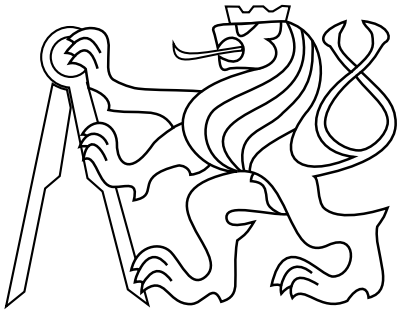
Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics
Multi-robot Systems



Nonlinear predictive control of unmanned aerial vehicle in environment with obstacles

Bachelor's Thesis

Jan Hřebec

Prague, May 2023

Study Program: Cybernetics and Robotics

Supervisor: Ing. Robert Pěnička, Ph.D.

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Hřebec Jan** Personal ID number: **498967**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Nonlinear Predictive Control of Unmanned Aerial Vehicle in Environments with Obstacles

Bachelor's thesis title in Czech:

Nelineární prediktivní řízení autonomního vzdušného robotu v prostředí s překážkami

Guidelines:

1) Study the state-of-the-art methods of Model Predictive Control (MPC) for unmanned aerial vehicles with an emphasis on approaches with time-optimal objective and collision-free flight constraints.
2) Propose and implement a new nonlinear MPC controller for real-time flight control of unmanned aerial vehicles using acados framework [1].
3) Integrate the proposed NMPC into the flight stack of MRS group and test the performance of the controller in flying along a given trajectory.
4) Compare the implemented NMPC controller with the existing control algorithms of the flight stack of MRS group.
5) Design a variant of the NMPC that avoids obstacles during flight.
6) (Optional) Test the controller on real unmanned aerial vehicles of the MRS group in the real world.

Bibliography / sources:

[1] R. Verschueren, G. Frison, D. Kouzoupis, et al., "acados—a modular open-source framework for fast embedded optimal control," Math. Prog. Comp. 14, 147–183, 2022.
[2] J. -R. Chiu, J. -P. Sleiman, M. Mittal, F. Farshidian and M. Hutter, "A Collision-Free MPC for Whole-Body Dynamic Locomotion and Manipulation," International Conference on Robotics and Automation, pp. 4686-4693, 2022.
[3] H. Nguyen, M. Kamel, K. Alexis and R. Siegwart, "Model Predictive Control for Micro Aerial Vehicles: A Survey," 2021 European Control Conference, pp. 1556-1563, 2021.
[4] M. Jacquet, M. Kivits, H. Das and A. Franchi, "Motor-Level N-MPC for Cooperative Active Perception With Multiple Heterogeneous UAVs," in IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 2063-2070, 2022.
[5] B. T. Lopez, J. -J. E. Slotine and J. P. How, "Dynamic Tube MPC for Nonlinear Systems," American Control Conference, pp. 1655-1662, 2019.
[6] J. Arrizabalaga and M. Ryll, "Towards Time-Optimal Tunnel-Following for Quadrotors," International Conference on Robotics and Automation, pp. 4044-4050, 2022.

Name and workplace of bachelor's thesis supervisor:

Ing. Robert Pěnička, Ph.D. Multi-robot Systems FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **31.01.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Ing. Robert Pěnička, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Jan Hřebec

May 26, 2023 in Prague

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my thesis supervisor, Robert Pěnička, for his invaluable guidance, professional advice, and constant support throughout the year. His patient and meticulous proofreading of my thesis, despite my last-minute approach, was especially appreciated and invaluable for the completion of this work.

I would also like to extend my sincere thanks to Parakh Gupta Manoj for his assistance with the integration into ROS, and for his very cooperative attitude throughout the year. His support during the drone testing in the real-world environment was instrumental and I am deeply grateful for it.

In addition, I would like to express my heartfelt gratitude to my family for providing a supportive environment that allowed me to fully commit to my studies. I am equally thankful to my classmates and friends, who have offered their advice and motivated me to stay focused and determined.

Lastly, I wish to acknowledge the chatGPT tool by OpenAI for its assistance with the translation into English and the rephrasing and smoothing of sentences. The service has been a great help in presenting my work in a comprehensive and articulate manner.

This endeavor has been made possible by the collective efforts and support of everyone mentioned, and for that, I am eternally grateful.

Abstract

This bachelor's thesis introduces a new Nonlinear Model Predictive Control (NMPC) for the navigation and obstacle avoidance of Unmanned Aerial Vehicles (UAVs), aimed at enhancing the controllers capabilities of the Multi-Robot Systems (MRS) group for agile flight scenarios. The thesis focuses on the design and implementation of the NMPC controller using the acados library, outlining its cost function and constraint formulation, alongside the methodology for obstacle management. To perform obstacle avoidance, drone flight is restricted to a flight corridor formed by a set of convex polyhedra, with an optimization penalty awarded for deviations from this corridor. This approach is evaluated through a series of simulation experiments that highlight the effectiveness of the proposed control architecture and demonstrate its potential to become a prominent controller in the MRS flight stack.

Keywords Unmanned Aerial Vehicles, Automatics Control, Nonlinear Model Predictive Control, Acados

Abstrakt

Tato bakalářská práce představuje nové Nelineární Modelové Prediktivní Řízení (NMPC) pro navigaci a vyhýbání se překážkám Bezpilotních Létajících Strojů (UAV), jehož cílem je rozšířit schopnosti řídicích systémů skupiny Multi-Robot Systems (MRS) pro scénáře obratných letů. Práce se zaměřuje na návrh a implementaci kontroléru NMPC pomocí knihovny acados, přičemž popisuje jeho penalizační funkci a formulaci omezení, spolu s metodologií pro zvládání překážek. Pro účely vyhýbání se překážkám je let drona omezen na letový koridor vytvořený množinou konvexních mnohostěnů, přičemž za odchylky od tohoto koridoru je v rámci optimalizace uložena penalizace. Tento přístup je vyhodnocen prostřednictvím několika simulačních experimentů, které zdůrazňují účinnost navrhované architektury řízení a ukazují jeho potenciál stát se významným kontrolérem ve vybavení MRS.

Klíčová slova Bezpilotní Prostředky, Automatické Řízení, Nelineární Prediktivní Řízení, Acados

Abbreviations

MAV Micro Aerial Vehicle

MPC Model Predictive Control

MRS Multi-robot Systems Group

NMPC Nonlinear Model Predictive Control

OCP Optimal Control Problem

PID Proportional-Integral-Derivative

RK4 Kunge-Kutta 4

ROS Robot Operating System

SQP Sequential Quadratic Programming

UAV Unmanned Aerial Vehicle

Contents

Chapter 1

Introduction

1.1 Motivation

The rapid growth and development of Unmanned Aerial Vehicle (UAV), commonly known as drones, have revolutionized various industries, including agriculture, surveillance, transportation, and photography [1]. One of the key challenges in the utilization of drones is ensuring precise and efficient flight control, especially in complex environments with obstacles and strict constraints on the drone's motion [2].

Traditional control methods, such as Proportional-Integral-Derivative (PID) controllers, may not be sufficient in achieving the desired performance for certain tasks or in challenging conditions, such as following very dynamic trajectory. Advanced control techniques, such as Model Predictive Control (MPC), have emerged as a promising alternative due to their ability to handle complex systems and incorporate constraints directly into the control formulation.

While drones can perform well using MPC, to fully exploit their agility and capabilities, it is necessary to model the nonlinear behavior of the drone using Nonlinear Model Predictive Control (NMPC). NMPC is a specific variant of MPC that deals with nonlinear systems, making it a suitable candidate for the control of drones, which exhibit nonlinear dynamics. The implementation of an NMPC controller using the acados library [3] is an approach that can potentially yield superior performance and enhanced robustness compared to traditional methods.

