On the implementation of an explicit MPC for a quadcopter via differential flatness

Huu-Thinh Do, Ionela Prodan

SAGIP
1er Congrès Annuel
Marseille 7-9 Juin 2023

On the implementation of an explicit MPC for a quadcopter via differential flatness

Huu-Thinh Do[1], Ionela Prodan[1]

Univ. Grenoble Alpes, Grenoble INP[†], LCIS, F-26000, Valence, France
[†]Institute of Engineering and Management Univ. Grenoble Alpes
{huu-thinh.do,ionela.prodan}@lcis.grenoble-inp.fr

KEYWORDS : *unmanned aerial vehicles, differential flatness, feedback linearization, quadcopter, explicit model predictive control.*

1 Introduction

With the popularity of Model Predictive Control (MPC) in both research and industrial applications, the explicit MPC [1] has promised to be a solution for surpassing the computational limitation of the implicit/on-line MPC (which has high requirement in computation time). However, real-time implementation of explicit MPC is rather challenging when it comes to handling the nonlinear nature of the model. In this paper, we show, as a first step, how to bypass such difficulty for a multicopter system via differential flatness properties. With the representation of all states and inputs in terms of a special output, called the *flat output*, and its derivatives, a coordinate change can be deduced to linearize the system's model via feedback [5, 3]. Furthermore, with a suitable inner-approximation of the feasible domain in the new convoluted coordinates [3], a standard explicit MPC setup is formulated, which, thus, promises a fast control calculation. Preliminary simulations, comparisons and discussions over applicability of the method are presented.

2 Explicit MPC setup and simulation for a quadcopter model

In this section, we first recall the coordinate change which linearizes the quadcopter model in closed-loop and the inner approximation of its corresponding new nonlinear input constraints. Then, with those ingredients, the explicit MPC setup is presented for later implementation.

2.1 System characterization with exact linearization deduced by flatness

Recall the discretized model of the translational dynamics of the quadcopter [5]:

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + Bh_\psi(\boldsymbol{u}_k) \tag{1}$$

where, at time step k, $\boldsymbol{x}_k = [x, y, z, \dot{x}_k, \dot{y}_k, \dot{z}_k]^\top \in \mathbb{R}^6$ gather the positions of the drone and their derivatives. $\boldsymbol{u}_k = [T_k, \phi_k, \theta_k]^\top \in \mathbb{R}^3$ denotes the three inputs including the normalized thrust, roll and pitch angles, while ψ refers to the yaw angle which is considered as a known parameter. Finally, g is the gravity acceleration, T_s is the sampling time and the remaining matrices are defined as:

$$A = \begin{bmatrix} \boldsymbol{I}_3 & T_s\boldsymbol{I}_3 \\ 0_{3\times 3} & \boldsymbol{I}_3 \end{bmatrix}, B = \begin{bmatrix} 0.5T_s^2\boldsymbol{I}_3 \\ T_s\boldsymbol{I}_3 \end{bmatrix}, h_\psi(\boldsymbol{u}_k) = \begin{bmatrix} T_k(\cos\phi_k \sin\theta_k \cos\psi + \sin\phi_k \sin\psi) \\ T_k(\cos\phi_k \sin\theta_k \sin\psi - \sin\phi_k \cos\psi) \\ -g + T_k \cos\phi_k \cos\theta_k \end{bmatrix}. \tag{2}$$

The quadcopter needs to respect the input constraints:

$$\boldsymbol{u}_k \in \mathcal{U} = \{0 \le T_k \le T_{\max}\ , |\phi_k| \le \epsilon_{max}, |\theta_k| \le \epsilon_{max}\} \tag{3}$$

where $T_{max} > 0, \epsilon_{max} \in (0; \pi/2)$ are constant bounds of the thrust and the angles, respectively.

By a change of control variables $\boldsymbol{u}_k = \varphi_\psi(\boldsymbol{v}_k)$ with:

$$\varphi_\psi(\boldsymbol{v}_k) \triangleq \begin{cases} T_k &= \sqrt{v_{1k}^2 + v_{2k}^2 + (v_{3k}+g)^2}, \phi_k = \arcsin\left((v_{1k}\sin\psi - v_{2k}\cos\psi)/T_k\right), \\ \theta_k &= \arctan((v_{1k}\cos\psi + v_{2k}\sin\psi)/(v_{3k}+g)), \end{cases} \tag{4}$$

the system (1) and the constraints as in (3) yield:

