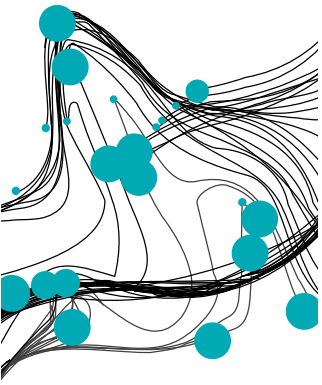
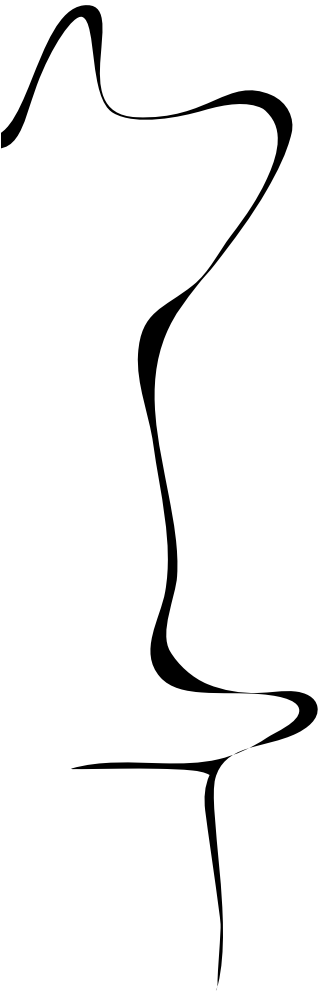


RAM ROBOTICS AND MECHATRONICS



CONTROL METHODS AND ARCHITECTURES FOR AERIAL MANIPULATORS IN PHYSICAL INTERACTION WITH HUMANS

M. (Mark) van Holland

MSC ASSIGNMENT

Committee:
A. Franchi, Ph.D HDR
ir. A.N.M.G. Afifi
dr. ir. J.J. de Jong

August 2021

055RaM2021
Robotics and Mechatronics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

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

Summary

The goal of this report is to investigate a possible methodology for human-aerial manipulator interaction. The control scheme presented in this report is inspired from different areas of aerial robotics literature, which will result in an innovative combined scheme. The result is a whole controller for an aerial manipulator, using optimization methods, which can behave in a compliant manner in response to external forces on its end-effector. The effectiveness of the approach is shown in a realistic simulation environment with human in the loop inputs. The robustness of the system is investigated with respect to measurement noise and parametric uncertainty.

Contents

1 Introduction

Nowadays unmanned aerial vehicles (UAVs) are used for a variety of applications, like contact free inspections tasks (Seo et al., 2018), photography, surveillance, and monitoring (de Oca et al., 2018). These applications all involve contact free situations. In recent years, more and more research is done in aerial manipulators which physically interact with the environment (Jimenez-Cano et al., 2015). In such situations the aerial manipulator needs to control the position of the contact point and the amount of force exerted by it.

This project is part of the Aerial-Core program. This is an EU funded initiative for the "development of core technology models and an integrated aerial cognitive robotic system that will have unprecedented capabilities on the operational range and safety in the interaction with people."(Ollero, 2019) At the University of Twente work is done in developing aerial manipulators that will cooperate with human workers in high-altitude work on power lines. For this purpose it is desirable to have a full pose controller which is predictable and safe for the human to be around.

The goal of this report is to design a controller to do cooperative tasks with a human worker. This controller will be tested on the FiberTHex. This is a fully actuated hexarotor, which is equipped with a three degree of freedom (DoF) arm. The whole system thus has nine DoF, which will make it redundant for postural tasks on the end-effector. In order to be safe for a human to be around the aerial manipulator, the behaviour of the system should be predictable for the human.

In order to achieve this goal a quadratic problem (QP) optimizer will be used to control the propeller speeds directly. A controller will be sending acceleration commands, for the FiberTHex platform and the joint angles, to the QP-optimizer. This will translate the postural task of the end-effector to the propeller speeds and joint velocities for the aerial manipulator. Because the system is redundant auxiliary tasks can be given to the system, which will be performed in the null-space. To make cooperation more natural for the human it is desirable for the system to be compliant when interacting with the human. In order to achieve this a admittance filter will be used.

Simulations of the system shows that the designed controller is able to take a pose for the end-effector and control the aerial manipulator to achieve this pose, while respecting system limits such as joint and propeller speed limits.

The rest of the report is structured as follows: chapter 2 will show related work that has been done in the field. Chapter 3 will give a model of the aerial manipulator. Chapter 4 explain the proposed controller, and the admittance filter. Chapter 5 will give the results of the simulations and the experiments on the physical drone, which are then interpreted in chapter 6. Chapter 7 will give the conclusions of this work, and lastly recommendations for future work are given in chapter 8

2 Related work

There is already research done on the topic of aerial manipulation. For example in (Ryll et al., 2019) they used a fully actuated aerial manipulator and equipped it with a fixed end effector. Because the platform was fully actuated, it was able to deliver forces and torques in all directions independently, making it a six DoF system. This meant the system was able to control not only the position of the end-effector, but also the position, while staying within the limits of the system. In (Ryll et al., 2019) they used a wrench observer to estimate the wrench on the system, assumed to be located at the end-effector. This wrench was used to implement a admittance filter in the control loop. (Ryll et al., 2019) Was able to create a 'flying end-effector' which was pose controlled. The end-effector was compliant, such that in a peg-in-hole task the system could correct for miss-alignment of the end effector.

In the work of (Nava et al., 2020) they used the same platform as in (Ryll et al., 2019), but instead of using a fixed end effector, they equipped the platform with a three DoF robotic arm. This way the system is redundant for full pose tasks at the end effector. The system controlled the propeller speeds, and the motor velocities at the the joints directly using a QP optimizer. The controller controlled the accelerations of the platform, the joint accelerations, and the forces applied by the end effector. The resulting signals were then converted by the QP optimizer to propeller speeds, and motor speeds. The wrench at the end effector was measured using a force torque sensor located at the end-effector. (Nava et al., 2020) Was able to control the force exerted by the end-effector. They were also able to use the arm to push of a surface to reach the waypoint faster.

In (Ryll et al., 2018) a fully actuated multi-rotor is modeled and controlled. The platform is equipped with a two DoF robotic arm and one with a four DoF robotic arm. The propeller speeds and the motor torques are obtained using model inversion. The controller on the end effector pose was in the task space and inverse kinematics was used to obtain the desired joint accelerations. For redundancy resolution they use the projected gradient method. This allows them to give auxiliary tasks to the system in the null-space of the task. The tasks given in the null space are to keep the platform horizontal, keep the joints as far from the joint limits as possible, and to avoid obstacles with the platform. (Ryll et al., 2018) showed in simulation that the system was able to keep the end-effector in position, or following a trajectory, while the platform moved around to perform the tasks given in the null-space.

In (Cataldi et al., 2016) an impedance control scheme is used to control a multi-rotor equipped with a six DoF manipulator. The controller they used was devided into three parts. Part one was the motion planner and the impedance control, the second part was the inverse kinematics, and the third part included the vehicle, and he manipulator controllers. The vehicle controller was split into three parts in order estimate the disturbance the manipulator would have on the platform and to counter this. In the inverse kinematics they used a weighted Jacobian in order to have the manipulator more compliant than the platform.

(Tzoumanikas et al., 2020) used an aerial manipulator to perform a writing task. The platform was a hexa-rotor equiped with a three Dof parrallel arm. An non-linear model predictive controller (NMPC) is used to calculate the body moments, collective thust, and the end-effector postition, from a desired trajectory and the current system state. The moments and thrust are converted to rotor commands via an allocation control, while the joint angles are calculated via inverse kinematics. Using this scheme Tzoumanikas et al. were able to write on a white board with a accuracy of 10 mm.

In (Ficuciello et al., 2015) a seven DoF fixed base serial manipulator was used to preform writing tasks in collaboration with a human. The redundancy was used to keep the robot's natural

behavior as close as possible to the desired impedance behavior. They also used a variable impedance to balance the accuracy of using high damping, and the speed of using low damping.

In this report different methods will be combined, to create one whole body controller. It will use a QP-optimizer as proposed in (Nava et al., 2020) to calculate the desired propeller speeds and joint positions, in order to achieve the desired joint accelerations. The QP-optimizer will do this while respecting the limit on the propeller speeds, and the joint limits. The QP-optimizer will also limit the rate of change of these variables, in order to more closely mimic real world behaviors.

Inverse kinematics will be used to translate the accelerations at the end-effector, given by the PD plus feed forward controller, to accelerations in the joints space of the aerial manipulator. To resolve the redundancy, the projected gradient method is used. This will allow to have some auxiliary tasks in the null-space, that will make the robot more predictable. The tasks in the the null-space will be to keep the platform horizontal, and to keep the joints away from their limits.

In order to have compliant behavior of the system, an admittance filter will be implement. This filter will be tuned to the desired tasks the drone is to perform. This will be done by shaping the apparent spring stiffness and damping. In this report for example the drone will have a region in which the human can interact with the system. In this region there will be no perceived spring and the system will behave as a mass-damper system. While if the aerial manipulator was to go outside this region there would be and apparent spring that will pull the system back into this region.

The main contribution of this paper will be to combine the methods of different control schemes, and make changes to them in order to suit the current framework, in order to get the desired behavior, with the goal of human-aerial manipulator interaction. To the best of the authors knowledge this is the first time a whole-body controller is created for human-aerial manipulator interaction.

3 Analysis

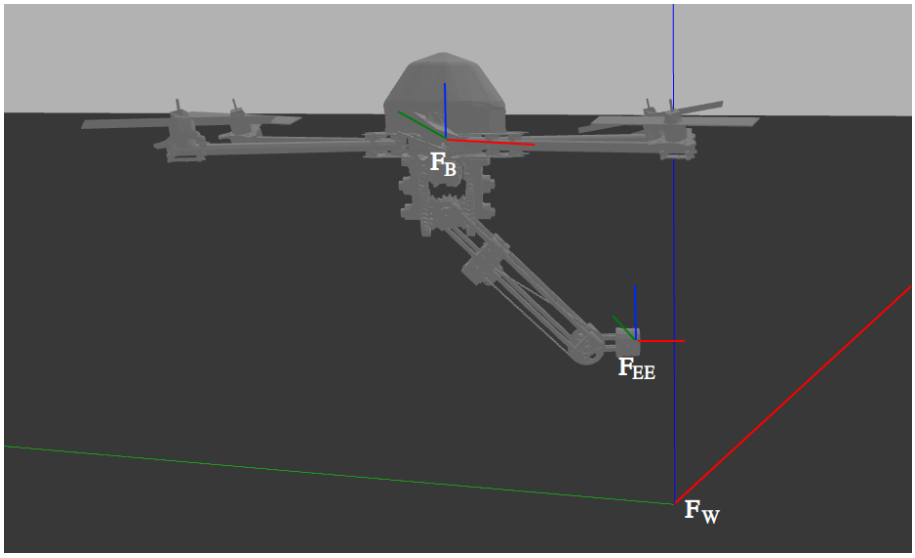


Figure 3.1: The FiberTHex aerial manipulator, with a graphical representation of the world frame, drone frame, and the end-effector frame.

We consider the aerial manipulator as shown in figure 3.1. This aerial manipulator consists of a fully actuated aerial platform (6 Dofs), which has a three DoF robotic arm mounted underneath. Assuming the arm has $n+1$ links, connected by n revolute joints, the aerial manipulator will have $6+n$ DoFs.

3.1 Notation

In order to describe the aerial manipulator different frames are defined, as depicted in figure 3.1:

- The world frame $\mathcal{F}_W$ which has unit axis $\{\boldsymbol{x_W}, \boldsymbol{y_W}, \boldsymbol{z_W}\}$ and origin $\boldsymbol{O_W}$. This frame is an inertial frame placed arbitrarily, and where axis $\boldsymbol{z_W}$ points in the opposite direction of gravity.
- The body frame $\mathcal{F}_B$ which has unit axis $\{\boldsymbol{x_B}, \boldsymbol{y_B}, \boldsymbol{z_B}\}$ and origin $\boldsymbol{O_B}$. The origin of this frame is rigidly attached to the center of mass (COM) of the platform, with $\boldsymbol{z_B}$ pointing upwards.
- The end-effector frame $\mathcal{F}_{EE}$ which has unit axis $\{\boldsymbol{x_{EE}}, \boldsymbol{y_{EE}}, \boldsymbol{z_{EE}}\}$ and origin $\boldsymbol{O_{EE}}$. The origin of this is rigidly attached to the end-effector, where $\boldsymbol{x_{EE}}$ is the approach direction.

The orientation of $\mathcal{F}_B$ with respect to $\mathcal{F}_W$ can be described by the rotation matrix $\boldsymbol{R_B^W} \in SO(3)$. The translation between these frames is given by $\boldsymbol{p_B^W} \in \mathbb{R}^3$

The state of the robot can be described by the state $\boldsymbol{x} = (\boldsymbol{p_B}, \boldsymbol{R_B}, \boldsymbol{q_A})$ where $\boldsymbol{p_B} \in \mathbb{R}^3$ is the position of $\boldsymbol{O_B}$ expressed in the world frame, $\boldsymbol{R_B} \in SO(3)$ is the orientation of frame $\mathcal{F}_B$ with respect to $\mathcal{F}_W$, and $\boldsymbol{q_A} \in \mathbb{R}^n$ is a vector of the joint angles characterizing the configuration robotic arm. The velocity of the system is given by the vector $\boldsymbol{v} = [\boldsymbol{v_B}^\top \boldsymbol{\omega_B}^\top \boldsymbol{v_A}^\top]^\top \in \mathbb{R}^{(6+n)}$, where $\boldsymbol{v_B} \in \mathbb{R}^3$ is the linear velocity of the aerial platform with respect to the world frame expressed in the world frame, $\boldsymbol{\omega_B} \in \mathbb{R}^3$ is the angular velocity of frame $\mathcal{F}_B$ with respect to $\mathcal{F}_W$ expressed in frame $\mathcal{F}_B$, and $\boldsymbol{v_A} \in \mathbb{R}^n$ are the joint velocities of the arm.

3.2 Dynamic model

The dynamic equations of the system can be computed using the Euler-Lagrange equations (Stramigioli, 2019):

$$\boldsymbol{M}(\boldsymbol{x})\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{x},\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{w_B} \\ \boldsymbol{\tau_A} \end{bmatrix} + \boldsymbol{w_e} \tag{3.1}$$

Where $\boldsymbol{M}(\boldsymbol{x}) \in \mathbb{R}^{(6+n)\times(6+n)}$ is the inertia matrix, $\boldsymbol{C}(\boldsymbol{x},\boldsymbol{\nu}) \in \mathbb{R}^{(6+n)\times(6+n)}$ is the matrix that accounts for the Coriolis and centrifugal effects, $\boldsymbol{g}(\boldsymbol{x}) \in \mathbb{R}^{(6+n)}$ is the gravity vector, $\boldsymbol{w_B} \in \mathbb{R}^6$ is the control wrench applied to the platform by the propellers, $\boldsymbol{\tau_A} \in \mathbb{R}^n$ is the vector of joint torques of the robotic arm, and $\boldsymbol{w_e}$ is the external wrench. It is assumed that aerodynamic effects such as propeller drag, and ground or wall effects can be neglected.

The control wrench on the platform is applied generated by the propellers. depending on the speed of the propellers the thrust they create changes. The standard quadratic relation between the propeller rates and the generated thrust is considered here: (Nava et al., 2020) (Bicego et al., 2020)

$$\boldsymbol{w_B} = \boldsymbol{G_w}(\boldsymbol{q})(\boldsymbol{\omega_p} \odot \boldsymbol{\omega_p}) \tag{3.2}$$

where $\boldsymbol{\omega_p} \in \mathbb{R}^P$ is a vector of the propellers spinning speeds, $\boldsymbol{G_w} \in \mathbb{R}^{(6\times P)}$ is the allocation matrix, which is a mapping between the propellers square spinning speeds and the generated wrench. The allocation matrix can be separated into two sub-matrices:

$$\boldsymbol{G_w} = \begin{bmatrix} \boldsymbol{G_{w1}} \\ \boldsymbol{G_{w2}} \end{bmatrix}$$

where $\boldsymbol{G_{w1}} \in \mathbb{R}^{(3\times P)}$ maps the square of the propeller spinning speeds to the force generated, and $\boldsymbol{G_{w2}} \in \mathbb{R}^{(3\times P)}$ maps to the moment generated. The rank of these matrices determine the controllability of the platform. An hexa-rotor with all thrust vectors pointing in the same direction, unidirectional-thrust, will have $\text{rank}(\boldsymbol{G_w}) = 4$, with $\text{rank}(\boldsymbol{G_{w1}}) = 1$ and $\text{rank}(\boldsymbol{G_{w2}}) = 3$. This means that that particular platform can independently apply a torque in all directions, but can only apply a force in one direction, resulting in an under-actuated vehicle. The platform used here is a fully-actuated hexa-rotor, meaning $\text{rank}(\boldsymbol{G_w}) = 6$

The platform has full rank for the G_w matrix because of the way the propellers are placed. The propellers are mounted on the platform in such a way that the z-axis of the propeller is rotated with respect to the the body, resulting into thrust directions are non-unidirectional. A detailed explanation is given in (Ryll et al., 2019).

We assume all interactions with the environment will be via the end-effector of the aerial manipulator. The interaction wrench can be measured using and sensor placed at the end-effector. The wrench at the end-effector can be mapped to a wrench on the body using the Jacobian:

$$\boldsymbol{w_e} = \boldsymbol{J_{EE}}(\boldsymbol{x})^\top \boldsymbol{w_{EE}} \tag{3.3}$$

where $\boldsymbol{J_{EE}}(\boldsymbol{x}) \in \mathbb{R}^{(6+n)\times(6+n)}$ is the Jacobian which maps the velocities $\boldsymbol{\nu}$ to velocities at the end-effector, and $\boldsymbol{w_{EE}} \in \mathbb{R}^6$ is the external wrench applied at the end-effector.

Rewriting equation 3.1 to include equations 3.2 and 3.3 gives:

$$\boldsymbol{M}(\boldsymbol{x})\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{x},\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{G}\boldsymbol{u} + \boldsymbol{J_{EE}}(\boldsymbol{x})^\top \boldsymbol{w_{EE}} \tag{3.4}$$

where $\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G_w} & 0_{(6\times n)} \\ 0_{(n\times P)} & I_{(n\times n)} \end{bmatrix}$ is the generalized allocation matrix, and $u = \begin{bmatrix} (\omega_p \odot \omega_p) \\ \tau_A \end{bmatrix}$ is the system input

4 Design

4.1 Controller

The task given to the system will be a pose for the end-effector, consisting of the position and orientation of the end-effector frame in the world frame. This task will have to be translated to the joint space in order to control the aerial manipulator. This can be done with the use of inverse kinematics. The joint space task will than be translated into system inputs. A general outline of the system is given in figure 4.1. The system can be divided into three parts: The first part includes the trajectory generator and the controller. The second part is the inverse kinematics together with the redundancy resolution. The Third part is the quadratic programming (QP) optimizer.