Furthermore, obstacle modeling is essential when the predefined trajectory passes close to obstacles, as disturbances or shortening of path while tracking of the reference point in high speeds may cause the drone to collide. Incorporating obstacle avoidance into the NMPC controller using linear constraints is a aspect of drone control, that enables safe navigation along trajectories within proximity to obstacles and reduces the risk of collision.

This thesis aims to develop and implement an NMPC controller for drone flight control along a predefined trajectory, taking into account obstacle modeling and avoidance, using the acados library. The motivation behind this research is to enhance the flight control performance of drones, enable agile flight maneuvers, and contribute to the advancement of control techniques used in UAV applications.

1.2 Problem statement

Current drone flight control systems, such as Model Predictive Control (MPC), face limitations in handling high-speed and dynamic maneuvers due to the linearization of the system, which can lead to inaccuracies when the operating point is far from the linearization point. This issue affects the drone's ability to perform agile maneuvers and maintain trajectory

tracking performance while navigating close to obstacles. NMPC is also better for handling actuation limits such as the motor max thrust.

Building upon the limitations of current NMPC controllers, it is important to note that existing solutions often do not consider obstacles in their formulations [2]. This omission can lead to a conservative flight strategy, requiring the drone to closely follow the predefined trajectory in the vicinity of obstacles to prevent collisions. In areas without obstacles, this conservative approach may negatively impact the drone's flight speed and limit its potential for agile maneuvers.

The problem this bachelor's thesis will address is the need to create a robust controller for the MRS group, capable of tracking very fast trajectories and accounting for obstacles.

1.3 Objectives and scope of the thesis

The primary objective of this thesis is to develop an NMPC controller that accurately captures the nonlinear dynamics of a drone and is suitable for flight control along a predefined trajectory. This will involve formulating the NMPC controller and implementing it using the acados library [3], ensuring real-time performance and efficient computations.

Another key objective is to incorporate obstacle modeling and avoidance in the NMPC controller. This will allow the drone to safely navigate in environments with obstacles while maintaining trajectory tracking performance. The performance of the developed NMPC controller will be assessed in various simulation scenarios, and its effectiveness and robustness will be compared with alternative control methods.

As an optional objective, the study will explore the possibility of conducting a proof-of-concept flight test on a real drone to demonstrate the feasibility and real-world performance of the NMPC controller. This will provide insights into potential limitations and areas for improvement in the proposed control approach.

The scope of this study is limited to the development and implementation of the NMPC controller using the acados library and the integration of obstacle modeling and avoidance. The primary focus will be on simulations, with the optional real-world flight test serving as a supplementary validation of the proposed control approach.

Chapter 2

State Of The Art

2.1 Model predictive control in drone control

MPC has emerged as a prominent control strategy for Micro Aerial Vehicle (MAV)s, particularly in multirotor configurations such as quadrotors [2]. The success of MPC is attributed to it being a model-based method, capable of exploiting the knowledge of the dynamic model of the system. Due to recent advancements, MPCs are applicable to both linear and nonlinear systems. By optimizing over the prediction horizon, MPC can track the reference and simultaneously satisfy constraints on the input while maintaining robust performance. Constraints can also include spatial constraints, thereby modeling obstacles and forbidden areas.

Another approach to achieve time-optimal flight involves the use of Model Predictive Contouring Control (MPCC) [4] [5]. This method is minimizing the Euclidean distance from a curve (as opposed to deviation from a sampled state at a shooting node in standard MPC) and maximizing the distance along the trajectory. This results in fast progress along given trajectory, while maintaining good trajectory tracking.

2.2 Existing NMPC implementations and applications

NMPC has become a popular choice for controlling drones due to its ability to handle nonlinear dynamics more effectively than linear MPC. NMPC has been successfully implemented in various drone applications, demonstrating its potential for enhancing flight performance and safety.

M. Jacquet et al. [6] introduces an application of NMPC for a group of UAVs aimed at optimizing observation capacity. The strategy employs a cooperative control structure which integrates a comprehensive nonlinear model of multirotor UAVs. This model considers their motor-level actuation elements and real-world constraints. Every UAV in the group is managed by its individual NMPC, which predicts the feature position utilizing a Kalman filter and shares these forecasted measurements with the rest of the group. Subsequently, each NMPC processes these aggregated measurements as external parameters for updating their internal Kalman filters, thereby enhancing their collective performance.

Recalde et al. [7] introduces a NMPC for rapid trajectory tracking in dynamic environments, particularly designed for hexacopter platforms. This control strategy optimizes a policy that guides the hexacopter along a pre-defined trajectory, simultaneously ensuring it evades obstacles. To achieve this, the authors developed a simplified dynamic model of the hexacopter, incorporating low-level PID schemes to enforce velocities generated by high-level NMPC controller. The obstacles were modeled as spherical constrains in the optimization

problem. This problem was solved using the CasADI optimization toolkit, ensuring fast computational times. The results demonstrated effective trajectory tracking and obstacle avoidance in a simulated environment. Despite external disturbances such as wind, the control errors tended towards zero, and the computational time remained consistently under 100 ms (which was the controller call period), showing the robustness and efficiency of the developed NMPC scheme.

2.3 Obstacle modeling and avoidance in drone control

Although the common procedure for collision-free flight is to plan a collision-free trajectory and then use a conventional control system, which does not require obstacle modelling, obstacle modelling can enhance safety and reliability in flight near obstacles. Various techniques have been developed to address these challenges, and this section highlights some of the prominent methods.

Although method [8] is designed for legged mobile robots, it employs MPC that takes into account obstacles. These obstacles are modeled using spheres, and distance to these spheres is penalized using a relaxed barrier function. This approach to modeling obstacles could potentially be applicable to the control of UAVs as well.

Another obstacles modelling strategy involves constraining the flight of UAVs to a tube along a trajectory. By incorporating uncertainties into the formulation of the MPC method, this approach, together with dynamic replanning of the tube, ensures system robustness even when confronted with previously unknown obstacles and state-dependent uncertainty [9]. Alternatively, this tube can be approximated using polyhedrons, and hard linear constraints can be employed to achieve time-optimal flight within the designated tube [10].

Chapter 3

Methodology

In this bachelor's thesis, the control of UAVs is adressed, specifically quadcopters, using NMPC. To achieve this, we require a state description of the quadcopter, a specified reference trajectory, and a method of calculating the cost function for the optimization algorithm.

3.1 Drone dynamics and mathematical models

For the state description, consider the quadcopter's position, velocity, orientation, and angular velocity as the state variables. An overwiew of used variables is in Table 3.1. The position is represented by a 3-dimensional Cartesian vector $\mathbf{p}$ in meters, while the velocity is a 3-dimensional vector $\mathbf{v}$. The orientation is represented by a quaternion $\mathbf{q}$, and the angular velocity is represented by a 3-dimensional vector $\boldsymbol{\omega}$ in the body frame. The inputs to the state model are the thrusts of the individual propellers T_i, which form a vector $\mathbf{T}$ of 4 values for the quadcopter in order shown in Figure 3.1.

As system parameters, we have the mass of the drone, the inertia tensor $\mathbf{I}$ (a 3×3 matrix), and an allocation matrix $\mathbf{A}$ that relates the thrusts of the propellers to the moments about the body axes. The allocation matrix is dependent on the length of the propeller arms l and the torque coefficient c_τ. This torque coefficient indicates the ratio between the thrust of the propeller and the moment of force that opposes the propeller's rotation.

