MPC with Learned Residual Dynamics with Application on Omnidirectional MAVs

Maximilian Brunner, Weixuan Zhang, Ahmad Roumie, Marco Tognon, Roland Siegwart

Abstract—The growing field of aerial manipulation often relies on fully actuated or omnidirectional micro aerial vehicles (OMAVs) which can apply arbitrary forces and torques while in contact with the environment. Control methods are usually based on model-free approaches, separating a high-level wrench controller from an actuator allocation. If necessary, disturbances are rejected by online disturbance observers. However, while being general, this approach often produces sub-optimal control commands and cannot incorporate constraints given by the platform design. We present two model-based approaches to control OMAVs for the task of trajectory tracking while rejecting disturbances. The first one optimizes wrench commands and compensates model errors by a model learned from experimental data. The second one optimizes low-level actuator commands, allowing to exploit an allocation nullspace and to consider constraints given by the actuator hardware. The efficacy and real-time feasibility of both approaches is shown and evaluated in real-world experiments.

I. INTRODUCTION

The advancement of micro aerial vehicles (MAVs) in the recent years has come with increasing focus on aerial physical interaction tasks. Investigations started with pick-and-place tasks, continuing with contact-based inspection and push-and-slide operations [1]–[5], all the way to the manipulation of the environment [6], [7]. Tasks and applications involving aerial physical interaction are getting more complex year by year [8].

Different platforms have been developed to cope with different challenges [9], some being generic for research or a broad spectrum of tasks, other being more specialized for certain applications. Generally, aerial interaction requires a flying platform equipped with a manipulator that is designed to interact according to the desired task. This manipulator can be passive or actively controlled. Depending on the task, the flying platform needs to meet certain requirements to compensate for wrenches (i.e., forces and torques) which arise during the interaction. While underactuated platforms are capable of compensating for some limited wrenches, fully actuated and overactuated platforms offer more freedom in this matter. Fully actuated platforms can compensate for any reaction wrenches that appear during interaction while overactuation adds the benefit of redundancy. Furthermore, we refer to Omnidirectional MAVs (OMAVs) as platforms that can generate thrust in any direction, providing sufficient lift force to hover in any possible orientation [10].

However, full actuation and/or overactuation comes with new challenges and opportunities. First, the higher number of actuators gives potential for more unmodeled disturbances, e.g., through inaccurate thrust (or other actuator) mappings or airflow interferences. Second, it increases the complexity of finding optimal control inputs. Therefore, we intend to explore the possibilities of finding the optimal control inputs for performing a flight task while rejecting internal disturbances.

Most common controllers of MAVs are divided into three parts: (i) a high level controller for pose and/or force tracking (e.g., PD, impedance) which produces linear/angular acceleration commands, (ii) a wrench estimator to observe unmodeled disturbances, and (iii) an actuator allocation to convert high level commands into actuator commands. While this structure has proven to work reliably, it comes with a few drawbacks. First, the separation of a pose tracking controller and the actuator allocation does not allow to fully optimize the actuator commands for the execution of a desired task. Second, employing an online wrench estimator introduces time delays that can impair the actual flight performance.

The controller (i) initially has often been implemented by a PD or impedance controller that generates acceleration commands based on the tracking errors. More recently, model-based optimal control approaches have been studied as well. Model predictive control (MPC) can be specifically useful in situations where accurate and fast trajectory tracking is needed in the presence of actuator constraints and external disturbances [11]. In this context, [12] uses a MPC on an underactuated quadrotor in strong wind gusts, comparing different Kalman filters for disturbance estimation. In this case, the authors aimed at improving the tracking performance by rejecting external disturbances.
work the model based approach leads to higher position tracking accuracy than PID controllers. In a different work, [13] models aerodynamic disturbances by Gaussian Processes and fuses them in an MPC to improve high-speed flight maneuvers with quadrotors. In [14], we have presented an MPC-based control framework for trajectory tracking of omnidirectional micro aerial vehicles (OMAVs). Optimizing on the wrench level, the individual rotor speed and tilt rotor inputs were then found through the same allocation process as presented in [2]. One difficulty of model-based approaches is their dependence on an accurate model of the system. If this is not known or contains unmodeled disturbances, these errors need to be either estimated online or compensated by an adaptive MPC.

