Linear and Nonlinear Model Predictive Control Strategies for Trajectory Tracking Micro Aerial Vehicles: A Comparative Study

I.K. Erunsal[1,2], J. Zheng[1], R. Ventura[2], A. Martinoli[1]

Abstract— **This paper presents a comparison of linear and nonlinear Model Predictive Control (MPC) strategies for trajectory tracking Micro Aerial Vehicles (MAVs). In this comparative study, we paid particular attention to establish quantitatively fair metrics and testing conditions for both strategies. In particular, we chose the most suitable numerical algorithms to bridge the gap between linear and nonlinear MPC, leveraged the very same underlying solver and estimation algorithm with identical parameters, and allow both strategies to operate with a similar computational budget. In order to obtain a well-tuned performance from the controllers, we employed the parameter identification results determined in a previous study for the same robotic platform and added a reliable disturbance observer to compensate for model uncertainties. We carried out a thorough experimental campaign involving multiple representative trajectories. Our approach included three different stages for tuning the algorithmic parameters, evaluating the predictive control feasibility, and validating the performances of both MPC-based strategies. As a result, we were able to propose a decisional recipe for selecting a linear or nonlinear MPC scheme that considers the predictive control feasibility for a peculiar trajectory, characterized by specific speed and acceleration requirements, as a function of the available on-board resources.**

I. INTRODUCTION

The recent technical advances on Micro Aerial Vehicles (MAVs) have motivated many researchers to deploy these vehicles into challenging real-world environments to accomplish complex missions, such as surveillance, search and rescue, object delivery, etc. [1]. Following a prescribed trajectory as closely as possible (trajectory tracking) is one of the most prominent and often essential components to complete such missions. In relation to this task, controllers must almost always meet four criteria: high trajectory tracking performance, satisfaction of constraints, robustness to unmodelled dynamics, and real-time, possibly on-board, operation [2].

Among possible control solutions, Model Predictive Control (MPC) stands out immediately when the first two criteria above are considered, since it optimizes the performance over a prediction horizon and integrates constraints into the problem explicitly [3]. However, unmodelled dynamics and perturbations usually generate significant issues, since MPC requires an accurate model of the system to operate, although it integrates the feedback into the control loop. In order to mitigate the impact of such inaccuracies, the system is typically identified as accurately as possible, and the use of a robust version of MPC and/or a disturbance estimator is recommended. Finally, performing the predictive control action in real-time and with on-board resources is challenging due to the inherent fast dynamics of MAVs and the high-level requirements of missions. While addressing this problem, researchers usually face an important dilemma: ”Should I use a linear or a nonlinear MPC strategy for my task?” Multiple contributions in the literature show that for highly nonlinear systems with fast dynamics, such as MAVs, Nonlinear MPC (NMPC) should be preferred in spite of its higher computational cost [4]. Nonetheless, multiple complex missions involve tasks that can be carried out in near-hovering conditions or with slow speed/acceleration. In such conditions, their dynamics and other constraints can be approximated by linear models and therefore a computationally light-weight Linear MPC (LMPC) scheme becomes competitive.

There are different successful real-world implementations of MPC for MAVs performing unified trajectory optimization and tracking [4], perception-aware tracking [5], six-dimensional tracking [6], time-optimal way-point tracking [7], Image-Based Visual Servo (IBVS) control [8] and local reference trajectory and tracking [9]. Furthermore, for interested readers, the studies in which the validation is performed only in simulation include [10], [11], [12], [13], [14]. Multiple previous contributions report comparative studies between LMPC and NMPC. For instance, [15] performs this analysis for rotary-winged MAVs engaged in trajectory tracking under nominal conditions and wind disturbance for several real scenarios. However, authors employ different solvers for LMPC and NMPC, a choice that does not allow for a fair benchmarking of the two strategies in terms of solver accuracy and computation time. In [2], the authors investigate the effectiveness of a flatness-based MPC approach and compare it to canonical LMPC and NMPC.

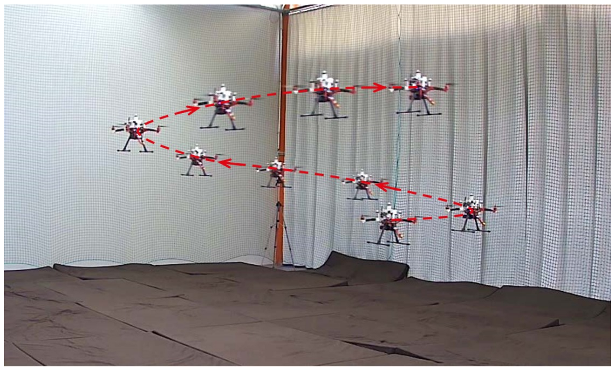
Fig. 1: A trajectory tracking experiment performed with a Helipal Storm Quadrotor; in this figure, a screw trajectory is followed.

[1]Distributed Intelligent Systems and Algorithms Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland kagan.erunsal@epfl.ch, jianhao.zheng@epfl.ch, alcherio.martinoli@epfl.ch

[2]Institute for Systems and Robotics, Instituto Superior Técnico, Lisbon, Portugal rodrigo.ventura@isr.tecnico.ulisboa.pt

While this contribution is not targeting a direct comparison between the two strategies, it indirectly contributes to a comparative performance evaluation. However, the insight is limited for our scope, as algorithms have not been deployed on-board, taking into account the typically scarce resources of a MAV. In [16], although the authors study the trajectory tracking control problem for an articulated Unmanned Ground Vehicle (UGV) and conduct an experimental benchmarking for LMPC and NMPC, they use different estimation frameworks and solvers in their implementation, making the comparison of the respective algorithms more difficult from a pure control perspective. Finally, although it is not a direct comparison between LMPC and NMPC, [17] analyzes two state-of-the-art controllers, i.e. NMPC and a differential flatness-based controller, in simulation and real-world experiments. Note that, in contrast to this paper, our focus is on the predictive controllers and we give insights about two different kinds of MPC methodologies, while the aforementioned paper includes a reactive controller in the comparison.