Table 3.1: Table of used variables

Symbol	dimension	units	description
$\mathbf{p}$	3	m	position
$\mathbf{v}$	3	m/s	velocity
$\mathbf{q}$	4	-	rotation quaternion
$\boldsymbol{\omega}$	3	rad/s	angular velocity
$\mathbf{T}$	4	N	thrusts
$\mathbf{x}$	13	-	drone state $(\mathbf{p}, \mathbf{v}, \mathbf{q}, \boldsymbol{\omega})$
m	1	kg	quadcopter mass
l	1	m	length of arm
$\mathbf{I}$	3×3	kg m^2	inertia tensor
$\mathbf{A}$	3×4	m	allocation matrix
c_τ	1	m	torque coefficient
c_d	1	s^{-1}	air drag coefficient
g	1	m/s^2	gravity acceleration

The allocation matrix defines the relationship between the thrusts of the individual

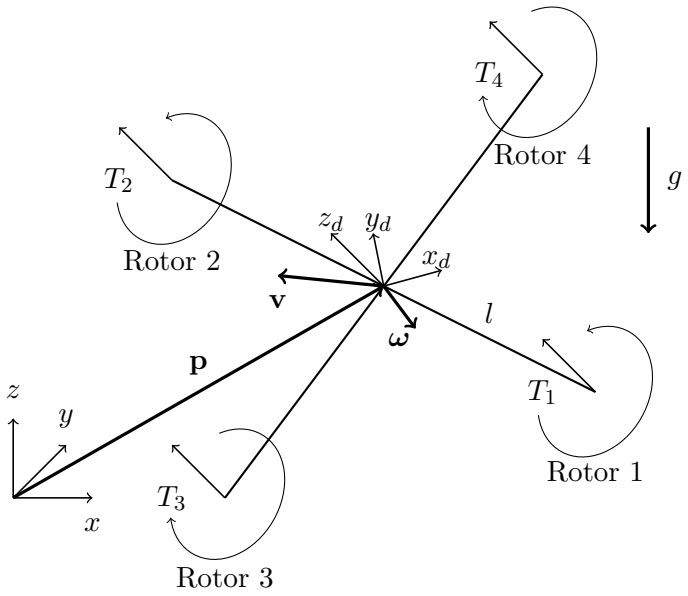


Figure 3.1: Schema of drone

motors and the moment of force acting on the quadcopter. It has the following form:

$$\mathbf{A} = \begin{bmatrix} -\frac{l}{\sqrt{2}} & \frac{l}{\sqrt{2}} & -\frac{l}{\sqrt{2}} & \frac{l}{\sqrt{2}} \\ -\frac{l}{\sqrt{2}} & \frac{l}{\sqrt{2}} & \frac{l}{\sqrt{2}} & -\frac{l}{\sqrt{2}} \\ -c_\tau & -c_\tau & c_\tau & c_\tau \end{bmatrix}. \tag{3.1}$$

The state model can be expressed in the following form:

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{3.2}$$

$$\dot{\mathbf{v}} = \mathbf{q} \circ \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} T_i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} - \mathbf{v} c_d, \tag{3.3}$$

$$\dot{\mathbf{q}} = 0.5 \cdot \mathbf{q} \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}, \tag{3.4}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{I}^{-1} \left(\mathbf{A}\mathbf{T} - \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} \right). \tag{3.5}$$

Where the $\circ$ operation represents rotation of vector by rotation quaternion. All symbols are explained in Table 3.1.

3.1.1 Delay and custom control command

The proposed NMPC controller produces motor thrusts T_n. This is important, because motor thrust constraints must be in the optimization constrains to achieve truly agile flight. But drones of the MRS group [11] have a pixhawk flight controller, which has collective thrust and body rates as the lowest-level command. This pixhawk flight controller has its own attitude rate PID, which run on higher frequency and commands power to individual rotors.

Therefore it's necessary to convert the thrusts of individual propellers into the collective thrust and attitude rate. Also the feedback loop delay was modeled to improve controller performance.

For modeling the delay, the last N values of motor thrusts $\mathbf{T}_n$ are stored in the cycle buffer. Before each call of the Optimal Control Problem (OCP) solver, the observed state is modified N times using the Kunge-Kutta 4 (RK4) method

$$\begin{aligned} &\text{N times repeat:} \\ &\mathbf{x} \leftarrow \text{RK4}\left(\mathbf{x}, \mathbf{T}_n, t_p\right), \end{aligned} \tag{3.6}$$

t_p is the period of controller calls.

After receiving the optimal thrust of the four propellers from NMPC, the total thrust is obtained simply by summing them. The attitude rate is obtained by simulating the current state with RK4 using the obtained thrust for a time t, which should roughly correspond to t_p, but it is precisely determined by fine-tuning.

3.2 Reference trajectory

It is assumed that the reference trajectory is already optimized according to certain parameters and is continuously differentiable. Thus the position, velocity, acceleration and jerk can be determined for each time step. From this acceleration, we can calculate the reference orientation except yaw and the reference thrust of the motors. Yaw can be given explicitly or simply set to zero. And from jerk and derivation of given yaw, angular speed can be calculated.

This all can be done, because drone system is differentially flat in its position and heading [12]. From these parameters and their derivatives all other state variables and system input can be computed.

Therefore, the reference trajectory is given by the function $f_{\text{traj}}(t) \rightarrow \mathbf{x}, \mathbf{T}$

3.2.1 Trajectory optimizer

For this work, the Large Scale Trajectory Optimizer [13] was chosen as a trajectory generator. This is a header library written in C language. It is capable of finding a trajectory with minimum jerk or snap for given sequence of points in a real time. This library operates by dividing the trajectory into individual segments between specified points. These segments are expressed as n-th order polynomials. It then imposes a continuity condition, stating that the polynomials at the junction points must have the first k derivatives identical (k=3 for jerk minimization, k=4 for snap minimization). Subsequently, it searches for coefficients for each polynomial that would minimize the integral under the above conditions:

$$\min \int_{t_0}^{t^m} \| \frac{d^k \mathbf{p}}{dt^k} \|^2 dt. \tag{3.7}$$

The library then transforms this problem into a Quadratic Program and, by solving it, obtains the desired trajectory. According to the desired tracking speed, it assigns time intervals for individual trajectory segments.

3.3 NMPC formulation and optimization

The main goal is to formulate an OCP, which can be solved by acados [14]. An OCP consists of finding the individual motor thrusts over time so that the deviation from the

desired trajectory is minimized

$$\min_{\mathbf{u}(t)} \quad \int_{t_0}^{t_e} f_{\text{cost}}\left(\mathbf{x}(\tau), \mathbf{u}(\tau), \mathbf{y}_{\text{ref}}(\tau)\right) d\tau \tag{3.8}$$

$$\text{subject to} \quad \dot{\mathbf{x}} = \mathbf{f}_{\text{dyn}}(\mathbf{x}, \mathbf{u}) \tag{3.9}$$

$$\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}.$$

Where $\mathbf{y}_{\text{ref}}(\tau)$ is reference state in time τ, $\mathbf{x}(\tau)$ is actual state in time τ (in our case it is composed by $\mathbf{p}, \mathbf{v}, \mathbf{q}$ and $\boldsymbol{\omega}$), $\mathbf{u}(\tau)$ are inputs to the system in time τ (in our case it is the vector $\mathbf{T}$ of all four thrusts), f_{cost} is cost function and $\mathbf{f}_{\text{dyn}}$ is a function describing dynamic model of system (in our case described by equations (3.2) - (3.5)). $\mathbf{u}_{\min}$ and $\mathbf{u}_{\max}$ are input constraints (for drone control $\mathbf{u}_{\min} = 0$ and $\mathbf{u}_{\max}$ is equal to maximum thrust of individual propellers).

3.3.1 Acados implementation

The acados library offers several possible forms of solution [14], each of which has certain advantages and requirements. The Nonlinear Least Squares was chosen to leverage the nonlinearity of the system. As a result, the $\mathbf{f}_{\text{dyn}}$ function can be any differentiable function, but f_{cost} must be in the quadratic form:

$$f_{\text{cost}}\left(\mathbf{x}, \mathbf{u}, \mathbf{y}_{\text{ref}}\right) = \frac{1}{2}\|\mathbf{Q}\mathbf{x} + \mathbf{R}\mathbf{u} - \mathbf{y}_{\text{ref}}\|^2, \tag{3.10}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are weight matrices, which map drone state and thrusts onto reference.

