A Chebyshev Pseudospectral Method-based Model Predictive Control of UAVs for Trajectory Tracking and Collision Avoidance

D. M. K. K. Venkateswara Rao[1], Hamed Habibi[1], and Holger Voos[1,2]

***Abstract*— In this paper, we address the Unmanned Aerial Vehicle trajectory (UAV) tracking and obstacle avoidance problem, by proposing a novel Chebyshev pseudospectral method-based Model Predictive Control (MPC) formulation that is real-time implementable. In this formulation, a continuous-time integral form of the quadratic tracking error cost function is considered and its exact solution is obtained by the Clenshaw-Curtis quadrature rule. The state and control histories are approximated by Lagrange interpolating polynomials, with their coefficients as decision variables. These polynomials are proven to yield smooth control histories, unlike piecewise constant control inputs in the standard MPC. The collocation method is used to satisfy the dynamics and avoid computationally expensive numerical integration. This also allows us for a longer prediction horizon with the same number of decision variables without affecting the computational speed. The MPC is designed by considering the translational dynamics for determining acceleration inputs, which are subsequently used by the low-level controller to obtain desired thrust, orientation, and angular speeds. The performance of the proposed MPC is validated by indoor experiments.**

SUPPLEMENTARY MATERIAL

Video: https://youtu.be/eBeCuGa7B3E

I. INTRODUCTION

Quadrotor Unmanned Aerial Vehicles (UAVs) have drawn significant attention, with a variety of applications *e.g.*, reconnaissance, detection, etc [1, 2]. Within these applications, the design of reliable control algorithms has been the main focus of operating UAVs autonomously in an often-cluttered environment [3, 4]. Therefore, safety and collision avoidance play vital roles. Regardless of the trajectory planning and corresponding solutions, the trajectory tracking problem still dominates to be an important performance measure for UAVs, which are inherently underactuated with fully nonlinear and coupled dynamics [5]. The controllers of many existing autopilots are linear [6], which imposes challenges for tracking complex trajectories and performing agile maneuvers [7]. On the other hand, nonlinear controllers are prone to saturation issues and cannot handle constraints such as collision avoidance. Artificial potential field methods that address collision avoidance, suffer from local minima and oscillating behavior issues [8], and navigation function methods are only for static environment [9].

[1]D.M.K.K. Venkateswara Rao (corresponding author), Hamed Habibi, and Holger Voos are with the Automation and Robotics Research Group, Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg. mohan.dasari@uni.lu, hamed.habibi@uni.lu, holger.voos@uni.lu

[2]Holger Voos is also with the Faculty of Science, Technology, and Medicine (FSTM), Department of Engineering, University of Luxembourg.

With advances in microprocessors, optimization-based Model Predictive Control (MPC) has become the main paradigm, satisfying safety, speed, state, and control input constraints [1, 10–12]. However, many optimization-based approaches are computationally expensive, and real-time implementation might impose safety risks while the solution is being computed. Some of the existing MPC-based solutions consider the approximated, linearized, or decoupled dynamics of the UAV for faster convergence of the solver [13–17]. However, these approaches cannot render the actual performance of the UAV in practice. Since high-level guidance commands, *e.g.*, position or velocity ones, are obtained via these approaches, they rely on existing onboard controls for the computation of low-level control inputs [18].

Most MPC problem formulations [14, 19] require an increase of sampling intervals for accurate model prediction over a longer horizon. This increases the number of decision variables, stemming from the numerical integration methods that are included in most solvers [18]. This increases the computational burden and convergence time. Moreover, the use of numerical integration accumulates the integration error over poorly sampled prediction horizons. Additionally, in standard MPC approaches, the control input is a piecewise constant between the sampling intervals within the prediction horizon. The unrealistic step changes of control inputs affect the accuracy of predictions [20]. To resolve this, some approaches, *e.g.*, [1, 5, 17] increase the number of sampling intervals, rendering more continuous-like control inputs within the prediction horizon. This further increases the number of decision variables, resulting in a heavy optimization problem.

Motivated by the aforementioned challenges, we address the UAV trajectory tracking and collision avoidance problem, considering its nonlinear underactuated dynamics. We propose an MPC formulation using a Chebyshev pseudospectral method to determine acceleration control inputs. We design low-level controllers for determining desired orientation, and angular speeds, which prevail the works [14–17], that only consider the approximated, linearized or decoupled models. The proposed MPC formulation also aims at real-time implementation. The formulation approximates state and control histories within a prediction horizon by Lagrange interpolating polynomials with coefficients as decision variables. It has a continuous-time integral form of the quadratic error cost function that is determined exactly using the Clenshaw-Curtis quadrature rule. We avoid computationally expensive numerical integration of dynamics for state prediction within each prediction horizon by using the collocation method. The main contributions of the paper are as follows.

- In the proposed MPC formulation, due to the use of the collocation method and exact integration, the accuracy of the solution is not affected if the prediction horizon is increased reasonably. Also, as the number of decision variables remains the same for a longer prediction horizon, computational speed is not affected, which allows to increase in the number of constraints.
- The choice of Chebyshev-Gauss-Lobatto (CGL) interpolation points allows a highly accurate approximation of state trajectories. The dense distribution of interpolation points at the initial and terminal boundary points avoids the Runge phenomenon.
- State and control histories within the prediction horizon are C^∞ continuous, and their time derivatives can be determined exactly. This allows the use of automatic differentiation and incorporation of complex forms of constraints and gradients, Jacobians, and Hessians are computed exactly. The smooth approximation of control histories is more realistic compared to piecewise constant control inputs in a standard MPC. Accordingly, we can also incorporate control input rate constraints.
- We generate a problem-specific MPC solver with real-time implementation capability which allows us to conduct experiments to highlight the applicability of the proposed approach in practice.