The wrench estimator (ii) can be implemented by a momentum-based observer (MBE) [2], [15], or another estimation framework such as a Kalman filter [12]. The estimator usually accounts for both external as well as internal disturbances. External disturbances can only be modeled to a certain degree (as they are caused by either interaction or unpredictable sources like wind gusts). Internal disturbances on the other hand are caused by unmodeled and unknown effects originating from the system itself. These can result from inaccurate hardware fabrication or complex aerodynamic effects. A common disadvantage of using an online observer is the inherent time delay as it requires state observations in order to estimate the current disturbance. Therefore, the disturbances can also be learned offline based on experimental data and then applied during a flight. This approach has been applied in [16] by using Gaussian Processes to learn the wrench residuals.

Lastly, the actuator allocation (iii) maps a wrench command into the actuator controls that will generate the desired wrench. It is determined by the particular actuation and geometry of the platform. According to the system, the actuator allocation problem can have a unique solution or, if the system is overactuated, an infinite number of solutions. This latter case can be exploited to achieve secondary objectives, like the minimization of energy or the optimization of actuation properties [17].

A. Related works

In order to cope with the above stated difficulties, various approaches have been investigated. In [18], the tracking controller and the allocation were in a single optimizer. This framework does not assume any linear model approximations nor does it depend on a cascaded control approach to decouple the translational and rotational dynamics of the rigid body. What is remarkable is the use of the derivatives of the individual propeller forces as control inputs which allows the direct translation of actual control inputs. Nevertheless, in the case of an OMAVs with actuated tilt angles, there is no explicit relationship between the generated individual forces and the actuator constraints. [19] presented a nonlinear model predictive control (NMPC) framework for overactuated MAVs with actively tiltable propellers. Two optimizers were compared, namely Interior Point Optimization (through IPOPT) and Sequential Quadratic Programming (SQP) in CasADI through ACADO. However, the framework is only evaluated in simulations in Gazebo.

Also different methods to deal with the problem of uncertain or unknown model dynamics have been presented, such as adaptive MPC, robust MPC (e.g. Tube-MPC [20], [21]), or learning-based MPC. A number of approaches exists to improve the MPC model by learning its true dynamics. [22] gives an overview on how MPC performance can be improved through learning from recorded data. Accordingly, this can be achieved by following the following approaches: (i) Learning the system dynamics, and/or (ii) learning the controller design, such as the optimal cost function or constraints. [23] followed the approach of Identification for control (IC4), which aims to not minimize the output prediction errors (i.e. to fit the MPC model to the real system as closely as possible), but rather to find a model that optimizes the control performance in closed loop. In this approach, the MPC acts as an outer loop (i.e., as a reference governor) that is based on a model of the inner loop, given by a fast PID controller. In this approach, closed loop experiments are repeated to find optimal control parameters through Bayesian Optimization. As mentioned in [24], “stochastic and robust MPC are suitable for handling unmodeled dynamics and rapidly changing disturbances”. However, they are conservative and not appropriate for adapting to constant parameters. In [25] the concept of learning-based MPC (LBMPC) was introduced and applied on a quadrotor in [26]. LBMPC uses a so-called oracle to learn the residual dynamics between a model and the true system dynamics. The oracle can be any linear or nonlinear parametric function whose parameters are adapted during the execution of the controller. In [26], an extended Kalman filter (EKF) is used for joint state and parameter estimation for a linear affine oracle. Other approaches such as Iterative Learning MPC [27], [28] are restricted to repetitive tasks, in which the performance can be improved by adapting the control inputs by learning from earlier iterations.

B. Contributions

In this work, we explore the methods of how to use MPC on an OMAV. Specifically, given by its tilt-rotor design, the OMAV of our work is capable of a high level of overactuation and potential internal disturbances, leading to model errors. We present the theory and experimental validation of an actuator-level NMPC that is able to generate real-time rotor speed and tilt angle commands for the task of trajectory tracking. Its knowledge of the actuator allocation allows to explore the actuation nullspace for an overactuated platform while respecting actuator constraints. Furthermore, we show how internal model disturbances can be learned and applied through a simple linear model inside a wrench-level MPC. Finally, we compare the performances of the different proposed approaches in various experiments.
II. MODELING

In this section we introduce the modeling of the system dynamics using Newton-Euler equations based on the following common assumptions: (i) The system is a single rigid body, (ii) mass and inertia matrix are constant, (iii) the center of mass (CoM) coincides with the geometric center of the system, and (iv) disturbance forces and moments can be reduced to a single wrench applied to the CoM.

