), \Delta u(k)) = ||\mathbf{x}(k)||^2_{\mathbf{Q}} + ||\Delta u(k)||^2_{\mathbf{R}} \tag{116}$$

$$V_f(\mathbf{x}(k)) = ||\mathbf{x}(k)||^2_{\mathbf{Q}_f} \tag{117}$$

In practice, the cost function may also be expressed in terms of the inputs and the initial condition.

$$J(\mathbf{x}_a(0), \Delta \mathbf{u}_{a,N}) = \begin{bmatrix} \mathbf{x}(0) \\ \vdots \\ \mathbf{x}(N) \end{bmatrix}^T \begin{bmatrix} \mathbf{Q} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{Q}_f \end{bmatrix} \begin{bmatrix} \mathbf{x}(0) \\ \vdots \\ \mathbf{x}(N) \end{bmatrix} \tag{118}$$

$$
\begin{aligned}
&+ \begin{bmatrix} \Delta u(0) \\ \vdots \\ \Delta u(N-1) \end{bmatrix}^T \begin{bmatrix} R & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & R \end{bmatrix} \begin{bmatrix} \Delta u(0) \\ \vdots \\ \Delta u(N-1) \end{bmatrix} \\
&= \mathbf{x}_{a,N}^T \tilde{\mathbf{Q}} \mathbf{x}_{a,N} + \Delta \mathbf{u}_{a,N}^T \tilde{\mathbf{R}} \Delta \mathbf{u}_{a,N} \\
&= (\mathbf{S}\mathbf{x}_a(0) + \mathbf{T}\Delta \mathbf{u}_{a,N})^T \tilde{\mathbf{Q}} && (119) \\
&\quad (\mathbf{S}\mathbf{x}_a(0) + \mathbf{T}\Delta \mathbf{u}_{a,N}) && (120) \\
&\quad + \Delta \mathbf{u}_{a,N}^T \tilde{\mathbf{R}} \Delta \mathbf{u}_{a,N} \\
&= \mathbf{x}_a(0)^T \mathbf{S}^T \tilde{\mathbf{Q}} \mathbf{S} \mathbf{x}_a(0) && (121) \\
&\quad + 2\mathbf{x}_a(0)^T \mathbf{S}^T \tilde{\mathbf{Q}} \mathbf{T} \Delta \mathbf{u}_{a,N} \\
&\quad + \Delta \mathbf{u}_{a,N}^T (\mathbf{T}^T \tilde{\mathbf{Q}} \mathbf{T} + \tilde{\mathbf{R}}) \Delta \mathbf{u}_{a,N} && (122) \\
&\approx \frac{1}{2} \Delta \mathbf{u}_{a,N}^T \mathbf{H} \Delta \mathbf{u}_{a,N} + \mathbf{f}^T \Delta \mathbf{u}_{a,N} && (123)
\end{aligned}
$$

In MATLAB, this formulation is useful for solving quadratic programming problems, which are often represented as follows:

$$
J_{\mathrm{M}} = \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x} + \mathbf{f}^T \mathbf{x} \tag{124}
$$

where x represents the optimization variable $\Delta \mathbf{u}_{a,N}$, $\mathbf{H} = 2(\mathbf{T}^T \tilde{\mathbf{Q}} \mathbf{T} + \tilde{\mathbf{R}})$, and $\mathbf{f}^T = 2\mathbf{x}_a(0)^T \mathbf{S}^T \tilde{\mathbf{Q}} \mathbf{T}$ Since $\mathbf{f}$ depends on $\mathbf{x}_a(0)$ it must be updated at the beginning of each optimization.