The integral (3.8) is solved by discretization to N time steps (shooting nodes), where $\mathbf{u}(\tau)$ is in each time step constant, approximating (3.9) by RK4. Acados employs a Sequential Quadratic Programming (SQP) method for optimization [3]. This approach iteratively linearizes the nonlinear problem, solving the resulting quadratic problem to compute an update of $\mathbf{u}(\tau)$. It is well-suited for nonlinear problems, allowing acados to efficiently compute optimal control inputs $\mathbf{u}(\tau)$ at each time step.

3.3.2 Quaternion error decomposition

It is favorable to assign different weights for controlling the drone's rotation around the z axis (yaw) and around the x and y axes (pitch and roll), because drone dynamics around z axis is much slower compared to x and y. We decompose the drone's deviation from the reference quaternion into the x, y, and z axes.

To do this we find error quaternion $\mathbf{q}_e$ as the shortest rotation between attitude quaternion $\mathbf{q}$ and reference $\mathbf{q}_r$ as

$$\mathbf{q} = \mathbf{q}_e \cdot \mathbf{q}_r, \tag{3.11}$$

$$\mathbf{q}_e = \mathbf{q} \cdot \mathbf{q}_r^{-1}. \tag{3.12}$$

$$\tag{3.13}$$

Then we decompose $\mathbf{q}_e$ to xy and z axes

$$\mathbf{q}_e = \mathbf{q}_z \cdot \mathbf{q}_{xy}, \tag{3.14}$$

$$\begin{bmatrix} q_{ew} \\ q_{ex} \\ q_{ey} \\ q_{ez} \end{bmatrix} = \begin{bmatrix} q_{wz} \\ 0 \\ 0 \\ q_z \end{bmatrix} \cdot \begin{bmatrix} q_{wxy} \\ q_x \\ q_y \\ 0 \end{bmatrix}. \tag{3.15}$$

By multiplying these quaternions, we obtain (only the first and last element of quaternion is important):

$$\begin{bmatrix} q_{ew} \\ \vdots \\ q_{ez} \end{bmatrix} = \begin{bmatrix} q_{wz}q_{wxy} \\ \vdots \\ q_z q_{wxy} \end{bmatrix} = q_{wxy} \begin{bmatrix} q_{wz} \\ \vdots \\ q_z \end{bmatrix}. \tag{3.16}$$

From this the rotation around z axis $\mathbf{q}_z$ is found (because $\|\mathbf{q}_z\| = 1$)

$$\begin{bmatrix} q_{wz} \\ 0 \\ 0 \\ q_z \end{bmatrix} = \begin{bmatrix} q_{ew} \\ 0 \\ 0 \\ q_{ez} \end{bmatrix} \frac{1}{\sqrt{q_{ew}^2 + q_{ez}^2}}. \tag{3.17}$$

Finaly from equation (3.14) we can compute $\mathbf{q}_{xy}$ and find minimal error coeficients:

$$\begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} = \frac{1}{\sqrt{q_{ew}^2 + q_{ez}^2}} \begin{bmatrix} q_{ew}q_{ex} + q_{ez}q_{ey} \\ q_{ew}q_{ey} - q_{ez}q_{ex} \\ q_{ez} \end{bmatrix}. \tag{3.18}$$

3.4 Linear constraints for obstacle modeling

In order to stick with Nonlinear Least Squares (in order to keep f_{cost} in format (3.10)), a reasonable option for modeling obstacles is to use linear constraints with the use of slack variables. Acados allows [14] adding the following linear constraints to the optimization problem:

$$\min_{\mathbf{u},\mathbf{s}} \quad \int_{t_0}^{t_e} f_{\text{cost}}\left(\mathbf{x}(\tau), \mathbf{u}(\tau), \mathbf{y}_{\text{ref}}(\tau)\right) + \frac{1}{2} \begin{bmatrix} \mathbf{s}(\tau) \\ 1 \end{bmatrix}^T \begin{bmatrix} \mathbf{Z} & \mathbf{z} \\ \mathbf{z}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{s}(\tau) \\ 1 \end{bmatrix} d\tau \tag{3.19}$$

$$\begin{aligned} \text{subject to} \quad & \dot{\mathbf{x}} = \mathbf{f}_{\text{dyn}}(\mathbf{x}, \mathbf{u}) \\ & \mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max} \\ & \mathbf{g}_l \leq \mathbf{C}\mathbf{x} + \mathbf{J}\mathbf{s} \\ & \mathbf{s} \geq 0, \end{aligned} \tag{3.20}$$

where $\mathbf{s}(\tau)$ are slack variables, $\mathbf{C}$ is a matrix with linear equation coefficients, $\mathbf{g}_l$ is a vector containing lower bounds for linear constrain inequality. Matrix $\mathbf{J}$ is assigning slack variables to individual inequalities (in this case it is identity matrix). The matrix $\mathbf{Z}$ is a diagonal matrix that contains weights used to penalize quadratic terms related to slack variables. Similarly, the vector $\mathbf{z}$ is a weight vector used to penalize linear terms associated with slack variables. Matrix $\mathbf{C}$ and vector $\mathbf{g}_l$ can be set individually for each time stamp in prediction horizon.

The inequality (3.20) describes these linear constrains:

$$ax + by + cz + d \geq 0. \tag{3.21}$$

Coefficients a, b and c ($a^2 + b^2 + c^2 = 1$) are in elements of the matrix $\mathbf{C}$ corresponding with position states, and $-d$ is in vector $\mathbf{g}$. In order to omit this linear constraint, we set a = b = c = d = 0. Consequently, this inequality is satisfied for every position of the drone.

These linear constraints are soft, meaning that their violation only results in an increase of the minimized function. Consequently, the resulting optimal trajectory of the drone may pass through the edge of the forbidden area. Therefore, it is necessary to place the linear constraints at a certain distance from the obstacle.

The number of linear constraints, which corresponds to the maximum number of faces of a polyhedron, was set to 6. This decision was made because a lower number of constraints might not be sufficient for modeling restrictions in enclosed spaces, while a higher number would result in increased computational complexity. However, this number can be adjusted according to the user's needs.

These linear constraints are assigned individually for each reference point in the prediction horizon. For each of these points is determined a polyhedron in which the point is situated most inward. This can be seen in figure 3.2. As linear constrains for points 1, 2 and 3 is used polyhedron A, for points 4 and 5 is used polyhedron B and for points 6 and 7 is used polyhedron C.

Theoretically, proposed algorithm can be modified to consider moving obstacles. This could be accomplished by making these linear constraints time-dependent. When assigning to individual shooting nodes, the corresponding time-dependent linear constraints would be used. This adjustment could potentially improve the algorithm's versatility and applicability in dynamic environments with moving obstacles.