4.1.1 Quadratic programming optimizer

A QP-optimizer is used in order to obtain the set of system inputs that best achieve the desired tasks. Let the set of desired tasks, desired system outputs, be denoted as $\boldsymbol{y} = [y_1 \;\; y_2 \;\; ... \;\; y_m]^\top \in \mathbb{R}^m$, where $m \in \mathbb{N}$ is the number of tasks. The second order derivative of these tasks are given by $\boldsymbol{a} = \left[\frac{d^2 y_1}{dt^2} \;\; \frac{d^2 y_2}{dt^2} \;\; ... \;\; \frac{d^2 y_m}{dt^2}\right]$. The controller assumes that this is directly controllable by the virtual input $\boldsymbol{a}^\star$:

$$\boldsymbol{a} = \boldsymbol{a}^\star \tag{4.1}$$

Using this assumption, the controller will be giving an $\boldsymbol{a}^\star$ that will steer the tasks $\boldsymbol{y}$ along a desired trajectory. In order for the controller to use the virtual inputs to steer the system, the QP-optimizer should ensure equation 4.1 holds true as much as possible. This will be done by solving the optimization problem:

$$\underset{\boldsymbol{u}}{\text{minimize}} \quad (\boldsymbol{a} - \boldsymbol{a}^\star)^\top \boldsymbol{W}_{\boldsymbol{a}} (\boldsymbol{a} - \boldsymbol{a}^\star) \tag{4.2}$$

where $\boldsymbol{W}_{\boldsymbol{a}} \in \mathbb{R}^{(m \times m)}$ is a positive definite diagonal matrix. This matrix holds the weights for the optimizer, which can be used to define priorities among the tasks. One of the advantages of this system is the possibility of adding or removing tasks to the system. Another advantage is the possibility of adding hard constraints to the optimizer. Hard constraints are constraints that can not be broken by the optimizer, in contrast to the soft constraints that are given by the controller and inverse kinematics.

The tasks for the aerial manipulator will exist of the desired joint space configuration, aerial platform position $\boldsymbol{p_B}$, orientation $\boldsymbol{R_B}$, and arm joint angles $\boldsymbol{q_a}$, a regularisation task for the

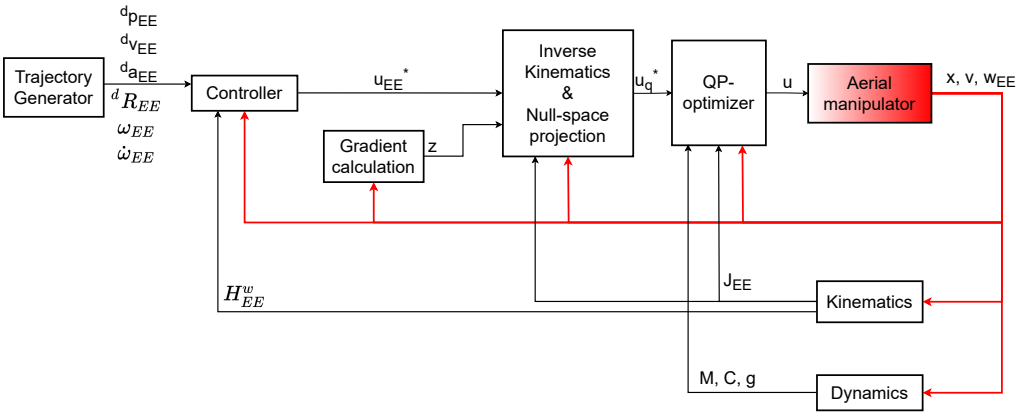


Figure 4.1: Overview of the control system.

propellers $\boldsymbol{r_p}$, and a task regulating the wrench at the end-effector $\boldsymbol{w_{EE}}$. The desired system output is thus given by $\boldsymbol{y} = \begin{bmatrix} \boldsymbol{p_B}^\top & \boldsymbol{R_B}^\top & \boldsymbol{q_a}^\top & \boldsymbol{r_p}^\top & \boldsymbol{w_{EE}}^\top \end{bmatrix}^\top$, and the corresponding time derivatives are given by $\boldsymbol{a} = \begin{bmatrix} \dot{\boldsymbol{v}}_{\boldsymbol{B}}^\top & \dot{\boldsymbol{\omega}}_{\boldsymbol{b}}^\top & \dot{\boldsymbol{v}}_{\boldsymbol{a}}^\top & \boldsymbol{r_p}^\top & \boldsymbol{w_{EE}}^\top \end{bmatrix}^\top$. The first three tasks can be used to obtain the desired pose at the end effector, and complete desired motions. The wrench task will be used to compensate the external wrench.

The propeller regulation task $\boldsymbol{r_p}$ tries to ensure that all the propellers deliver the same amount of force. The goal of this tasks is to balance the forces delivered by the propeller, which will avoid solutions which are unfavorable from an energetic viewpoint. In other words with this task the optimizer will try to avoid solutions where some propellers speeds are close to the lower limit, while other propellers are almost saturated. The task is defined as follow:

$$\boldsymbol{r_p} = \boldsymbol{D}_{\boldsymbol{r_p}}(\omega_p \odot \omega_p) \tag{4.3}$$

where $\boldsymbol{D}_{\boldsymbol{r_a}} \in \mathbb{R}^{(p-1\times p)}$ is a matrix with 1 on the diagonal, -1 elements right above the diagonal, end zeros elsewhere.

The tasks for the propeller regulation, and the aerial manipulator configuration can be expressed in terms of system inputs in the absence of the external wrench. Extending the system input with the external wrench $\boldsymbol{u'} = \begin{bmatrix} \boldsymbol{u}^\top \boldsymbol{w_{EE}}^\top \end{bmatrix}^\top$ allows for all tasks to be expressed in terms of the outputs of the QP-optimizer. Using this extended input, and equations 3.4, and 4.3, $\boldsymbol{a}$ is given by:

$$\boldsymbol{a} = \boldsymbol{H}(\boldsymbol{x})\boldsymbol{u'} + \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{v}) \tag{4.4}$$

Where $\boldsymbol{H}(\boldsymbol{q}) \in \mathbb{R}^{(m\times(P+n+6))}$ is the called the hessian of the QP-optimizer, $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{v}) \in \mathbb{R}^m$ is the bias vector of the optimizer. $\boldsymbol{H}(\boldsymbol{q})$ and $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{v})$ are given by:

$$\boldsymbol{H}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{M}(\boldsymbol{x})^{-1}\boldsymbol{G} & \boldsymbol{M}(\boldsymbol{x})^{-1}\boldsymbol{J_{EE}}(\boldsymbol{x})^\top \\ \boldsymbol{D}_{\boldsymbol{r_p}} & 0_{(p-1\times 6)} \\ 0_{(6\times(P+n))} & 1_{(6\times 6)} \end{bmatrix} \tag{4.5}$$

where $\boldsymbol{D}_{\boldsymbol{r_a}}$ is extended with zeros.

$$\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{v}) = \begin{bmatrix} \boldsymbol{M}(\boldsymbol{x})^{-1}(-\boldsymbol{C}(\boldsymbol{x}, \boldsymbol{v})\boldsymbol{v} - \boldsymbol{g}(\boldsymbol{x}) \\ 0 \\ 0 \end{bmatrix} \tag{4.6}$$

Constraints are added to the optimization problem of 4.2, in order to obtain the desired behaviour. One of the constraints that is added to the QP-optimizer is the dynamic equation. This will ensure that the solution to the minimization problem is dynamically feasible. Other constraints will be the system input limits, these are the propeller spinning speed limits, and the joint torque limits of the robotic arm. The outputs of the QP-optimizer are propeller speeds and joint torques, this output can theoretically jump from one limit to the other in one optimization step. This is something that is not feasible to do for the real platform, the motors don't have limits of how fast they can accelerate/change the delivered torque. Because of this it is desirable to also limit the rate of change of the output given by the optimizer. This can be done by giving an additional constraint to the optimizer which limit how much the output can change from one optimization step to the next.

In the presence of the given constraints it might be that a solution that guarantees equations 4.1 does not exist.In this case the optimizer will prioritize higher priority tasks, while relaxing lower priority tasks. The priority of the tasks can be tuned by tuning the $\boldsymbol{W_a}$ matrix. The weights of the optimizer will dictate how the aerial manipulator behaves.

Including the constraints in the optimization problem, the problem can be formulated as:

$$\underset{\boldsymbol{u'}}{\text{minimize}} \quad (\boldsymbol{a} - \boldsymbol{a}^\star)^\top \boldsymbol{W_a}(\boldsymbol{a} - \boldsymbol{a}^\star) \tag{4.7a}$$

$$\text{subject to:} \qquad \boldsymbol{M}(\boldsymbol{x})\dot{\boldsymbol{v}} + \boldsymbol{C}(\boldsymbol{x},\boldsymbol{v})\boldsymbol{v} + \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{G}\boldsymbol{u} + \boldsymbol{J_{EE}}(\boldsymbol{x})^\top \boldsymbol{w_{EE}} \tag{4.7b}$$

$$\boldsymbol{u_l} \leq \boldsymbol{u} \leq \boldsymbol{u_u} \tag{4.7c}$$

$$\boldsymbol{\lambda_l} \leq \dot{\boldsymbol{u}} \leq \boldsymbol{\lambda_u} \tag{4.7d}$$

where $\lambda_l \in \mathbb{R}^{(P+n+6)}$ is the limit on the rate of change of the system input. This is a constant in this paper, but it can be a function of the current system input. In (Bicego et al., 2020) it is explained why it is favourable to do make the propeller speed rate of change depended on the spinning speed of the propellers, and how to determine the function experimentally.

4.1.2 Inverse kinematics and redundancy resolution

The platform configuration will be given to the QP-optimizer by the inverse kinematics. (see figure 4.1) Because the system is redundant for a pose task at the end effector, there are an infinite amount of possible solutions to achieve the pose task. In order to make the system more predictable auxiliary tasks are given to the system which it should complete, as best as possible, alongside the main task. These tasks are completed in the null-space. The null space is the space in which the aerial manipulator can change configurations, while not influencing the main task, for example the aerial manipulator can keep the end-effector in the same pose while moving the platform around by exploiting the redundancy of the system.

The differential kinematics of the aerial manipulator can be written as: (De Luca, 2020)

$$\boldsymbol{u_{EE}}^\star = \boldsymbol{J_{EE}}(\boldsymbol{x})\boldsymbol{u_q}^\star + \dot{\boldsymbol{J}}_{\boldsymbol{EE}}(\boldsymbol{v},\boldsymbol{x})\boldsymbol{v} \tag{4.8}$$

where $\boldsymbol{u_{EE}} \in \mathbb{R}^6$ is the vector of linear and angular accelerations of the end-effector given by the controller, and $\boldsymbol{u_q} \in \mathbb{R}^{(6+n)}$ are the joint accelerations needed, given the current joint velocities $\boldsymbol{v}$, in order to achieve the accelerations $\boldsymbol{u_{EE}}$. In order to obtain the desired joint accelerations the equation can rewritten in the form:

$$\boldsymbol{u_q}^\star = \boldsymbol{J_{EE}}^\dagger(\boldsymbol{u_{EE}}^\star - \dot{\boldsymbol{J}}_{\boldsymbol{EE}}(\boldsymbol{v},\boldsymbol{x})\boldsymbol{v}) + \left(I - \boldsymbol{J_{EE}}^\dagger \boldsymbol{J_{EE}}\right) z \tag{4.9}$$

where $\boldsymbol{J_{EE}}^\dagger$ represents the pseudo inverse of the Jacobian $\boldsymbol{J_{EE}}$, and $z \in \mathbb{R}^{(6+n)}$ is an additional virtual representing the auxiliary tasks. These auxiliary tasks will be projected into the null space by the projection matrix $\left(I - \boldsymbol{J_{EE}}^\dagger \boldsymbol{J_{EE}}\right)$.

In this report the weighted pseudo inverse will be used:

$$\boldsymbol{J_{EE}}^\dagger = \boldsymbol{W}^{-1}\boldsymbol{J}^\top\left(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^\top\right) \tag{4.10}$$

where $\boldsymbol{W}$ is the weight matrix. The weight is symmetrical and in this case also diagonal. Giving a larger weight $\boldsymbol{W}_i$ corresponding to a state variable $\boldsymbol{x}_i$ means that the inverse kinematic solution is less likely to use this state. This behavior will be used to have the platform of the aerial manipulator be more likely to move than the arm. This makes the system more predictable for the human cooperator. A larger weight is also given to the orientation of the platform, in order to make it so that the platform does not want to pitch or roll.

For the redundancy resolution, the projected gradient method will be used. This method constructs a gradient function of desired tasks for the aerial manipulator and projects these into the null space of the main tasks. The virtual input $\boldsymbol{z}$ given in equation 4.9 is the gradient function that will be projected into the null space. $\boldsymbol{z}$ will be determined by the following tasks.

1. Keep the platform horizontal: When in rest ($\boldsymbol{v} = \boldsymbol{0}$), the most energy efficient configuration for the platform is when the pitch and the roll are both zero.($\theta = 0$, and $\phi = 0$) This is

because of the way the propellers are orientated, if the platform is horizontal all the propellers will have the same spinning speeds. The cost function corresponding with this task will be given by $\boldsymbol{H_1}$

2. Minimize the joint angles of the robotic arm: Minimizing the joint angles helps with the maximizing the range of possible movements of the arm and also keeps the joint angles away form the joint limits as much as possible. The cost function corresponding with this task will be given by $\boldsymbol{H_2}$

The gradient with these tasks can be written as:

$$\boldsymbol{z} = -\nabla_x \boldsymbol{H_1} - \nabla_x \boldsymbol{H_2} - \boldsymbol{K_d}\,\boldsymbol{v} \tag{4.11}$$

where $\boldsymbol{K_d} \in \mathbb{R}^{(6+n)\times(6+n)}$ is a positive definite diagonal damping matrix. This last term is needed to stabilize the self-motions in the null-space.

$$\boldsymbol{H_1} = \frac{1}{2N} k_1 \sum_{i=4}^{5} \left(\frac{x_i - \bar{x}_i}{x_{u,i} - x_{l,i}} \right)^2 \tag{4.12}$$

with $\bar{x}_i = \frac{x_{u,i}+x_{l,i}}{2} \in \mathbb{R}$ the middle point of the joint range, $x_{u,i} = -x_{l,i} \in \mathbb{R}$ being the upper and lower limit respectively, $k_1 \in \mathbb{R}$ is a gain factor, and $N \in \mathbb{N}$ is the total number of degrees of freedom of the aerial manipulator.

$$\boldsymbol{H_2} = \frac{1}{2N} k_2 \sum_{i=7}^{N-1} \left(\frac{x_i - \bar{x}_i}{x_{u,i} - x_{l,i}} \right)^2 \tag{4.13}$$

where k_2 is a gain factor. The gain factors can be used to tune how the system reacts to these tasks.

4.1.3 Controller design

The desired end-effector accelerations $\boldsymbol{u_{EE}}$ are calculated by the controller. The controller will steer the system in such a way that the pose at the end effector follows the desired trajectory given to the controller, by the trajectory generator. the controller can be described as follows:

$$\boldsymbol{u_{EE}} = \begin{bmatrix} \boldsymbol{u_{EE_1}} \\ \boldsymbol{u_{EE_2}} \end{bmatrix} \tag{4.14}$$

with:

$$\boldsymbol{u_{EE_1}} = {}^{\boldsymbol{d}}\boldsymbol{a_{EE}} + \boldsymbol{K_{p_2}} \left({}^{\boldsymbol{d}}\boldsymbol{v_{EE}} - \boldsymbol{v_{EE}} \right) + \boldsymbol{K_{p_1}} \left({}^{\boldsymbol{d}}\boldsymbol{p_{EE}} - \boldsymbol{p_{EE}} \right) \tag{4.15}$$

where ${}^{\boldsymbol{d}}\boldsymbol{a_{EE}} \in \mathbb{R}^3$, ${}^{\boldsymbol{d}}\boldsymbol{v_{EE}} \in \mathbb{R}^3$, and ${}^{\boldsymbol{d}}\boldsymbol{p_{EE}} \in \mathbb{R}^3$ are the desired linear acceleration, velocity and position of the end effector respectively, and $\boldsymbol{K_{p_1}} \in \mathbb{R}^{(3\times3)}$, and $\boldsymbol{K_{p_2}} \in \mathbb{R}^{(3\times3)}$ are the controller gains.

$$\boldsymbol{u_{EE_2}} = {}^{\boldsymbol{d}}\dot{\boldsymbol{\omega}} + \boldsymbol{K_{\omega_2}} \left({}^{\boldsymbol{d}}\boldsymbol{\omega_{EE}} - \boldsymbol{\omega_{EE}} \right) + \boldsymbol{K_{\omega_1}}\boldsymbol{e_R} \tag{4.16}$$

with the orientation error $\boldsymbol{e_R}$ defined as:

$$\boldsymbol{e_R} = \frac{1}{2} \left[R_{EE}^{\top}\, {}^{d}R_{EE} - {}^{d}R_{EE}^{\top} R_{EE} \right]_{\vee} \tag{4.17}$$

where ${}^{\boldsymbol{d}}\dot{\boldsymbol{\omega}}_{\boldsymbol{EE}} \in \mathbb{R}^3$, ${}^{\boldsymbol{d}}\boldsymbol{\omega_{EE}} \in \mathbb{R}^3$, and ${}^{\boldsymbol{d}}\boldsymbol{R_{EE}} \in \mathbb{R}^{(3\times3)}$ are the desired angular acceleration, velocity and orientation of the end effector respectively, $\boldsymbol{K_{\omega_1}} \in \mathbb{R}^{(3\times3)}$, and $\boldsymbol{K_{\omega_2}} \in \mathbb{R}^{(3\times3)}$ are the controller gains, and $[]_{\vee}$ represents the map from so(3) to $\mathbb{R}^3$.

The trajectory that the end-effector has to follow is given by the trajectory generator. The trajectory generator takes waypoint that the end effector will have to make, and designs a smooth trajectory between these points. The waypoints are specified by a position and an rotation. The rotation is described in the angle axis notation, meaning an axis is specified around which the rotation should take place and an angle for how much it rotates around this axis. These waypoints are translated into a minimal jerk trajectory. The system outputs the linear and angular accelerations and velocities, as well as the position and orientation at each time step.

4.2 Admittance filter

In addition to the controller an admittance filter is implemented into the system. This filter will be used to make the system compliant to interaction with the human cooperator. The filter will only be used for the translational domain, meaning it will not effect the orientation of the system. This is shows that the filter can be applied to the translational and rotational domain independently, which is useful when it is important when either the position or the orientation needs to be unchanged. It is also possible to have the admittance filter act in both the translation and rotational domains. This filter will be placed in between the trajectory generator and the controller as shown in figure 4.2

Given the desired trajectory of the trajectory generator in terms of position, velocity, and acceleration (${}^{d}\boldsymbol{p}_{EE}$, ${}^{d}\boldsymbol{v}_{EE}$, and ${}^{d}\boldsymbol{a}_{EE}$), the corresponding set of variables (${}^{r}\boldsymbol{p}_{EE}$, ${}^{r}\boldsymbol{v}_{EE}$, and ${}^{r}\boldsymbol{a}_{EE}$) can be generated by the admittance filter. The resulting trajectory is called the reference trajectory. This set of variables is then fed to the controller (see figure 4.2). The admittance filter is characterized by the following dynamics:

$$\boldsymbol{M_a}\left({}^{d}\boldsymbol{a}_{EE} - {}^{r}\boldsymbol{a}_{EE}\right) + \boldsymbol{D_a}\left({}^{d}\boldsymbol{v}_{EE} - {}^{r}\boldsymbol{v}_{EE}\right) + \boldsymbol{K_a}\left({}^{d}\boldsymbol{p}_{EE} - {}^{r}\boldsymbol{p}_{EE}\right) = -\boldsymbol{F}_{EE} \tag{4.18}$$

where $\boldsymbol{M_a} \in \mathbb{R}^{(6\times6)}$ is the apparent mass, $\boldsymbol{D_a} \in \mathbb{R}^{(6\times6)}$ is the apparent damping, and $\boldsymbol{K_a} \in \mathbb{R}^{(6\times6)}$ is the apparent stiffness. These Matrices are all positive definite and can be tuned in order to get different behaviors of the system. This will make the system behave like a mechanical mass spring damper system, where the internal force is equal but opposite to the interaction force at the end-effector. The output of the filter is given by:

$${}^{r}\boldsymbol{a}_{EE} = {}^{d}\boldsymbol{a}_{EE} + \boldsymbol{M_a}^{-1}\boldsymbol{D_a}\left({}^{d}\boldsymbol{v}_{EE} - {}^{r}\boldsymbol{v}_{EE}\right) + \boldsymbol{M_a}^{-1}\boldsymbol{K_a}\left({}^{d}\boldsymbol{p}_{EE} - {}^{r}\boldsymbol{p}_{EE}\right) + \boldsymbol{M_a}^{-1}\boldsymbol{F}_{EE} \tag{4.19}$$

The reference acceleration is integrated to get the reference velocity, and again to get the reference position.