A. Notation

We denote scalars by lowercase symbols, vectors \( v \) by lowercase bold symbols, and matrices \( M \) by uppercase bold symbols. If not specified differently, we use subscripts to indicate the frame of a vector \( Wv \). To represent orientations, we use unit quaternions \( q = \begin{bmatrix} q_w & q_x & q_y & q_z \end{bmatrix}^T \in \mathbb{R}^4 \), such that \( \|q\| = 1 \), as well as rotation matrices \( R_B \in SO(3) \). Quaternions and rotation matrices can be used interchangeably, such that they act as vector transformations, i.e., \( Wv = q \otimes Bv \otimes q^{-1} = R_B Bv \), where \( \otimes \) represents the quaternion multiplication.

B. Frame definitions

We will refer to three reference frames: the inertial world frame \( F_W = \{O_W, x_W, y_W, z_W\} \), the body frame \( F_B = \{O_B, x_B, y_B, z_B\} \) which is fixed to the geometric center of the OMAV, and the local frame \( F_L = \{O_L, x_L, y_L, z_L\} \), which is obtained by a pure yaw rotation of the platform yaw angle from the world frame. \( O \), represents the center of the generic frame \( F_* \), while \( \{x_*, y_*, z_*\} \) represents its unit axes. \( F_W \) is defined s.t. \( z_W \) is aligned with the gravity vector \( g = \begin{bmatrix} 0 & 0 & -9.81 \text{ m/s}^{-2} \end{bmatrix}^T \). Figure 2 gives an overview of the frames used in this work.

\[
\begin{align*}
\dot{p} &= R_B v \quad \text{(1a)} \\
\dot{q} &= \frac{1}{2} q \otimes 0 \quad \text{(1b)} \\
\dot{v} &= m^{-1}(f_a + \Delta f) + R_B g - \omega \times v \quad \text{(1c)} \\
\dot{\omega} &= J^{-1}(\tau_a + \Delta \tau - \omega \times (J\omega)) \quad \text{(1d)}
\end{align*}
\]

where we considered a residual wrench \( \Delta w = [\Delta f^T \Delta \tau^T]^T \in \mathbb{R}^6 \) acting on the CoM. The residual (or disturbance) forces and torques \( \Delta f \in \mathbb{R}^3 \) and \( \Delta \tau \in \mathbb{R}^3 \) account for all unmodeled effects, both internal and external, such as airflow interference (within rotor groups and between different rotor groups), hardware misalignments, or slightly different propeller characteristics. We assume that no further external or time-varying disturbances are present. Defining a state vector as \( x = [p^T q^T v^T \omega^T]^T \) we can then write the dynamic equations as

\[
\dot{x} = f_R(x, w_a, \Delta w).
\]

D. Allocation of actuator commands

The OMAV actuation can be described as follows: There are \( n_a \) tiltable arms attached to the body core, with each arm carrying \( n_{rra} \) rotors, resulting in a total number of \( n_r = n_a n_{rra} \) rotors. The total actuator wrench \( w_a \) is the result of the commanded rotor thrusts \( t \in \mathbb{R}^{n_r} \) and the current tilt angle configuration, given by the tilt angles \( \alpha \in \mathbb{R}^{n_a} \). The geometry of the platform determines the relation between the actuator commands and the total actuator wrench \( w_a \). In the following we present a method for the actuator allocation which computes actuator commands from a reference actuator wrench, i.e., \( (\alpha, t) = f_{alloc}(w_a) \).

**Actuator allocation**: We define the vector \( \tilde{t}(\alpha, t) \in \mathbb{R}^{2n_r} \) that describes the vertical and lateral thrust components of each propeller in the body frame. We can then write the relation between \( w_a \) and \( \tilde{t} \) by a linear function:

\[
\begin{align*}
w_a(\alpha, t) &= \begin{bmatrix} f_a \\ \tau_a \end{bmatrix} = A \tilde{t}(\alpha, t), \quad \text{(3a)} \\
\tilde{t}(\alpha, t) &= \begin{bmatrix} f_{1,t} \\ f_{1,v} \\ \vdots \\ f_{n_r,t} \\ f_{n_r,v} \end{bmatrix} = \begin{bmatrix} \sin(\alpha_1) t_{1l} \\ \cos(\alpha_1) t_{1l} \\ \vdots \\ \sin(\alpha_{n_r}) t_{nl} \\ \cos(\alpha_{n_r}) t_{nl} \end{bmatrix}.
\end{align*}
\]