The following notations are used. Underlined variables, *e.g.*, $\underline{x}$, and $\underline{x}^B$ represent vectors in the inertial and body frames of reference, respectively. $\mathbb{R}$, $\mathbb{R}_{>0}$, and $\mathbb{Z}_{\geq 0}$, are the real numbers, positive real numbers, and no negative integer numbers sets, respectively. $\mathbb{P}_N$ denotes the sets of polynomials order of N. $\mathsf{SO}(3)$ represents special orthogonal matrices, *i.e.*, $\mathsf{SO}(3)=\{R|R\in\mathbb{R}^{3\times 3}, R^TR=RR^T=I_3, det(R)=1\}$. $g=9.81(m/s^2)$ represents gravity. Symbol $\times$ denotes the cross product of vectors. $\underline{x}\leq\underline{y}$ denotes element-wise inequality for vectors $\underline{x}$ and $\underline{y}$. $R\succeq 0$ denoted matrix inequality.

II. MODEL DESCRIPTION AND PRELIMINARIES

A. UAV Dynamics

The UAV's nonlinear dynamics can be presented as [21]

$$\dot{\underline{p}}(t)=\underline{v}(t), \tag{1a}$$
$$\dot{\underline{v}}(t)=\frac{1}{m}\left(R(t)\underline{F}^B_{th}(t)-m\underline{g}\right), \tag{1b}$$
$$\dot{\underline{\Phi}}(t)=R_q^{-1}(t)\underline{\omega}^B(t), \tag{1c}$$
$$\dot{\underline{\omega}}^B(t)=J^{-1}\left(\underline{\tau}^B(t)-\underline{\omega}^B(t)\times J\underline{\omega}^B(t)\right), \tag{1d}$$

where, $\underline{p}(t)=[p_x(t),p_y(t),p_z(t)]^T\in\mathbb{R}^3$ and $\underline{v}(t)=[v_x(t),v_y(t),v_z(t)]^T\in\mathbb{R}^3$ are the center of mass' position and velocity vectors, respectively. $\underline{\omega}^B(t)=[p(t),q(t),r(t)]^T\in\mathbb{R}^3$ is the angular velocity in the body frame of reference. m is the mass and $J\in\mathbb{R}^{3\times 3}$ is the mass moment inertia. $\underline{\Phi}(t)=[\phi(t),\theta(t),\psi(t)]^T\in\mathbb{R}^3$ is the Euler angle vector and $\underline{F}^B_{th}(t)=[0,0,f_{th}(t)]^T\in\mathbb{R}^3$ is the thrust vector in the body frame of reference. $\phi(t)$, $\theta(t)$, and $\psi(t)$ denote roll, pitch, and yaw angles, respectively. $f_{th}(t)\in\mathbb{R}_{\geq 0}$ is thrust and $\underline{g}=[0,0,g]^T\in\mathbb{R}^3$ is the vector of gravity in the inertial frame of reference, whilst $\underline{\tau}^B(t)\in\mathbb{R}^3$ is the torque vector in the body frame of reference. The rotation matrix $R(t)\in\mathsf{SO}(3)$ transforms a vector from body to inertial frame of reference, and the transformation matrix $R_q(t)\in\mathbb{R}^{3\times 3}$ are given as:

$$R(t)=\begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix}, \tag{2a}$$

$$R_q(t)=\begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{bmatrix}, \tag{2b}$$

where, C_α and S_α denote cosine and sine of $\alpha(t)$, respectively. For the sake of notational simplicity, we present the translational dynamics, *i.e.*, (1b), as

$$\dot{\underline{v}}(t)=\frac{1}{m}\underline{F}(t)-\underline{g}, \tag{3}$$

where, $\underline{F}(t)=R(t)\underline{F}^B_{th}(t)=[f_x(t),f_y(t),f_z(t)]^T\in\mathbb{R}^3$. Therefore, (1a) and (1b), can be readily represented as

$$\dot{\underline{x}}(t)=\underline{f}(\underline{x}(t),\underline{u}(t)), \tag{4}$$

where $\underline{x}(t)=[\underline{p}(t)^T,\underline{v}(t)^T]^T\in\mathbb{R}^6$, $\underline{u}(t)=[u_x(t),u_y(t),u_z(t)]^T=\underline{F}(t)/m\in\mathbb{R}^3$ is the acceleration input vector, and $\underline{f}(\underline{x}(t),\underline{u}(t))=[\underline{v}(t)^T,\underline{u}(t)^T-\underline{g}^T]^T$.

B. Technical preliminaries

The following lemmas are used in the design procedure.