In this paper, our aim is to provide an as fair as possible comparison between LMPC and NMPC strategies on a real MAV engaged in trajectory tracking, in order to help the reader to respond in an as educated way as possible to the previously mentioned dilemma. To achieve a performance well-tuned to the underlying hardware platform for both strategies, we employ the system identification results of a previous study [18], and integrate a disturbance observer based on an Extended Kalman Filter (EKF) to mitigate the issues related to model uncertainty and perturbations. We choose a Real Time Iteration (RTI) strategy based on Sequential Quadratic Programming (SQP) for the NMPC solution in order to easily bridge the gap between the nonlinear and linear counterparts, as explained in [19]. Our approach consists of three stages, eventually resulting in a decisional recipe for selecting the most appropriate MPC-based scheme for the targeted application. In the first stage, we leverage a realistic and calibrated robotic simulator, namely, Webots [20], in order to tune the parameters of both controllers, i.e. control weights, control/estimation frequency and horizon length, to obtain the best performances in terms of trajectory tracking accuracy. Note that we use the same cost penalty parameters and tolerances for both controllers, allocate similar computational budgets, and employ the same low-level QP solver to be objective. One big advantage of LMPC here is its ability to increase both frequency and horizon length due to the relative simplicity and convexity of the optimal control problem solved for a given computational budget. For the second stage, we switch to physical reality, choose a generic type of trajectory and conduct various experiments with increasing difficulty in terms of the feasibility metrics to investigate the relation between the feasibility of a controller and tracking performance. In order to validate this approach, in the third stage, we evaluate the performance of both MPC-based strategies on a representative set of trajectories and show the effectiveness of the aforementioned decision process.

The main contribution of this paper lies in the objective benchmarking campaign designed for LMPC and NMPC, which is, to the best of our knowledge, missing in the literature. Leveraging the very same low-level convex solver, tolerances, maximum computational budget, cost weights, estimation/control framework and trajectory types reveals the underlying characteristics of both controllers and makes the comparison between them as fair as possible. Furthermore, the extensive experiments conducted for testing and validation increase the credibility of the campaign. Finally, a significant contribution is that we present a quantitative procedure to facilitate the decisional process for a given system and targeted application in order to select MPC scheme. This recipe includes the computation of two novel metrics: the feasibility margin and the average feasibility. Both metrics natively consider the robot's resources and the characteristics of the targeted trajectory.

II. METHODOLOGY

Our comparative study is carried out on a multi-rotor MAV operating indoor and performing a trajectory tracking mission. Assume that the mission must be carried out with high accuracy due to the cluttered nature of the environment and timing constraints. The vehicle has access to its absolute position through a global localization system such as a Motion Capture System (MCS), is equipped with an integrated optical flow and distance sensor to obtain linear velocities and an IMU to acquire linear accelerations, rotational velocities and Euler angles. The MAV has two processors, one of them is a commercially available board running an open-source autopilot and the other one is a high-level on-board computer.

A. Notation and Plant Model

Similar to [21], we will adopt the following notation for vectors and rotation matrices assuming that x is the name of the vector and $\{a\},\{b\},\{c\}$ are the coordinate frames:

$\boldsymbol{x}^c_{a/b}$:	Vector quantity $x \in \mathbb{R}^n$ of the origin of $\{a\}$ with respect to the origin of $\{b\}$ expressed in $\{c\}$.
$\boldsymbol{R}^b_a$:	Rotation matrix $R \in \mathbb{SO}^3$ that transforms a vector expression from $\{a\}$ to $\{b\}$.

$\| \, . \, \|_2$ represents the Euclidean norm of a vector or the spectral norm of a real matrix and $\| \, . \, \|_P := \sqrt{x^T P x}$ (with $P \in \mathbb{R}^{n \times n}$ and $P \succeq 0$) denotes the weighted Euclidean norm of $x \in \mathbb{R}^n$. Note that bold letters represent matrices or vectors.

The state of a MAV can be described by the position of the center of gravity $\boldsymbol{x}^n_{b/n} \in \mathbb{R}^3$, the linear velocity $\boldsymbol{v}^n_{b/n} \in \mathbb{R}^3$, the attitude Euler angles $\boldsymbol{t}^n_{b/n} \in \mathbb{R}^3$ and the angular speeds $\boldsymbol{w}^b_{b/n} \in \mathbb{R}^3$. Here, $\{n\}$ is the Earth-fixed inertial frame and $\{b\}$ is the body-fixed frame of the MAV.

We adopt the full dynamics of the multi-rotor MAV including dominant (propeller) and auxiliary (drag) aerodynamic effects based on [22] and [23]. Note that although this model can be used as a plant model for any simulation, it is too complex for the MPC running onboard. We will explain the abstract model employed for MPC in the following sections.

The cascaded architecture displayed in Fig. 2 is designed for both strategies in order to separate high frequency tasks (attitude control) from the low frequency tasks (trajectory control). We adopted this structure due to three main reasons. First, [17] indicates the necessity of using an inner-loop controller for high performance trajectory tracking by validating

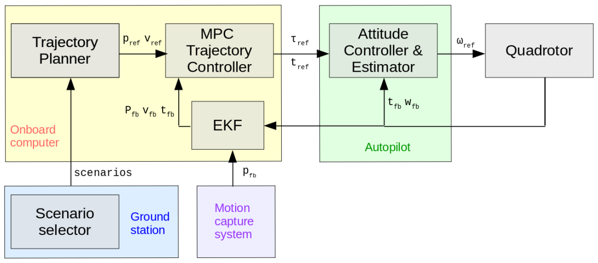

Fig. 2: Control and estimation architecture of the MAV.

it with real experiments. Second, this separation lets us use the full potential of the autopilot and onboard computer by assigning the tasks to the dedicated hardware. Third, MPC can still reach/manipulate the attitude dynamics variables as inputs, thus, it has a good control over the inner-loop. The next parts will explain the estimation algorithm briefly and present the trajectory controllers in detail.

B. Vehicle Model and Estimation Scheme

In order to reduce the computational load of MPC and handle the control actions with different rates, we take advantage of the autopilot where a high frequency attitude controller is implemented. Consequently, a first order, simplified representation of the attitude subsystem is employed. Based on the model presented in [15], we propose the following dynamic equations (or prediction function) for a multirotor MAV:

$$\dot{\boldsymbol{x}}^n_{b/n} = \boldsymbol{v}^n_{b/n} \tag{1a}$$

$$\dot{\boldsymbol{v}}^n_{b/n} = \frac{1}{m_b}\left(\boldsymbol{R}^n_b \begin{bmatrix} 0 \\ 0 \\ T_{cmd} \end{bmatrix} - \boldsymbol{F}_{aero} + \boldsymbol{d}^b_{b/n}\right) + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \tag{1b}$$

$$\dot{\boldsymbol{t}}^n_{b/n} = \frac{1}{\boldsymbol{\tau}_t}\left(\boldsymbol{K}_t \boldsymbol{t}_{cmd} - \boldsymbol{t}^n_{b/n}\right) \tag{1c}$$

where m_b is the mass of the MAV, $\boldsymbol{d}^b_{b/n} \in \mathbb{R}^3$ is the perturbation estimated by the EKF (assumed constant over the horizon) , $\boldsymbol{t}_{cmd} \in \mathbb{R}^3$ is the commanded attitude angles $[\phi_{cmd} \;\; \theta_{cmd} \;\; \psi_{cmd}]^T$, $\boldsymbol{F}_{aero} \in \mathbb{R}^3$ is the drag forces acting on the body, $\boldsymbol{\tau}_t = diag(\tau_\phi, \tau_\theta, \tau_\psi)$ and $\boldsymbol{K}_t = diag(k_\phi, k_\theta, k_\psi)$ are respectively the gains and time constants of first-order attitude subsystem. For the parametric estimation of the drag coefficient, attitude gains and time constant, please refer to [18].