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{v}_k \tag{5a}$$ $$\boldsymbol{v}_k \in \mathcal{V} \triangleq \{\boldsymbol{v} : \varphi_\psi(\boldsymbol{v}) \in \mathcal{U}\} \tag{5b}$$

with $\boldsymbol{v}_k = [v_{1k}, v_{2k}, v_{3k}]^\top$ denotes the new input. Next, as presented in [3], we adopt the zonotope-based inner approximation procedure to construct a ψ-independent convex subset of $\mathcal{V}$ as in (5b), which is denoted as $\mathcal{S}_v$ (See FIG. 1). Note that, for simplicity, we choose to construct $\mathcal{S}_v$ with only 6 vertices. Regardless, the control problem now is reduced to the governing of the model (5a) while the constraint $\boldsymbol{v}_k \in \mathcal{S}_v$ is respected.

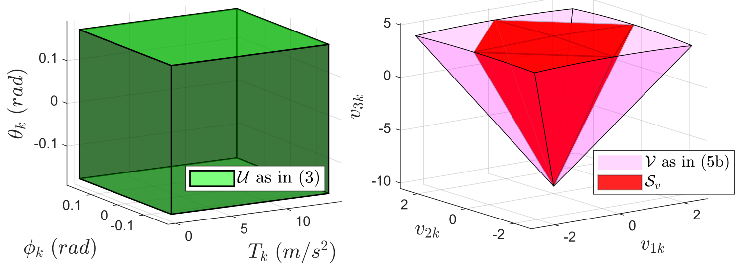

FIG. 1: The original input constraint set $\mathcal{U}$, new constraints $\mathcal{V}$ and its inner-approximation $\mathcal{S}_v$

2.2 Explicit MPC setup

With the above ingredients, we formulate the stabilizing MPC problem as follows:

$$J^*(\boldsymbol{x}_k) \triangleq \min \sum_{i=0}^{N_p-1} (\|\boldsymbol{x}_{i+k}\|_Q^2 + \|\boldsymbol{v}_{i+k}\|_R^2) \quad \text{s.t} : \begin{cases} \boldsymbol{x}_{i+k+1} = A\boldsymbol{x}_{i+k} + B\boldsymbol{v}_{i+k},\ \boldsymbol{v}_{i+k} \in \mathcal{S}_v, \\ \boldsymbol{x}_{i+k} \in \mathcal{X},\ i \in \{0, ..., N_p - 1\} \end{cases} \tag{6}$$

where $\mathcal{S}_v$ and $\mathcal{X}$ are the inner-approximation of $\mathcal{V}$ as in (5b) and the state constraints, respectively. N_p is the prediction horizon and Q, R are the positive definite weighting matrices.

Denote $\boldsymbol{v}^*(\cdot|\boldsymbol{x}_k)$ as the optimal sequence of $\boldsymbol{v}_{i+k}$ in the problem (6), then the solution of (6) can be represented as a piece-wise function of $\boldsymbol{x}_k$ as follows [1]:

$$\underset{j\in\{1,\dots,M\}}{\left(\boldsymbol{v}^*(\cdot|\boldsymbol{x}_k); J^*(\boldsymbol{x}_k)\right)} = \begin{cases} \left(F_1\boldsymbol{x}_k + \mu_1; \boldsymbol{x}_k^\top\gamma_1\boldsymbol{x}_k + \alpha_1\boldsymbol{x}_k + \beta_1\right) & , \boldsymbol{x}_k \in \mathcal{R}_1; \\ & \dots \\ \left(F_M\boldsymbol{x}_k + \mu_M; \boldsymbol{x}_k^\top\gamma_M\boldsymbol{x}_k + \alpha_M\boldsymbol{x}_k + \beta_M\right) & , \boldsymbol{x}_k \in \mathcal{R}_M. \end{cases} \tag{7}$$

where $\mathcal{R}_j = \{\boldsymbol{x} : H_j\boldsymbol{x} \le h_j\}$, the j-th polyhedral critical region, together with the parameters $F_j, \mu_j, \gamma_j, \alpha_j, \beta_j$ are numerically available within the parametric programming framework [2, 6].

2.3 Simulation study

In this paper, we employed the Multi-Parametric Toolbox for MATLAB [4, 6] to compute the piece-wise function as in (7). The system parameters are adopted from [3] as $T_s = 0.1$ (sec), $g = T_{max}/1.45 = 9.81 m/s^2, \epsilon_{max} = 0.1745$ rad while the controller (6) is set up with $Q = \text{diag}(50\boldsymbol{I}_3, 5\boldsymbol{I}_3), R = 5\text{diag}(\boldsymbol{I}_3)$, $\mathcal{X} = \{\boldsymbol{x} : |q| \le 1.5, |\dot{q}| \le -1, q \in \{x, y, z\}\}$ and $N_p = 3$. Numerical results can be found in TAB. 1 and FIG. 2 where the system's state is color-coded with the corresponding critical region.