Lemma 1. *Clenshaw-Curtis quadrature rule [22]: Consider $t\in[-1,1]$ and the polynomial $p(t)\in\mathbb{P}_N$, where $N\in\mathbb{Z}_{\geq 0}$. Then,*

$$\int_{-1}^{1}p(t)dt=\sum_{k=0}^{N}p(t_k)w_k, \tag{5}$$

is exact, where t_k are CGL points chosen as

$$t_k=\cos\left(\frac{\pi k}{N}\right), \tag{6}$$

for $k=0,...,N$. For even N, the weights w_k are obtained as

$$w_0=w_N=\frac{1}{N^2-1}, \tag{7a}$$
$$w_k=\frac{2}{N}\left[\gamma(0,k)+\gamma(N/2,k)+2\sum_{i=1}^{N/2-1}\gamma(i,k)\right], \tag{7b}$$

for $k=1,...,N-1$, and for odd N as

$$w_0=w_N=\frac{1}{N^2}, \tag{8a}$$
$$w_k=\frac{2}{N}\left[\gamma(0,k)+2\sum_{i=1}^{(N-1)/2}\gamma(i,k)\right], \tag{8b}$$

where

$$\gamma(i,k)=\frac{1}{1-4i^2}\cos\frac{2\pi ik}{N}. \tag{9}$$

Lemma 2. *[21] Consider $\underline{F}(t)$ in (3), obtained for the desired trajectory $\underline{p}_d(t)=[p_{x_d}(t),p_{y_d}(t),p_{z_d}(t)]^T\in\mathbb{R}^3$. Compute the commanding thrust $f_{th}(t)$, the desired Euler angle*

$\underline{\Phi}_d(t) = [\phi_d(t), \theta_d(t), \psi_d(t)]^T$, *and desired angular velocities* $\underline{\omega}_d^B(t)$, *as*

$$f_{th}(t) = \sqrt{f_x^2(t) + f_y^2(t) + f_z^2(t)}, \tag{10a}$$

$$\theta_d(t) = tan^{-1}\left(\frac{C_{\psi_d}(t)f_x(t) + S_{\psi_d}(t)f_y(t)}{f_z(t)}\right), \tag{10b}$$

$$\phi_d(t) = sin^{-1}\left(\frac{S_{\psi_d}(t)f_x(t) - C_{\psi_d}(t)f_y(t)}{f_{th}(t)}\right), \tag{10c}$$

$$\underline{\omega}_d^B(t) = -\lambda_\omega R_q(\underline{\Phi}(t))\,(\underline{\Phi}(t) - \underline{\Phi}_d(t)), \tag{10d}$$

respectively, for a given yaw angle $\psi_d(t)$, *where* $\lambda_\omega \in \mathbb{R}_{>0}$ *is a design parameter. Then, the rotational dynamics* (1c) *and* (1d) *is stable, which further yields* $\underline{p}(t)$ *to converge to* $\underline{p}_d(t)$.

Assumption 1. *It is assumed that* $\underline{p}$, $\underline{v}$, *and* $\underline{\Phi}$ *are all available [23]. The desired trajectory* $\underline{p}_d(t)$ *is known,* C^2 *continuous and feasible so that the desired accelerations are within the control input bounds [21]. The fast low-level onboard angular-rate controller tracks the commanded angular velocities* $\underline{\omega}_d(t)$ *accurately.*

C. Problem statement

Considering (1), we propose an MPC formulation to steer the UAV position $\underline{p}(t)$ towards the desired trajectory $\underline{p}_d(t)$, for any initial condition $\underline{p}(0)$ and satisfying the constraints, aimed at faster convergence, and real-time implementation. Then, the proposed MPC problem is solved in real-time to obtain the $\underline{u}(t)$ in (4). Subsequently, the commanding thrust $f_{th}(t)$, the desired Euler angles $\underline{\Phi}_d(t)$ and the angular velocities $\underline{\omega}_d^B(t)$, are determined. Finally, the motor commands are determined by the low-level onboard angular-rate controller.

III. MPC DESIGN

A. MPC Formulation

The MPC problem is formulated as a trajectory optimization problem as

$$[\underline{x}^*(t), \underline{u}^*(t)] = \underset{[\underline{x}(t), \underline{u}(t)]}{\text{argmin}}\, \mathcal{J}(.), \tag{11}$$

with a continuous-time integral form of the cost function as

$$\mathcal{J}(.) = \int_{t_0}^{t_0+T} \Big((\underline{x}(t) - \underline{x}_d(t))^T Q(\underline{x}(t) - \underline{x}_d(t)) + (\underline{u}(t) - \underline{u}_d(t))^T R(\underline{u}(t) - \underline{u}_d(t))\Big)dt, \tag{12}$$

where $\underline{x}_d(t) = [\underline{p}_d(t)^T, \dot{\underline{p}}_d(t)^T]^T \in \mathbb{R}^6$, and $\underline{u}_d(t) = \ddot{\underline{p}}_d(t) + \underline{g}$ subject to the dynamics, nonlinear, bound, and initial condition constraints as

$$\dot{\underline{x}}(t) = \underline{f}(\underline{x}(t), \underline{u}(t)), \tag{13}$$

$$\underline{h}(\underline{x}(t), \underline{u}(t), \underline{\rho}(t)) \leq 0, \tag{14}$$

$$\underline{x}_L \leq \underline{x}(t) \leq \underline{x}_U, \tag{15}$$

$$\underline{u}_L \leq \underline{u}(t) \leq \underline{u}_U, \tag{16}$$

$$\underline{x}(t_0) = \underline{x}_{t_0}, \tag{17}$$

where t_0 is the initial time, T is the prediction horizon, $Q \succ 0 \in \mathbb{R}^{6\times6}$ and $R \succeq 0 \in \mathbb{R}^{3\times3}$ are positive definite and semi-definite weighting matrices, respectively. $\underline{h}(\cdot)$ and $\underline{\rho}(t)$ are the constraints and parameters vectors, respectively, of appropriate dimensions. $\underline{x}_L$ and $\underline{x}_U$ are the lower and upper bound vectors of states, respectively, $\underline{u}_L$ and $\underline{u}_U$ are the lower and upper bound vectors of control inputs, respectively, and $\underline{x}_{t_0}$ is the initial state. The inequalities (14)-(16) denote element-wise inequality for vectors.