It is not necessary for the apparent matrices to be linear, although for interaction with people it might be beneficial for the mass to stay constant throughout the interaction. In this paper an interaction space has been defined, where the filter will be active. In this space $\boldsymbol{K_a}$ will be zero, and the system will behave like a mass damper system. This will allow for the human cooperator to freely move the robot around in the space. When the robot moves out of this space, the stiffness and the damping of the system will increase in order to push the robot back into the interaction space. Figure 5.31 shows the interaction space in the simulation environment, the space is marked by the transparent green area.

4.3 System overview

An overview of the complete system, admittance filter included, is shown in figure 4.2. The system consists of various sub systems which all have to work together to get the desired behavior at the output. The behavior of the system can be tuned in different ways depending on the desired behavior. A list of parameters that can be tuned (going from input to output) is:

- Trajectory generator:

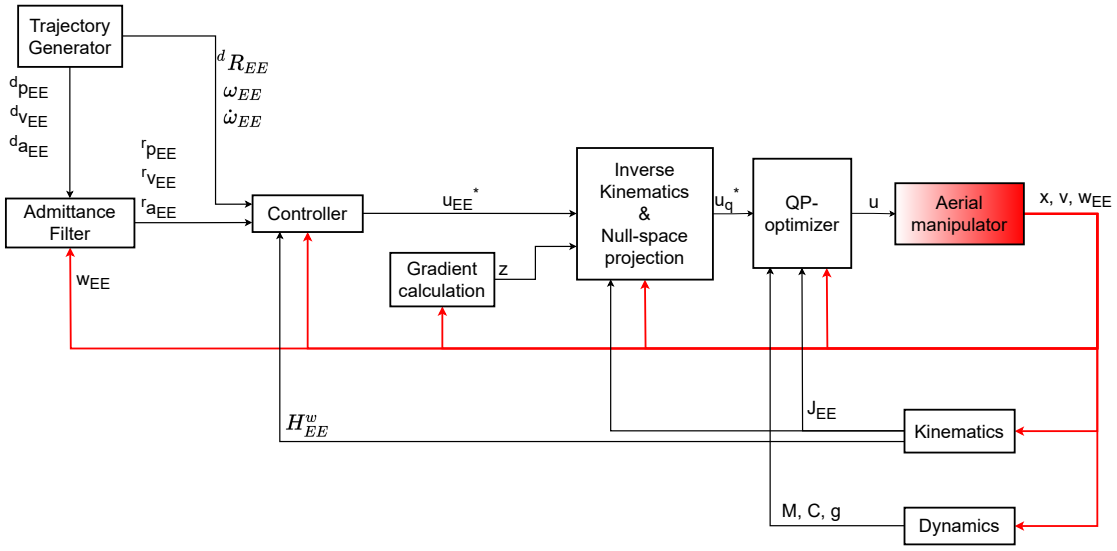


Figure 4.2: An overview of the total system.

- $^{d}v_{EE_{max}}$: This is the maximum linear velocity of the trajectory.
- $^{d}a_{EE_{max}}$: This is the maximum linear acceleration of the trajectory.
- $^{d}\omega_{EE_{max}}$: This is the maximum angular velocity of the trajectory.
- $^{d}\dot{\omega}_{EE_{max}}$: This is the maximum angular acceleration of the trajectory.

- Admittance filter:
  - $\boldsymbol{M_a}$: The apparent mass of the system as seen by the human cooperator.
  - $\boldsymbol{D_a}$: The apparent damping of the system as seen by the human cooperator.
  - $\boldsymbol{K_a}$: The apparent stiffness of the system as seen by the human cooperator.
- Controller:
  - $\boldsymbol{K_{p_1}}$: The proportional gain on the translational part of the controller.
  - $\boldsymbol{K_{p_2}}$: The differential gain of the translational part of the controller.
  - $\boldsymbol{K_{\omega_1}}$: The proportional gain on the rotational part of the controller.
  - $\boldsymbol{K_{\omega_2}}$: The differential gain of the rotational part of the controller.
- Gradient calculation:
  - k_1: Gain for the platform orientation task.
  - k_2: Gain for the center arm task.
- Inverse kinematic and null space projection:
  - $\boldsymbol{W}$: The weighting factor for the pseudo inverse Jacobian.
- Quadratic programming optimizer:
  - $\boldsymbol{W_a}$: Weights of the tasks.
  - λ: Limit on the rate of change of the output.

In addition to these parameters additional tasks can be added to the gradient calculation, and additional tasks and constraints can be added to the QP-optimizer.

Note that the limits on the accelerations and velocities in the trajectory generator only apply on the trajectory outputted by the trajectory generator. It is still possible, due to a force on the end effector, for the admittance filter to give a reference trajectory which has higher accelerations and velocities then the before mentioned limits.

Because the real system will not be able to use torque control for the arm, the joint torques will be translated into joints positions. This will be done by using forward Dynamics in order to get the joint accelerations, which will be integrated twice to get the desired joint positions of the arm. The forward dynamics equations is given in equation 3.4

The forces and the torques on the end effector can be measured like done in (Nava et al., 2020), where a force/torque sensor is placed at the end effector in order to measure the forces. Another possibility is to use a wrench estimator like in (Ryll et al., 2019). In this report a force torque sensor is used in order to measure the forces and the torques on the end effector. The main for this is the ease of implementation of a force/torque sensor in the simulation.

5 Simulation and Results

The Aerial manipulator considered will be the FiberTHex (see figure 3.1). This is an fully actuated hexa-rotor equipped with a three DoF robotic arm. The dynamics of the aeial manipulator will be simulated using the Gazebo simulator. (Koenig and Howard, 2004) A model of the aerial manipulator is described in the URDF file format. This model includes the joints at the rotors and at the robotic arm joints. Using Genom3 (Foughali et al., 2018) the rotors can be controlled by Matlab/Simulink (MAT, 2020), the propellers can be controlled using the angular rates. The robotic arm will be controlled using the controlboard plugin of YARP (Metta et al., 2006) This plugin simulates an PID controller with motors for the joints of the arm. This plugin is able to accept joint torques, velocities, or positions, and is able return joint velocities, and positions. The state of the robot will also be obtained using YARP.

The controller will be running in Matlab/Simulink. The pose of the base, and its velocities are obtained from YARP, as well as the joint positions and velocities of the arm. The controller will calculate the required angular rates for the propellers and joint positions, which will be sent to Gazebo using Genom3 and YARP respectfully.

In order to apply forces on the system during simulation, a controller plugin is used. This plugin allows the user to move a joystick, which will exert a force on the system. The user is also able to change the amount of force applied by using the D-pad on the joystick, in order to increase and decrease the force. The forces created using this plugin will be applied to the end-effector of the aerial manipulator.

In order to test the system, it will first be tested without the admittance filter. This will show how well the whole body controller works. After this the admittance filter will be added and tested, to see the behavior of the whole system. Lastly the robustness of the system will be tested. The parameters of the system are given in table 5.1. The upper and lower limits of the propeller speeds are $16Hz$ and $100Hz$ respectfully. The joint limits for the first joint are $-45°, 45°$, for the second joint $-90°, 90°$ and for the third joint $-105°, 90°$.

5.1 Controller simulations

5.1.1 Trajectory

First the ability of the controller to follow a trajectory will be tested. During this test no forces will be applied to the system. The trajectory represents a pick and place like motion. The end-effector will move to $(1,0,1)$ where it will move to the ground in order to "pick up" something, after this it will move to $(-2,3,1)$, where it will "place" the object.

The results are show in figures 5.1 to 5.4. Figure 5.1 shows the state of the system, this plot shows the platform is horizontal for the complete trajectory. Figure 5.2 shows the pose of the end effector together with the trajectory of the trajectory generator. (The desired trajectory.) This shows the end-effector is able to follow the trajectory. In the lower plot it can be seen that the orientation deviates from the desired when the orientation needs to change. Figure 5.3 shows the error in the position and in the orientation of during the simulation. The position error is smaller then $5mm$ while following the trajectory, and the orientation error is smaller then $5°$. Figure 5.4 shows the system inputs sent to Genom3 and YARP. The upper plot shows the propeller speeds will be between 65 and $80Hz$ when no forces are applied to the system.

Table 5.1: System parameters during simulations

Parameter	Value
${}^{d}v_{EE_{max}}$	0.8
${}^{d}a_{EE_{max}}$	0.4
${}^{d}\omega_{EE_{max}}$	20
${}^{d}\dot{\omega}_{EE_{max}}$	10
$\boldsymbol{M_a}$	Cartesian mass
$\boldsymbol{D_a}$	3
$\boldsymbol{K_a}$	5
$\boldsymbol{K_{p_1}}$	20
$\boldsymbol{K_{p_2}}$	10
$\boldsymbol{K_{\omega_1}}$	20
$\boldsymbol{K_{\omega_2}}$	10
k_1	100
k_2	70
$\boldsymbol{W}$	blckdiag([1 1 1 10000 10000 10000 1000 1000 1000])
$\boldsymbol{W_a}$	blckdiag([8 0.1 0.8 0.0001 10])
λ	[0.1296 0.1]

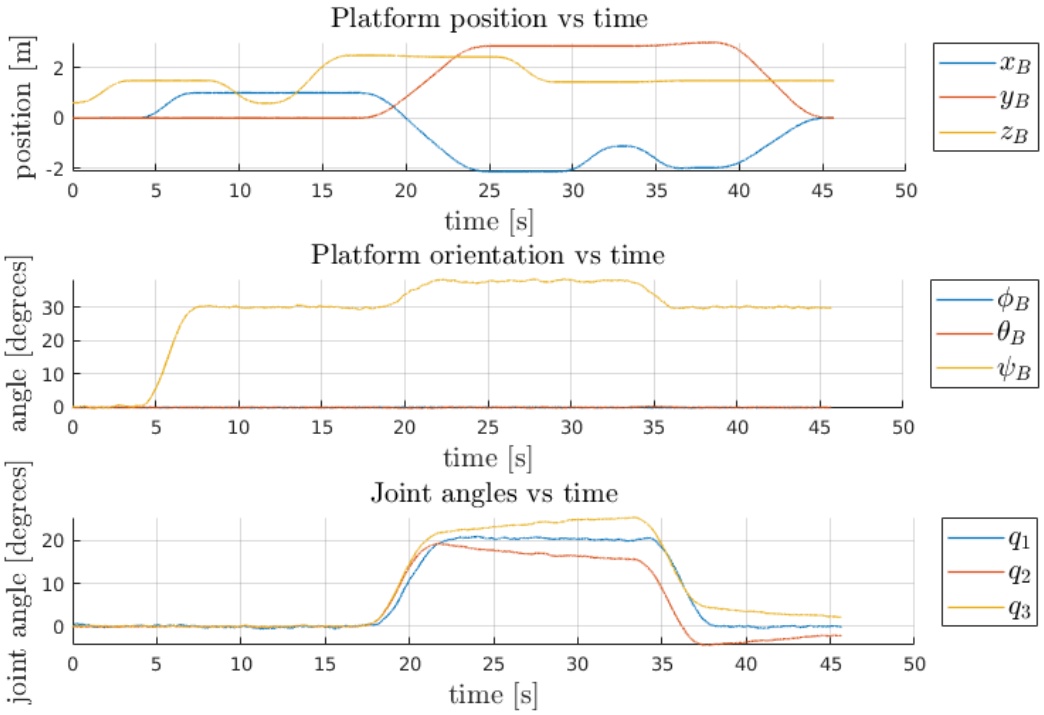


Figure 5.1: System state during trajectory simulation. The first plot shows the position of the base, the second plot the orientation of the base in roll, pitch, and yaw. The third plot shows the joint positions of the arm joints.

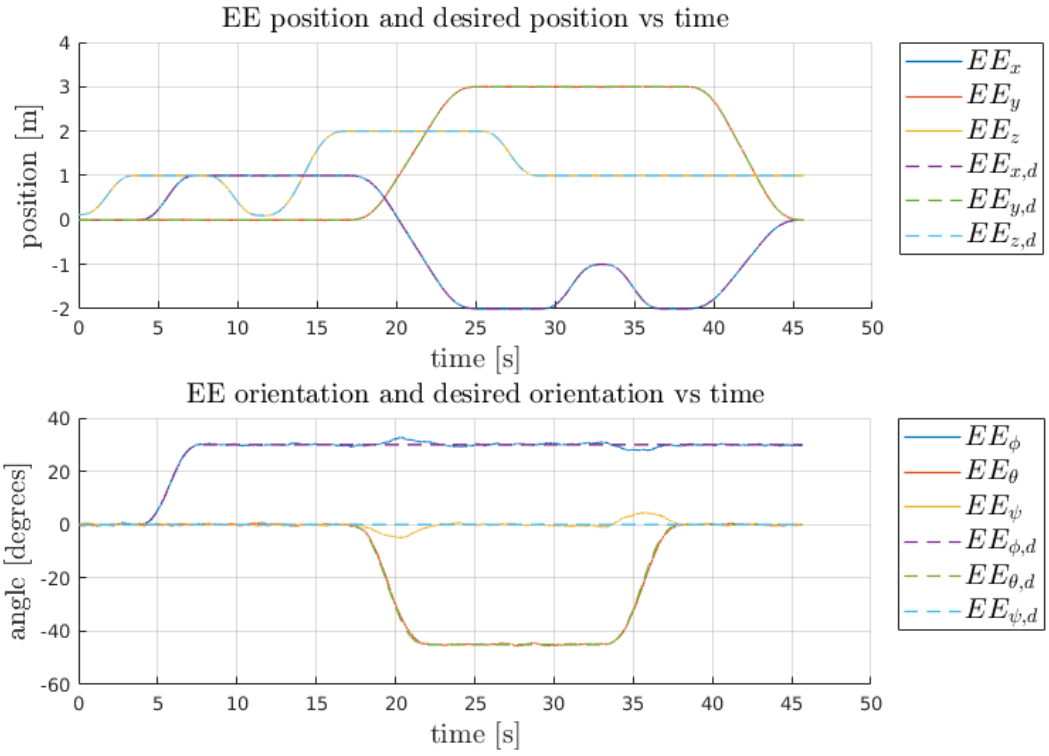


Figure 5.2: End-effector position during trajectory simulation. The first plot shows the position of the end-effector, solid line, together with the desired position, dashed line. The second plot shows the orientation of the end-effector, solid line, together with the desired orientation, dashed line.

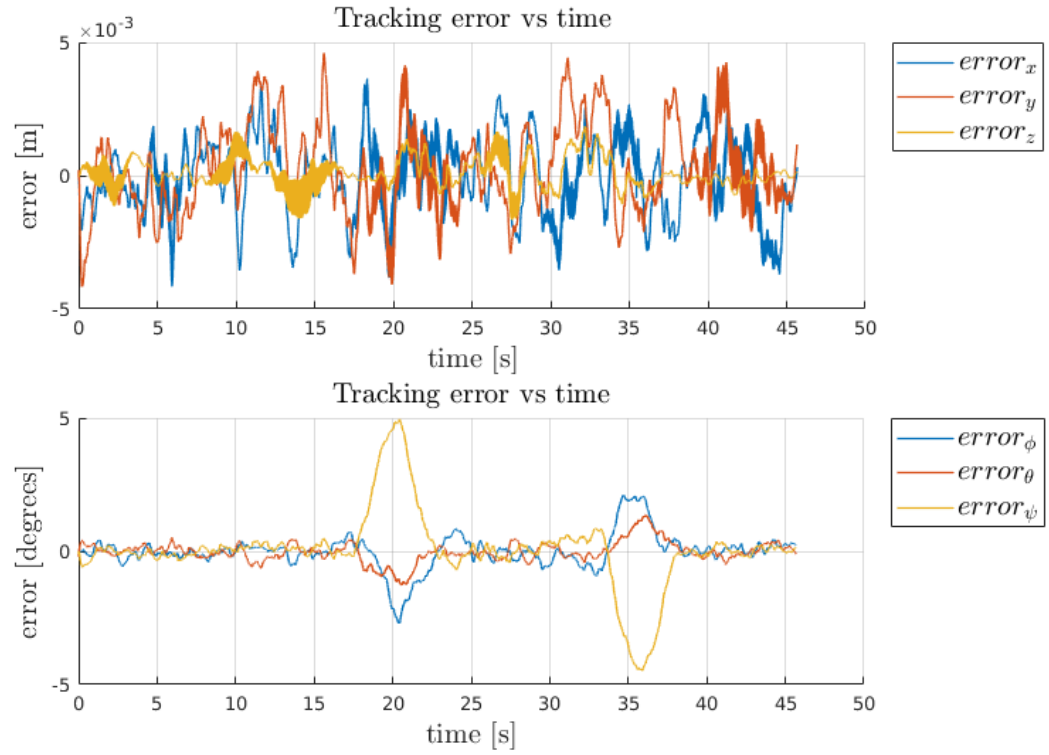


Figure 5.3: error of the end-effector pose during the trajectory simulation. The first plot shows the position error, and the second plot the orientation error.

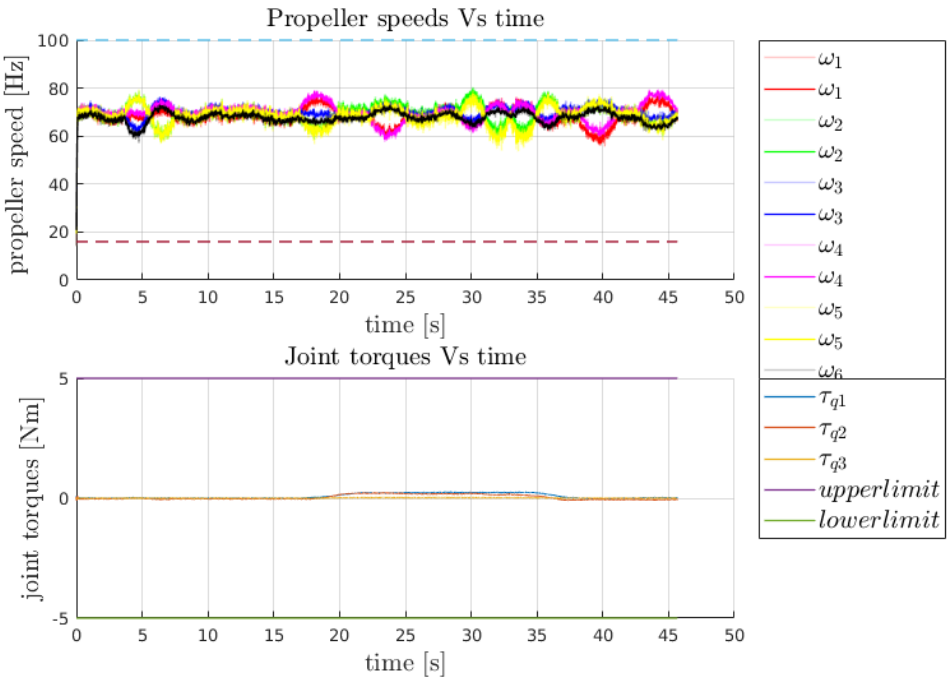


Figure 5.4: System inputs during trajectory simulation. The First plot shows the angular rates of the propellers, and the second plot shows the joint torques.