E. Optimal Control Problem - Reference Tracking

The MPC problem, with reference tracking, is then reformulated as the following optimal control problem.

$$
\mathbb{P}_N(\mathbf{x}_a(0)) : \begin{cases} \min\limits_{\Delta u_0, \ldots, \Delta u_{N-1}} & J_{\mathrm{R}}(\mathbf{x}_a(0), \Delta \mathbf{u}_{a,N}) \\ s.t & \mathbf{x}_a(k+1) = \mathbf{A}_a \mathbf{x}_a(k) + \mathbf{B}_a \Delta u(k) \\ & \mathbf{x}_a(0) = \mathbf{x}_a(0) \in \mathbb{X} - x_r, \\ & \mathbf{x}_a(k+1) \in \mathbb{X} - x_r, \\ & \qquad k = 0, \ldots, N \\ & \Delta u(k) \in \mathbb{U} - u_r, \\ & \qquad k = 0, \ldots, N-1 \end{cases} \tag{125}
$$

The objective function $J_{\mathrm{R}}(\cdot)$ is adjusted slightly in comparison to $J(\cdot)$ for the reference tracking problem. The stage cost $\ell(\cdot)$ and the terminal cost $V_f(\cdot)$ of the objective function are now defined relative to the reference.

$$
\begin{aligned}
\ell(\mathbf{x}(k), \Delta u(k)) = &||\mathbf{x}(k) - \mathbf{x}_{ref}(k)||_{\mathbf{Q}}^2 + \ldots && (126) \\
&||\Delta u(k) - \Delta u_{ref}(k)||_{\mathbf{R}}^2 \\
V_f(\mathbf{x}(k)) = &||\mathbf{x}(k) - \mathbf{x}_{ref}(k)||_{\mathbf{Q}_f}^2 && (127)
\end{aligned}
$$

Rewriting this cost function is a similar process as done for Equation 123. In practice, the cost function may also be expressed in terms of the inputs and the initial condition.

$$\begin{aligned} J_{\mathrm{R}}(\mathbf{x}_a(0), \Delta\mathbf{u}_{a,N}) &= (\mathbf{x}_{a,N} - \mathbf{x}_{ref,N})^T \tilde{\mathbf{Q}} (\mathbf{x}_{a,N} - \mathbf{x}_{ref,N}) \\ &\quad + (\Delta\mathbf{u}_{a,N} - \Delta\mathbf{u}_{ref,N})^T \tilde{\mathbf{R}} (\Delta\mathbf{u}_{a,N} - \Delta\mathbf{u}_{ref,N}) \\ &\approx \Delta\mathbf{u}_{a,N}^T (\mathbf{T}^T \tilde{\mathbf{Q}} \mathbf{T} + \tilde{\mathbf{R}}) \Delta\mathbf{u}_{a,N} \\ &\quad + 2(\mathbf{x}_a(0)^T \mathbf{S}^T \tilde{\mathbf{Q}} \mathbf{T} && (128) \\ &\quad - \mathbf{x}_{\text{ref},N}^T \tilde{\mathbf{Q}} \mathbf{T} - \Delta\mathbf{u}_{\text{ref},N}^T \tilde{\mathbf{R}}) \Delta\mathbf{u}_{a,N} && (129) \\ &\approx \Delta\mathbf{u}_{a,N}^T \mathbf{H} \Delta\mathbf{u}_{a,N} + 2\mathbf{f}^T \Delta\mathbf{u}_{a,N} && (130) \end{aligned}$$

where $\mathbf{H}$ remains the same, but $\mathbf{f}^T = 2(\mathbf{x}_a(0)^T \mathbf{S}^T \tilde{\mathbf{Q}} \mathbf{T} - \mathbf{x}_{\text{ref},N}^T \tilde{\mathbf{Q}} \mathbf{T} - \Delta\mathbf{u}_{\text{ref},N}^T \tilde{\mathbf{R}})$ which depends on $\mathbf{x}_a(0)$, $\mathbf{x}_{\text{ref},N}$, and $\Delta\mathbf{u}_{\text{ref},N}$. Where $\mathbf{x}_{\text{ref},N} \in \mathbb{R}^{nN}$, and $\Delta\mathbf{u}_{\text{ref},N} \in \mathbb{R}^{mN}$ are the reference vectors stacked.

V. Appendix VI - Additional Results

The main results of the complete framework are presented in section III. The following section presents some additional results and measurements that tie into subsection III-C, specifically related to Figure 15, providing more details and context.

A. Pilot Experiment

The main results of the complete framework are presented in section III. The following section presents some additional results and measurements that tie into the pilot experiment. During the pilot experiment, the human participant was provided with two different references, which are both depicted below. For additional clarity, the figures have also been separated into passive participation and active participation. Reference 1 is the same as depicted in section III. However, reference 2 is an additional set of data taken.