Remark 1. *It should be noted that* (14) *and* (15) *may result in infeasibility sometimes, and it can be resolved by adding soft constraints and penalizing the slack variables by including them in the cost function as weighted quadratic terms.*

B. NLP Transcription

The continuous-time form of the MPC problem (11) is transcribed into an NLP problem with a finite number of decision variables using a Chebyshev pseudospectral method. To do so, we transform time into a non-dimensionalized time variable, $\tau \in [-1, 1]$, as

$$\tau = \frac{2(t - t_0)}{T} - 1. \tag{18}$$

Using this in (4) yields a non-dimensionalized dynamics as

$$\dot{\underline{x}}(\tau) = \frac{T}{2}\underline{f}(\underline{x}(\tau), \underline{u}(\tau)). \tag{19}$$

The state and control histories of (19) are approximated by Lagrange interpolating polynomials as

$$\underline{x}(\tau) = \sum_{i=0}^{N} \underline{x}_i \phi_i(\tau), \tag{20a}$$

$$\underline{u}(\tau) = \sum_{i=0}^{N} \underline{u}_i \phi_i(\tau), \tag{20b}$$

where N is the polynomial order, $\underline{x}_i \in \mathbb{R}^6$ and $\underline{u}_i \in \mathbb{R}^3$ are node point vectors, and ϕ_i are Lagrange basis functions as [24]

$$\phi_i(\tau) = \prod_{\substack{j=0 \\ j\neq i}}^{N} \frac{\tau - \tau_j}{\tau_i - \tau_j}, \tag{21}$$

in which τ_i is the i^{th} interpolating point. The time-derivative of (20a) can be obtained analytically as

$$\dot{\underline{x}}(\tau) = \sum_{i=0}^{N} \left(\underline{x}_i \phi_i(\tau) \sum_{\substack{j=0 \\ j\neq i}}^{N} \frac{1}{\tau_i - \tau_j}\right). \tag{22}$$

For $k = 0, ..., N$, choose CGL points as the interpolation points given by

$$\tau_k = \cos\left(\frac{\pi k}{N}\right). \tag{23}$$

By definition, the value of Lagrange polynomials is exactly equal to the node points at interpolations points as

$$\begin{aligned} \underline{x}_k &= \underline{x}(\tau_k), \\ \underline{u}_k &= \underline{u}(\tau_k). \end{aligned} \tag{24}$$

By using Lemma 1, transformation (18), (20), and (24) the exact solution of the continuous-time integral cost function (12) is determined as a finite sum as

$$\mathcal{J}(.) = \frac{T}{2}\sum_{k=0}^{N} w_k \Big((\underline{x}_k - \underline{x}_d(\tau_k))^T Q (\underline{x}_k - \underline{x}_d(\tau_k)) + (\underline{u}_k - \underline{u}_d(\tau_k))^T R (\underline{u}_k - \underline{u}_d(\tau_k)) \Big). \tag{25}$$

On the other hand, collocation of the approximating state and control polynomials in (20) at the interpolation points results in a finite set of nonlinear algebraic equations. Similarly, nonlinear constraints in (14) are also chosen to satisfy at the interpolation points using node points. Considering (24), the bound constraints on states and controls directly translate to bounds on node points. Finally, using (25), the MPC problem of (11) is transcribed into the following NLP problem:

$$[\underline{x}_k^*, \underline{u}_k^*] = \underset{[\underline{x}_k, \underline{u}_k]}{\operatorname{argmin}} \mathcal{J}(.), \tag{26}$$

subject to

$$\sum_{i=0}^{N} \left(\underline{x}_i \phi_i(\tau_k) \sum_{\substack{j=0 \\ j\neq i}}^{N} \frac{1}{\tau_i - \tau_j} \right) = \frac{1}{2} T \underline{f}(\underline{x}_k, \underline{u}_k), \tag{27}$$

$$\underline{h}(\underline{x}_k, \underline{u}_k, \underline{\rho}_k) \leq 0, \tag{28}$$

$$\underline{x}_L \leq \underline{x}_k \leq \underline{x}_U, \tag{29}$$

$$\underline{u}_L \leq \underline{u}_k \leq \underline{u}_U, \tag{30}$$

$$\underline{x}_0 = \underline{x}_{t_0}, \tag{31}$$

for $k = 0, ..., N$, and $\underline{\rho}_k = \underline{\rho}(\tau_k)$.

C. Controller Design

By solving an instance of (26), at the time t_0, the commanding acceleration input vector $\underline{u}(t_0)$ is computed in a receding fashion as

$$\underline{u}(t_0) = \underline{u}_0^*. \tag{32}$$

Then, using Lemma 2, we compute the commanding thrust $f_{th}(t_0)$, the desired Euler angles $\underline{\Phi}_d(t_0) = [\phi_d(t_0), \theta_d(t_0), \psi_d(t_0)]^T$, and angular velocities $\underline{\omega}_d^B(t_0)$, for a given yaw angle $\psi_d(t_0)$ as

$$f_{th}(t_0) = \sqrt{f_x^2(t_0) + f_y^2(t_0) + f_z^2(t_0)}, \tag{33a}$$

$$\theta_d(t_0) = \tan^{-1}\left(\frac{C_{\psi_d}(t_0) f_x(t_0) + S_{\psi_d}(t_0) f_y(t_0)}{f_z(t_0)} \right), \tag{33b}$$