5.1.2 Hovering simulation

During the hovering test, the end-effector is asked to go to $(0,0,1)$, and keep the end effector orientations identity. During this simulation a force is applied to the end-effector in order to see what the system can handle.

The results of the hovering simulation are shown in figures 5.5 to 5.8. Figure 5.5 shows the system state, figure 5.6 shows the pose of the end-effector (solid line), and the desired pose of the end-effector (dashed line), figure 5.7 shows the interaction wrench on the end-effector, and figure 5.8 show the inputs to the system with the input limits.

Looking at figures 5.6 and 5.7, it can be seen that when the force on the end-effector is $6N$, in x or y-direction, the error increases. Figure 5.8 shows this is when some of the propellers are saturated.

When the system is pushed with $6N$ the QP-optimizer will relax the constraints on the lower priority tasks to satisfy the higher priority ones, this effect can be seen in the results. The highest priority task beside the propeller legalization, is the orientation of the platform. While the platform rotates more than $10°$ (see figure 5.5) the roll and the pitch of the system remain close to zero, this is also partly because of the redundancy resolution.

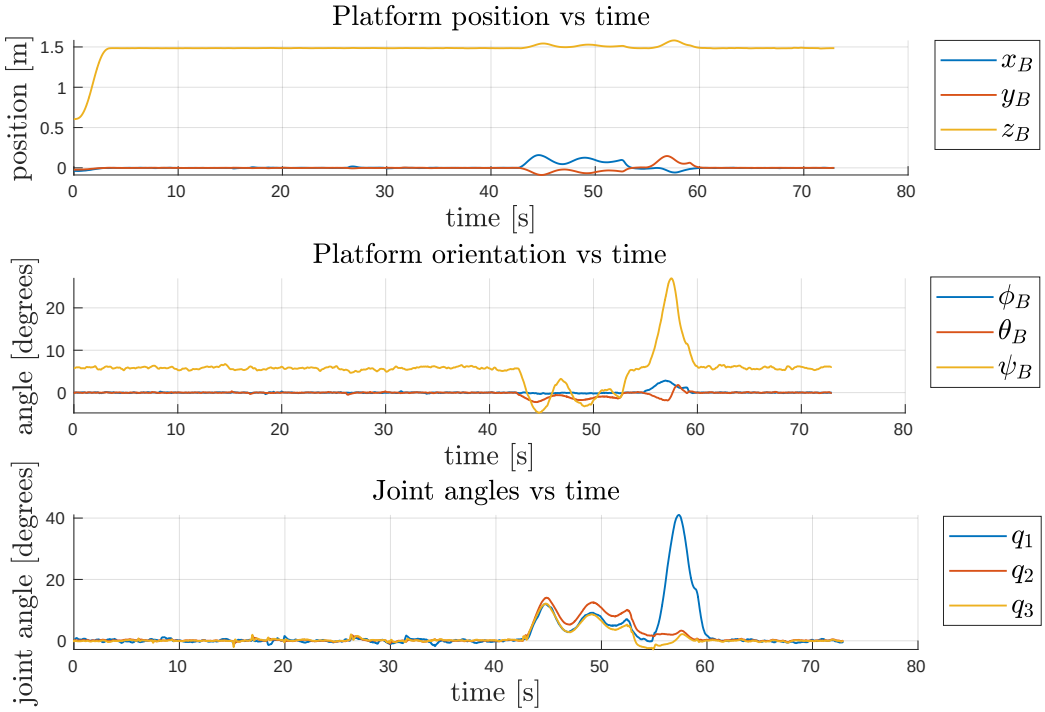


Figure 5.5: System state during hovering simulation. The first plot shows the position of the base, the second plot the orientation of the base in roll, pitch, and yaw. The third plot shows the joint positions of the arm joints.

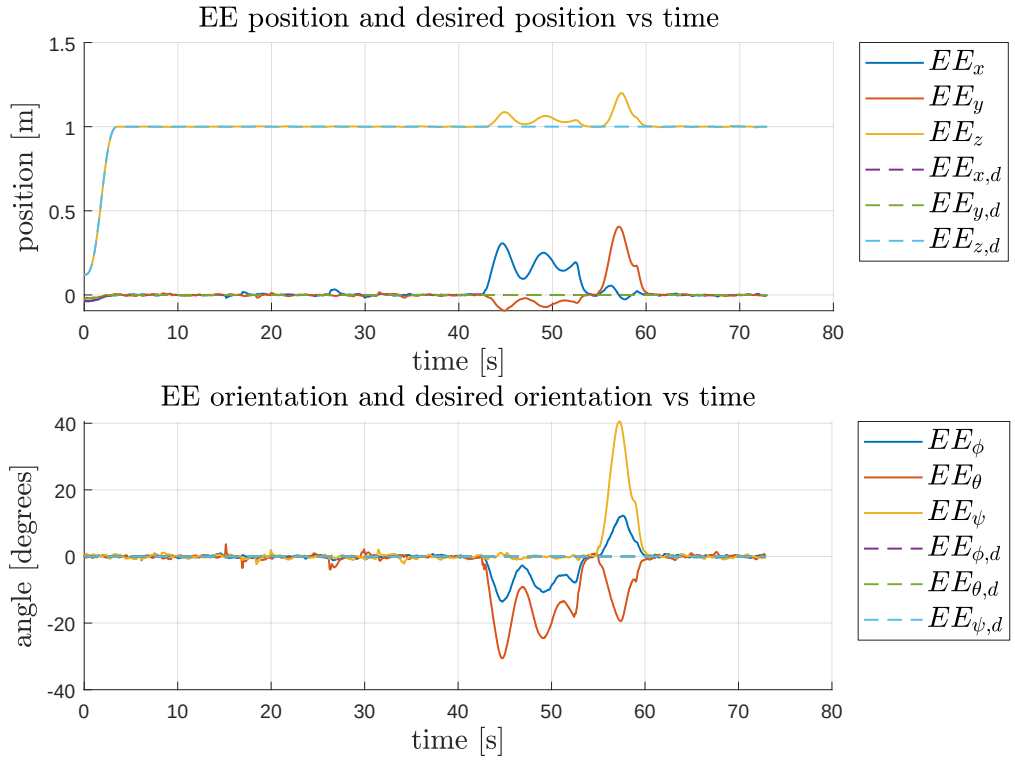


Figure 5.6: End-effector position during hovering simulation. The first plot shows the position of the end-effector, solid line, together with the desired position, dashed line. The second plot shows the orientation of the end-effector, solid line, together with the desired orientation, dashed line.

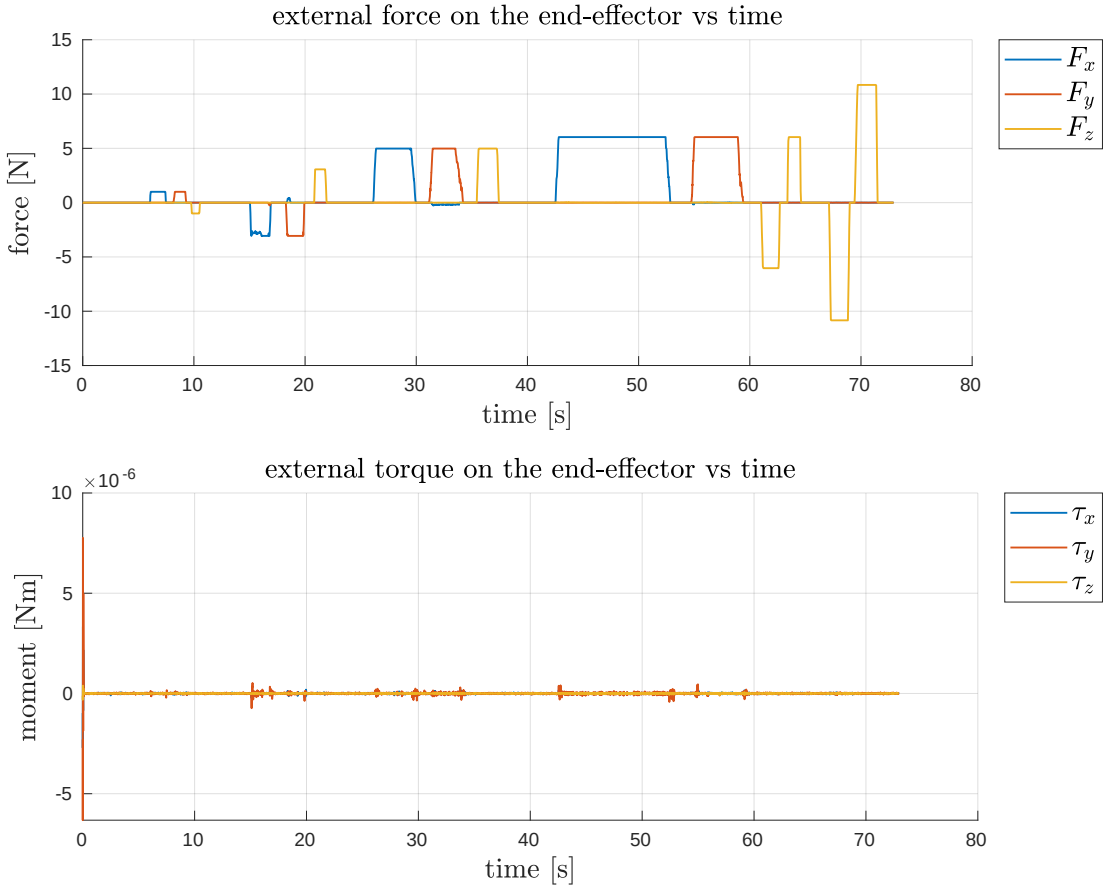


Figure 5.7: Wrench on the end-effector during hovering simulation. The first plot shows the measured forces, the second plot shows the measured torque.

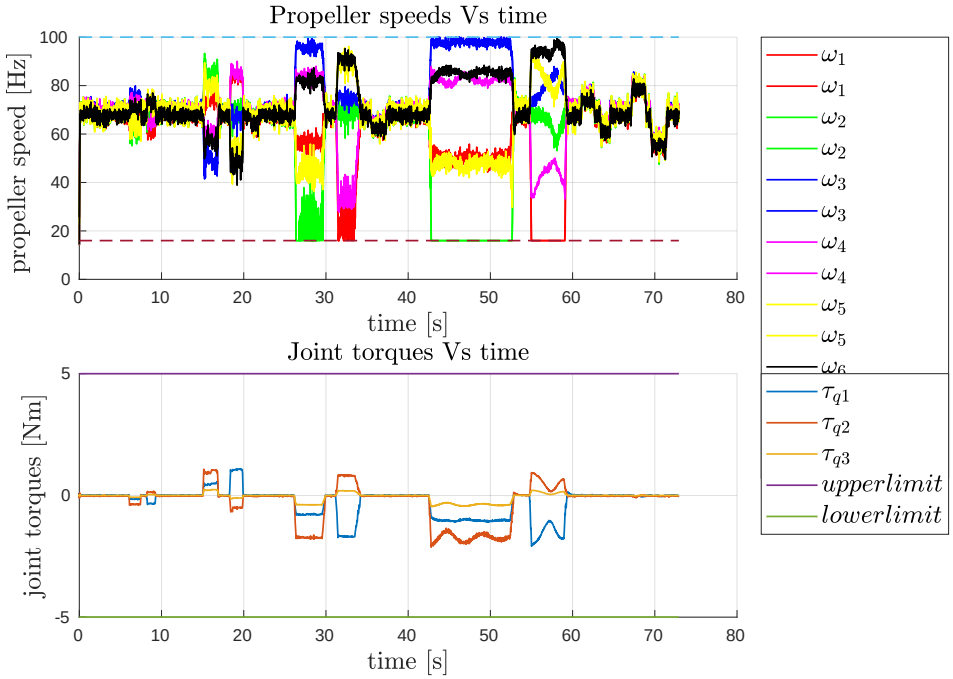


Figure 5.8: System inputs during hovering simulation. The First plot shows the angular rates of the propellers, and the second plot shows the joint torques.

5.1.3 Orientation trajectory

During the orientation trajectory simulation, the end-effector is asked to go to $(0,0,1)$, and keep this position, while changing the orientation. The redundancy is trying to have the joint angles as far from the joint limits as possible, the effect of this can be seen in figure 5.9. Joint three has a larger angle than joint two, this is because the limit of joint three is at $-105°$, while the joint limit of joint two is at $-90°$. The angles in figures 5.9, and 5.10 are mapped between $-180°$ and $180°$, this wrap around can be seen in both figures.

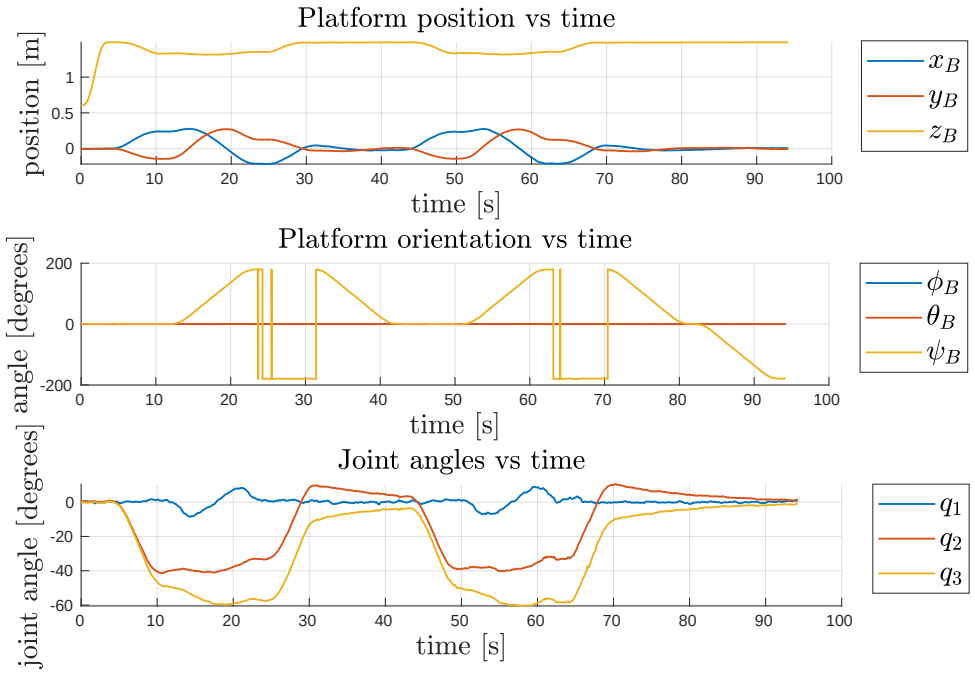


Figure 5.9: System state during orientation trajectory simulation. The first plot shows the position of the base, the second plot the orientation of the base in roll, pitch, and yaw. The third plot shows the joint positions of the arm joints.

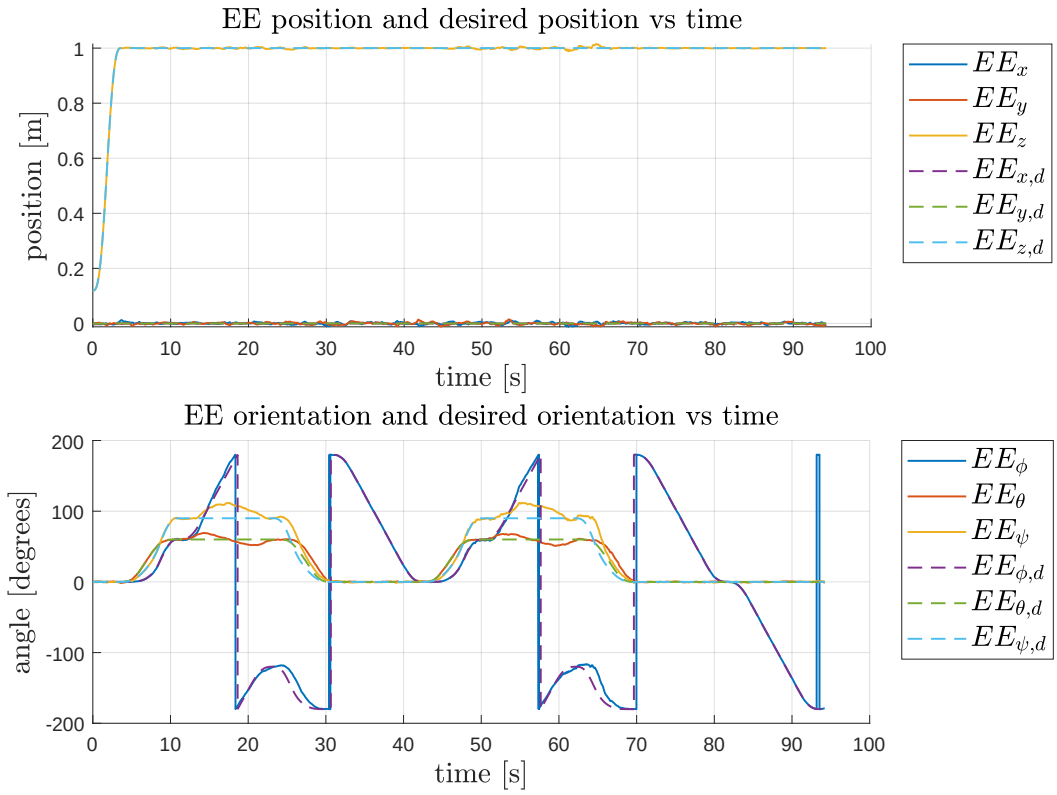


Figure 5.10: End-effector position during orientation trajectory simulation. The first plot shows the position of the end-effector, solid line, together with the desired position, dashed line. The second plot shows the orientation of the end-effector, solid line, together with the desired orientation, dashed line.

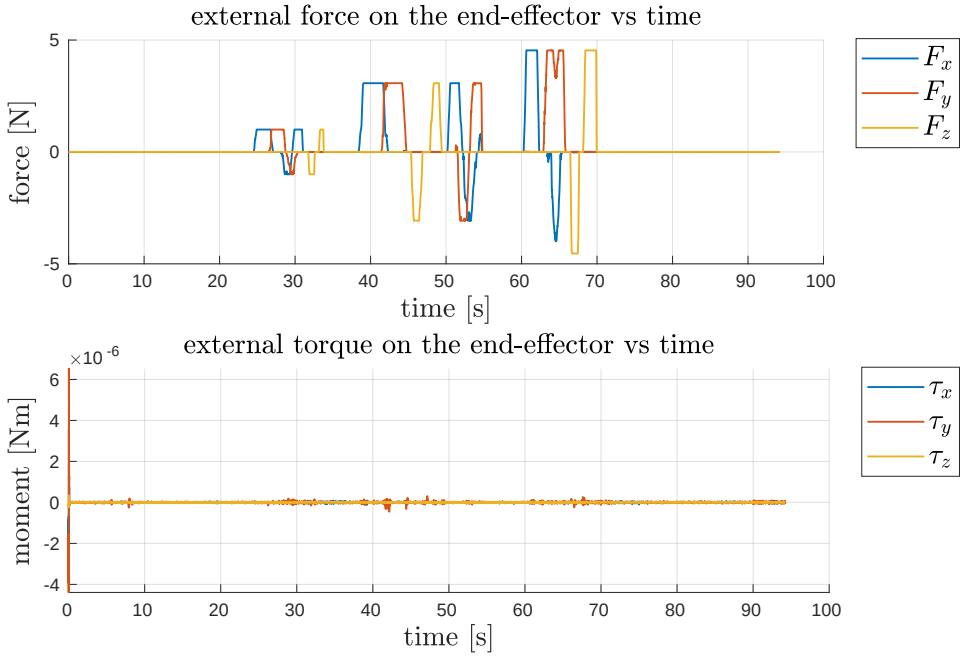


Figure 5.11: Wrench on the end-effector during orientation trajectory simulation. The first plot shows the measured forces, the second plot shows the measured torque.