Given this relation, the allocation matrix \( A \in \mathbb{R}^{6 \times 2n_r} \) is constant and can be obtained from the platform geometry. In order to compute the actuator commands \( u_a := [\tilde{t}^T]^T \).
from a given actuator wrench \( w_a \), we apply the Moore-Penrose Inverse:

\[
\dot{i} = A^\dagger w_a + (I - A^\dagger A) b
\]  

\[
\alpha_i = \tan^2 \left( \sum_{j} f_{j,l} \right) \quad \forall i = 1 \ldots n_a
\]  

\[
t_i = \sqrt{f_{i,l}^2 + f_{i,v}^2} \quad \forall i = 1 \ldots n_r.
\]

If the system is overactuated, the vector \( b \in \mathbb{R}^{nr} \) can be used to find solutions in the nullspace of \( A \). For \( b = 0 \) the norm of \( \dot{t} \) is minimized. Note that, due to the geometrical relation between \( \dot{t} \) and \( t \) in (3b), the minimization of \( \dot{t} \) corresponds to a minimization of \( t \). We will refer to this solution as the minimum norm or optimal allocation, resulting in the optimal commands denoted by \( w^*_a = [\alpha^* \; t^*]^\top \).

This procedure has the following properties: (i) the norm of the resulting thrusts is minimized, (ii) it is instantaneous, (iii) for \( b = 0 \) the mapping \( w_a \rightarrow (\alpha^*, t^*) \) is bijective, i.e. for each \( w_a \) there is a unique set of commands \( (\alpha^*, t^*) \).

While (i) and (iii) often provide advantages, (ii) can cause difficulties during fast motions. When \( w_a \) changes rapidly, it can result in unfeasible fast changes of actuator commands. This can pose difficulties in the common separation of a high-level wrench generation controller and a low-level allocation. In the following section, we will address this problem by introducing the allocation in the MPC formulation to impose cost and constraints on the actuator dynamics.

### III. Model Predictive Control Framework

In this section, we present two different model-based controllers for the task of free flight trajectory tracking with an overactuated aerial vehicle. Both methods are based on the same formalisms to model the system dynamics but differ in the level of detail to which the models are embedded and how residual wrenches are handled. Specifically, the first approach optimizes wrench commands and uses a default allocation, while the second approach operates on the actuator level.

We refer to these two proposed methods as Wrench-MPC (WMPC) and Actuator-MPC (AMPC), respectively. Both controllers rely on residual wrench estimates to account for otherwise unmodeled disturbances. These estimates can either be provided by an online estimator (EKF) or by a Residual Dynamics Model (RDM). Figure 3 illustrates the two different methods in a control block diagram and Table I compares the similarities and differences between the two methods.

#### A. MPC formulation

We first introduce a general formulation of the MPC problem. To this end, we define the state vector as \( x \in \mathcal{X} \subseteq \mathbb{R}^n \) and the input vector as \( u \in \mathcal{U} \subseteq \mathbb{R}^m \). We further assume that states and control inputs are constrained by the polytopes \( \mathcal{X} \) and \( \mathcal{U} \). The system is subject to its dynamics \( \dot{x} = f(x,u) \) discretized as \( x_{k+1} = g(x_k, u_k) \).

We also define the stage cost \( h(x, x_r) \) and terminal cost \( h_N(x_N, x_{r,N}) \). The discrete-time MPC problem is then formulated as the minimization of a cost function over a finite time horizon of \( N \) steps:

\[
\min_{u} \sum_{k=0}^{N-1} \left( h(x_k, x_{r,k}) \right)^2 + \left( u_k \right)^2_R
\]

subject to \( x_k \in \mathcal{X}, u_k \in \mathcal{U} \)

\[
x_{k+1} = g(x_k, u_k)
\]

\[
x_0 = x(t).
\]

The matrices \( Q, Q_N \in \mathbb{R}^{n \times n} \) and \( R \in \mathbb{R}^{m \times m} \) represent the state, terminal state, and input cost matrices, respectively. For the remainder of this section we present the details and differences of the two approaches and how (5) is adapted accordingly.

#### B. State and input vectors for different model formulations

For the two formulations of the WMPC and AMPC we use different definitions of state and input vectors.