B. Reference 1 - Passive Participation

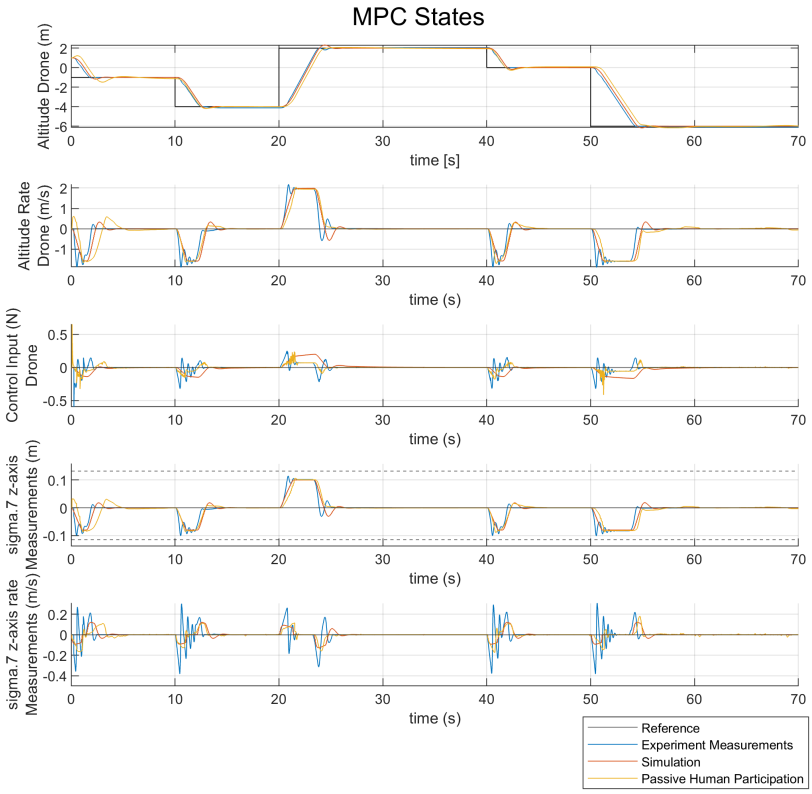

Fig. 37. A set of repeated step references provided as an altitude reference to the MPC (depicted by the black line) to create an overview of the effects of friction on the system. The blue line and red lines present the experimental and simulation results, respectively, for comparison. The yellow line indicated results by the human participant.

b

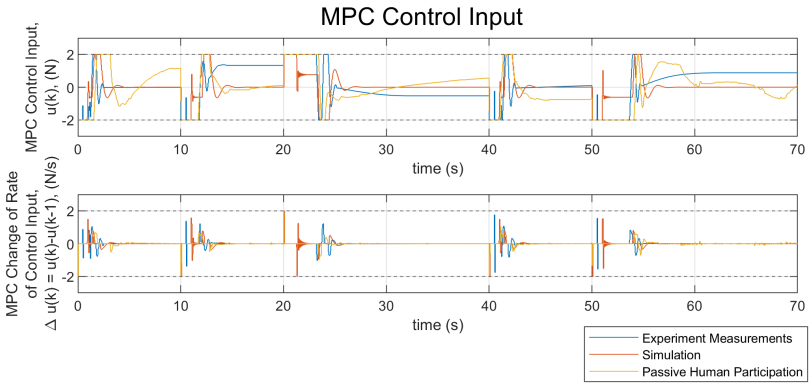

Fig. 38. A companion figure to Figure 37 depicting the control input determined by the MPC.