$$\phi_d(t_0) = \sin^{-1}\left(\frac{S_{\psi_d}(t_0) f_x(t_0) - C_{\psi_d}(t_0) f_y(t_0)}{f_{th}(t_0)} \right), \tag{33c}$$

$$\underline{\omega}_d^B(t_0) = -\lambda_\omega R_q(\underline{\Phi}(t_0)) (\underline{\Phi}(t_0) - \underline{\Phi}_d(t_0)), \tag{33d}$$

where $m\underline{u}(t_0) = [f_x(t_0), f_y(t_0), f_z(t_0)]^T$, and $\lambda_\omega \in \mathbb{R}_{>0}$. After commanding the control inputs, a new instance of NLP problem (26) at $t_0 + \delta t$ is solved for the new initial condition $x(t_0 + \delta t)$, where $\delta t \in \mathbb{R}_{>0}$ is the time-step of the controller, and the process is repeated. The main properties of the designed controller are summarized in the following proposition.

Proposition 1. *Consider the UAV dynamics (1), represented as (4) and satisfying Assumption 1. Design the control law (33), by solving the NLP problem (26) with constraints (27)-(31) and using (32). Then, for any initial condition $\underline{x}(t_0)$,*

i) The solution of (12) is exact and not affected by any numerical integration errors.

ii) The increase of prediction horizon T does not change the number of decision variables. Therefore, the computational complexity remains the same for a longer prediction horizon for same level of accuracy.

iii) The control histories of (33) are C^∞ within the prediction horizon of every instance of MPC problem (11), transcribed into an NLP problem (26).

iv) The UAV trajectory $\underline{p}(t)$ converges to feasible $\underline{p}_d(t)$.

Proof:

i) The cost function of (25) is the exact solution of the continuous-time integral form (12), in accordance to Lemma 1. Also, the numerical integration of the dynamics (13), is not required since the state histories are represented by Lagrange interpolating polynomials (20) and the dynamics are enforced in the form of collocation constraints in (27).

ii) In standard MPC, to maintain the same level of accuracy, one needs to increase the number of intervals to increase the prediction horizon. This in turn increases the number of decision variables involved in any numerical integration method that increases the computational complexity and time. In our approach, the number of decision variables is fixed for a reasonably longer horizon. Also, the collocation method ensures that the accuracy is not affected due to a longer prediction horizon, hence the computational complexity remains the same.

iii) The control histories are approximated by Lagrange interpolating polynomials (20), which are continuous and of finite order. The time derivatives of the Lagrange interpolating polynomials are exact and also yield continuous polynomials. Therefore, the control histories are C^∞. Compared to the standard MPC, in which control histories are piecewise constant, our approximation of control inputs is more realistic, and control-rate constraints can be easily incorporated.

iv) The control inputs (33) are designed in accordance to Lemma 2, in a receding fashion for each instance of NLP problem (26). Therefore, the convergence is guaranteed for feasible $\underline{p}_d(t)$. □

Remark 2. *In the NLP problem (26) with constraints (27)-(31), we have encapsulated the nonlinear inequalities in (28). This can represent the safety constraints such as avoiding an obstacle at $\underline{p}_o(t)$ as $\underline{h}(.) \triangleq R_o - \left\| \underline{p}(t) - \underline{p}_o(t) \right\|$ with the safety radius R_o, as studied in the Sec. IV.*

Remark 3. *It should be noted that, although, increasing the prediction horizon does not affect the computational requirements, higher-order approximation of states and inputs is necessary to accurately capture rapidly evolving scenarios, with a negligible increase in computational requirements.*

IV. EXPERIMENTAL RESULTS AND VALIDATION

A. Experimental setup

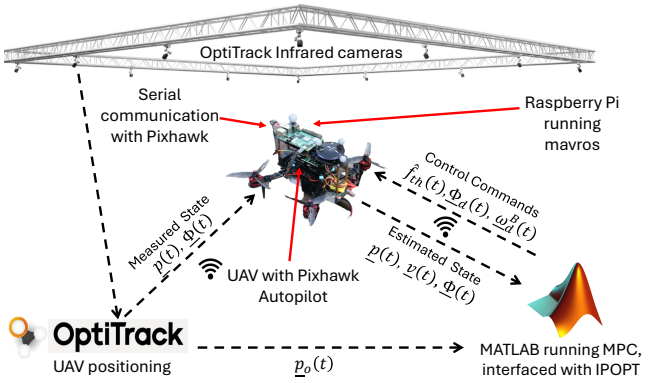

Fig. 1: Illustration of the experimental setup.

The experimental setup is illustrated in Fig. 1. The UAV is a customized QAV250 model equipped with a Pixhawk autopilot and Raspberry Pi onboard computer. An optitrack system is used for localization at a rate of 120 Hz. The autopilot receives the position and orientation information via the onboard computer and fuses them with accelerometer and gyroscope measurements to estimate the state of the UAV. A Robot Operating System (ROS) and `mavros` are used for communication between different components over a local network. The MPC is implemented at a rate of 100 Hz on a ground computer running Matlab software interfaced with an IPOPT solver. The computed $f_{th}(t)/m$ is scaled to $[0,1]$, representing the null and full speed of motors, respectively. The scaling factor is obtained experimentally by using the commanded normalized thrust f_{hover}, obtained by checking the ROS topic `/mavros/target_actuator_control` while hovering the UAV at a fixed point as $a_{scale} = g/f_{hover}$. Then, the computed $f_{th}(t)$ is normalized as $\hat{f}_{th}(t) = \frac{f_{th}(t)}{a_{scale} m}$. The normalized thrust, desired Euler angles, and angular-rate commands are sent to the autopilot, and the onboard low-level attitude-rate controller determines motor commands.