In order to filter out the sensor noise and estimate the perturbations, an extended-state EKF inspired by [24] is implemented. The state for the estimator can be written as follows:

$$\hat{\boldsymbol{\xi}} = [\hat{\boldsymbol{x}}^b_{b/n}{}^T \; \hat{\boldsymbol{v}}^b_{b/n}{}^T \; \hat{\boldsymbol{t}}^b_{b/n}{}^T \; \hat{\boldsymbol{d}}^b_{b/n}{}^T]^T \tag{2}$$

where $\hat{\boldsymbol{d}}^b_{b/n}$ is the perturbations acting on the body due to modelling inaccuracies and external disturbance. The process model is chosen based on Eq. (1) and a slowly varying disturbance model in three linear dimensions is concatenated to the overall process. The measurement model includes the position information from the MCS, body velocities from the optical flow-distance sensor module and Euler angles from the IMU. The parameters of the estimator are tuned in real experiments considering the relative magnitudes of process and sensor noises.

C. Linear MPC

To eliminate the direct dependency on the yaw angle ψ, we introduce a heading-free reference frame $\{hf\}$ where the x-axis is aligned with the heading direction of the vehicle ($\psi^{hf} = 0$). The transformation of the attitude angles between Earth-fixed inertial and heading-free frame is given by:

$$\begin{bmatrix} \phi \\ \theta \end{bmatrix} = \begin{bmatrix} cos\psi & sin\psi \\ -sin\psi & cos\psi \end{bmatrix} \begin{bmatrix} \phi^{hf} \\ \theta^{hf} \end{bmatrix} \tag{3}$$

Based on that, we define the following state and control input vector for LMPC:

$$\boldsymbol{\xi}_L = [\boldsymbol{x}^b_{b/n}{}^T \; \boldsymbol{v}^b_{b/n}{}^T \; \phi^{hf} \; \theta^{hf}]^T \tag{4}$$

$$\boldsymbol{u}_L = [\phi^{hf}_{cmd} \; \theta^{hf}_{cmd} \; T_{cmd}]^T \tag{5}$$

where T_{cmd} is the commanded thrust generated by four propellers, ϕ^{hf}_{cmd} and θ^{hf}_{cmd} are the commanded roll and pitch angle in the heading-free frame.

Inspired by [15], the vehicle model is linearized around hovering condition assuming small attitude angles. Next, we discretize the continuous-time model by the Zero Order Hold (ZOH) method, knowing that this is an highly accurate analytical method for linear systems if the sample time is sufficiently small. The linearized discrete state-space model (or prediction function) can be written as:

$$\Delta\boldsymbol{\xi}_L[k+1] = \boldsymbol{A}\Delta\boldsymbol{\xi}_L[k] + \boldsymbol{B}\Delta\boldsymbol{u}_L[k] + \boldsymbol{B}_d\Delta\boldsymbol{d}[k] \tag{6}$$

where $\Delta\boldsymbol{\xi}_L[k] = \boldsymbol{\xi}_L[k] - \boldsymbol{\xi}^{ref}_L[k]$ is the deviation between the variable $\boldsymbol{\xi}_L[k]$ and the reference state $\boldsymbol{\xi}^{ref}_L[k]$; the same notations hold for $\Delta\boldsymbol{u}_L[k]$ and $\Delta\boldsymbol{d}[k]$. In our implementation, only the reference position $[\boldsymbol{x}^b_{b/n}]^{ref}$ is provided. All other reference vectors are given as they all refer to hovering condition. Since the matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{B}_d$ are invariant to position, the dynamic model in Eq. (6) still holds even if it is linearized around the hovering condition instead of references.

In the remaining sections, we will use the following notation to distinguish the predicted values from the actual ones: $\boldsymbol{x}[m|n]$ is the value of variable $\boldsymbol{x}$ at discrete instant $k = m$, predicted at $k = n$ where $m \geq n$. Moreover, N is the prediction (and control) horizon and Δt is defined as the sampling time. At time k with initial $\hat{\boldsymbol{\xi}}_L[0]$, the LMPC controller solves the following Open-Loop Control Problem (OCP):

$$\min_{\Delta\Xi_L, \Delta\boldsymbol{U}_L} \sum_{n=0}^{N-1} (\|\Delta\boldsymbol{\xi}_L[k+n|k]\|^2_{\boldsymbol{Q}} + \|\Delta\boldsymbol{u}_L[k+n|k]\|^2_{\boldsymbol{R}}) + \|\Delta\boldsymbol{\xi}_L[k+N|k]\|^2_{\boldsymbol{P}} \tag{7}$$

subject to the following constraints:

$$\begin{aligned}
&\Delta\boldsymbol{\xi}_L[k+n+1|k] = \boldsymbol{A}\Delta\boldsymbol{\xi}_L[k+n|k] + \boldsymbol{B}\Delta\boldsymbol{u}_L[k+n|k] + \boldsymbol{r}[k+n|k]\\
&\boldsymbol{r}[k+n|k] = \boldsymbol{A}\boldsymbol{\xi}_L^{ref}[k+n] + \boldsymbol{B}\boldsymbol{u}_L^{ref}[k+n]\\
&\qquad\qquad + \boldsymbol{B}_d\boldsymbol{d}[k+n|k] - \boldsymbol{\xi}_L^{ref}[k+n+1]\\
&\boldsymbol{d}[k+n|k] = \hat{\boldsymbol{d}}[0], \quad \Delta\boldsymbol{\xi}_L[k+n|k] + \boldsymbol{\xi}_L^{ref}[k+n] \in \mathbb{X}_L\\
&\Delta\boldsymbol{u}_L[k+n|k] + \boldsymbol{u}_L^{ref}[k+n] \in \mathbb{U}_L, \quad n = 0,...,N-1\\
&\boldsymbol{\xi}_L[k|k] = \hat{\boldsymbol{\xi}}_L[0]
\end{aligned} \tag{8}$$

where $\boldsymbol{Q} \succeq 0$ and $\boldsymbol{R} \succeq 0$ are respectively the penalty on the state error and input error, $\boldsymbol{P}$ is the terminal penalty on state error, $\boldsymbol{r}$ is the affine term to correct the infeasible trajectory, $\mathbb{X}_L$ is the state constraint due to small attitude angle assumption and safety consideration and $\mathbb{U}_L$ is the input constraint from the physical limits of the propellers and electric motors.