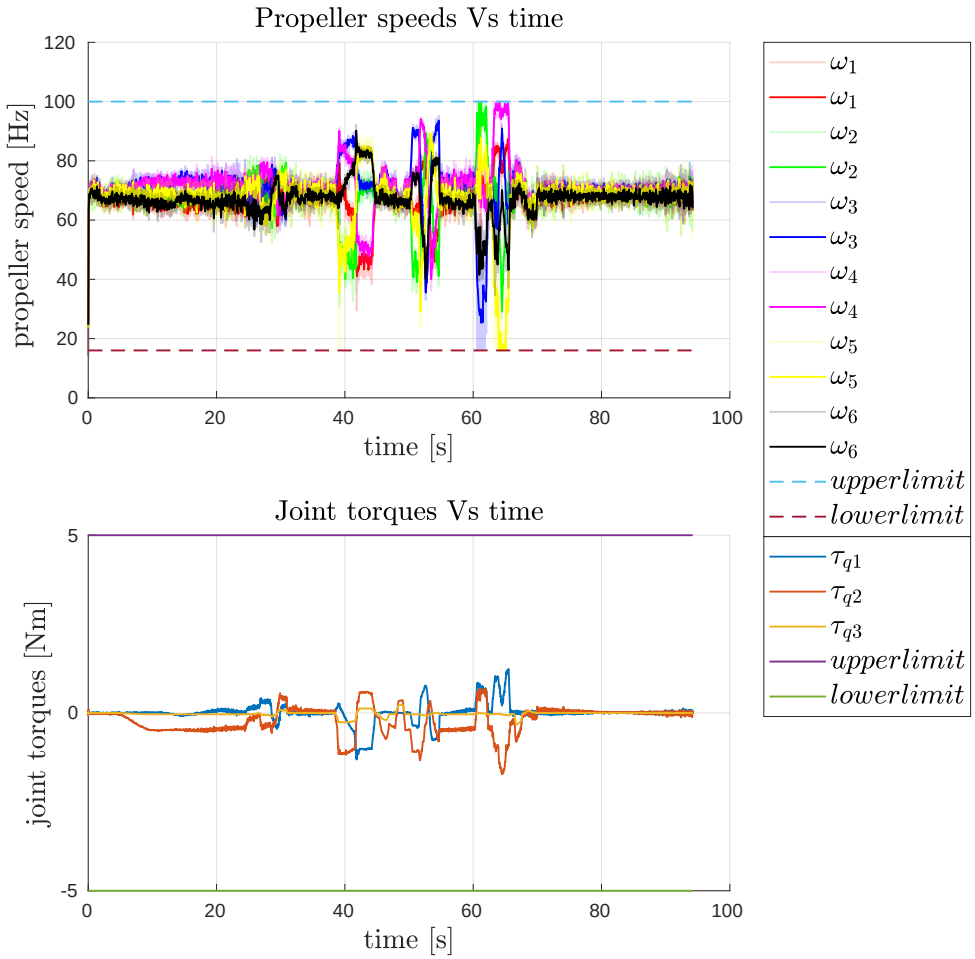


Figure 5.12: System inputs during orientation trajectory simulation. The First plot shows the angular rates of the propellers, and the second plot shows the joint torques.

5.1.4 Trajectory with disturbance

This simulation is the same as the Trajectory simulation, only this time force will be applied to the end-effector. These forces will be applied with a maximum amplitude of $3N$ in x, y, or z-direction. The forces will be applied between the "pick up" and the "place" parts of the trajectory.

The results of the third test are shown in figures 5.13 to 5.18. Here the velocity of the end-effector is included, figure 5.15, to show how the velocity setpoints of the trajectory generator are followed. This plot shows that the velocity of the platform follows the velocity of the trajectory generator when there is no force on the end-effector. When there is interaction with the end-effector, the velocity changes, but still average around the trajectory velocity. The tracking error is shown in figure 5.16, this shows that the system is able to track the position with an accuracy of $6cm$ when forces are applied at the end effector. The orientation of the end effector can be tracked with an accuracy of 9° with disturbance.

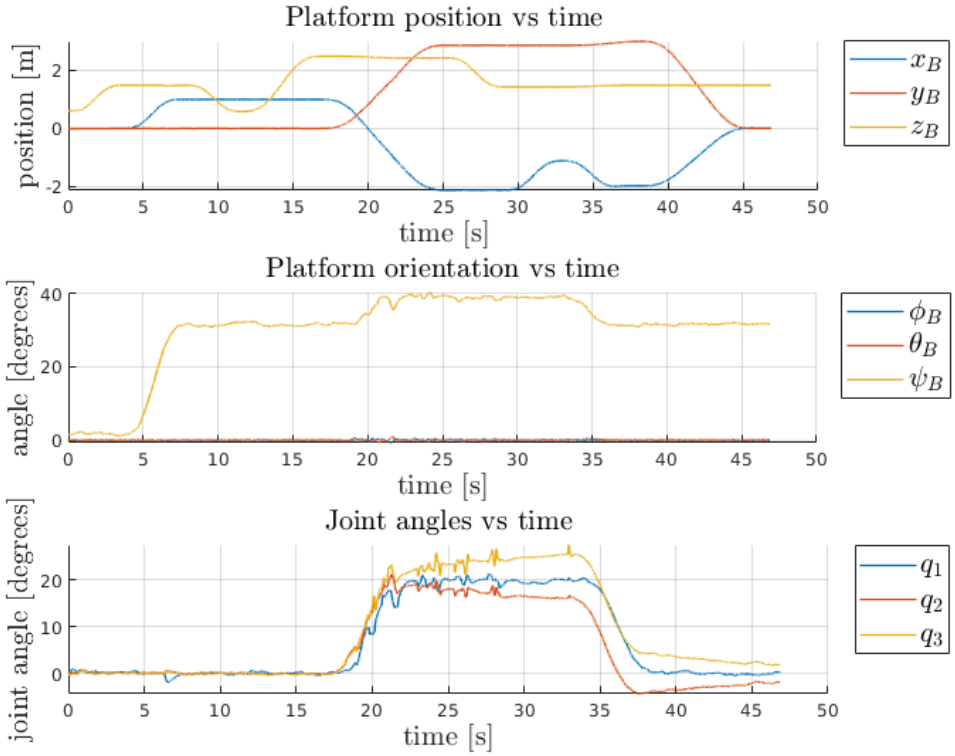


Figure 5.13: System state during Trajectory with disturbance simulation. The first plot shows the position of the base, the second plot the orientation of the base in roll, pitch, and yaw. The third plot shows the joint positions of the arm joints.

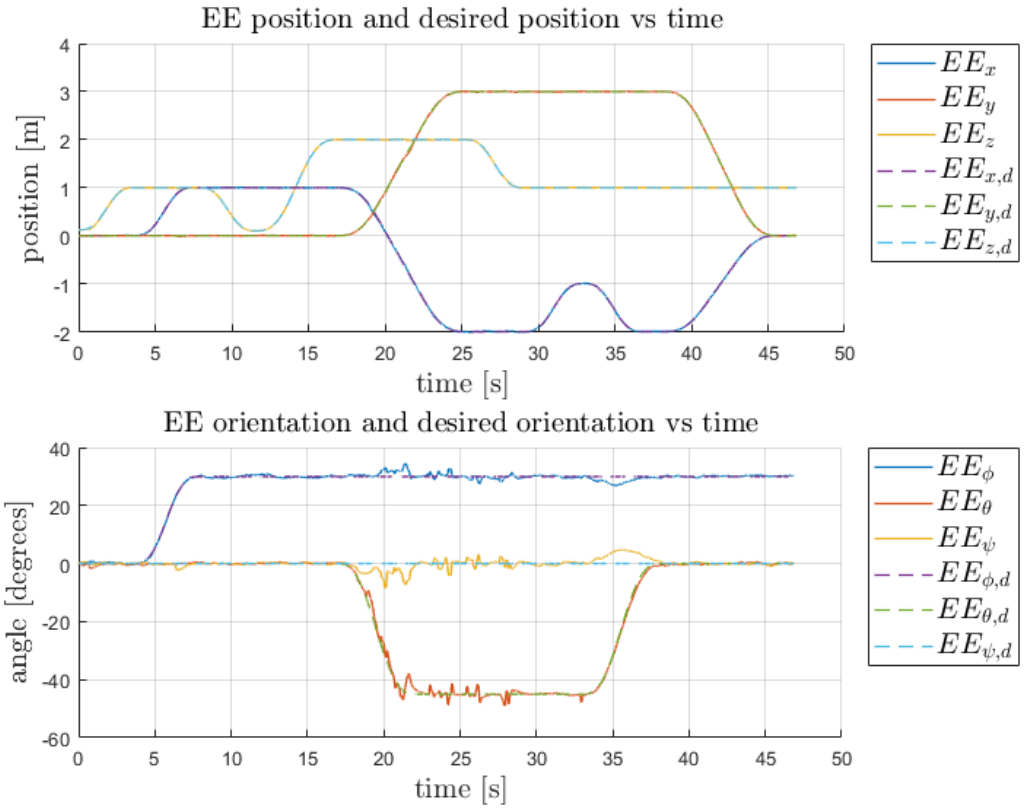


Figure 5.14: End-effector position during Trajectory with disturbance simulation. The first plot shows the position of the end-effector, solid line, together with the desired position, dashed line. The second plot shows the orientation of the end-effector, solid line, together with the desired orientation, dashed line.

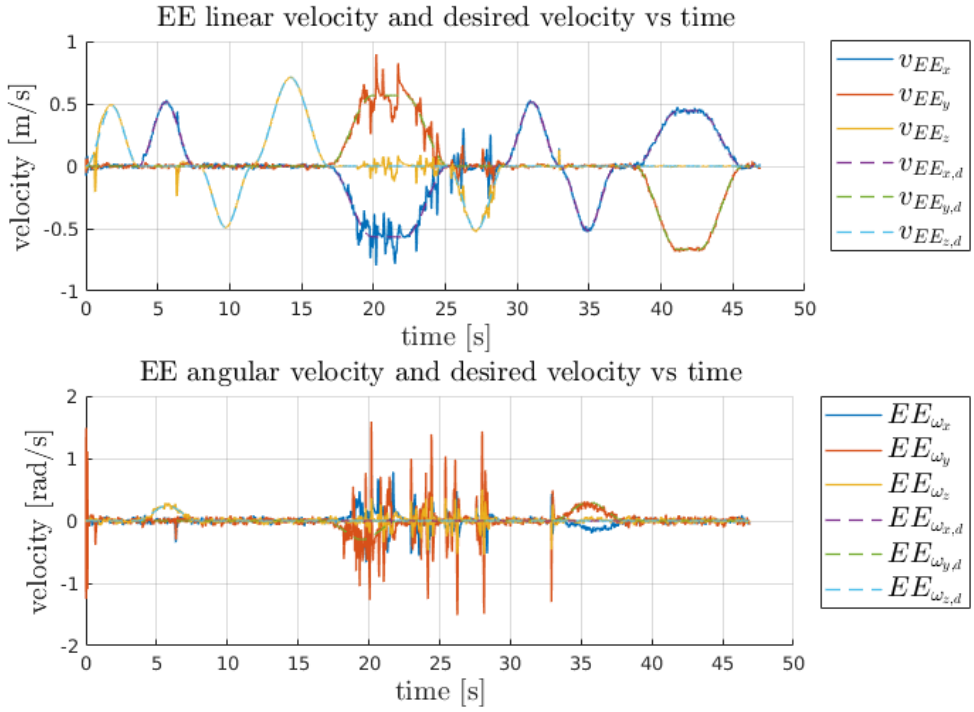


Figure 5.15: End-effector velocities during Trajectory with disturbance simulation. The first plot shows the linear velocity of he end-effector, solid line, and the desired linear velocity, dashed line. The second plot shows the angular velocity of the end-effector, solid line, and the angular desired angular velocity, dashed line.

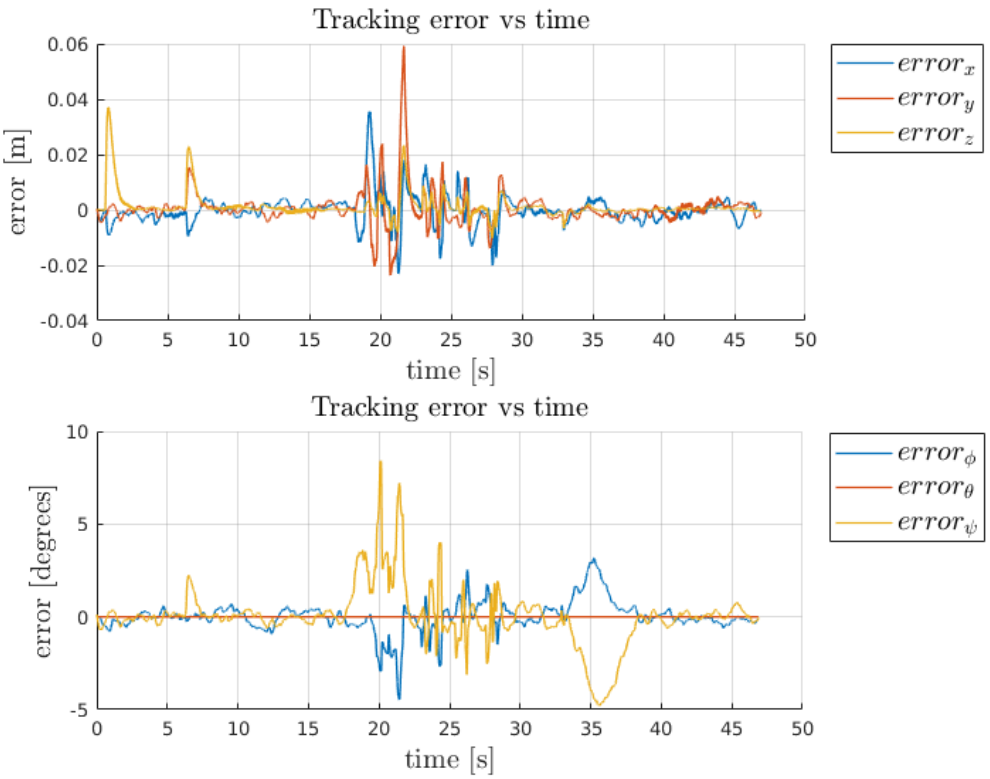


Figure 5.16: error of the end-effector pose during the Trajectory with disturbance simulation. The first plot shows the position error, and the second plot the orientation error.

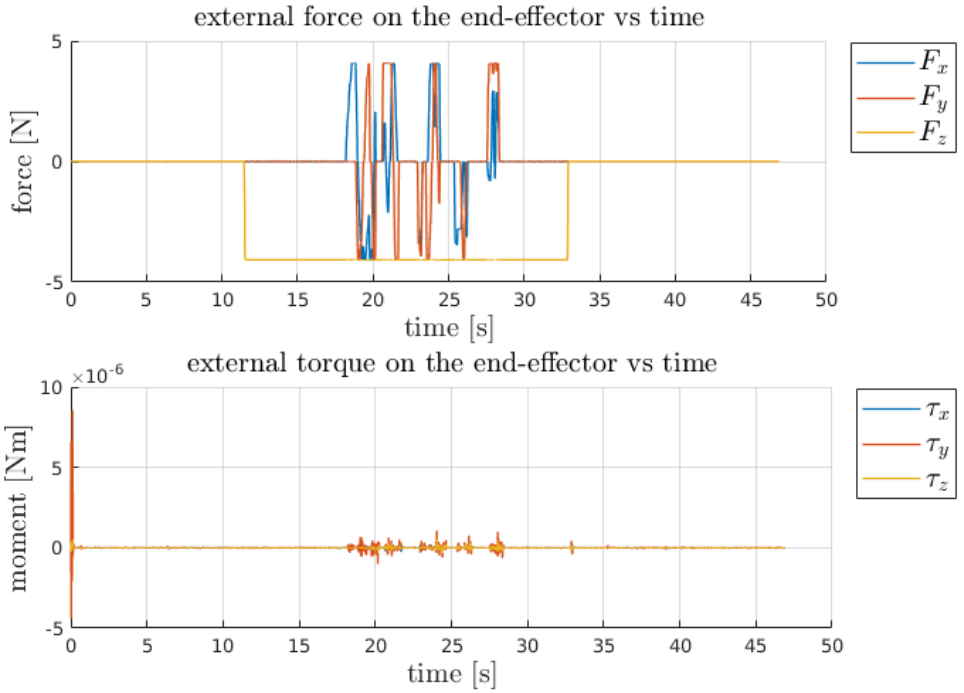


Figure 5.17: Wrench on the end-effector during Trajectory with disturbance simulation. The first plot shows the measured forces, the second plot shows the measured torque.

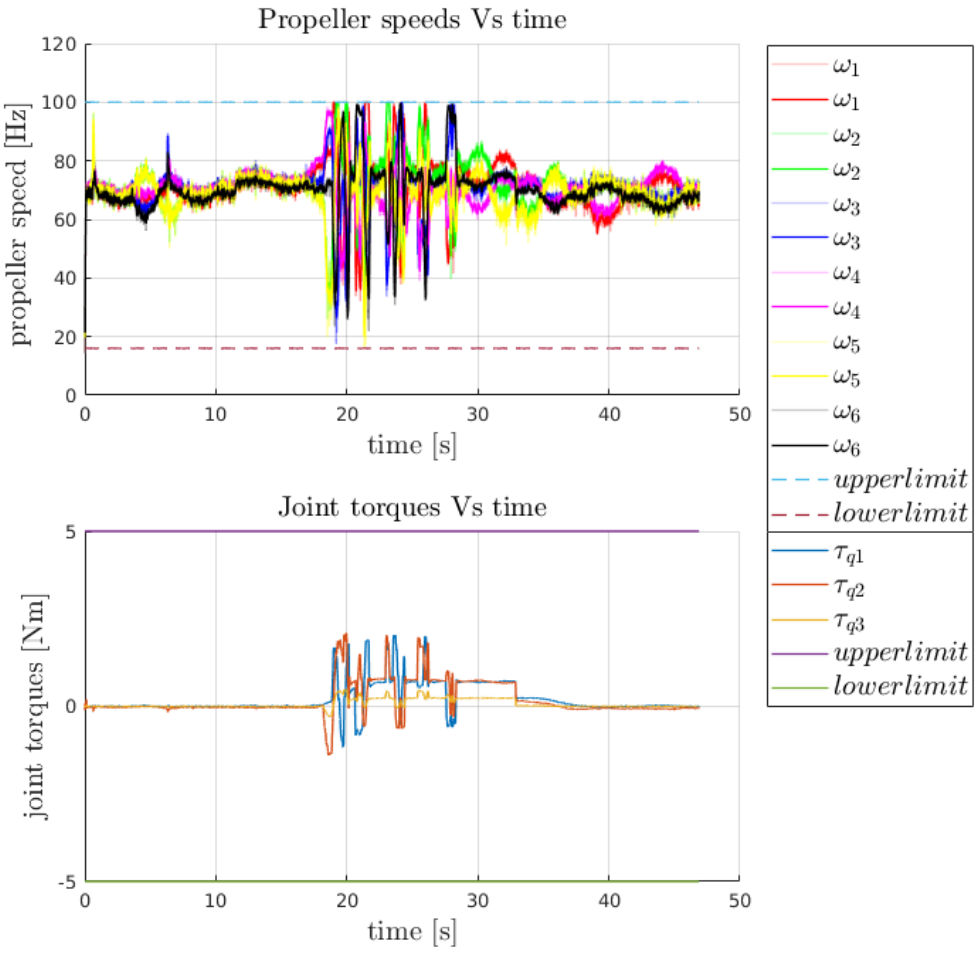


Figure 5.18: System inputs during Trajectory with disturbance simulation. The First plot shows the angular rates of the propellers, and the second plot shows the joint torques.