Results	Number of regions (M)	Average computation time	Size of exported data
Value	19612	34.79 ms	45 MB

TAB. 1: Numerical specifications and results

As can be seen from FIG. 2, the computation time appears to be relatively small compared to other implementation of non-linear MPC in the literature [5] ($\ge$ 80 ms). However, when it comes to the conventional MPC where problem (6) is solved online, the constructed MPC setup show no improvement, computationally, compared to the similar settings in [3] (with ≤ 30 ms). This burden comes directly from the heavy linear search (i.e., a sequential search) for the critical region among an enormous number of sets ($M = 19612$) in the 6D space. Besides, as suggested in TAB. 1, the size of the data exported is relatively large, this may create certain hardware requirement on the vehicle's embedded computer in real implementation.

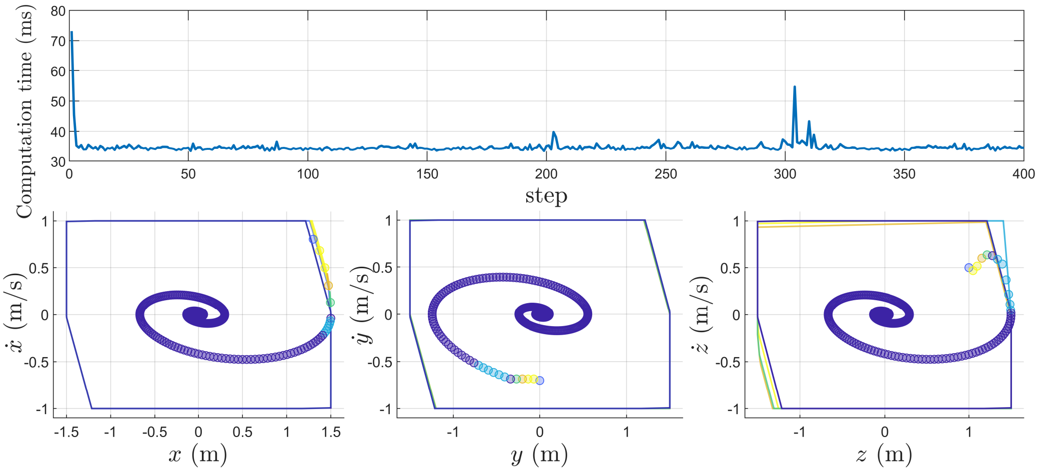

FIG. 2: Computation time (top) and the system's state color-coded by critical regions (bottom).

Theoretically, prior to this point, the difficulty of non-linearity was surpassed for the control of the multicopter dynamics (1), which hence, allows us to explicitly compute the solution for an MPC setup via a piece-wise function and to have a fast MPC implementation. However, computationally, there still exist several challenges that can possibly be improved with a better formulated problem as well as stronger computation power.

3 Conclusion

In this paper, we presented how to formulate the explicit MPC for a multicopter vehicle by exactly linearizing the system via its differential flatness properties and an inner approximation of the feasible domain. Simulations show the validation and applicability of the procedure, however, computational difficulty remains a hindrance for real-time implementation.

References

[1] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[2] Francesco Borrelli. *Constrained optimal control of linear and hybrid systems*, volume 290. Springer, 2003.

[3] Huu Thinh Do and Ionela Prodan. Indoor experimental validation of MPC-based trajectory tracking for a quadcopter via a flat mapping approach. Accepted to: the 2023 European Control Conference (ECC), Available: `https://youtu.be/1a1K6R6__3s`.

[4] Martin Herceg, Michal Kvasnica, Colin N Jones, and Manfred Morari. Multi-parametric toolbox 3.0. In *2013 European control conference (ECC)*, pages 502–510. IEEE, 2013.

[5] Ngoc Thinh Nguyen, Ionela Prodan, and Laurent Lefèvre. Stability guarantees for translational thrust-propelled vehicles dynamics through nmpc designs. *IEEE Transactions on Control Systems Technology*, 29(1):207–219, 2020.

[6] Bálint Takács, Juraj Števek, Richard Valo, and Michal Kvasnica. Python code generation for explicit MPC in MPT. In *2016 European Control Conference*. IEEE, 2016.