Note that the OCP can be cast as a Quadratic Programming (QP) problem, by considering $\Delta\Xi_L = [\Delta\boldsymbol{\xi}_L[k+1|k],...,\Delta\boldsymbol{\xi}_L[k+N|k]]^T$ and $\Delta\boldsymbol{U}_L = [\Delta\boldsymbol{u}_L[k|k],...,\Delta\boldsymbol{u}_L[k+N-1|k]]^T$. The QP solver we employed is qpOASES [25], which is not an ideal solver to tackle the sparse optimization problem described by Eqs. (7) and (8). Thus, the condensing procedure described in Section II of [26] is implemented to reduce the dimension. The overall procedure is summarized in Algorithm 1.

Algorithm 1 LMPC at the discrete time t_k

```
Initial preparation phase before start (at $t_k < 0$)

0. **Linearize** the system around hovering conditions $(\Xi_L^h(t_k), U_L^h(t_k))$ and obtain the sensitivities to construct the linear model and constraints
1. **Discretize** the system by a proper integration method (ZOH)
2. **Define** the QP problem similar to (7) and (8) omitting $\boldsymbol{\xi}_L[k|k] = \hat{\boldsymbol{\xi}}_L[0]$
3. **Condense** QP to reduce sparsity

Feedback phase performed at $t_k$ upon arrival of $\boldsymbol{\xi}_L[k|k]$

4. **Input** $\boldsymbol{\xi}_L[k|k]$ to the QP problem
5. **Iterate and solve** the QP problem until obtaining a sufficiently low tolerance, check processor limits and control sample
    do while $(t_k == 0 \ || \ t_{CPU} \le t_s)$ && $tol \le threshold$
        $(\Xi_L^*(t_k), U_L^*(t_k)) \leftarrow QP(\Xi_L^g(t_k), U_L^g(t_k), \hat{\Xi}_L(t_k))$
    end while
```

Only the first control input $\boldsymbol{u}_L[k|k] = \Delta\boldsymbol{u}_L[k|k] + \boldsymbol{u}_L^{ref}[k]$ is applied to the vehicle. However, the commanded roll ϕ_{cmd}^{hf} and pitch θ_{cmd}^{hf} angles need to be transformed into the Earth-fixed inertial frame by Eq. (3). Furthermore, the non-zero attitude effects on thrust can be compensated by the following equation involving roll and pitch angles:

$$\tilde{T}_{cmd} = \frac{T_{cmd}}{cos\phi cos\theta} \tag{9}$$

D. Nonlinear MPC

We choose a RTI-based SQP approach to obtain the solution because of two main reasons. First, it is arguably a most successful (in terms of speed and accuracy) and largely used strategy for real applications [19]. Second, it is very straightforward to comprehend the connection between the QP solution of LMPC and the SQP solution of NMPC.

In contrast to LMPC, the state and input are redefined by the following expressions:

$$\boldsymbol{\xi}_N = [\boldsymbol{x}_{b/n}^{b\ T} \ \boldsymbol{v}_{b/n}^{b\ T} \ \phi \ \theta]^T \tag{10}$$

$$\boldsymbol{u}_N = [\phi_{cmd} \ \theta_{cmd} \ T_{cmd}]^T \tag{11}$$

Based on the Eq. (1) and a discretization methodology suitable for real-time applications (Runge-Kutta-4), the following discrete-time nonlinear model (or prediction function) is obtained:

$$\boldsymbol{\xi}_N[k+1] = \hat{f}(\boldsymbol{\xi}_N[k], \boldsymbol{u}_N[k], \boldsymbol{d}[k]) \tag{12}$$

Similar to the linear counterpart, the OCP can now be written as follows:

$$\begin{aligned}
\min_{\Xi_N, \boldsymbol{U}_N} \sum_{n=0}^{N-1} (\|\boldsymbol{\xi}_N[k+n|k] - \boldsymbol{\xi}_N^{ref}[k+n]\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u}_N[k+n|k] - \boldsymbol{u}_N^{ref}[k+n]\|_{\boldsymbol{R}}^2)\\
+ \|\boldsymbol{\xi}_N[k+N|k] - \boldsymbol{\xi}_N^{ref}[k+N]\|_{\boldsymbol{P}}^2
\end{aligned} \tag{13}$$

subject to the following constraints:

$$\begin{aligned}
&\boldsymbol{\xi}_N[k+n+1|k] = \hat{f}(\boldsymbol{\xi}_N[k], \boldsymbol{u}_N[k], \boldsymbol{d}[k])\\
&\boldsymbol{d}[k+n|k] = \hat{\boldsymbol{d}}[0]\\
&\boldsymbol{\xi}_N[k+n|k] \in \mathbb{X}_N, \quad \boldsymbol{u}_N[k+n|k] \in \mathbb{U}_N, \quad n = 0,...,N-1\\
&\boldsymbol{\xi}_N[k|k] = \hat{\boldsymbol{\xi}}_N[0]
\end{aligned} \tag{14}$$

The details of the solution strategy for NMPC are reported in Algorithm 2.