1) Wrench-MPC: In the case of WMPC we define the state vector to comprise both the OMAV and the wrench states. The modeling of the state dynamics is equal to the rigid body dynamics in (2).

\[
x_W = [w_a^\top \; p^\top \; v^\top \; q^\top \; \omega^\top]^\top \in \mathbb{R}^{19}
\]

\[
u_W = \dot{w}_a \in \mathbb{R}^6.
\]

2) Actuator-MPC: While the system dynamics in AMPC are described equivalently to WMPC, we include the actuator commands \( u_a = [\alpha^\top \; t^\top]^\top \) in the state as well as the allocation (3a) in the MPC system dynamics.

\[
x_A = [\alpha^\top \; t^\top \; p^\top \; v^\top \; q^\top \; \omega^\top]^\top \in \mathbb{R}^{31}
\]

\[
u_A = \dot{u}_a \in \mathbb{R}^{18}.
\]

By including the actuator commands in the model, the MPC does not rely on the allocation procedure (4), thus allowing a larger exploration space of possible solutions. Additionally, we can impose constraints on actuator velocities and ensure continuity of the commands.
C. Cost vector

The cost vector \( h(x, x_r) \) is designed slightly different for the two different approaches of WMPC and AMPC. Both have in common that a reference trajectory is to be tracked, given by the time dependent variables \( p_r, v_r, q_r, \omega_r \).

1) Wrench-MPC: We employ a common definition of tracking errors to write the cost vector:

\[
\begin{align*}
    h(x_k, x_r, k) &= \begin{bmatrix}
        p_k - p_{r,k} \\
        v_k - v_{r,k} \\
        q_{e,k} \\
        \omega_k - R_{B}^{w} R_{B}^{w} \omega_{r,k}
    \end{bmatrix},
\end{align*}
\]

(8)

with \( q_e \in \mathbb{R}^3 \) as the vector that describes the rotation error between \( q \) and \( q_r \), such that

\[
q_r = q \otimes \Delta q, \quad \Delta q := \begin{bmatrix}
    1 \\
    q_e
\end{bmatrix}.
\]

(9)

2) Actuator-MPC: The actuator-based MPC further includes the tilt angles and propeller thrusts as state variables:

\[
\begin{align*}
    h(x_k, u_k) &= \begin{bmatrix}
        p_k - p_{r,k} \\
        v_k - v_{r,k} \\
        q_{e,k} \\
        \omega_k - R_{B}^{w} R_{B}^{w} \omega_{r,k} \\
        \alpha_k - \alpha_{r,k} \\
        t_k - t_{r,k}
    \end{bmatrix},
\end{align*}
\]

(10)

Generally, we aim to allow the actuator dynamics to evolve as freely as possible. There are different possibilities to penalize these states. Not penalizing the actuator states would allow the largest exploration freedom but would also lead to long optimization times of the MPC. Therefore, we chose to penalize the deviation of the actuator commands from the minimum norm commands (\( \alpha^*, t^* \)):

\[
\begin{align*}
    \alpha_{r,k} &= \alpha^*_k \\
    t_{r,k} &= t^*_k.
\end{align*}
\]

(11a)

(11b)

Tuning of the weight matrix \( Q \) then allows to give more or less range in deviating from the optimal allocation. The minimum norm commands (\( \alpha^*_k, t^*_k \)) are obtained assuming static hover, i.e., only exerting the force required to hold the platform weight in the current attitude.

D. Constraints

In order to obtain smooth control inputs and to account for unmodeled dynamic effects, we can employ hard constraints on any of the states and inputs.

1) WMPC: In the case of WMPC, we constrain the total actuator wrench \( w_a \) and its derivative \( \dot{w}_a \). This allows us to ensure that both the total wrench and the wrench rate remain in feasible bounds. Note that employing constraints on the wrench rate implicitly constrains the actuator rates due to the relation in (3a).

\[
\begin{align*}
-w_{a,max} \leq w_a \leq w_{a,max} \\
-w_{a,max} \leq \dot{w}_a \leq w_{a,max}
\end{align*}
\]

(12a)

(12b)

2) AMPC: One major advantage of AMPC is the possibility to constrain actuator commands directly. Therefore, we employ hard constraints on the thrusts, the thrust rates, and the tilt angle rates:

\[
\begin{align*}
    t_{min} \leq t &\leq t_{max} \\
    -t_{max} \leq \dot{t} &\leq t_{max} \\
    -\dot{\alpha}_{max} \leq \dot{\alpha} &\leq \dot{\alpha}_{max}
\end{align*}
\]

(13a)

(13b)

(13c)

E. Optimal problem result

Each MPC iteration of solving (5) returns a sequence of optimal inputs \( U = [u_0^*, \ldots, u_N^*] \) and associated states \( X = [x_0^*, \ldots, x_N^*] \). We use this state sequence to extract the optimal inputs and apply them as control inputs to the system — either \( w_a \) in the case of WMPC or \( u_a \) in the case of AMPC.