c

C. Reference 1 - Active Participation

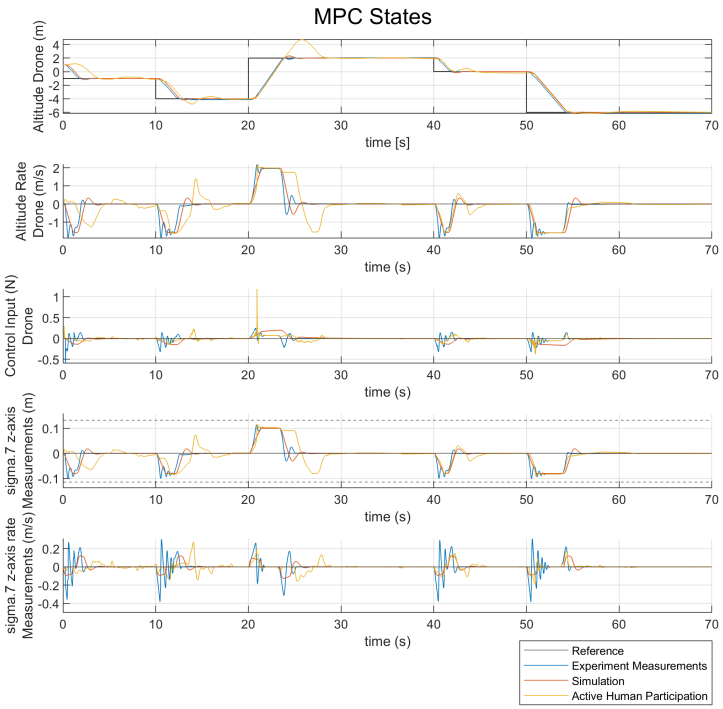

Fig. 39. A set of repeated step references provided as an altitude reference to the MPC (depicted by the black line) to create an overview of the effects of friction on the system. The blue line and red lines present the experimental and simulation results, respectively, for comparison. The yellow line indicated results by the human participant.

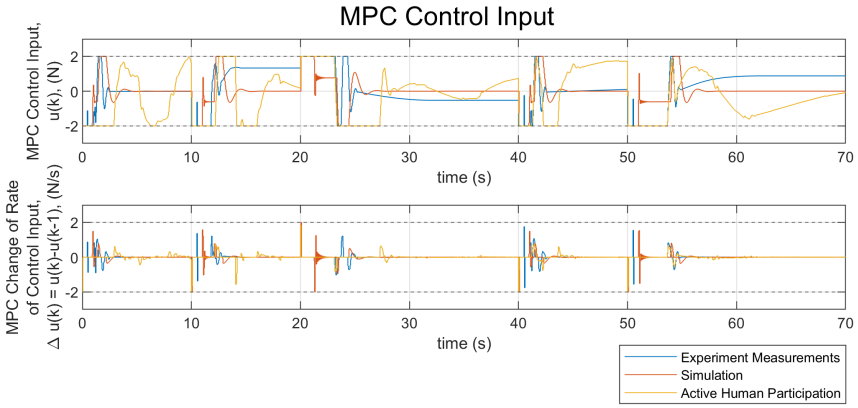

Fig. 40. A companion figure to Figure 39 depicting the control input determined by the MPC.

D. Reference 2 - Passive Participation

During the pilot experiment, the response of the participant to reference 2 was quite interesting. Between 10–20 seconds, they only noticed the forces exerted by the Sigma.7 when they started moving the end-effector up slightly. It was only by feeling a force differential that they were able to determine that there was indeed a force pushing in another direction. As may be observed in Figure 41, the forces command provided by the MPC in that time frame was the maximum of 2 N.