5.1.5 Zero motions

The zero motions simulation shows how the redundancy resolution can move the system without interfering with the main objective of the end-effector pose. An additional term is added to the redundancy gradient calculation, to let the platform x-position move sinusoidal. The results of the forth test are shown in figures 5.19 and 5.20. The pose changes within a range of $2cm$ and within 0.02° while the platform moves in total $18cm$ from side to side.

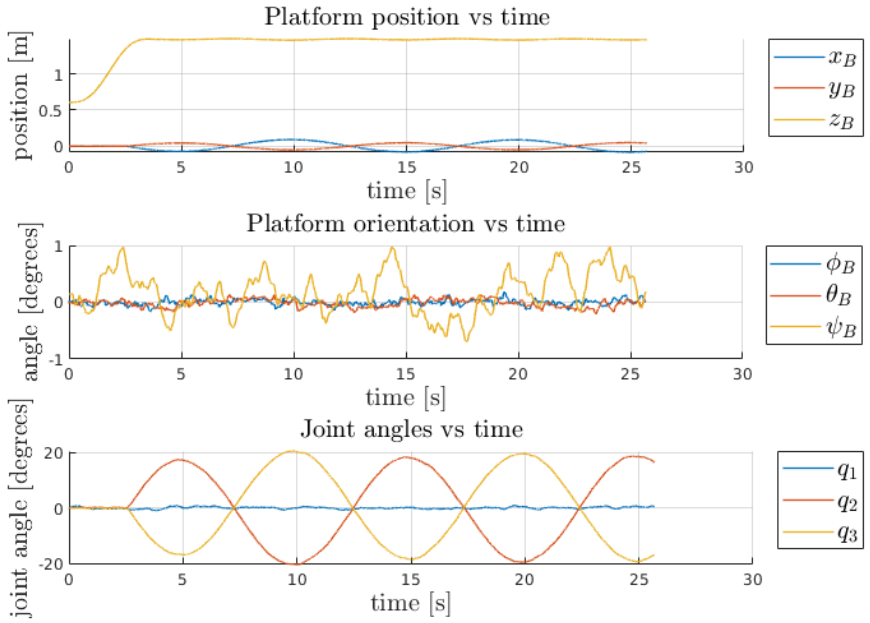


Figure 5.19: System state during zero motions simulation. The first plot shows the position of the base, the second plot the orientation of the base in roll, pitch, and yaw. The third plot shows the joint positions of the arm joints.

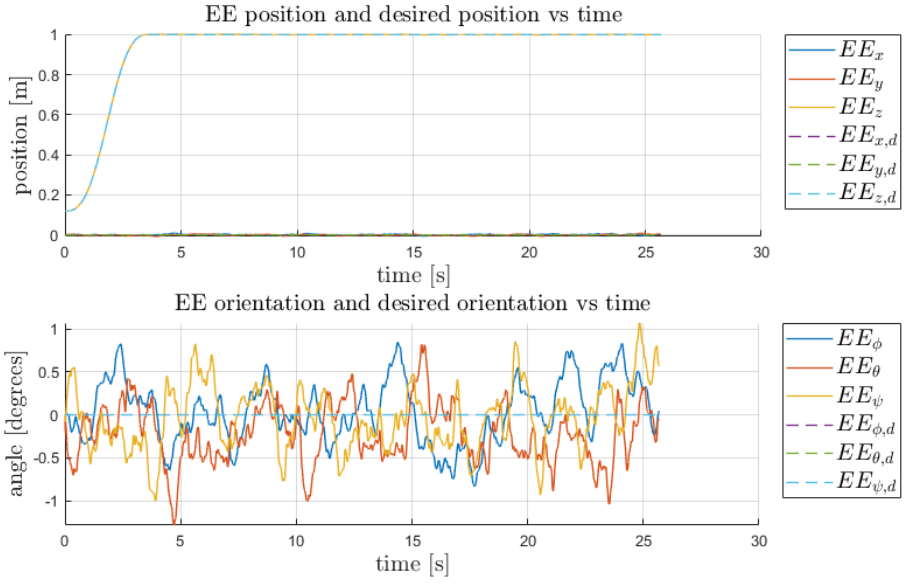


Figure 5.20: End-effector position during zero motions simulation. The first plot shows the position of the end-effector, solid line, together with the desired position, dashed line. The second plot shows the orientation of the end-effector, solid line, together with the desired orientation, dashed line.

5.2 Including the admittance filter

5.2.1 Mass spring damper simulation

During this simulation the system will tasked to do the same thing as in the hovering test, but this time the admittance filter is active. Forces will be applied to the end effector and the system should behave like a mass-spring-damper system.

The results of this test are shown in figures 5.21 to 5.25, where figure 5.23 shows the reference position, velocity, and acceleration from the admittance filter (solid line), with the desired position, velocity, and acceleration from the trajectory generator (dashed line). Notice that the reference velocities, and accelerations can be larger than the maximums defined in table 5.1. The mass spring damper parameters form an under damped system. In figure 5.22 (and figure 5.24) you can see that when the force on the end-effector is $6N$ the position of the end effector deviates from the reference position, and also the orientation deviates from the reference. This again corresponds to some of the propellers being saturated. (figure 5.25)

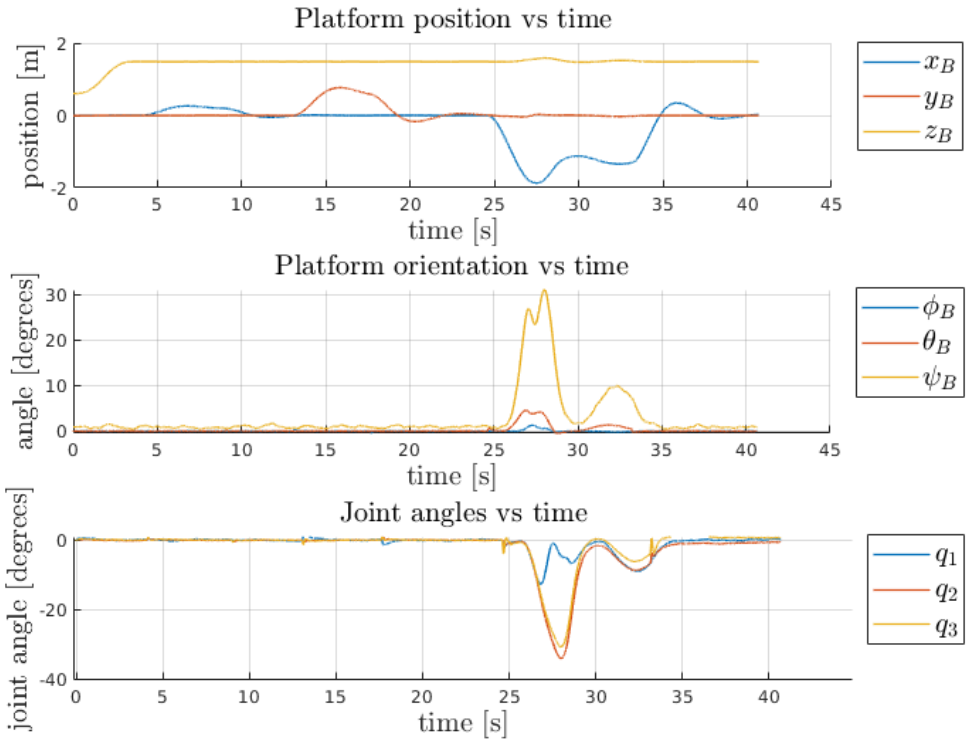


Figure 5.21: System state during mass spring damper simulation. The first plot shows the position of the base, the second plot the orientation of the base in roll, pitch, and yaw. The third plot shows the joint positions of the arm joints.

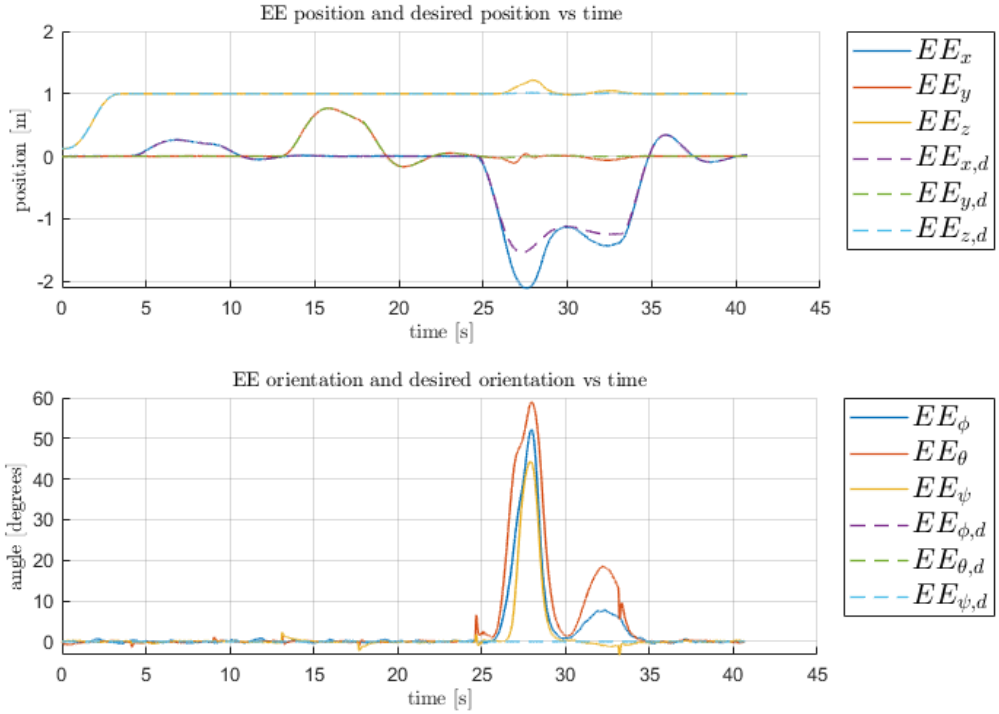


Figure 5.22: End-effector position during mass spring damper simulation. The first plot shows the position of the end-effector, solid line, together with the desired position, dashed line. The second plot shows the orientation of the end-effector, solid line, together with the desired orientation, dashed line.

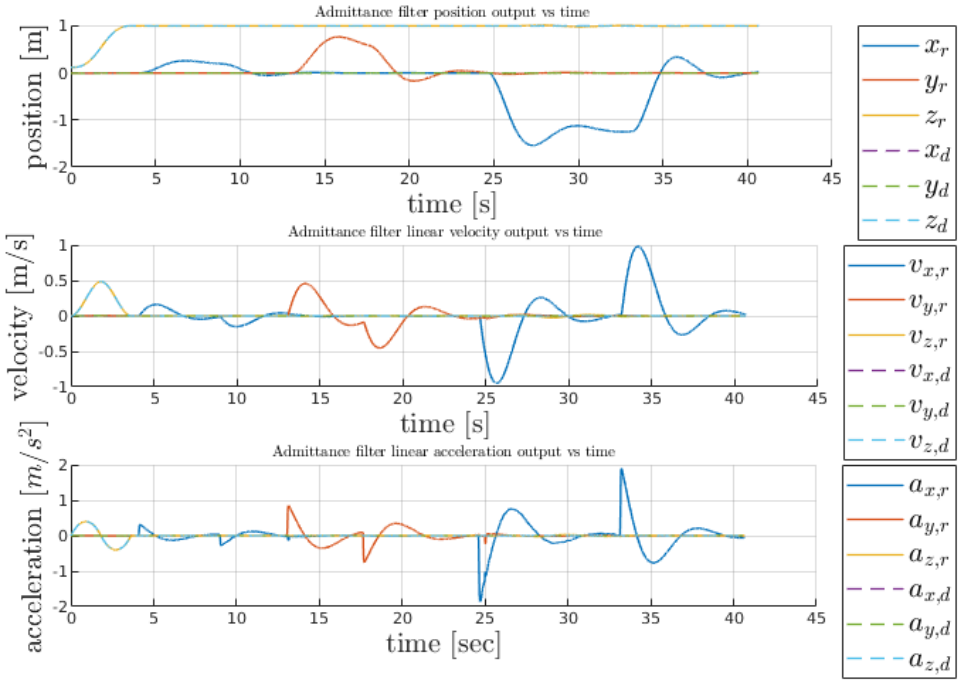


Figure 5.23: The position, velocity, and accelerations given by the admittance filter, solid line, Together with the position, velocity, and acceleration from the trajectory generator, dashed line, during the mass spring damper simulation.

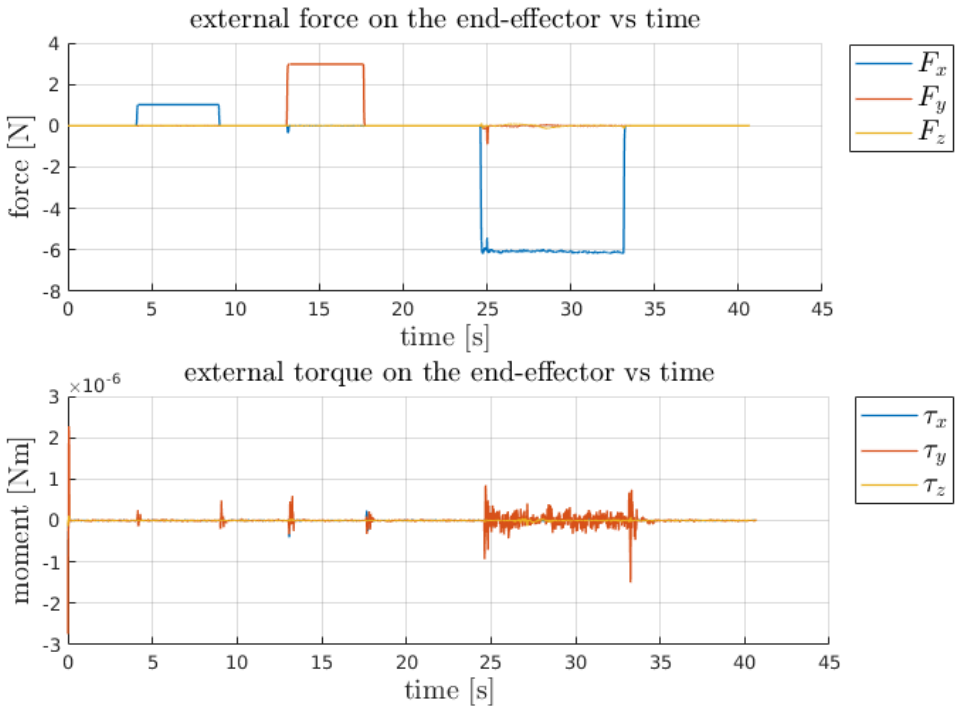


Figure 5.24: Wrench on the end-effector during mass spring damper simulation. The first plot shows

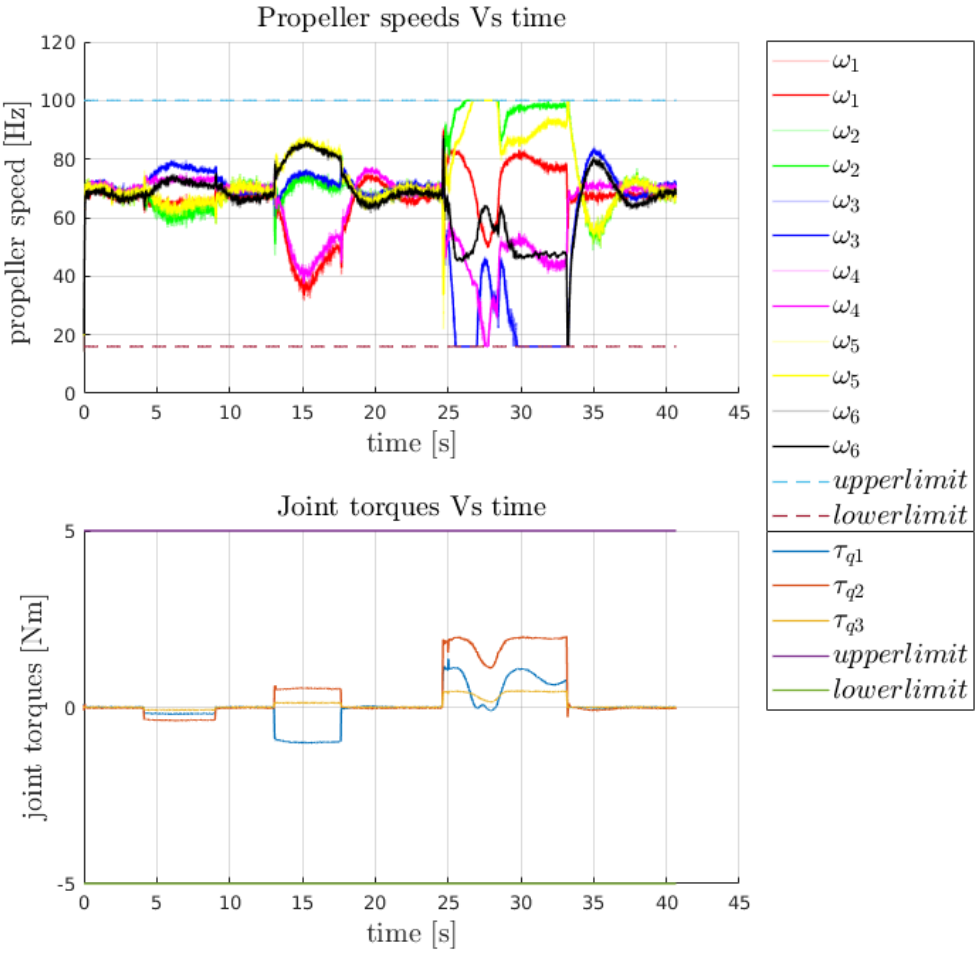


Figure 5.25: System inputs during mass spring damper simulation. The First plot shows the angular rates of the propellers, and the second plot shows the joint torques.

5.2.2 Admittance trajectory

During the admittance trajectory simulations, the system will follow the same trajectory as in the trajectory simulation. This time the admittance filter will be active, and interaction forces will be applied to the end-effector.

The results of this test are shown in figures 5.21 to 5.25. Again, the interaction force is applied between the pick and the place parts of the trajectory. In figure 5.28 the output of the admittance filter, and the trajectory generator are show. This plot shows that the reference position changes around the desired position, because of the applied force. This is also the case for the velocity and acceleration, but the changes here are more extreme.