Algorithm 2 NMPC at the discrete time t_k

```
Initial preparation phase before start (at $t_k < 0$)

0. **Discretize** the system by a proper integration method (i-RK-4)

Preparation phase over the time interval $[t_{k-1}, t_k]$

1. **Time-shift** optimal solution at the previous time step to obtain initial guess, $(\Xi_N^g(t_k), U_N^g(t_k)) \leftarrow (\Xi_N^*(t_{k-1}), U_N^*(t_{k-1}))$
2. **Linearize** the system around $(\Xi_N^g(t_k), U_N^g(t_k))$ and obtain the sensitivities to construct the linear model and constraints
3. **Define** the QP problem similar to (13) and (14) omitting $\boldsymbol{\xi}_N[k|k] = \hat{\boldsymbol{\xi}}_N[0]$
4. **Condense** the QP problem to reduce the sparsity

Feedback phase performed at $t_k$ upon arrival of $\boldsymbol{\xi}_N[k|k]$

5. **Input** $\boldsymbol{\xi}_N[k|k]$ to the QP problem
6. **Iterate and solve** the QP problem until obtaining a sufficiently low KKT result, for the next iterations check processor limits, control sample time and desired running KKT limit to finish
    do while $(t_k == 0 \ || \ t_{CPU} \le t_s)$ && $KKT \ge threshold$
        $(\Xi_N^*(t_k), U_N^*(t_k)) \leftarrow QP(\Xi_N^g(t_k), U_N^g(t_k), \hat{\Xi}_N(t_k))$
    end while
```

To realize Algorithm 2, the nonlinear program solver ACADO [27] is integrated with the necessary modifications. Note that ACADO employs qpOASES as convex solver which is the same the solver leveraged in LMPC. This allows us to equivalently setup the problems and compare the LMPC and NMPC in an objective fashion.

E. Algorithmic comparison of LMPC and NMPC

By comparing Algorithm 1 and 2, the following algorithmic differences can be pointed out:

- LMPC can be considered as following a RTI scheme where the preparation phase is performed only once and usually offline. This reduces the computational cost significantly.

- The linearization step for LMPC is performed with respect to the reference trajectory in contrast to NMPC, where an initial guess based on the previous optimal solution is leveraged.
- While the analytical discretization methods such as ZOH can be employed for LMPC (only once during the initial preparation phase), the approximate numerical strategies such as implicit Runge-Kutta must be applied for NMPC which generates considerable overhead for the computation. Our objective in selecting a discretization method is to obtain an as accurate as possible integration while taking computational resources into account.
- The termination condition for LMPC includes the error tolerance for the QP solver. On the other hand, NMPC uses Karush-Khun-Tucker (KKT) conditions to decide the optimality of the solution.

F. *Feasibility Margin and Average Feasibility*

Two important definitions should be given here in order to facilitate the comprehension of next sections. These definitions will be adopted to decide the controller type by taking the target application into account.

Consider two compact sets $X \subset \mathbb{R}^n$ and $Y \subset \mathbb{R}^n$. Assume that sets $P \subset \mathbb{R}^n$ and $Q \subset \mathbb{R}^n$ are the bound sets of X and Y, respectively. Note that, the bound set can be defined as the set of points on the contour of the given set. Then the *Feasibility Margin* of set X with respect to Y can be defined as follows:

$$FM_{X,Y} = \min_p (I_Y(p) \min_q \|p-q\|) \tag{15}$$

where $p \in P$, $q \in Q$ and $I_Y(p)$ is the signed-indicator function whose definition is given as follows:

$$I_Y(p) = \begin{cases} +1, & \text{if } p \in Y \\ -1, & \text{otherwise} \end{cases} \tag{16}$$

Note that this margin is negative if $X \not\subset Y$ and can also be expressed as the *Percentage Feasibility Margin*, $PFM_{X,Y} = FM_{X,Y}/\|q^*\|$ where q^* is solution to $FM_{X,Y}$. Fig. 3 helps the reader to visualize this concept.

The *Average Feasibility* of X with respect to Y is defined as follows:

$$AF_{X,Y} = 1/L \int_P I_Y(p) \min_q \left(\frac{\|p-q\|}{\|q\|} \right) dp \tag{17}$$

where $p \in P$, $q \in Q$ and $L = \int_P dp$.

Assuming that there is no limit velocity for the MAV's working regime, in the context of this paper, the set Y represents the feasible (achievable) accelerations of the MAV and X stands for the set of target accelerations. The set of achievable accelerations Y can be calculated by Eq. (1b) and the knowledge of thrust, angle limits and maximum velocity during flight. Furthermore, the reference accelerations can be obtained by differentiating the position reference twice in time.

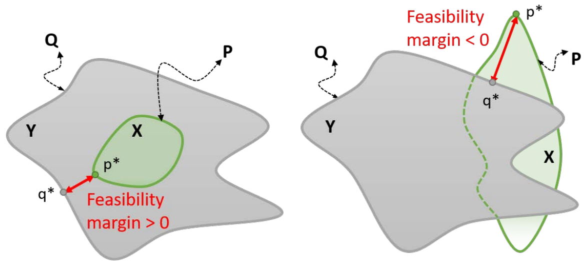

Fig. 3: Feasibility margin explained in two different scenarios: $X \subset Y$ and $X \not\subset Y$ where p^* and q^* are the solutions to the optimization problem in Eq. (15).

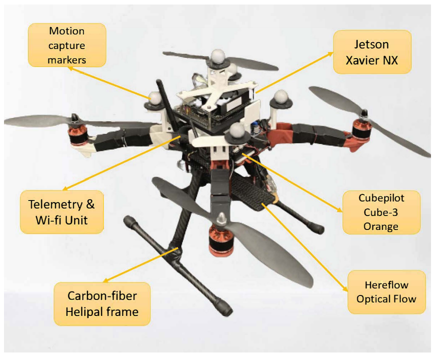

Fig. 4: The Helipal quadrotor and its components.

III. EXPERIMENTS AND RESULTS

We applied a three-stages experimental procedure. As mentioned before, the first stage leveraged simulation experiments and was used to tune the meta-parameters of both controllers. The remaining stages were performed in the physical reality and their purposes were to obtain a procedure to decide on the type of MPC, validate the results and give further insights about the performance of the controllers in a variety of conditions. Before presenting the details of the stages, it is worth giving additional details about the MAV used in our experimental campaign.

The quadrotor is a modified Helipal Storm Drone-4 v3, endowed with a Cube Orange autopilot, a Jetson Xavier NX on-board computer, an IMU, and an optical flow, as shown in Fig. 4. It weighs 1.55 kg and has a center to propeller distance of 21 cm. The 3D position information is obtained with the aid of an MCS with millimeter accuracy. All computations are performed on-board by leveraging the Robot Operating System (ROS). The Jetson Xavier NX computer has six parallel NVIDIA Carmel ARM v8.2 (max. 1.9 GHz) cores and one dedicated GPU (Volta architecture with 384 NVIDIA CUDA cores and 48 Tensor cores). For the experiments, all CPUs are activated with maximum power. The trajectories are generated inside an indoor volume of 27 m^3. The quadrotor can reach up to 6.4 m/s^2 acceleration and 10 m/s speed. The attitude controller runs at 100 Hz and it receives the autopilot references from the onboard computer at the running frequency of the NMPC.