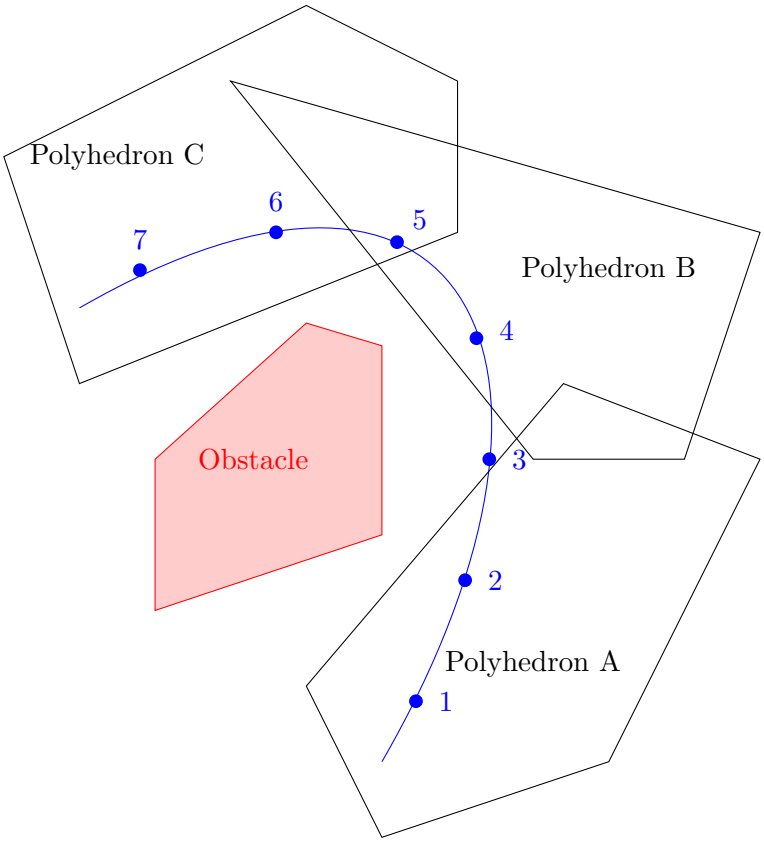


Figure 3.2: Example of flight corridor composed of convex polyhedrons. Blue line is reference trajectory and points on it are shooting nodes of NMPC.

Chapter 4

Simulation and results

4.1 Simulation setup and environment

The controller was tested in three different environments: firstly, in a simple environment that was implemented for purposes of this thesis, then in Gazebo [15], and finally on a real drone.

The simple environment was a simulation that simulated the system's behavior in discrete time steps of 10 ms using the RK4 method. The inputs to the system were directly the motor commands. This simulation did not model any input or output disturbances, nor did it simulate signal delays.

The Gazebo environment was used within a Singularity container [16]. The communication between the controller and the environment was facilitated through Robot Operating System (ROS) [17]. The controller receives the drone's state and the desired trajectory as inputs and returns the total motor thrust and desired drone angular velocity. The simulation of the drone flight and the calculation of the optimal control action occur asynchronously, and the simulation models noise in both the input and output. In this simulation, the control loop delay is similar to the delay experienced on a real drone.

The drone T650 [18], which has a mass of 3.5 kg and is capable of generating a total thrust of 92 N, was utilized for a comparative analysis of the controllers. A drone with identical parameters was used in the RK4 simulation. For testing of point tracking (Figure 4.1 and 4.2) and for real world deployment we used drone F450 [18], which has a mass of 1.9 kg and a total thrust of 32 N. For flights in Gazebo and in real-world flight a controller was used with parameters according to Table 4.1. The Q and R parameters are weights in diagonal matrices $\mathbf{Q}$ and $\mathbf{R}$ from equation (3.10) in corresponding positions. For flights in RK4 used parameters were according to Table 4.2. The parameter Z is value of diagonal of matrix $\mathbf{Z}$ from integral (3.19) and z is value of each element in vector $\mathbf{z}$ in the same integral.

Table 4.1: Parameters of NMPC controller used in Gazebo.

parameter	value
Number of shooting nodes	30
Prediction horizon	2 s
Q position xy	0.5
Q position z	1
Q rotation xy	0.2
Q rotation z	3
Q velocity	0.2
Q omega	0.5
R thrust	0

Table 4.2: Parameters of NMPC controller used in simple RK4 simulation.

parameter	value
Number of shooting nodes	30
Prediction horizon	2 s
Q position xy	100
Q position z	100
Q rotation xy	0
Q rotation z	0
Q velocity	0
Q omega	0
R thrust	0
Z	0
z	0

4.2 Results and analysis of the NMPC performance

4.2.1 Time requirement

In acados, two hyperparameters can be adjusted: the Prediction horizon and the number of shooting nodes. The Prediction horizon specifies how far into the future we look - this corresponds to the integration length $t_e - t_0$ from equation (3.8). The larger the prediction horizon, the further into the future the drone can see, but it also results in more dispersed shooting nodes, which may impair the tracking of rapid dynamics. The number of shooting nodes determines how finely the integral (3.8) will be approximated. A larger number accelerates the dynamics of the control, but also increases the computational demand. The following Table 4.3 and Table 4.4 shows the dependence of the average and maximum computation time on the number of shooting nodes. This results were achieved on PC with processor AMD Ryzen 5 5500U (6 cores 2.1 GHz) and 8 GB RAM.

Table 4.3: Table of required computation time with a prediction horizon of 1 second.

Shooting nodes	average time [ms]	max time [ms]
20	1.61ms	5.4ms
30	2.74ms	9.7ms
40	3.31ms	12.2ms
50	4.32ms	14.2ms

Table 4.4: Table of required computation time with a prediction horizon of 2 second.

Shooting nodes	average time [ms]	max time [ms]
20	1.34	5.2
30	2.26	10.1
40	3.03	13.6
50	4.32	16.5

From these tables, it can be seen that the time complexity depends almost linearly on the number of shooting nodes and does not depend on the prediction horizon.

When the OCP was adjusted to include obstacles (3.19), the time complexity increased. With 6 linear constraints, the time complexity was more than double, as can be seen in table 4.5.

Table 4.5: Table of required time with prediction horizon 1 second with and 6 linear constrains.

Shooting nodes	average time [ms]	max time [ms]
10	2.16	3.5
20	3.94	9.9
30	5.24	16.5
40	6.87	25.2
50	8.67	27.4

Such high computational complexity is limiting for larger numbers of shooting nodes when deploying on real drones, but for a smaller number of shooting nodes, it is usable.

To make this approach more usable, we can take into account the fact that usually not all linear constraints in a convex polyhedron are limiting for a particular shooting node, and therefore we can consider only the 3 closest linear constraints, thereby reducing the time complexity 4.6.

Table 4.6: Table of required time with prediction horizon 1 second with 3 linear constrains.

Shooting nodes	average time [ms]	max time [ms]
10	1.29	4.7
20	2.67	9.5
30	4.18	15
40	5.7	21
50	6.8	20

4.2.2 Point tracking

Firstly, the accuracy of point tracking (That means that the reference trajectory has all positions in one point.) was tested, to see the precision of tracking, and the controller's response to a reference jump to assess the speed of return to the desired position after deviation.

Point reference tracking in simple simulation is accurate (the error is at the level of rounding error). Therefore, the investigation of point reference tracking is done in the Gazebo

simulator. This is shown in Figure 4.1. The controller's task is to track the position (5,0,5). However, my controller makes a slight systematic error, likely due to a fact, that we do not model the motor dynamics, or mismatch between the real parameters of the drone and the model, or the drone has some other unmodeled feature.

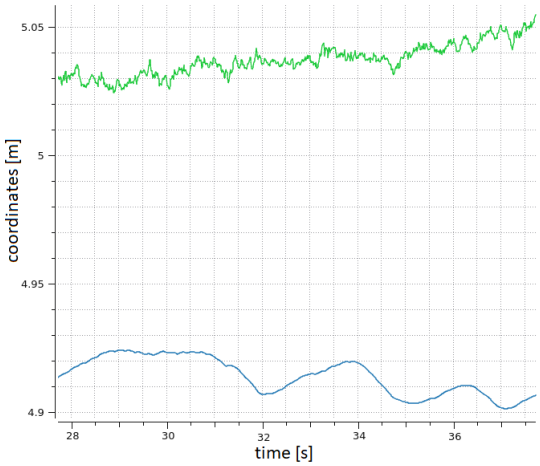


Figure 4.1: Coordinates x (blue) and z (green) of drone hovering with my NMPC.