IV. COMPENSATION FOR DISTURBANCES

We identify model mismatches as a main cause that leads to non-optimal tracking of reference trajectories. Therefore, we introduce two methods to tackle this challenge: (i) an EKF-based disturbance observer, and (ii) a linear model that predicts disturbance wrenches based on experimental data. Both methods intend to predict the residual wrench \( \Delta w = [\Delta f^T, \Delta \tau^T]^T \) for each flight configuration.

A. Disturbance observer

We employ an EKF to estimate the disturbance force and torque in real time. We assume that most disturbances originate from internal model errors, e.g., from interfering air flows, inaccurate rotor-speed/thrust mapping, or misaligned tilt arms. Furthermore, we assume that these internal errors are independent of the platform yaw angle \( \psi \). Therefore, we estimate the disturbance force in the local frame \( F_L \), which is obtained by a pure yaw rotation of the platform yaw angle from the world frame, i.e., \( R_L = R_z(\psi) \). We use the following state vector \( \hat{x} \in \mathbb{R}^{19} \), inputs \( u_{EKF} \in \mathbb{R}^5 \), and

\[
\begin{array}{|c|c|}
\hline
\text{WMPC} & \text{AMPC} \\
\hline
\text{Output allocation} & \text{Actuator wrench derivative } w_a \\
\text{Minimum norm allocation after MPC} & \text{Actuator commands derivative } \dot{\alpha}, \dot{t} \\
\text{Disturbance compensation} & \text{Implicitly in MPC} \\
\text{Disturbance observer or model-based} & \text{Disturbance observer} \\
\hline
\end{array}
\]

TABLE I: Comparison of WMPC and AMPC.
measured wrench residuals for the IMU bias.

In order to compute the model parameters we use ridge regression to minimize the training error. We train each row of the model matrix individually. Let us define the feature matrix for \( n_s \) samples as \( X \in \mathbb{R}^{n_s \times n_f} \), and the vector of measured residuals for the \( i \)-th component of the wrench \( y_i \):

\[
X = \begin{bmatrix}
\tilde{x}_0 \\
\tilde{x}_1 \\
\vdots \\
\tilde{x}_{n_s-1}
\end{bmatrix}, \quad y_i = \begin{bmatrix}
\Delta w_{i,0} \\
\Delta w_{i,1} \\
\vdots \\
\Delta w_{i,n_s-1}
\end{bmatrix}, \quad C = \begin{bmatrix}
c_0 \\
c_1 \\
\vdots \\
c_5
\end{bmatrix}
\]

where \( c_i \in \mathbb{R}^{1 \times n_f} \) represents the \( i \)-th row of the model matrix. We can then find the model coefficients for each wrench component individually:

\[
c_i = \arg \min_{c_i} \| y_i - X c_i^\top \| + \lambda \| c_i \|, \quad i \in \{0, \ldots, 5\}.
\]

Note that (22) is a ridge regression with \( \lambda \) as the regularization parameter that helps avoid overfitting to the training data.

5) Application of the residuals in the control loop:
Generally, we only employ the residual model in the WMPC formulation only and not in AMPC. This is because the allocation nullspace exploitation in AMPC can result in various different actuator commands \( u_a \) for the same states, leading to different wrench residuals as a consequence and making a parametric model infeasible.

Within WMPC, we investigate two different methods of applying the residual model in the control framework.

a) In-MPC: In this approach, the model is implemented in the state dynamics of the MPC formulation. This allows the controller to respect the residual dynamics while computing an optimal input trajectory.
b) Post-MPC: In this approach, the MPC is agnostic of any model inaccuracies. Instead, the resulting optimal wrench commands \( \mathbf{w}_a^* \) are corrected after the MPC optimization.

C. Discussion

The first approach of employing an EKF-based disturbance observer has the advantage of being simple to implement while being able to adapt to most disturbances. On the other hand, it introduces a time delay into the system as it requires sensor measurements to adapt its estimates.