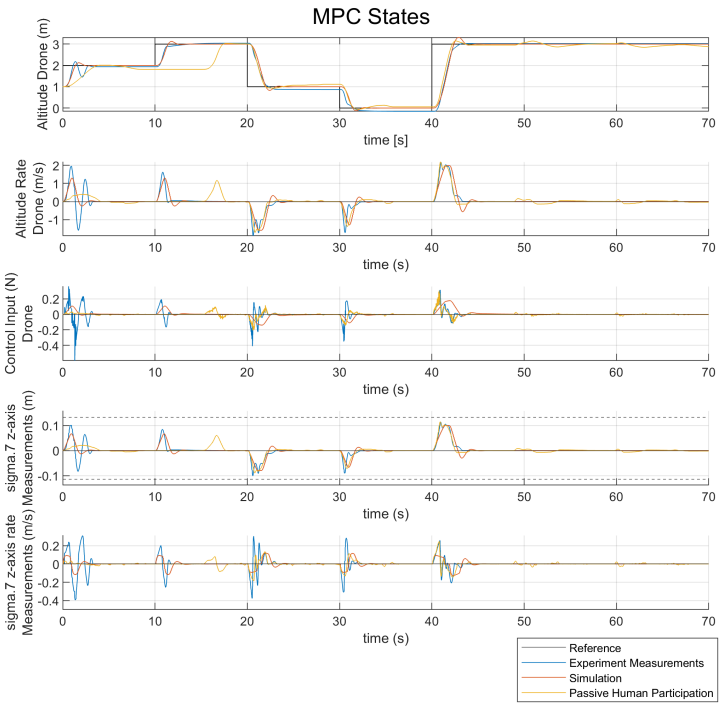

Fig. 41. A set of repeated step references provided as an altitude reference to the MPC (depicted by the black line) to create an overview of the effects of friction on the system. The blue line and red lines present the experimental and simulation results, respectively, for comparison. The yellow line indicated results by the human participant.

e

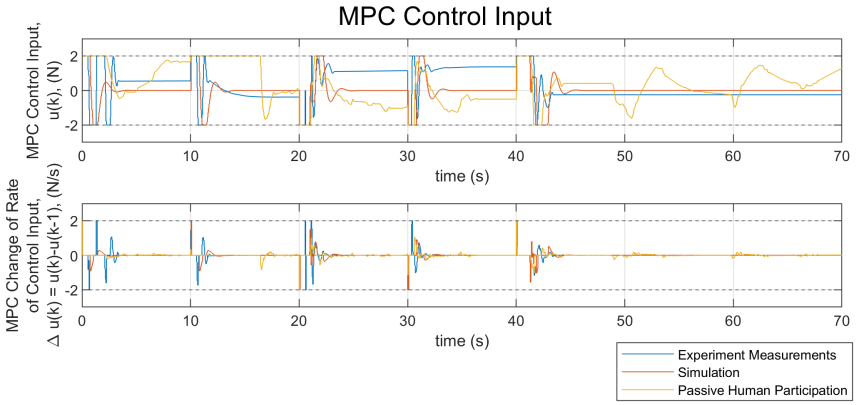

Fig. 42. A companion figure to Figure 41 depicting the control input determined by the MPC.

f

E. Reference 2 - Active Participation

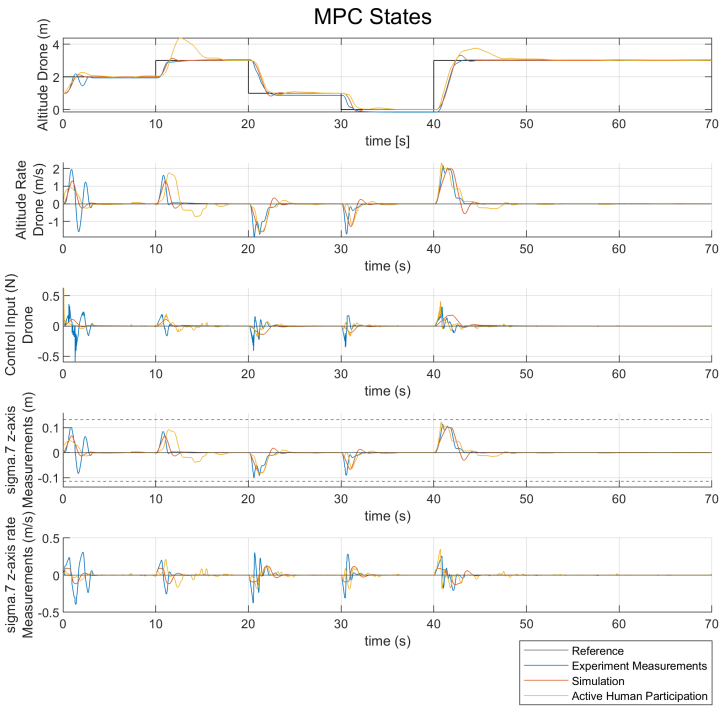

Fig. 43. A set of repeated step references provided as an altitude reference to the MPC (depicted by the black line) to create an overview of the effects of friction on the system. The blue line and red lines present the experimental and simulation results, respectively, for comparison. The yellow line indicated results by the human participant.