B. Solver Implementation

Several software strategies are designed and executed to facilitate real-time implementation of the proposed MPC. The code to transcribe the MPC problem into an NLP

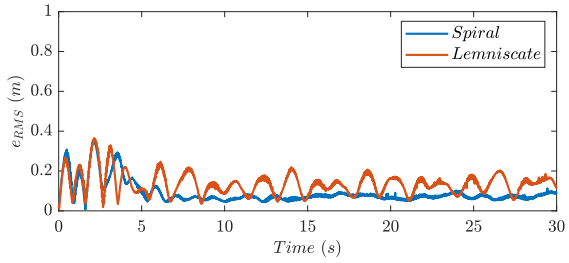

Fig. 2: Time histories of RMS errors for Spiral and Lemniscate maneuvers.

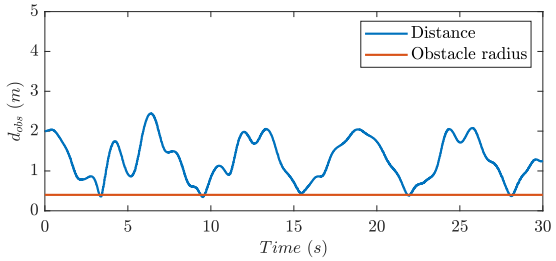

Fig. 3: Time history of distance of UAV to obstacle for Circle maneuver.

problem is written in Matlab. IPOPT solver is used to solve the resulting NLP problem. To speed up the computational time, the gradient of the cost function, Jacobian of constraints, and Hessian of the augmented cost function are determined analytically using CasADi [25]. To prevent frequent and time-consuming callbacks between IPOPT and Matlab, CasADi was used to generate C code for the cost function, constraints, gradient, Jacobian, and Hessian functions. A problem-specific MEX interface is created combining the generated C code and IPOPT solver. Static pointers are used to store the previous solution on the MEX side and use the warm-start capabilities of IPOPT. The software architecture, generation, and implementation of the proposed MPC solver are illustrated in Fig. 4.

C. Results

To study the proposed MPC performance and its real-time capabilities, two trajectory tracking maneuvers and one obstacle avoidance maneuver are carried out. A spiral trajectory with $\underline{p}_d(t) = [\cos(t), \sin(t), 0.75 + 0.025t]^T$, a Lemniscate trajectory with $\underline{p}_d(t) = [\cos(t), \cos(t)\sin(t), 0.75 + 0.025t]^T$, and a circle trajectory with $\underline{p}_d(t) = [\cos(t), \sin(t), 0.75]^T$ are chosen as reference trajectories. A Tello drone is made to hover at $[0, -1, 0.75]^T$ as an obstacle for the Circle maneuver. The MPC is designed with a prediction horizon of $T = 1$ s. The weighting matrices in the cost function (25) are chosen as $Q = diag([10, 10, 10, 1, 1, 1])$ and $R = diag([0.1, 0.1, 0.1])$. The Lagrange interpolating polynomials in (20) are chosen of order $N = 10$. Therefore, $N+1$ CGL points are chosen using (23) and quadrature weights w_k in (25) are obtained using (7). The bounds in (29) and (30), are chosen as $\underline{x}_L = [-2, -2, 0, -5, -5, -5]^T$, $\underline{x}_U = [2, 2, 3, 5, 5, 5]^T$, $\underline{u}_L = [-3, -3, 0]^T$, and $\underline{u}_U = [3, 3, 14]^T$. The parameter in the angular-rate controller (33d) is chosen as $\lambda_\omega = 1$. For the Circle maneuver, the obstacle avoidance constraint is as

$$\underline{h}(.) \triangleq 0.16 - (p_x(t) - p_{x_o})^2 - (p_y(t) - p_{y_o})^2. \qquad (34)$$

a_{scale} is determined experimentally as 35.30. The video of the experimental validation is available in this link[1].

The RMS errors histories calculated as $e_{RMS}(t) = \sqrt{\underline{p}(t) - \underline{p}_d(t)}$ for the Spiral and Lemniscate maneuvers are presented in Fig. 2. It can be seen that in both maneuvers, it