The model-based approach can be instantly applied in the controller framework, not adding any time delays. However, it requires a rich dataset upon which the model parameters can be fit. Additionally, selecting the correct features and a reasonable regularization parameter is not straightforward. Because of its linear formulation it can also only cover a limited area around a specific operating point.

V. EXPERIMENTAL VALIDATION

A. Implementation

1) Flying platform: We perform all experiments on our custom built OMAV. This platform is designed with 6 arms equally spaced around its body center. Each arm can be tilted individually by a Dynamixel XL430-W250-T servo. At the end of each arm a double rotor group, containing two KDE2315XF-885 motors with counter rotating 9x4.5 in propellers is mounted. The counter rotation of each rotor group minimizes the net torque of each arm, but the exact influence of the airflow interferences is unknown. The entire system is powered by a single 7000 mAh battery. Fully prepared for a flight its mass is 4.36 kg.

2) Software: The entire controller is implemented in ROS on an Intel NUC that is mounted on the platform. Reference trajectories are transmitted via WiFi from an offboard computer. The onboard computer runs the MPC solver and publishes either wrench or actuator commands (according to WMPC or AMPC), which are forwarded to a Pixhawk flight controller. For WMPC, the flight controller computes the optimal actuator commands and sends them to the actuators, while for AMPC it solely passes the commands through to the actuators.

The MPC optimizer is implemented using the ACADO framework. The system dynamics are discretized through direct multiple shooting and solved through an Implicit Runge Kutta method (Gauss-Legendre integrator of order 6). We use qpOASES as the QP solver.

B. Model for WMPC

We have tested different feature sets for the model. For the experiments we used a set made up from the commanded wrench and the roll and pitch angle, encoded by the 3rd row of the rotation matrix \( \mathbf{R}_B \), resulting in 9 features:

\[
\hat{x} = [\mathbf{w}_a^\top, \sin(\theta), \cos(\theta) \sin(\phi), \cos(\theta) \cos(\phi)]^\top.
\]  

(23)

It turned out that the regularization parameter \( \lambda \) is an essential tuning parameter for the closed loop stability. Low values of \( \lambda \) lead to a better model fit to the training data, but also to high residual predictions, resulting in strong countersteering and increasing instabilities. We therefore converged to choosing a high value of \( \lambda = 1 \times 10^5 \).

We recorded a training dataset which consists of two trajectories: the first one containing pure pitching and rolling motions of with a duration of 167 s and the second one tracking out a horizontal square, lasting 178 s. This dataset therefore contains both angular as well as linear acceleration data. The results of the model fit are presented in Fig. 4 and Table II. Both the table and the figure show the RMSE of the training data, i.e. of \( e_{\text{train},i} \). It can be seen that the model is able to compensate especially static offsets in both force and torque.

C. Experiments

The presented building blocks of WMPC/AMPC and disturbance observer/parametric model are combined in different ways to evaluate their performances in real-world experiments.

Specifically, we aim to analyze the following characteristics: (i) Position and attitude tracking performance, and (ii) velocity of actuator commands as well as nullspace exploitation of the AMPC.

We evaluate the controllers by comparing their capabilities to track given 6-degrees of freedom (DoF) trajectories in free space with an OMAV. We use four different trajectories and different velocities to evaluate the tracking performance: (i) square trajectory: horizontal square with 1 m leg length and reference velocities up to 3 m s\(^{-1}\), (ii) attitude trajectory: pure pitching and rolling up to 45\(^\circ\) while hovering at a fixed position with a duration of 27 s, (iii) lemniscate trajectory: combined position and attitude trajectory tracking a bent lemniscate with a two possible speeds (slow: 15 s duration,
In-MPC - D/o N/c

In-MPC

Post-MPC

Roll/pitch angles. Accordingly, the attitude RMSE is the attitude trajectory in different configurations.