Subsequently, the response to a reference step, that can be seen in Figure 4.2, was tested. The aforementioned issue is even more apparent in this case.

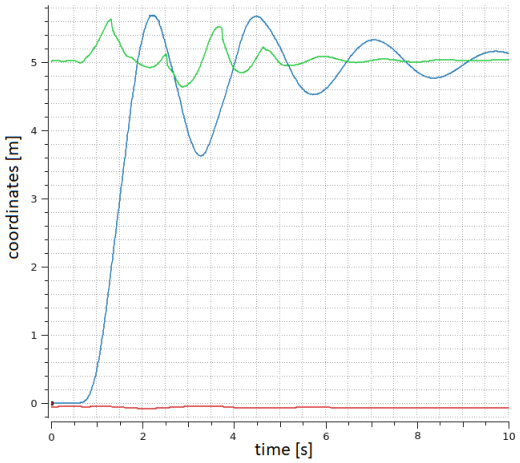


Figure 4.2: Coordinates x (blue), y (red) and z (green) of drone responding to step change of reference with proposed NMPC in Gazebo.

When testing the response to a step in the reference in my simple simulator, the obtained response can be seen in Figure 4.3. No oscillations are visible here, so there is no fundamental error in the controller.

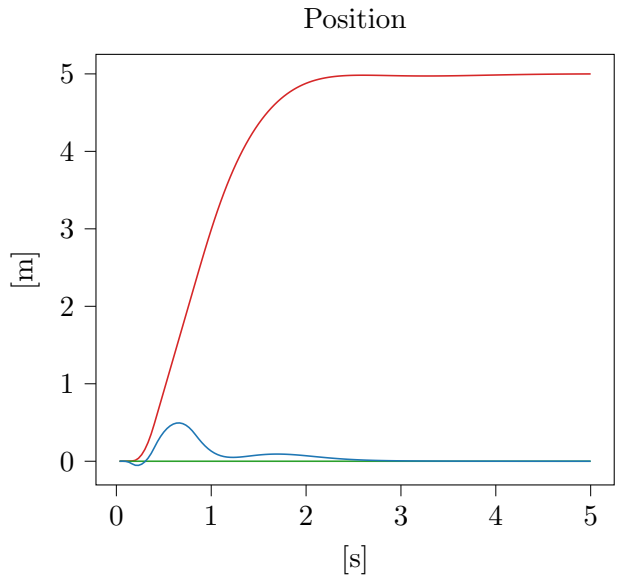


Figure 4.3: Coordinates x (red), y (green) and z (blue) of drone responding to step change of reference with proposed NMPC in simple simulation.

4.2.3 Trajectory tracking

To verify trajectory tracking, a trajectory was generated in my simulator using the Large Scale Trajectory Optimiser [13]. Minimum-jerk trajectory was calculated and the transit points was set as per Table 4.7. The produced trajectory can be seen in Figure 4.4. The speed parameter was adjusted so that the requirement was to traverse this trajectory in 6 seconds.

Table 4.7: Transit points used for trajectory generation in meters.

x	y	z
0	0	0
10	0	0
10	10	0
0	10	0
0	10	10
0	10	20
10	10	20

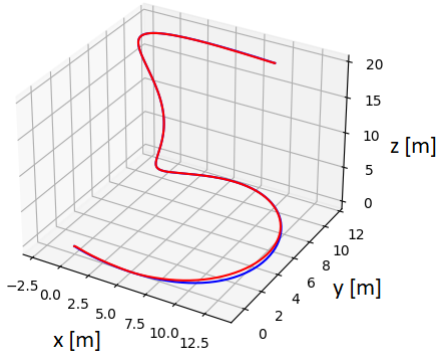


Figure 4.4: 3D plot of reference trajectory (blue) and flown path (red).

The flight has high dynamics where maximum achieved speed was 60 km/h (Figure 4.6), despite this the deviation from the reference points on the trajectory remained within 0.5 meters (Figure 4.5), even though reference trajectory was not tailor-made for this specific drone.

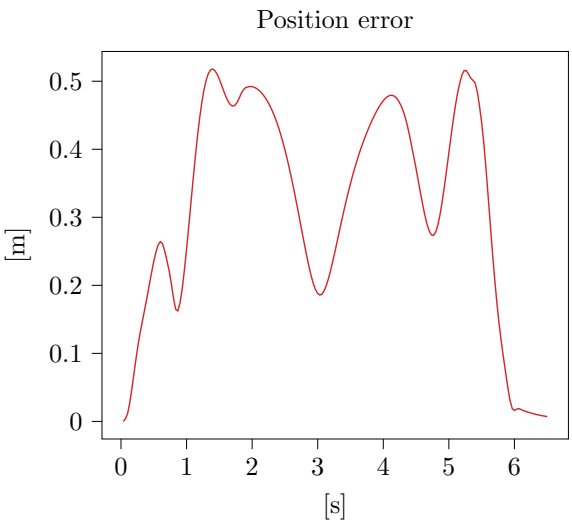


Figure 4.5: Time course of position error size

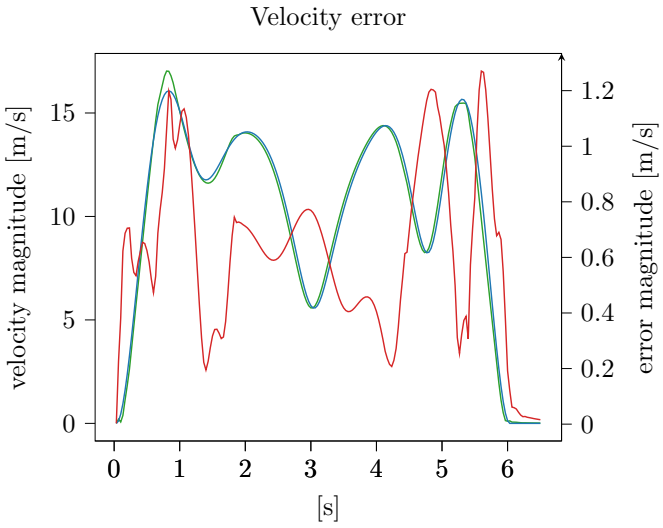


Figure 4.6: Velocity magnitude: reference (blue), actual (green) and error (red - axis on right).

4.2.4 Obstacle avoidance

To test obstacle avoidance, the drone was instructed in simple simulator to follow a trajectory that passes through the edges of the penalized area, and the trajectory tracking was set to be fast (to fly through whole trajectory in 7 seconds). The top down view can be seen in Figure 4.7 and the magnitude of position error is in Figure 4.8. I set up 4 convex polyhedra that formed a flight corridor. Their complement constituted the penalized area.

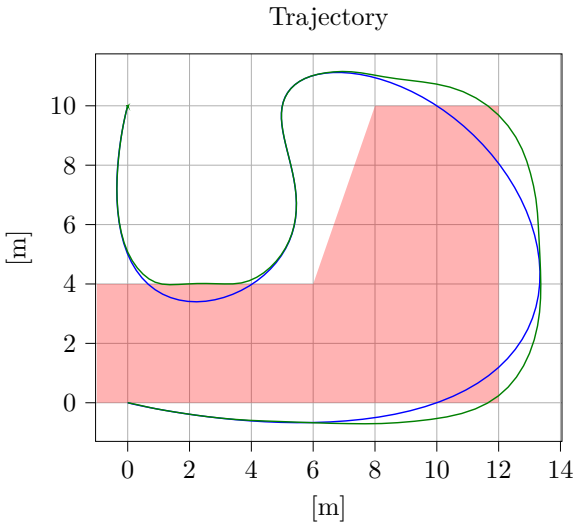


Figure 4.7: Trajectory following with penalized area. Blue is reference trajectory and green is flown path.