[1] https://youtu.be/eBeCuGa7B3E

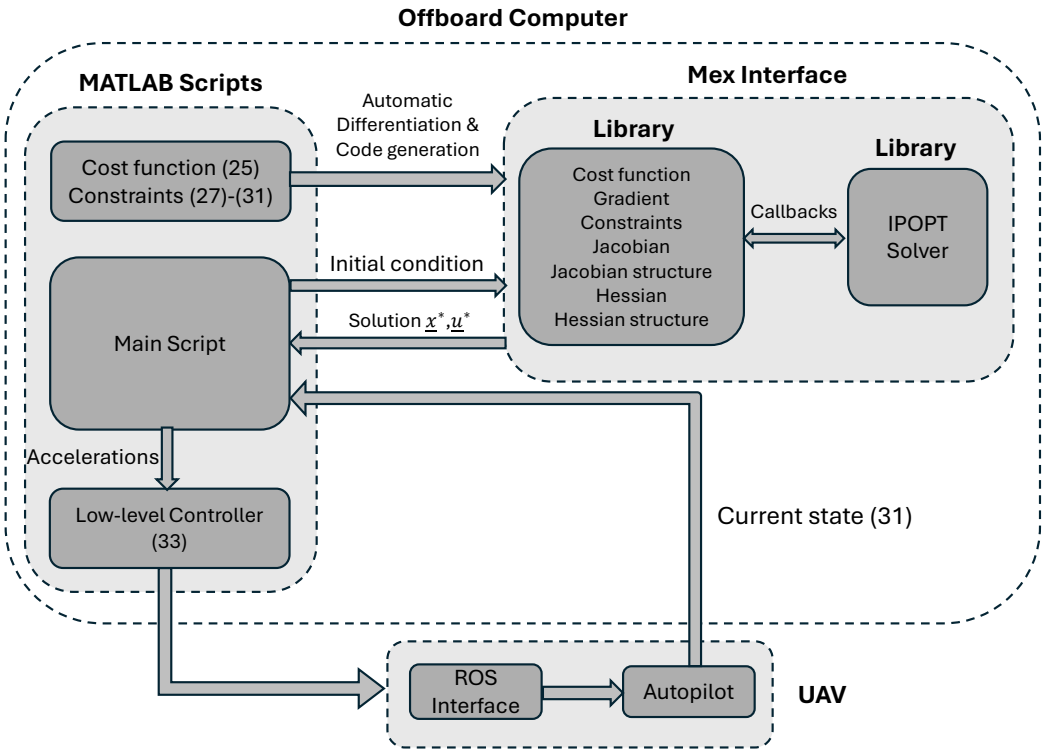

Fig. 4: Software architecture, generation, and implementation of the proposed MPC solver.

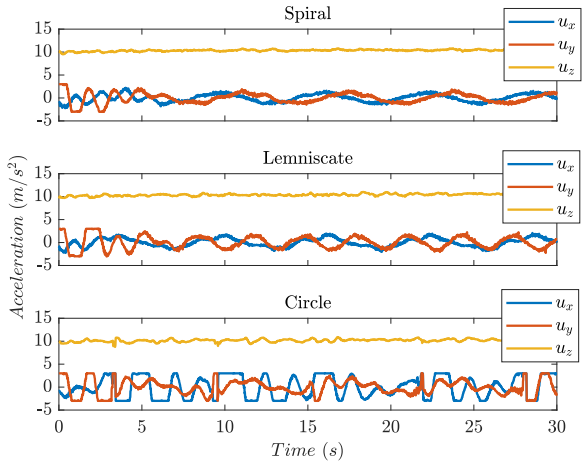

Fig. 5: Time histories of control inputs for Spiral, Lemniscate, and Circle maneuvers.

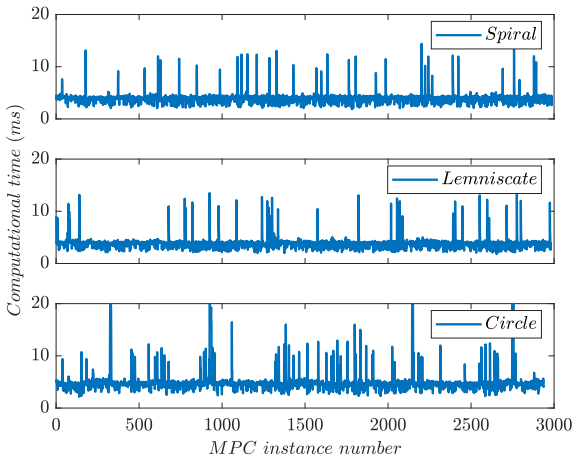

Fig. 6: MPC computational times.

took around 5 s to converge to the reference trajectory. After convergence, the average error remained the same around 0.1 m for the Spiral maneuver, and varied between 0.1 and 0.2 m for the Lemniscate maneuver. The steady-state errors are less in x and y directions compared to z direction primarily due to model uncertainties arising from non-consideration of aerodynamic lift forces in the dynamic model and delay by the attitude controller in orienting the UAV's thrust in the desired direction determined by MPC. The time history of the distance of the UAV to the obstacle is presented in Fig. 3 for the Circle maneuver. It can be seen that the UAV stayed away from the obstacle throughout the maneuver, except in a few instances. Even in these instances, the maximum constraint violation is 0.058 m, which is partially due to model uncertainty, but it can be addressed by adding a safety margin. It is an inherent problem of MPC to violate constraints partially in dynamic scenarios due to its short-term predictive nature. It is important to highlight the robustness of IPOPT solver to compute a valid solution even in the presence of marginally infeasible scenarios. With a sufficient amount of safety margin and the use of soft constraints,

the problem of collision avoidance and infeasible situations can be handled more elegantly. Control input histories for all three maneuvers are presented in Fig. 5. For Spiral and Lemniscate maneuvers, some of the acceleration inputs reached their bound limits during the convergence period but were smooth for the rest of the maneuvers. In the Circle maneuver, bound limits were often reached primarily due to large acceleration inputs computed by the solver for collision avoidance and quick convergence to the reference trajectory after circumventing the obstacle. The computational times for each instance of the MPC problem for each maneuver are presented in Fig. 6. It can be seen that, except in a few instances, the solver was able to compute the solutions in around 4 *ms* for the Spiral and Lemniscate maneuvers, and 5 *ms* for the Circle maneuver. This highlights the real-time capabilities of the proposed MPC.