A. *Stage 1: Tuning Meta-Parameters*

In order to perform the comparison of LMPC and NMPC in a fair fashion, a number of criteria can be enforced to be the same: estimation framework, cost penalty, solver type, error tolerance and allocated computational budget

Freq. (Hz) \ Hor. (steps)	20	30	40	50	60	70
20	6.6 ± 2.1	3.8 ± 1.6	3.5 ± 1.5	3.2 ± 1.4	3.2 ± 1.4	3.4 ± 1.6
30	11.9 ± 3.1	5.5 ± 1.7	3.2 ± 1.2	2.5 ± 1.2	2.4 ± 1.3	2.3 ± 1.2
40	18.1 ± 4.5	10.6 ± 2.9	5.0 ± 1.5	3.1 ± 1.3	2.3 ± 1.2	**2.1 ± 1.1**
50	24.5 ± 6.1	13.2 ± 3.4	7.7 ± 2.1	4.7 ± 1.4	3.1 ± 1.2	2.3 ± 1.1
60	30.9 ± 7.6	17.5 ± 4.4	10.7 ± 2.9	6.8 ± 1.9	4.5 ± 1.4	3.1 ± 1.2

TABLE I: Mean and standard deviations of position tracking RMSE [cm] for LMPC with the different frequency and horizon lengths.

Freq. (Hz) \ Hor. (steps)	20	30	40
20	6.5 ± 1.9	**3.8 ± 1.6**	**3.6 ± 1.7**
30	12.0 ± 3.1	5.5 ± 1.7	-
40	18.7 ± 4.5	10.7 ± 2.8	-
50	25.3 ± 6.0	-	-
60	32.1 ± 7.5	-	-

TABLE II: Mean and standard deviations of position tracking for NMPC [cm] with the different frequency and horizon lengths.

(to the possible extent). However, the controller-estimation frequency and prediction horizon can be different for LMPC and NMPC: in this way, LMPC can benefit from its lower computational cost. In theory, we expect that, first, increasing total prediction time would improve the performance up to a point where the model mismatch becomes significant; second, having high control/estimation frequency would help to reject the measured and unmeasured disturbances up to a certain frequency, where the impact of such mechanism on the performance would saturate. Following this reasoning, firstly, we should find the optimal frequency and horizon length for both controllers without exceeding the maximum computational budget. For our implementation, while the optimal controller frequency was limited by the maximum sensing frequency of absolute positioning information (60 Hz), the horizon length was upper-bounded by the capabilities of the QP solver (70 steps). Hence, a systematic search was performed inside these bounds. Note that the metric we used to find the optimal parameters is the position tracking Root Mean Square Error (RMSE).

We run the LMPC and NMPC with the chosen frequencies and horizons on the on-board computer and extracted the pairs whose CPU usage was less than 60% of the total budget. This percentage was selected as a safety factor. Afterwards, we designed a generic and feasible set of Lissajous trajectories whose acceleration and speed profiles are smooth, and tested these pairs ten times in Webots [20], an open-source, high-fidelity robotic simulator. We list the results in Table I for LMPC, in Table II for NMPC, and highlight the optimal frequency and horizon pairs. As a result, the frequency-horizon pair of $\{40,70\}$ and $\{20,30\}$ are selected for LMPC and NMPC, respectively. Note that we discard the pairs $\{20,40\}$ for NMPC since it was very close to the allocated CPU limit.

B. Stage 2: Investigating Feasibility

Before starting this stage, our hypothesis was that the feasibility margin and average feasibility of a predictive control scheme in carrying out a given trajectory tracking mission has a significant effect on the performance. In order to validate this claim, we designed five Lissajous trajectories with decreasing feasibility margin by increasing the maximum acceleration and speed of the reference trajectory, and

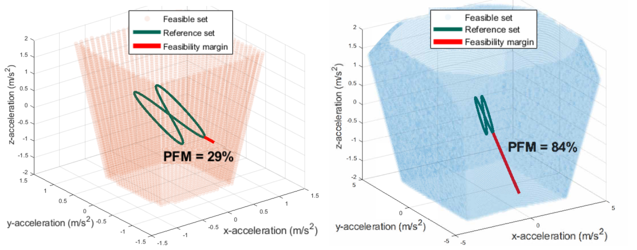

Fig. 5: Comparison of percentage feasibility margin of LMPC (left) and NMPC (right) for a Lissajous trajectory.

Prop. \ Type	Lissajous (1)	Lissajous (2)	Lissajous (3)	Lissajous (4)	Lissajous (5)
$\boldsymbol{a}_{max}$ (m/s^2)	0.17	0.37	0.76	1.48	4.12
$\boldsymbol{v}_{max}$ (m/s)	0.49	0.74	1.05	1.47	2.45
PFM-LMPC (%)	84.7	65.0	28.9	-31.2	-121.1
AF-LMPC	88.9	75.2	48.5	8.6	-69.1
PFM-NMPC (%)	97.0	93.2	84.6	62.6	-0.1
AF-NMPC	99.9	99.5	90.4	77.5	34.8

TABLE III: Properties of the designed Lissajous trajectories.

conducted five trials for each with the real hardware.

Fig. 5 is obtained purely by theoretical computation and demonstrates the feasible (achievable) set of accelerations for LMPC (orange) and NMPC (blue) for a specific Lissajous trajectory. Here, we also indicated the reference acceleration (green) of the prescribed trajectory we tested. As can be seen from the figure, since a small-angle assumption is needed for LMPC, the feasible set is quite narrow and the feasibility margin (red) is very small, amounting to 29%. On the other hand, NMPC has quite a large margin: 85% for the very same trajectory.

Table III provides the properties of the designed Lissajous trajectories, again obtained exclusively by theoretical computation, and Fig. 6 presents the mean, standard deviation and maximum tracking errors for the same trajectories with decreasing feasibility margin in physical reality. This data is quite revealing in several ways. First of all, for the high average feasibility (AF >50%) for LMPC, it is apparent that LMPC is more performant than NMPC when we consider the average RMSE. Next, if the average feasibility of LMPC is small (AF <50%), NMPC starts to outperform LMPC and the performance of LMPC degrades rapidly until failing completely to track the trajectory. Next, we see that PFM is significantly correlated with the standard deviation of the RMSE tracking error and is relatively large for $PFM < 0$ for the selected controller. As a result, although one can pay attention to AF if the average error is concerned, PFM is a strong indication of low tracking performance in specific regions of trajectory, hence affecting mainly the spread of error.

Table IV lists the average and standard deviations of CPU solver times for the differently parametrized Lissajous trajectories. As can be seen, all solutions were obtained inside the dedicated sample time with enough accuracy; note also that there is a 2-5 times CPU time difference between the solutions of LMPC and NMPC.