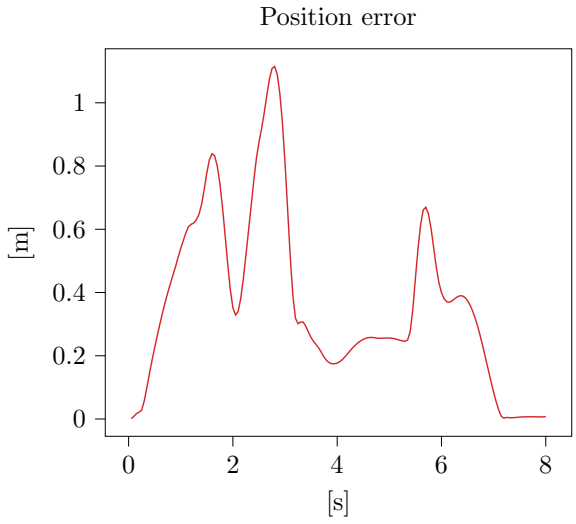


Figure 4.8: Position error of drone while following trajectory intersecting penalized area.

4.3 Comparison with other control approaches

This section compares three controllers in the Gazebo simulator. These include the SE3 controller (based on geometric state feedback [19]), the MPC, and presented implementation of the NMPC. The comparison will be conducted on a reference trajectory created by the tracker for a move from coordinates (0,0,5) to (5,0,5). For this tracker, so-called "fast constraints" were set to create a quick trajectory.

The MPC is part of the current flight stack of the Multi-robot Systems Group (MRS), but it does not have trajectory tracking implemented. This means that it uses only the first point of this trajectory as a reference. As a result, the trajectory tracking in this manner is slow and very inaccurate (Figure 4.9).

The SE3 controller is the most accurate controller from the current flight stack of the MRS, but according to the MRS website [20], it is sensitive to measurement noise, requires a feasible and smooth reference, and needs to be tuned for each individual drone. However, these disadvantages did not become evident in this comparison (Figure 4.10).

The proposed NMPC takes into account the entire trajectory. But for an aforementioned issue, it begins to oscilate for a quick trajectory (Figure 4.11). For the trajectory generated with "slow constraints", the dynamics are slow enough and the drone follows the trajectory smoothly (Figure 4.12).

Comparison of various controllers for trajectory tracking. Green represents the position of the drone, blue represents the first point of the reference trajectory.

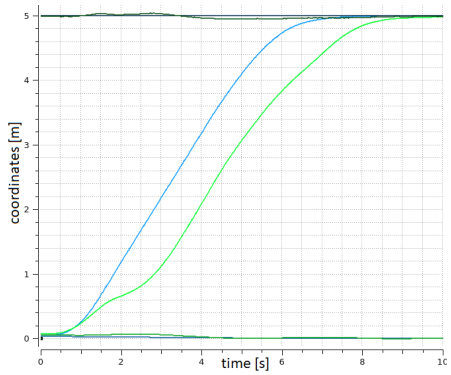


Figure 4.9: MPC

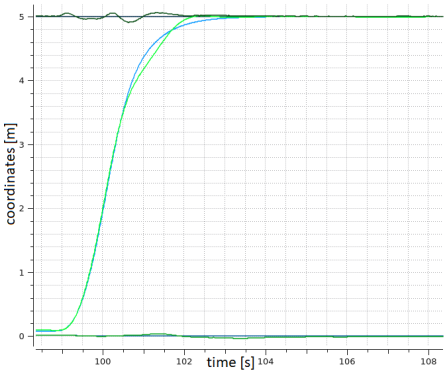


Figure 4.10: SE3

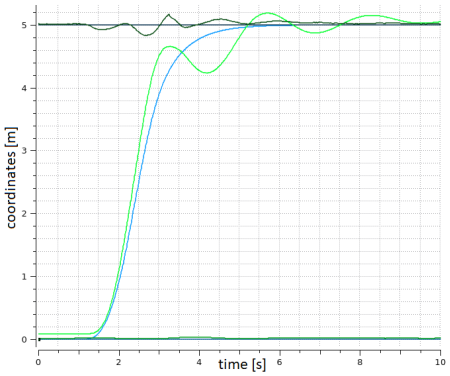


Figure 4.11: NMPC

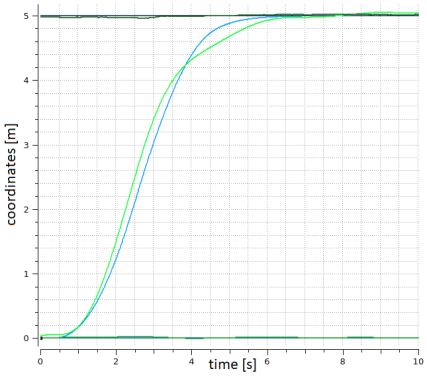


Figure 4.12: NMPC slow

4.4 Results from flight on real drone

The presented NMPC was tested on a real drone. During the testing period, which was month before this thesis submission, only the point-tracking version was implemented. This reference point was subsequently manually moved, achieving the drone's navigation through environment. Due to the lack of information about the entire reference trajectory, the flight performance was limited. Nevertheless, the experiment demonstrated the usability of drone flight using the proposed NMPC under real conditions. Photo of the drone F450 used in this experiment is in Figure 4.13.

Video of the experiment is available on youtube: `https://youtu.be/ht1Rcb-7Fzk`

Figure 4.13: Drone F450 is relying on proposed NMPC.

From the flight log, we obtained both the drone's actual positions and the commanded positions. During the displayed time (Figure 4.14, Figure 4.15, Figure 4.16), the drone was given a command to move in the xy plane. The graphs reveal that the drone's position is temporally lagging behind the reference point, as the drone does not receive the entire trajectory but only a single reference point.

Record of x, y, and z coordinates of drone F450 in real environment. Red is reference and green is recorded position.