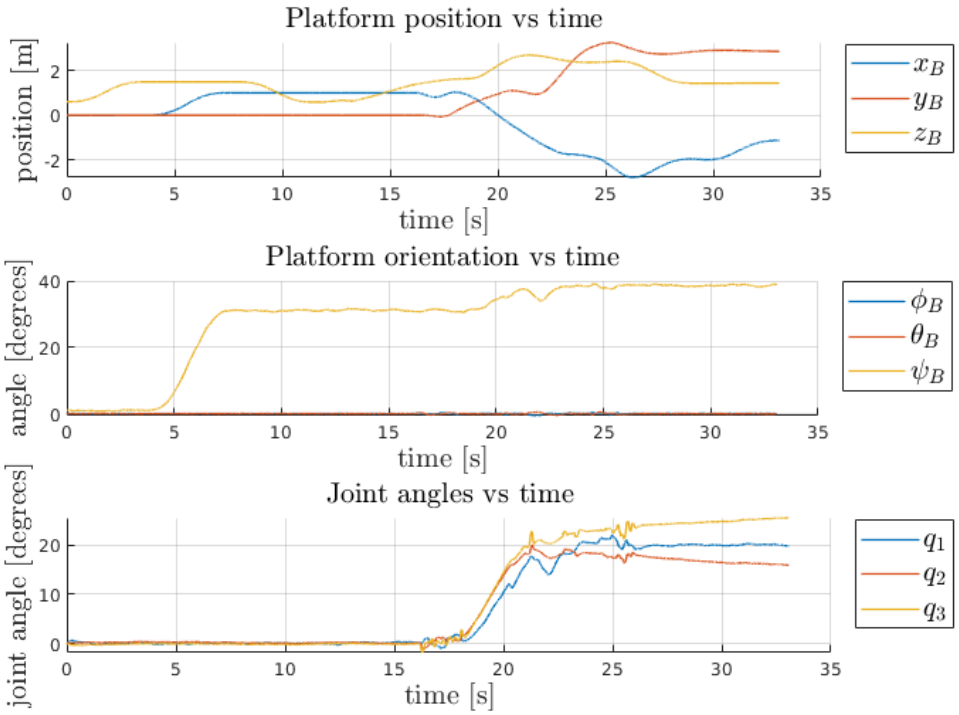


Figure 5.26: System state during admittance trajectory simulation. The first plot shows the position of the base, the second plot the orientation of the base in roll, pitch, and yaw. The third plot shows the joint positions of the arm joints.

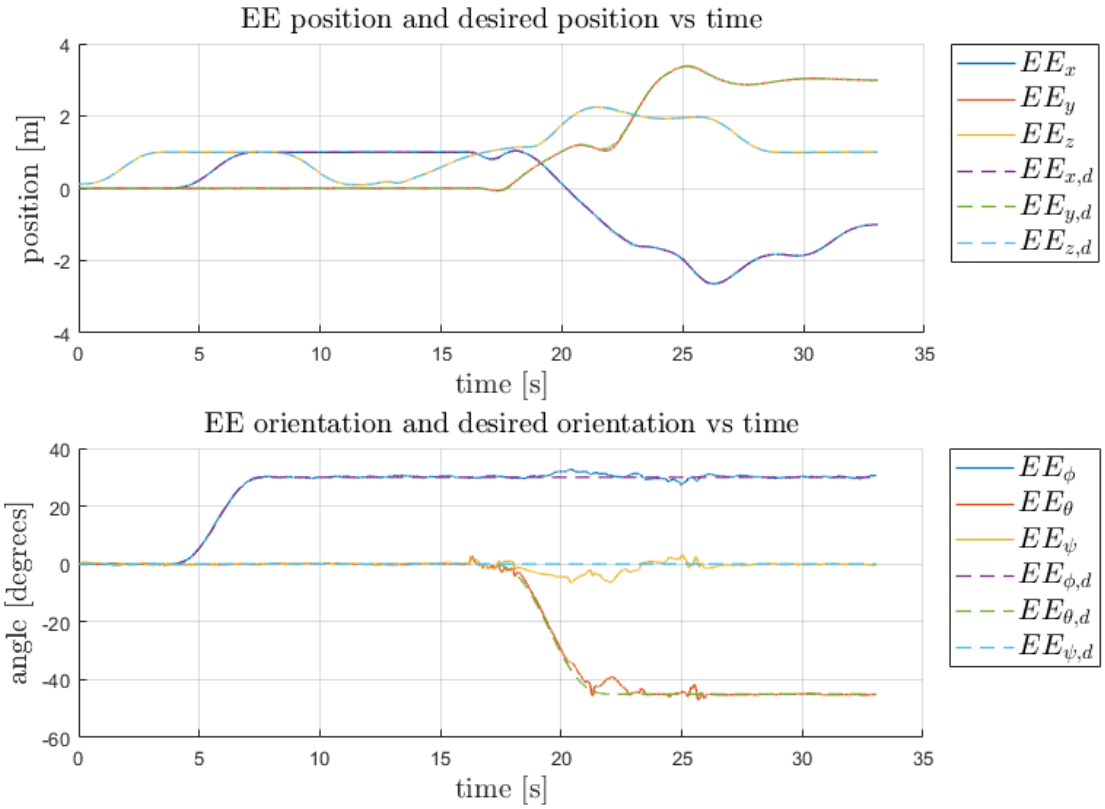


Figure 5.27: End-effector position during admittance trajectory simulation. The first plot shows the position of the end-effector, solid line, together with the desired position, dashed line. The second plot shows the orientation of the end-effector, solid line, together with the desired orientation, dashed line.

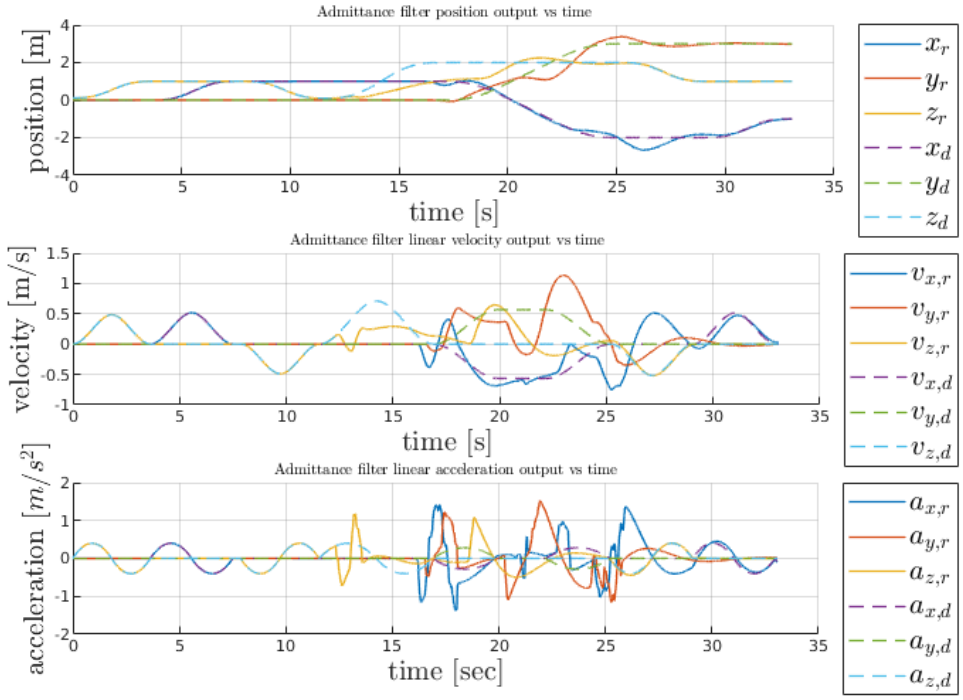


Figure 5.28: The position, velocity, and accelerations given by the admittance filter, solid line, Together with the position, velocity, and acceleration from the trajectory generator, dashed line, during the admittance trajectory simulation.

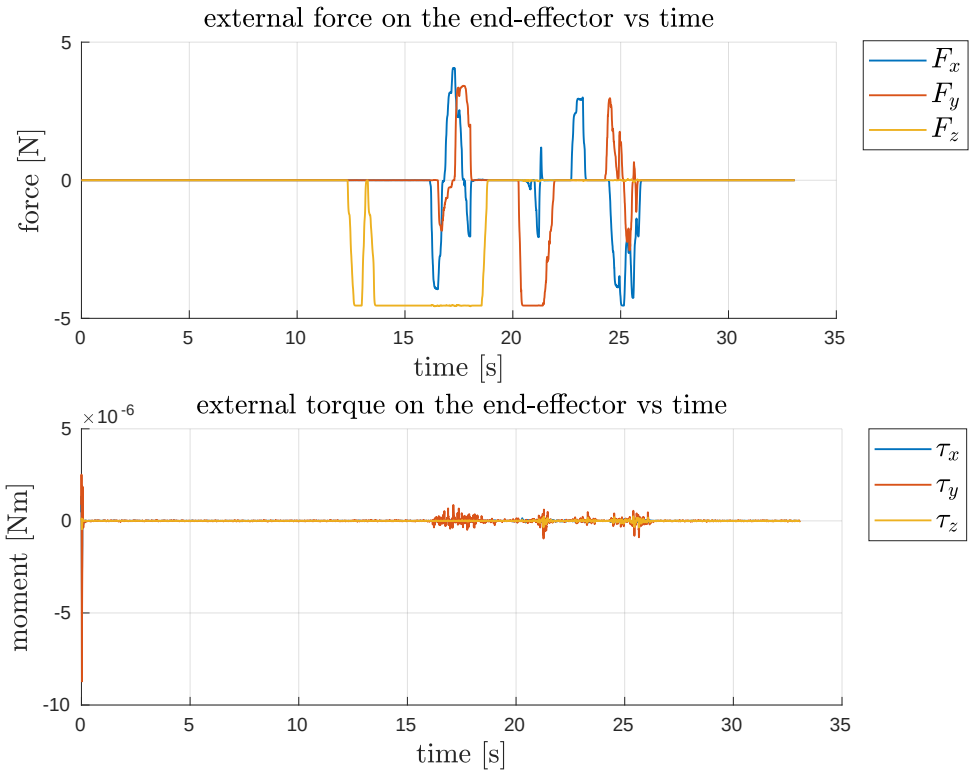


Figure 5.29: Wrench on the end-effector during admittance trajectory simulation. The first plot shows the measured forces, the second plot shows the measured torque.

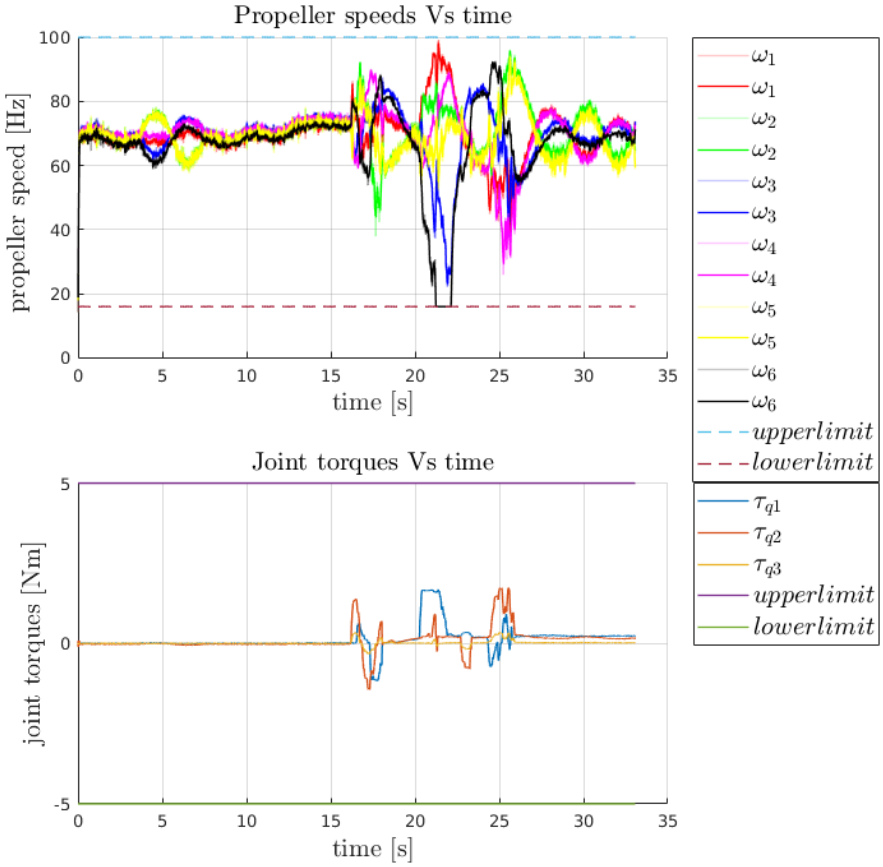


Figure 5.30: System inputs during admittance trajectory simulation. The First plot shows the angular rates of the propellers, and the second plot shows the joint torques.

5.3 Interaction space

During the interaction space simulation the aerial manipulator will be tasked to fly into the interaction space, where it will switch profile in the admittance filter. The end-effector should remain in the interaction area, where it can be freely moved around by the human operator, when the end-effector is pushed outside the interaction space, the system will push it back into the area. The interaction space is represented by the transparent green area in figure 5.31. corners of the space in (x, y)-coordinates, starting from the one closest to the origin and going counterclockwise, are: $(0,2), (3,2), (3,4), (2,4), (2,3)$, and $(0,3)$ and in the z-direction the space is between 1.5 and 2.5.

Looking at figure 5.33 it can be seen that when the end-effector moves beyond the boundaries at $19s$ the system moves back into the interaction zone. Also moving the system inside the interaction space makes the reference position move with the system.

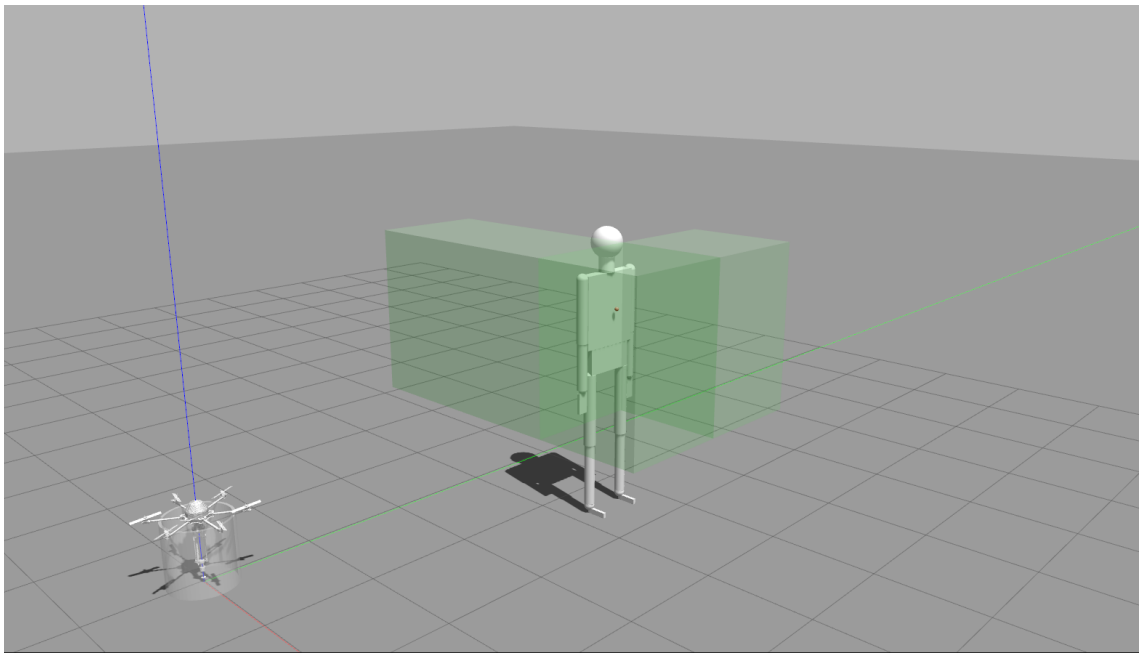


Figure 5.31: Screenshot from the interaction environment within gazebo. The transparent green area represents the interaction space

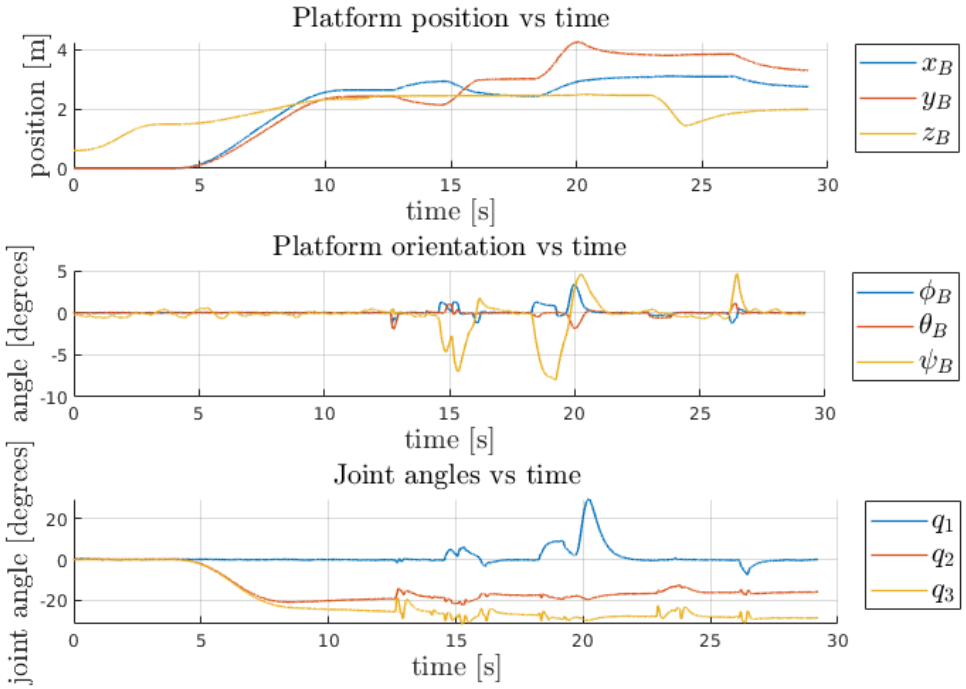


Figure 5.32: System state during interaction space simulation. The first plot shows the position of the base, the second plot the orientation of the base in roll, pitch, and yaw. The third plot shows the joint positions of the arm joints.

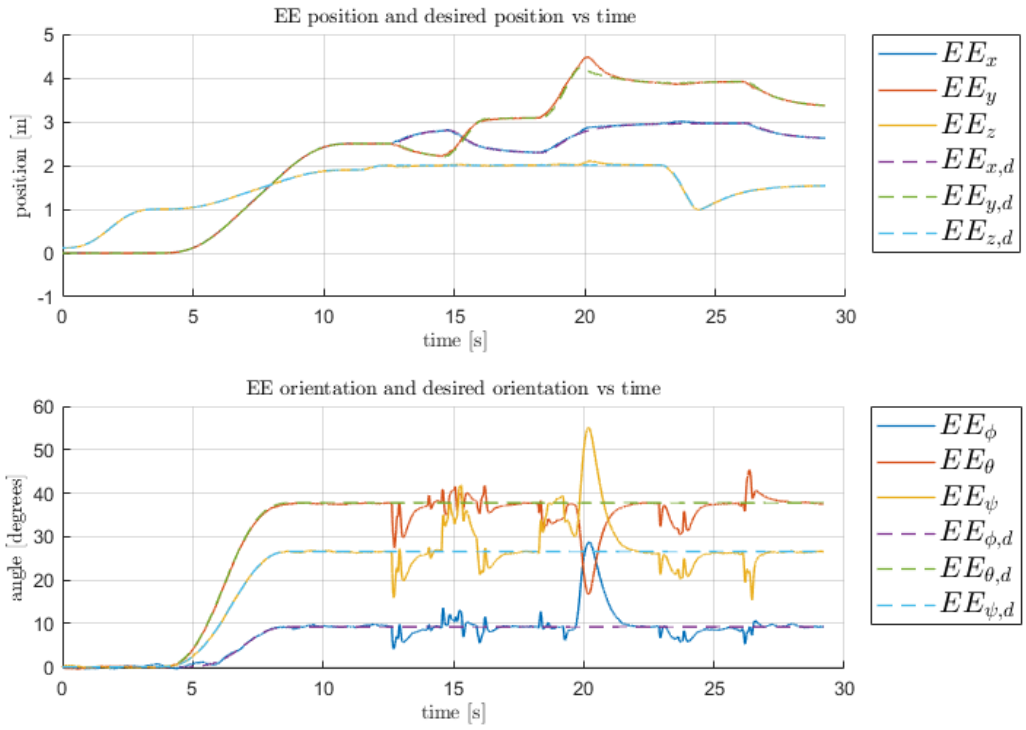


Figure 5.33: End-effector position during interaction space simulation. The first plot shows the position of the end-effector, solid line, together with the desired position, dashed line. The second plot shows the orientation of the end-effector, solid line, together with the desired orientation, dashed line.