C. Stage 3: Validation on Arbitrary Trajectories

For this stage, various arbitrary trajectories involving different feasibility margins were designed in order to verify the

Cont. \ Type	Lissajous (1)	Lissajous (2)	Lissajous (3)	Lissajous (4)	Lissajous (5)
LMPC (40 Hz)	5.0 ±0.8	4.9 ±0.8	5.6 ±2.0	9.7 ±3.6	-
NMPC (20 Hz)	19.5 ±3.9	20.4 ±2.7	21.2 ±2.30	21.3 ±1.7	21.5 ±1.7

TABLE IV: Solver CPU times [ms] of the design trajectories.

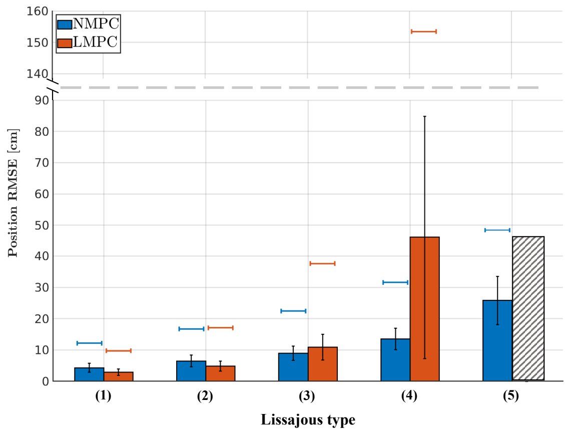

Fig. 6: Tracking errors obtained with the real quadrotor for Lissajous trajectories categorized by different maximum velocities and accelerations. Colored horizontal bar shows the maximum error, the black vertical bar represents the standard deviation while the end of a column indicates the mean error. Note that LMPC fails to track Lissajous (5).

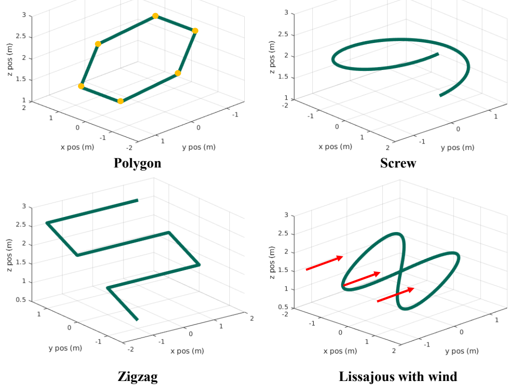

Fig. 7: Validation trajectories (green), disturbance vector (red). Note that when following the Polygon trajectory, the drone will stop for 2 seconds after each linear segment (at orange point). In contrast, it moves continuously without any pause when following the Zigzag trajectory.

Prop. \ Type	Polygon (mild)	Polygon (agile)	Screw (mild)	Screw (agile)	Zigzag	Lissajous (wind)
$\boldsymbol{a}_{max}$ (m/s^2)	6.1	10.1	0.35	0.73	10.11	0.37
$\boldsymbol{v}_{max}$ (m/s)	0.5	1.1	0.73	1.05	0.47	0.74
PFM-LMPC (%)	-504.3	-748	61.9	19.2	-348.1	65.0
AF-LMPC	85.1	46.6	62.2	21.9	91.4	75.2
PFM-NMPC (%)	-40.6	-102.2	98.9	84.2	-84.8	93.2
AF-NMPC	95.6	89.2	99.0	88.3	96.5	99.5

TABLE V: Properties of the validation trajectories.

findings of the previous stage. Fig. 7 illustrates graphically these trajectories: Polygon, Screw, Zigzag and Lissajous with emulated wind disturbance (average $\sim 1.2m/s$). The properties of the validation trajectories are given in Table V and the tracking errors are displayed in Fig. 8. The results obtained here mostly confirm the previous stage with one exception. First, AF for LMPC (with 50% threshold) determines the selection of MPC controller type if the average tracking error considered. This statement is accurate even for the second Lissajous trajectory with the wind disturbance, although the degradation of performance due to wind is more apparent for LMPC, i.e. large maximum error. Note that, for these experiments, the mean and standard deviations of the estimated disturbance forces are 0.43 N and 0.11 N, respectively. Second, small PFM ($< 30\%$) for LMPC is an effective indication of large maximum error in all trajectories except for the Polygon trajectory with mild dynamics. A possible reason for this exception is that there were pausing points at the each corner of the polygon, which reduce the maximum error occurred for slow speeds. As a result, if the maximum error is critical for the mission, NMPC should be preferred under these conditions.

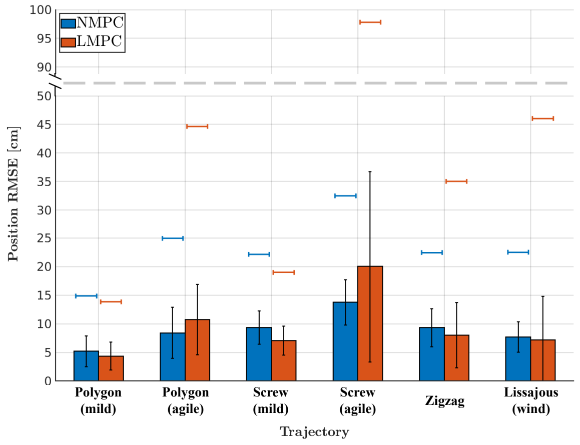

Fig. 8: Tracking errors for different validation trajectories in physical reality.

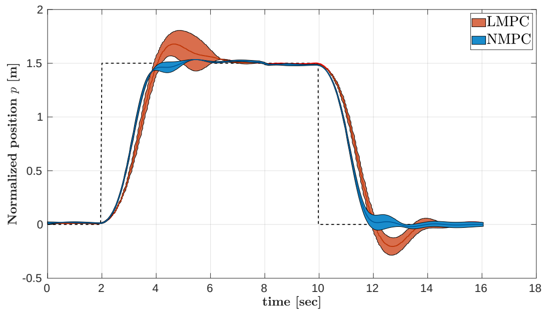

Fig. 9: Way-point tracking response for LMPC and NMPC in physical reality.

Finally, it is also crucial to compare the two controllers in more extreme conditions, where the feasibility margin is heavily negative. Figure 9 illustrates the mean and standard deviation of the position evolution for LMPC and NMPC for two arbitrary way-points. It is worth noting that each way-point is tracked five times in physical reality. Closer inspection of the plot reveals that NMPC has better step response characteristics: lower rise time, settling time, error variation and overshoot. Numerical values of this way-point experiment are reported in Table VI.