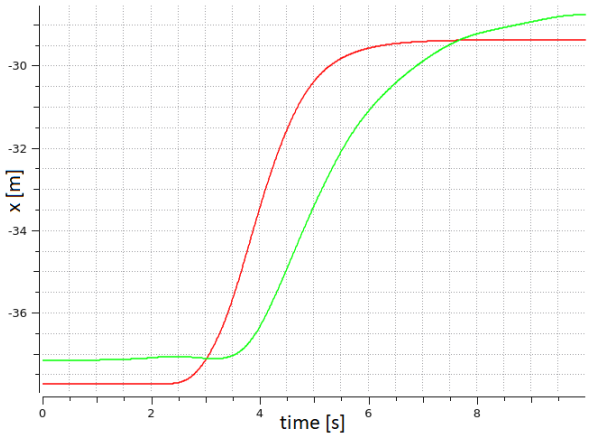


Figure 4.14: Axis x

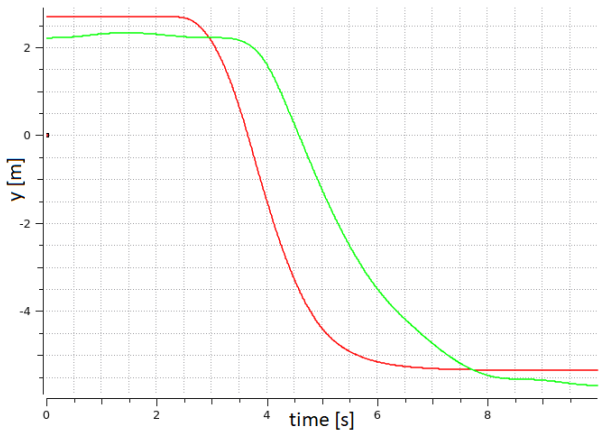


Figure 4.15: Axis y

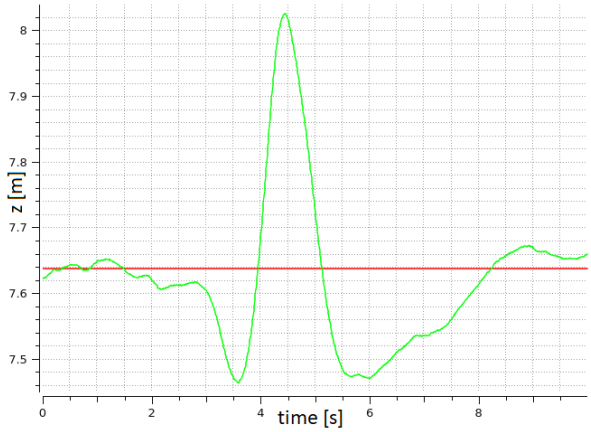


Figure 4.16: Axis z

Chapter 5

Conclusion

This work has discussed current methods for controlling drones based on MPC. New NMPC controller meant for agile drone flight was designed and implemented using the acados library. The quality of dynamic properties of the NMPC was tested with a simple RK4-based simulator. The controller was integrated into the MRS system and tested in the Gazebo simulator. The tests indicated that the controller functions as intended, however, the system exhibits some unmodelled dynamics or other issues that result in compromised control.

Further in the project, the optimization function was enhanced with the ability to consider obstacles and assessed the impact on computational demand and functionality in an RK4 simulation. Then, the proposed controller was compared to the current flight stack of the MRS and found that it has the potential to become the preferred controller for high-speed flights if current issues are addressed. Lastly, the NMPC was deployed on a drone in a real environment and real applicability was confirmed.

Future work will include identifying and addressing these shortcomings, as well as testing obstacle avoidance in the Gazebo system. Given the high versatility of the acados library, the developed controller could be extended to incorporate several additional features, such as movable obstacles or variable time steps for individual shooting nodes, which could result in increased modeling accuracy for the near future while still maintaining a broad future outlook using a reasonably small number of shooting nodes.

Chapter 6

References

[1] M. Sivakumar and N. M. TYJ, "A literature survey of unmanned aerial vehicle usage for civil applications," *Journal of Aerospace Technology and Management*, vol. 13, e4021, 2021, ISSN: 2175-9146. DOI: 10.1590/jatm.v13.1233.

[2] H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, "Model predictive control for micro aerial vehicles: A survey," in *2021 European Control Conference (ECC)*, 2021, pp. 1556–1563. DOI: 10.23919/ECC54610.2021.9654841.

[3] R. Verschueren, G. Frison, D. Kouzoupis, *et al.*, "Acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021, ISSN: 1867-2957. DOI: 10.1007/s12532-021-00208-8.

[4] A. Romero, R. Penicka, and D. Scaramuzza, "Time-optimal online replanning for agile quadrotor flight," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7730–7737, 2022. DOI: 10.1109/LRA.2022.3185772.

[5] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3340–3356, 2022. DOI: 10.1109/TRO.2022.3173711.

[6] M. Jacquet, M. Kivits, H. Das, and A. Franchi, "Motor-level n-mpc for cooperative active perception with multiple heterogeneous uavs," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2063–2070, 2022. DOI: 10.1109/LRA.2022.3143218.

[7] L. F. Recalde, B. S. Guevara, C. P. Carvajal, V. H. Andaluz, J. Varela-Aldás, and D. C. Gandolfo, "System identification and nonlinear model predictive control with collision avoidance applied in hexacopters uavs," *Sensors*, vol. 22, no. 13, 2022, ISSN: 1424-8220. DOI: 10.3390/s22134712.

[8] J.-R. Chiu, J.-P. Sleiman, M. Mittal, F. Farshidian, and M. Hutter, "A collision-free mpc for whole-body dynamic locomotion and manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4686–4693. DOI: 10.1109/ICRA46639.2022.9812280.

[9] B. T. Lopez, J.-J. E. Slotine, and J. P. How, "Dynamic tube mpc for nonlinear systems," in *2019 American Control Conference (ACC)*, 2019, pp. 1655–1662. DOI: 10.23919/ACC.2019.8814758.

[10] J. Arrizabalaga and M. Ryll, "Towards time-optimal tunnel-following for quadrotors," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4044–4050. DOI: 10.1109/ICRA46639.2022.9811764.

[11] T. Baca, M. Petrlik, M. Vrba, *et al.*, "The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 1, p. 26, 2021, ISSN: 1573-0409. DOI: 10.1007/s10846-021-01383-5.

[12] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2018. DOI: 10.1109/LRA.2017.2776353.

[13] Z. Wang, H. Ye, C. Xu, and F. Gao, "Generating large-scale trajectories efficiently using double descriptions of polynomials," in *IEEE International Conference on Robotics and Automation*, IEEE, Xi'an, China, 2021, pp. 7436–7442.

[14] J. Frey, *Optimalization problem formulation*, https://github.com/acados/acados/blob/master/docs/problem_formulation/problem_formulation_ocp_mex.pdf.

[15] O. Robotics, *Gazebo*, https://gazebosim.org, Accessed on 25.5.2023, 2022.

[16] M. robot Systems (MRS) group, *Singularity container*, https://github.com/ctu-mrs/mrs_singularity, Accessed on 25.5.2023, 2023.

[17] M. Quigley, K. Conley, B. P. Gerkey, *et al.*, "Ros: An open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[18] D. Hert, T. Baca, P. Petracek, *et al.*, "Mrs modular uav hardware platforms for supporting research in real-world outdoor and indoor environments," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2022, pp. 1264–1273. DOI: 10.1109/ICUAS54217.2022.9836083.

[19] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se(3)," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5420–5425. DOI: 10.1109/CDC.2010.5717652.

[20] *Multi-robot systems group unmanned aerial vehicle controllers*, Accessed on 25.5.2023. [Online]. Available: https://ctu-mrs.github.io/docs/software/uav_core/mrs_uav_controllers/.

Chapter A

Appendix A

This appendix provides an overview of the codebase created for the Nonlinear Model Predictive Control (NMPC) controller, developed as part of this Bachelor's thesis.

A.1 Python Scripts

The Python scripts are used for generation of required C files:

1. `main_build.py` The primary build script.
2. `quadrotor_model.py` Contains the state equations for quadrotor.
3. `quaternion.py` Implements Quaternion calculations necessary for the controller.
4. `utils.py` ontains various utility functions used by aforementioned scripts.

A.2 Core Library

The core of the library is composed of C++ files placed in `cpp` and `hpp` directories:

A.2.1 CPP Directory

1. `acados_wrapper.cpp`: A wrapper script for Acados, providing an interface to use the Acados library with this project.
2. `drone.cpp`: Contains the implementation of drone behavior and RK4.
3. `traj_optim_wrapper.cpp`: A wrapper script for Trajectory optimizer library, providing an interface to use of trajectory generation with this project.
4. `simulation.cpp`: This file includes a sample code demonstrating the use of the library.

A.2.2 HPP Directory

1. `acados_wrapper.hpp`: The header file for `acados_wrapper.cpp`.
2. `drone.hpp`: The header file for `drone.cpp`.
3. `common.hpp`: Contains definitions and declarations used throughout the library.

These files require the C files generated as mentioned above.

A.3 Configuration Files

YAML files are used for configuration purposes in this project. These files allow users to define the controller parameters for the NMPC controller, simulation parameters and obstacles.

A.4 Visualization Scripts

Two Python scripts have been written to visualize the results obtained from the simulation:

1. `visualize_output.py`: Plots 3D figure of drone trajectory.
2. `visualize_output_graphs.py`: Renders time evolution of drone state in graphs.

A.5 Large Scale Trajectory Optimizer Library

The `large_scale_traj_optimizer` directory contains an additional library that provides generation of trajectories.