Table III: Control parameters for WMPC and AMPC.

| Parameter          | WMPC          | AMPC          |
|--------------------|---------------|---------------|
| N                  | 10            | 20            |
| $\Delta t$         | 0.05 s        | 0.05 s        |
| $f_{a,max}$        | 20 N          | 20 N          |
| $\tau_{a,max}$     | 20 N m        | 20 N m        |
| $\alpha_{max}$     | n/a           | 10 rad s$^{-1}$|
| $\beta_{max}$      | n/a           | 16 N          |
| $\gamma_{max}$     | n/a           | 0.1 N         |
| $\tau_{max}$       | n/a           | 29 N s$^{-1}$  |

Table IV: Trajectory tracking RMSE of the WMPC approach for different controller configurations and trajectories. N/c = “No correction” (i.e., no disturbance compensation).

| Position err. [m] | N/c | In-MPC | Post-MPC | D/o |
|-------------------|-----|--------|----------|-----|
| Squares           | 0.198 | 0.172 | 0.150 | -   |
| Attitude trajectory | 0.139 | 0.088 | 0.104 | 0.095 |
| Lemniscate        | 0.137 | 0.091 | 0.085 | 0.100 |
| Lemniscate fast   | 0.146 | 0.109 | 0.108 | -   |

| Attitude err. [rad] | N/c | In-MPC | Post-MPC | D/o |
|---------------------|-----|--------|----------|-----|
| Squares             | 0.210 | 0.174 | 0.167 | -   |
| Attitude trajectory | 0.152 | 0.150 | 0.105 | 0.116 |
| Lemniscate          | 0.167 | 0.106 | 0.105 | 0.123 |
| Lemniscate fast     | 0.215 | 0.140 | 0.156 | -   |

Fig. 6: Tracking errors of the WMPC during the attitude trajectory for different controller configurations. The violins show the position RMSE on the left (in [m]) and the attitude RMSE on the right (in [rad]), respectively.

E. AMPC

For AMPC, we analyze the influence of the controller tuning (i.e., the actuator input weights) on the pose tracking performance and on the exploitation of the allocation nullspace. To this end, we perform two sets of experiments. In both sets, we first track the reference trajectory with high weights $w_\alpha$ and then with low weights. 1) We first apply this procedure on the square trajectory. The numerical results in Table V show that the tracking accuracy is higher for high actuator weights. 2) We then also use this procedure on tracking horizontal position reference steps of 1 m. Horizontal steps are particularly challenging as they require a sudden thrust direction change that can lead to infeasibly high tilt angle speeds. Figure 7 presents the results with a focus on the tilt angle commands. It shows that for low actuator weights, the tilt angle commands change rapidly and exhibit infeasibly high velocities. However, for higher weights, the tilt angle speeds are significantly lower while the position tracking is only slightly affected, resulting in 0.328 m in the first and 0.351 m in the latter case, respectively. Furthermore, note the tilt angle drift in the second period. As the weights $w_\alpha$ are lowered, the allocation nullspace is explored more freely, neglecting the objective to achieve maximum power efficiency.

Fig. 5: Bent lemniscate trajectory. Note that this trajectory also involves pitch angles up to 30°.
| High alpha cost | \( w_y \) | \( w_\alpha \) | \( w_{\alpha_k} \) | Pos. err. [m] | Att. err. [rad] |
|-----------------|--------|--------|---------------|-------------|-------------|
|                 | 1.0    | 10     | 10            | 0.111       | 0.212       |
| Low alpha cost  | 0.1    | 0.1    | 10            | 0.136       | 0.279       |

TABLE V: Trajectory tracking RMSE for different controllers and trajectories. For both AMPC tunings, the square trajectory was tracked three times, resulting in evaluation times of 30s.

Fig. 7: Position tracking, tilt angle commands and tilt angle speeds for lateral position steps. The actuator weights are high in the first highlighted period and low in the second highlighted period.

Fig. 8: Comparison of MPC solver times for WMPC and AMPC.

VI. CONCLUSION

We have presented a model predictive control framework for fully actuated or overactuated MAVs. Within this framework, we have employed two MPCs that optimize different control inputs and that use different approaches to cope with disturbances that arise from unknown internal effects.

The first one, WMPC, optimizes actuator wrenches which are thereafter translated by an optimal allocation into actuator commands. WMPC can consider disturbances either inside the model formulation (In-MPC) or as a posteriori correction of the optimal wrench commands (Post-MPC). The disturbances can either be estimated by an EKF or by an approximation through a linear model that is trained on experimental data.

On the other hand, AMPC optimizes actuator commands and relies on an EKF as a disturbance estimator. Due to its knowledge of the actuator allocation, it can exploit the allocation nullspace and direct constraints on actuator commands.

Finally, we have conducted experiments to show the performances of the two controllers and their respective up- and downsides. While the AMPC approach in theory models the system more accurately, it suffers from the higher complexity (both in tuning, repeatability, and in computation). Therefore, the WMPC approach remains the preferred method for most use cases.