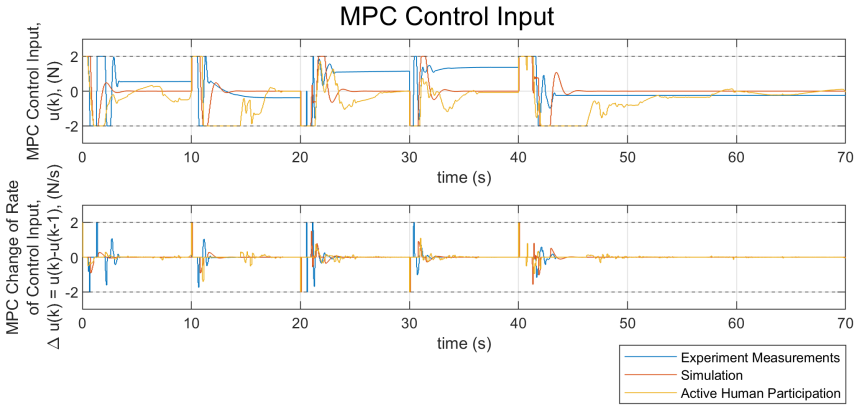

Fig. 44. A companion figure to Figure 43 depicting the control input determined by the MPC.

VI. Appendix VII - Haptic Rendering

Haptic rendering is the process that allows for the sense of touch to be reproduced when interacting with virtual objects. In 1995, Salisbury et al. defined haptic rendering as "the process of computing and generating forces in response to user interactions with virtual objects" [37]. There are a few studies that propose the addition of haptic rendering supports the user during motor task execution, and are cautiously hopeful that the additional haptic information itself may be beneficial to the motor learning process [38, 39, 40]. As a result, the topic was an additional point of interest during this study. However, due to the very nature of quadcopter dynamics being rather fast, the focus of the project quickly came to lie with the MPC. We made the choice to include the work and observations made on the addition of Haptic Rendering in the appendix.

In this section, we briefly explore the use of haptic rendering in combination with quadcopter dynamics to provide real-time feedback on the drone's states. The concept was to project the forces exerted on the quadcopter frame during flight, onto the haptic interface, such that the human operator may feel the influence of their action during task execution. The haptic feedback is delivered through the Force Dimension Sigma.7, a seven-degree-of-freedom haptic interface. This device is capable of rendering haptic information across all six degrees of freedom of a quadcopter, making it a suitable choice.

As mentioned in subsection II-B, the quadcopter flight mode is in *Stabilized Mode* which require a set of PID controllers governing the roll, pitch, yaw rate, and altitude rate axes, influencing the systems' dynamics (see Figure 8). As a result of the control infrastructure onboard the drone, the responsiveness of the quadcopter to human control input is increased. These controllers govern the forces exerted by the quadcopter motors such that the desired rotations and velocities are reached. As a result, it is actually the onboard PID controllers that become the governing source for haptic rendering.

The forces exerted on the quadcopter frame are, effectively, governed by Newton's law $F = ma$. Accelerations are often very noisy signals, both as measurements or modelled variables, and require additional filtering. As is, these accelerations cannot be provided as a signal to the haptic interface. However, other than this fundamental issue, there is also the very nature of the quadcopter system dynamics to consider — it is fast. By design, quadcopter respond and move rather fast in 3D space.

Consider the following input sequence, depicted in Figure 45, that we provided to the Sigma.7 end-effector. This is an input to the z-axis, axis 0, axis 1, and axis 2 in an attempt to simply fly the quadcopter through the virtual environment. The resulting accelerations, depicted in Figure 46, are very noisy signals. The linear vertical of the quadcopter offers a relatively slow signal overall. Provided the signal can be filtered properly without introducing significant delays, it could prove to be a useful axis for haptic rendering.

However, rotational accelerations produce a more erratic and short-lived signal, rarely lasting more than a fraction of a second. Many are in the order of 0.1 s long. which

makes it challenging for users to interpret useful feedback regarding the quadcopter's orientation. In fact, it is so brief that the aforementioned problem of the signal containing high-frequency content becomes the problem. The Sigma.7 is extremely sensitive in its rotational axis, and directly providing the unfiltered accelerations to the end-effector causes it to respond to its own force commands.