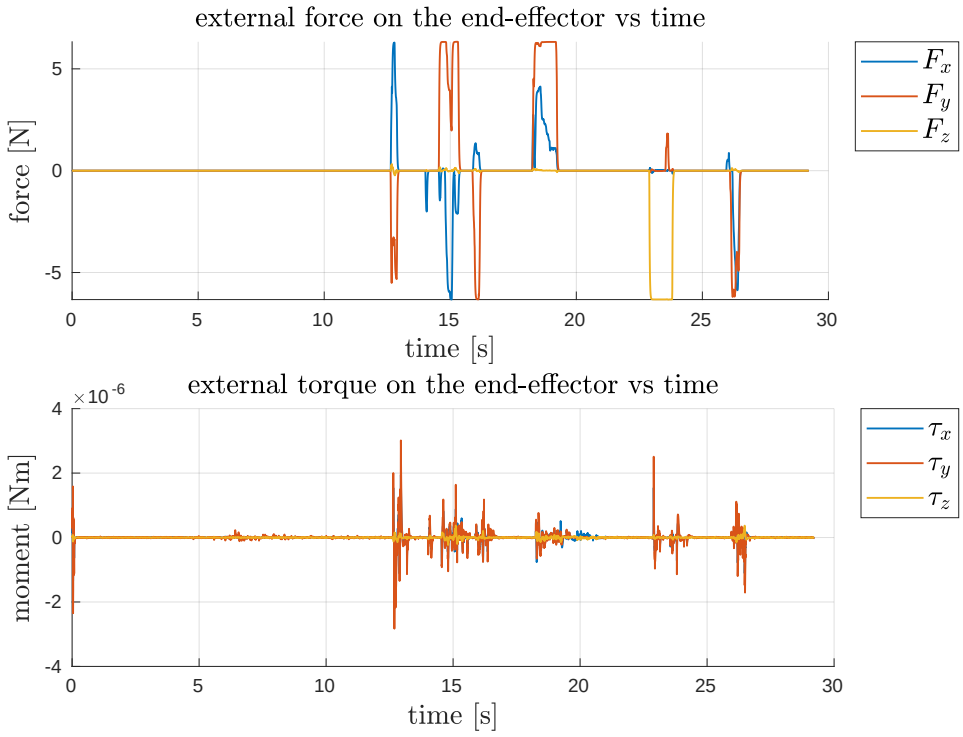


Figure 5.34: Wrench on the end-effector during interaction space simulation. The first plot shows the measured forces, the second plot shows the measured torque.

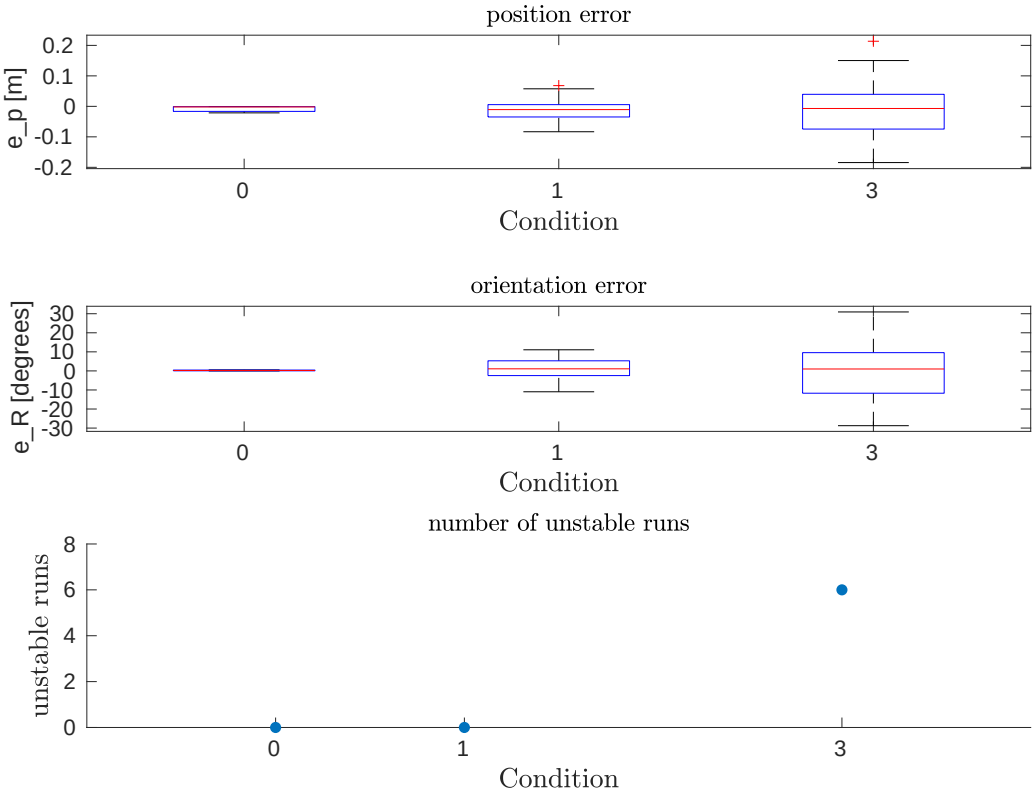


Figure 5.35: The first figure shows the box plot of the Monte Carlo simulations with the resulting position error for the different conditions k. The second figure shows the box plot of the Monte Carlo simulation with the resulting orientation errors for the different conditions k. The last plot shows the number of unstable trails for the different conditions k.

5.3.1 System robustness

In order to test the robustness of the system a Monte Carlo simulation is performed. In this simulation, different parameters can vary with around the value known by the controller. The variations are a Guassian distribution around the known value. For the mass of the body the standard deviation is: $\sigma_1 = 0.01$, meaning 68% of the masses are within $0.01kg$ of the known value. The other masses have a standard deviation of: $\sigma_2 = 0.005$. The positions of the arm joints and propellers can have a standard deviation of $\sigma_3 = 0.01$, their orientation have a standard deviation of $\sigma_4 = 0.015$. All inertias have a standard deviation of 5% of their original value. During these simulations noise will be added to all measurements, system state, force and torque measurements.

During the simulation the admittance filter is not active, and no interaction forces are applied. The system is tasked to fly in a circle, and change the orientation to $45°$, and then to $-45°$. This simulation will be run 25 times and in every run the above mentioned parameters are changed. This simulation is ran two times, ones with the above mentioned standard deviations, and once with three times as large standard deviations. A simulation where the controller model is exactly the gazebo model is used as basis.

Figure 5.35 shows the results of the Monte Carlo simulation. The different conditions are given by $k[\sigma_1\ \sigma_2\ \sigma_3\ \sigma_4\ \sigma_5]$ with $k = [0\ 1\ 3]$. These conditions represent varying degrees of modeling uncertainties. As can be seen in figure 5.35 when the uncertainty increases, so does the error and the number of unstable trails. The controller is able to handle modeling errors correlating to the first condition. Here the position error is still within $1cm$.

6 Discussion

6.1 Controller simulations

The results of the hovering and orientation trajectory simulations show that the system is able to hold the end-effector at the desired pose, when the forces at the end-effector do not exceed $5N$. Looking at the system inputs 5.8 shows this is because from this moment the propellers will saturate when more force is applied. This plot also shows that the constraints in the QP-optimizer on the propeller limits will not be violated by the system, making these hard constraints. The optimizer can however violate the tasks, it sacrifices the platform position to keep the platform orientation for example (see figure 5.6), given to the QP-optimizer.

One thing to note about how the robotic arm is controlled is that while the QP-optimizer can ask for certain joint torques, the arm is position controlled. There is a position controller (the YARP controlboard plugin) which can only give a limited rate of change of the torque on the arm. Tuning the λ value in the QP-optimizer can make the QP the limiting factor on this rate of change. Doing this will ensure that what the QP-optimizer thinks the system does, is indeed what the system does.

The redundancy resolution tries to keep the joint angles from reaching its limits, and it tries to keep the platform horizontal. Figure 5.5 shows that the platform does not tilt more than 5°, in roll and in pitch, even when some of the propellers are saturated, see figure 5.8. That the redundancy resolution keeps the joint angles as much from the joint limits as possible can best be seen in figure 5.9, here the angle required to complete the task is split over the last two joint angles. It allows joint three to bend more that joint two, because the joint limit of the former is farther away. ($-105°$ vs. $-90°$)

The ability of the redundancy resolution to change the configuration of the aerial manipulator can best be seen in zero motions simulation. Where the redundancy resolution wants to change the x-position of the platform sinusoidally. Figures 5.19 ans 5.20 shows the platform moving both in x and in y, while the position of the end-effector stays within $2cm$ of the desired position. Also the orientation of the end-effector orientation stays within 1°, while the rest of the system moves.

The results of the trajectory with disturbance simulation show that the system is able to stay within $6cm$ of the desired position, as long as the disturbance is not larger than the earlier established $5N$, where the propellers begin to saturate. When there is no disturbance, the system is however able to track the position with an average error of maximal $0.5cm$, see figure 5.3, and peak errors of maximal $4cm$ during startup as seen in figure 5.16. The orientation of the end-effector can be tracked with a maximum error of 9°.

6.2 Including the admittance filter

Including the admittance filter in the control loop, made the system behave differently. During the mass spring damper simulation, the admittance filter was tuned as stated in table 5.1, this made the aerial manipulator act like an under actuated mass spring damper system. The can best be seen in figures 5.22 and 5.23. These plots show that the reference position (admittance filter output) changes while the desired position (trajectory generator output) stays constant. As seen in figure 5.22 together with figure 5.24, when the disturbance on the system increases above $5N$ the position of the end effector will not follow the reference position anymore, but it still shows similar behavior. The orientation tracking during this large disturbance becomes worse with the admittance filter, see figure 5.6.

The admittance trajectory simulation shows that the admittance filter will still follow the desired trajectory, but will change around this trajectory according to the forces on the system. This can best be seen in figure 5.28, in the position plot. This plot shows the reference and the desired trajectory are the same if no forces are applied to the system, but when there is a force on the end-effector, the two will deviate. This shows that the admittance filter will not interfere with the trajectory from the trajectory generator, but it will impose compliant behavior while still following the trajectory.

The interaction space simulation shows how an interaction space can be defined in the admittance filter. The admittance filter will allow the aerial manipulator to be freely moved within the space, but once it leaves the space the admittance filter will pull the aerial manipulator back in. This can be seen in figure 5.33, where the system is moved around in the space and does not move back to its original position. The same plot also shows when the end-effector leaves the interaction space, it is pulled back into the space. This means that the system is compliant, and can be freely moved by the human inside the interaction space, while still trying to keep the end effector inside the interaction space.

The interaction space used in the simulation is an example of how the admittance filter can be used to get a certain behavior out of the system. The interaction space could for example also be made into a long rectangular space with a small height. This could allow for cooperatively transporting objects between the human and the aerial manipulator. As the aerial manipulator can not go down outside the interaction space it can carry a load, while the human can still make the aerial manipulator move from side to side to avoid objects.

The Monte Carlo simulation shows that the system is able to handle modeling uncertainties to some degrees. The relative (x, y, z) positions for example can vary $1cm$(for 68% of the cases) around the actual values, and the system will still be able to handle it with an error of $10cm$. The problems only arise when the error starts to increase.

7 conclusion

The goal of this report is to design a controller which will allow for human-aerial manipulator interaction. In order to achieve this goal a full pose controller is designed, which will take a desired pose of the end-effector and will send control signals to the propellers and robotic arm in order to reach this pose. The system will also have to be compliant in order for it to cooperate with the human cooperator in a natural way. The aerial manipulator used in this paper is the FiberTHex. This a fully actuated hexa-rotor, which is equipped with an 3 DoF robotic arm. This gives the aerial manipulator 9 DoF which is redundant for the pose task of the end effector.

In order to achieve this goal, a full pose controller is designed with an admittance filter to give the compliant behavior to the system. The admittance filter takes the desired trajectory of the trajectory generator and the forces measured at the end-effector, in order to shape the trajectory such that the system is compliant to the human cooperator. The full pose controller has three main parts: The first part is the PD-controller with feed-forward, which will take the position, velocity, and acceleration of the admittance filter in order to control the end effector position and orientation. This controller will output the desired linear and angular acceleration of the end effector. The second part is the inverse kinematics part. This will take the accelerations of the controller and translate the end-effector accelerations to joint space acceleration. Because the aerial manipulator is redundant, redundancy resolution is also used to give more predictable behavior to the system. The redundancy resolution is done using the projected gradient method. The third part is the QP-optimizer, this will take the joint space accelerations, and together with other tasks will calculate the propeller speeds and joint torques needed to best achieve the desired task. The QP-optimizer also imposes hard constraints on the outputs of the system.

In the end they system designed in this paper was able was able to control the pose of the end effector with a maximum error of $6cm$ and 9° with an interaction force, and an error of $0.5cm$ and 5° without disturbance. This error will increase if the sideways interaction force goes above $5N$, as from this point some propellers will start to saturate. The admittance filter was able to make the system compliant to the human cooperator. Defining an interaction space in the admittance filter allowed for the end-effector to be placed freely in the space by the human, and once the end-effector leaves the space it would be pulled back into it by the admittance filter.

8 Recommendations

The admittance filter can ask for infinitely high accelerations, and velocities, which the aerial manipulator can not do. Because of this it can be useful to limit the accelerations, and velocities the admittance filter can ask for. It can also be investigated to see if the admittance filter can be used to improve the stability of the system. As the system is now, higher interaction forces can make the system go unstable, because the propellers saturate. Using the admittance filter to change the reference trajectory with the force in such a way to keep the propellers from saturating might make the whole system more robust against disturbances.

As was shown in the interaction space simulation, an interaction space can be defined where the aerial manipulator can interact with the human cooperator. One might investigate if this can also be used to do obstacle avoidance, by excluding regions around obstacles from the interaction space.

The controller now runs on a separate computer using MATLAB/Simulink. The speed of the system can be increased by making the controller run in C++. This will also allow the controller to run on the aerial manipulator itself, which would eliminate the need of running a cable from the aerial manipulator to an off-board computer. This will give the system more freedom of movement as it is no longer restrained by the cable.

In order to measure the interaction forces at the end-effector, a force torque sensor can be used. This however adds weight to the aerial manipulator, and extra wiring. Because of this it would be beneficial to measure the interaction force using the sensors already on the platform. One way of doing this would be to design a wrench observer, which can estimate the wrench at the end-effector using the already available sensor data.

Bibliography

(2020), *MATLAB version 9.8.0.1538580 (R2020b) update 6*, The Mathworks, Inc., Natick, Massachusetts.

Bicego, D., J. Mazzetto, M. Farina, R. Carli and A. Franchi (2020), Nonlinear Model Predictive Control with Enhanced Actuator Model for Multi-Rotor Aerial Vehicles with Generic Designs, *Journal of Intelligent and Robotic Systems: Theory and Applications*, **vol. 100**, p. 1213–1247, ISSN 0921-0296, doi:10.1007/s10846-020-01250-9.

Cataldi, E., G. Muscio, M. A. Trujillo, Y. Rodriguez, F. Pierri, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini and A. Ollero (2016), Impedance Control of an aerial-manipulator: Preliminary results, in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3848–3853, doi:10.1109/IROS.2016.7759566.

De Luca, A. (2020), Kinematic redundancy 2, Robotics 2.

Ficuciello, F., L. Villani and B. Siciliano (2015), Variable Impedance Control of Redundant Manipulators for Intuitive Human–Robot Physical Interaction, *IEEE Transactions on Robotics*, **vol. 31**, pp. 850–863, doi:10.1109/TRO.2015.2430053.

Foughali, M., F. Ingrand and A. Mallet (2018), GenoM3 Templates: from Middleware Independence to Formal Models Synthesis, *ArXiv*, **vol. abs/1807.10154**.

Jimenez-Cano, A. E., J. Braga, G. Heredia and A. Ollero (2015), Aerial manipulator for structure inspection by contact from the underside, in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1879–1884, doi:10.1109/IROS.2015.7353623.

Koenig, N. and A. Howard (2004), Design and use paradigms for Gazebo, an open-source multi-robot simulator, in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pp. 2149–2154 vol.3, doi:10.1109/IROS.2004.1389727.

Metta, G., P. Fitzpatrick and L. Natale (2006), YARP: Yet another robot platform, *International Journal of Advanced Robotic Systems*, **vol. 3**, doi:10.5772/5761.

Nava, G., Q. Sablé, M. Tognon, D. Pucci and A. Franchi (2020), Direct Force Feedback Control and Online Multi-Task Optimization for Aerial Manipulators, *IEEE Robotics and Automation Letters*, **vol. 5**, pp. 331–338, doi:10.1109/LRA.2019.2958473.

de Oca, A. M., L. Arreola, A. Flores, J. Sanchez and G. Flores (2018), Low-cost multispectral imaging system for crop monitoring, in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 443–451, doi:10.1109/ICUAS.2018.8453426.

Ollero, P. A. (2019), Aerial-core objectives.
https://aerial-core.eu/objectives/

Ryll, M., D.Bicego and A. Franchi (2018), A Truly Redundant Aerial Manipulator exploiting a Multi-directional Thrust Base, 12th International IFAC Symposium on Robot Control, SYROCO 2018, SYROCO ; Conference date: 27-08-2018 Through 30-08-2018.

Ryll, M., G. Muscio, F. Pierri, E. Cataldi, G. Antonelli, F. Caccavale, D. Bicego and A. Franchi (2019), 6D Interaction Control with Aerial Robots: The Flying End-Effector Paradigm, *International journal of robotics research*, **vol. 38**, pp. 1045–1062, doi:10.1177/0278364919856694.

Seo, J., L. Duque and J. Wacker (2018), Drone-enabled bridge inspection methodology and application, *Automation in Construction*, **vol. 94**, pp. 112–126, ISSN 0926-5805, doi:https://doi.org/10.1016/j.autcon.2018.06.006.
https://www.sciencedirect.com/science/article/pii/S0926580517309755

Stramigioli, S. (2019), lecture 4, Modern robotics.

Tzoumanikas, D., F. Graule, Q. Yan, D. Shah, M. Popovic and S. Leutenegger (2020), Aerial Manipulation Using Hybrid Force and Position NMPC Applied to Aerial Writing, *CoRR*, **vol. abs/2006.02116**.
https://arxiv.org/abs/2006.02116