D. Discussion

Based on all the results above, considerations can be made in view of choosing the appropriate type of MPC controller:

- The biggest constraint is the computational budget to

	NMPC	**LMPC**
Rise time [s]	1.2 ± 0.1	1.3 ± 0.1
Overshoot [%]	2.5 ± 0.7	13.0 ± 6.2
Settling time [s]	4.1 ± 0.5	5.4 ± 0.5

TABLE VI: Step response metrics for goal tracking.

solve the MPC problem. Since, on average, generating a NMPC solution takes two to five times longer duration than computing a LMPC solution, for the applications where the robot has limited processing power, LMPC should be preferred.

- For the simpler trajectories (in terms of velocity and acceleration) with $AF > 50\%$, LMPC might reach or even outperforms NMPC in terms of average tracking error due to the capability to handle a higher control/estimation frequency and longer horizon length. The selection of LMPC could also be advantageous since it allows the inclusion of other essential computations.
- For more dynamic trajectories for LMPC with $AF < 50\%$, NMPC would be a better candidate since the performance of LMPC degrades significantly after this point.
- Very small PFM of a controller for the given trajectories is highly correlated with the maximum error that can occur during the tracking. NMPC seems to be a better option if the maximum or standard deviation of error is crucial for the mission.
- Time-critical and highly dynamic tasks such as waypoint tracking should be performed by NMPC due to its inherent solution quality and relaxed constraints.
- The mild wind disturbance combined with the trajectory did not change our selection criteria considering average RMSE. However, the effect of disturbance degrades the performance LMPC more compared to NMPC, increasing significantly the spread of tracking error.

IV. CONCLUSIONS

In this paper, we have designed various experiments in order to compare linear and nonlinear MPC for a MAV carrying out trajectory tracking maneuvers. We performed such comparison in a fair way by considering multiple aspects: numerical solver, trajectory richness, optimization of algorithmic meta-parameters, computational budget. The results of our experimental campaign not only report quantitative results for multiple representative trajectories but also allow us to shed light on relevant criteria for choosing one of the two schemes. In particular, by defining quantitative metrics considering the feasibility of a targeted trajectory with respect to the on-board resources and controller's capabilities, we have been able to propose a decisional recipe for the selection of the most appropriate MPC-based scheme for minimizing the RMSE position tracking error.

ACKNOWLEDGMENT

This work has been partially sponsored by the FCT grant [PD/BD/135151/2017], the FCT doctoral program RBCog and the FCT project [UIDB/50009/2013]

REFERENCES

[1] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE (3)," in *IEEE CDC*, 2010, pp. 5420–5425.

[2] M. Greeff and A. P. Schoellig, "Flatness-based MPC for quadrotor trajectory tracking," in *IEEE/RSJ IROS*, 2018, pp. 6740–6745.

[3] U. Eren, A. Prach, B. B. Koçer, S. V. Raković, E. Kayacan, and B. Açıkmeşe, "MPC in aerospace systems: Current state and opportunities," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 7, pp. 1541–1566, 2017.

[4] M. Neunert, C. De Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast NMPC for unified trajectory optimization and tracking," in *IEEE ICRA*, 2016, pp. 1398–1404.

[5] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-aware MPC for quadrotors," in *2018 IEEE/RSJ IROS*, 2018, pp. 5200–5207.

[6] M. Brunner, K. Bodie, M. Kamel, M. Pantic, W. Zhang, J. Nieto, and R. Siegwart, "Trajectory tracking NMPC for an overactuated MAV," in *IEEE ICRA*, 2020, pp. 5342–5348.

[7] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for near-time-optimal quadrotor flight," *arXiv preprint arXiv:2108.13205*, 2021.

[8] P. Roque, E. Bin, P. Miraldo, and D. V. Dimarogonas, "Fast model predictive image-based visual servoing for quadrotors," in *IEEE/RSJ IROS*, 2020, pp. 7566–7572.

[9] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi, "NMPC with enhanced actuator model for multi-rotor aerial vehicles with generic designs," *Journal of Intelligent & Robotic Systems*, vol. 100, no. 3, pp. 1213–1247, 2020.

[10] J. Schlagenhauf, P. Hofmeier, T. Bronnenmeyer, R. Paelinck, and M. Diehl, "Cascaded NMPC for realtime quadrotor position tracking," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 7026–7032, 2020.

[11] M. A. Gomaa, O. De Silva, G. K. Mann, and R. G. Gosine, "NMPC without terminal costs or constraints for multi-rotor aerial vehicles," *IEEE Control Systems Letters*, pp. 440–445, 2021.

[12] M. Bangura and R. Mahony, "Real-time MPC for quadrotors," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11 773–11 780, 2014.

[13] G. V. Raffo, M. G. Ortega, and F. R. Rubio, "An integral predictive/nonlinear h control structure for a quadrotor helicopter," *Automatica*, vol. 46, no. 1, pp. 29–39, 2010.

[14] J. C. Pereira, V. J. Leite, and G. V. Raffo, "NMPC on se (3) for quadrotor aggressive maneuvers," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 3, pp. 1–15, 2021.

[15] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.

[16] E. Kayacan, W. Saeys, H. Ramon, C. Belta, and J. M. Peschel, "Experimental validation of linear and nonlinear MPC on an articulated unmanned ground vehicle," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 5, pp. 2023–2030, 2018.

[17] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of NMPC and differential-flatness-based control for quadrotor agile flight," *arXiv preprint arXiv:2109.01365*, 2021.

[18] I. K. Erunsal, R. Ventura, and A. Martinoli, "NMPC for formations of multi-rotor micro aerial vehicles: An experimental approach," in *ISER*. Springer, 2020, pp. 449–461.

[19] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, 2020.

[20] O. Michel, "Webots: Professional mobile robot simulation," *Journal of Advanced Robotics Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[21] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011, pp. 15–41, 15-41.

[22] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[23] S. Omari, M.-D. Hua, G. Ducard, and T. Hamel, "Nonlinear control of VTOL UAVs incorporating flapping dynamics," in *IEEE/RSJ IROS*, 2013, pp. 2419–2425.

[24] D. Dias, "Distributed state estimation and control of autonomous quadrotor formations using exclusively onboard resources," Ph.D. dissertation, EPFL–IST, No. 9224, 2019.

[25] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[26] M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl, "Auto-generated algorithms for nonlinear MPC on long and on short horizons," in *IEEE CDC*, 2013, pp. 5113–5118.

[27] B. Houska, H. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.