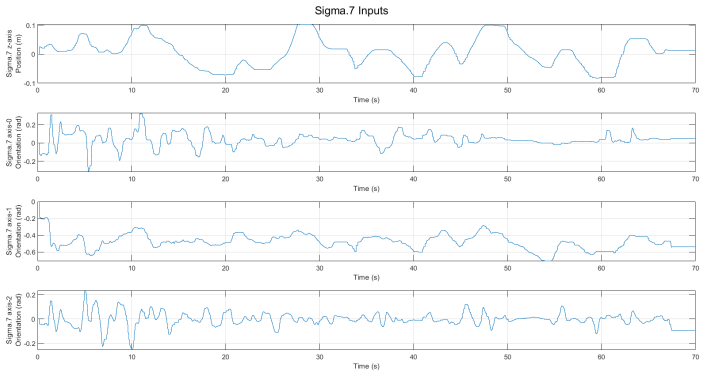

Fig. 45. This figure shows the measurements taken by the Sigma.7 as the author flew the quadcopter through the virtual environment. From top to bottom they correspond to references for the quadcopter as the desired altitude rate, roll angle, pitch angle, and yaw rate.

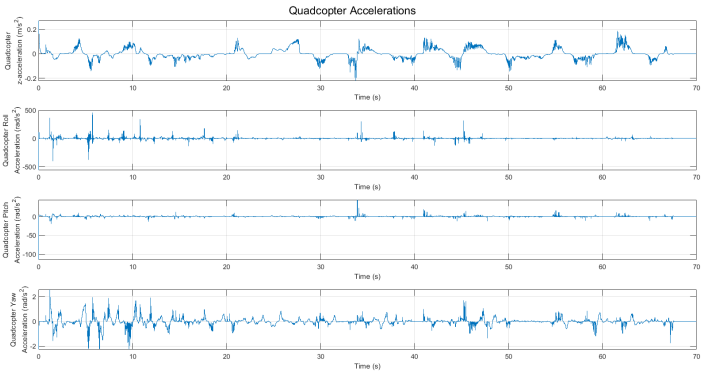

Fig. 46. This figure shows the resulting acceleration of the quadcopter from the input provided in Figure 45.

As a result, while the linear accelerations offer a potentially reliable basis useful for haptic rendering, the rotational accelerations pose challenges for consistent and actionable feedback.

As a last brief experiment, the linear accelerations of the z-axis were provided as force feedback, using the aforementioned $F = ma$ as a preliminary test. The results of which are depicted in Figure 47. In this figure, we can see that the influence of

my arm had minimal effect in stabilizing the noisy nature of the signal. It might be that the high-frequency content of the accelerations cannot be counteracted by the natural stiffness of my arm. But by providing this noisy signal as force-feedback, the Sigma.7 responds to its own force commands, exacerbating the effect. This makes for very uncomfortable force feedback.

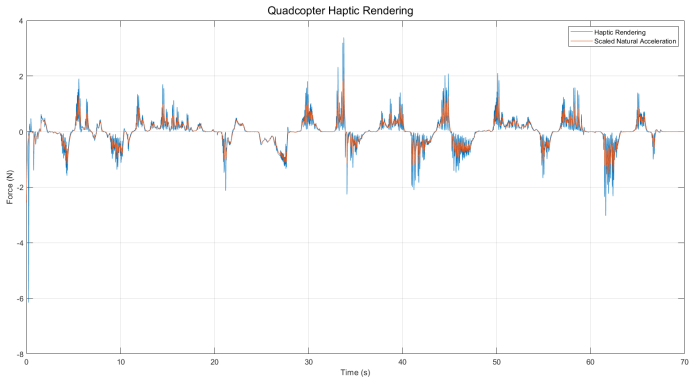

Fig. 47. This figure depicts the impact of the human arm on the haptic rendering should the z-acceleration be used unfiltered.

Further research into the topic was halted, however, some suggestions can be made. The linear accelerations may still be an interesting source for haptic rendering, however, additional filtering is required. As this method must be applicable real-time, as well as with minimal time delays involved, a potential suitable method could be the application of a Kalman Filter.