Remark 4. *Even though the proposed MPC solver was implemented in an offboard computer as in Fig. 4, it can also be implemented on a smaller onboard computer with real-time capabilities.*

V. CONCLUSIONS

In this paper, we addressed the Unmanned Aerial Vehicle (UAV) trajectory tracking and obstacle avoidance problem, by proposing a Model Predictive Control formulation with a continuous-time integral form of the quadratic tracking error cost function. The problem was transcribed into a nonlinear programming problem using a Chebyshev pseudospectral method. Automatic differentiation was used to exactly determine the gradients, Jacobians, and Hessians, and code generation was used to generate a problem-specific MPC solver for real-time implementation. The Clenshaw-Curtis quadrature rule was used to compute the exact solution of the cost function. Therefore, the computationally expensive numerical integration was avoided. This also allows us for a longer prediction horizon with the same number of decision variables. Indoor UAV experiments were used to validate the proposed formulation. In future, we explore the robustness of the proposed MPC formulation under the model uncertainties, wind disturbances, sensor noise, etc., and evaluate its computational load on smaller onboard computers to demonstrate broader applicability, especially in outdoor environments.

REFERENCES

[1] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," *IEEE Trans. Robotics*, vol. 38, no. 6, pp. 3357–3373, 2022.

[2] T. Tomic *et al.*, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE Robot. Autom. Mag.*, vol. 19, no. 3, pp. 46–56, 2012.

[3] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, 2015.

[4] E. Restrepo, I. Sarras, A. Loria, and J. Marzat, "3d uav navigation with moving-obstacle avoidance using barrier lyapunov functions," *IFAC-PapersOnLine*, vol. 52, no. 12, pp. 49–54, 2019.

[5] B. Lindqvist, S. S. Mansouri, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Nonlinear mpc for collision avoidance and control of uavs with dynamic obstacles," *IEEE Robot. Autom. Lett*, vol. 5, no. 4, pp. 6001–6008, 2020.

[6] H. Habibi, A. Safaei, H. Voos, M. Darouach, and J. L. Sanchez-Lopez, "Safe navigation of a quadrotor uav with uncertain dynamics and guaranteed collision avoidance using barrier lyapunov function," *Aerosp. Sci. Technol.*, p. 108 064, 2023.

[7] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for mavs in dynamic environments," *IEEE Robot. Autom. Lett*, vol. 4, no. 2, pp. 776–783, 2019.

[8] D. V. Rao, H. Habibi, J. L. Sanchez-Lopez, and H. Voos, "An integrated real-time uav trajectory optimization and potential field approach for dynamic collision avoidance," in *2023 Int. Conf. Unmanned Aircraft Systems (ICUAS)*, IEEE, 2023, pp. 79–86.

[9] J. Hong, Y. Kim, and H. Bang, "Cooperative circular pattern target tracking using navigation function," *Aerosp. Sci. Technol.*, vol. 76, pp. 105–111, 2018.

[10] "Forces nlp: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, vol. 93, no. 1, pp. 13–29, 2020.

[11] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "Qpoases: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, pp. 327–363, 2014.

[12] G. Frison and M. Diehl, "Hpipm: A high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[13] J. L. Sanchez-Lopez, M. Castillo-Lopez, M. A. Olivares-Mendez, and H. Voos, "Trajectory tracking for aerial robots: An optimization-based planning and control approach," *J. Intell. Robot. Syst.*, vol. 100, no. 2, pp. 531–574, 2020.

[14] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time uav replanning using guided gradient-based optimization and topological paths," in *2020 IEEE Int. Conf. Robotics Automation (ICRA)*, IEEE, 2020, pp. 1208–1214.

[15] M. Castillo-Lopez, P. Ludivig, S. A. Sajadi-Alamdari, J. L. Sanchez-Lopez, M. A. Olivares-Mendez, and H. Voos, "A real-time approach for chance-constrained motion planning with dynamic obstacles," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3620–3625, 2020.

[16] Z. Xu, D. Deng, Y. Dong, and K. Shimada, "Dpmpc-planner: A real-time uav trajectory planning framework for complex static environments with dynamic obstacles," in *2022 Int. Conf. Robotics Automation (ICRA)*, IEEE, 2022, pp. 250–256.

[17] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *2018 Int. Conf. Intelligent Robots Systems (IROS)*, IEEE, 2018, pp. 1–8.

[18] Y.-W. Chen, M.-L. Chiang, G.-R. Kuo, C.-J. Chung, and L.-C. Fu, "Model predictive control with reference path planning for multi-uav formation control system," in *2024 American Control Conf. (ACC)*, 2024, pp. 279–284.

[19] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Trans. Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021.

[20] A. M. Ali, H. A. Hashim, and C. Shen, "Mpc based linear equivalence with control barrier functions for vtol-uavs," in *2024 American Control Conf. (ACC)*, 2024, pp. 1–6.

[21] D. V. Rao, H. Habibi, J. L. Sanchez-Lopez, P. P. Menon, C. Edwards, and H. Voos, "Adaptive super-twisting controller design for accurate trajectory tracking performance of unmanned aerial vehicles," *IEEE Trans. Control Syst. Tech.*, 2024.

[22] L. N. Trefethen, *Spectral methods in MATLAB*. SIAM, 2000.

[23] L. Besnard, Y. B. Shtessel, and B. Landrum, "Quadrotor vehicle control via sliding mode controller driven by sliding mode disturbance observer," *J. Franklin Inst.*, vol. 349, no. 2, pp. 658–684, 2012.

[24] D. Garg, M. Patterson, W. W. Hager, A. V. Rao, D. A. Benson, and G. T. Huntington, "A unified framework for the numerical solution of optimal control problems using pseudospectral methods," *Automatica*, vol. 46, no. 11, pp. 1843–1851, 2010.

[25] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, pp. 1–36, 2019.