Application of an Approximate Model Predictive Control Scheme on an Unmanned Aerial Vehicle

Matthias Hofer, Michael Muehlebach, and Raffaello D'Andrea

***Abstract*— An approximate model predictive control approach is applied on an unmanned aerial vehicle with limited computational resources. A novel method using a continuous time parametrization of the state and input trajectory is used to derive a compact description of the optimal control problem. Different first order methods for the online optimization are discussed in terms of memory requirements and execution time. The generalized fast dual gradient method, as presented in [1], is implemented on the aerial vehicle. The approximate model predictive control algorithm runs on an embedded platform with a STM32 Cortex M4 processor. Simulation studies show that the model predictive controller outperforms a linear quadratic regulator in aggressive maneuvers. The model predictive control approach is evaluated in practice and shown to yield satisfactory flight behavior.**

I. Introduction

Model Predictive Control (MPC) provides the possibility to systematically incorporate input and state saturations imposed by the physical system. Penalizing state and input deviations from reference trajectories allows for an intuitive tuning of the control strategy. These benefits come with an increased computational demand compared to other control algorithms such as proportional-integral-derivative control. An extensive overview about MPC can be found for example in [2] and [3]. Recently, significant progress has been made in terms of real-time capability. However, for systems with fast dynamics, the required sampling rates are still challenging, particularly for resource-constrained embedded platforms.

A. Related Work

A method to avoid the online computational burden is to use explicit MPC, [4]. Multiparametric programming allows the offline precomputation of a lookup table from initial state to optimal control input, which has to be evaluated online. This approach is viable for small problem sizes (up to around 5 states), but becomes prohibitive for higher state and input spaces due to rapidly growing memory requirements.

When it comes to online MPC, second order methods incorporate the advantage of a reduced number of iterations through the usage of curvature information of the objective function. However, the computational effort per iteration increases, and upper bounds on the number of iterations needed, tend to be conservative.

The authors of [5] suggest an interior point method, which exploits the specific problem structure inherent to the MPC formulation. Software frameworks (CVXGEN, FORCES) are introduced in [6] and [7], which allow for the generation of real-time solvers for embedded hardware, tailored to a family of optimization problems. In [8], FORCES is applied to control autonomous racing cars of scale 1:43. The online solver is running at 50 Hz on an embedded platform using an ARM A9 chip (1.7 GHz).

An alternative to second order methods tailored to specific problem structures are first order methods. An efficient evaluation and straightforward implementation are beneficial, with the major drawback being large iteration numbers especially for ill-conditioned problems. In [9], the fast gradient method, [10], is applied to linear quadratic MPC problems. Lower iteration bounds are derived for getting a solution of predefined suboptimality. An alternative to the fast gradient method represents the generalized fast dual gradient method, which is introduced in [1].

MPC has been previously applied in the field of UAVs. In [11], MPC is used for state interception maneuvers of a quadrocopter. The MPC is running offboard, while a rate controller is employed onboard. A decoupled model of the dynamics is used to generate interception trajectories at 50 Hz. A learning-based MPC controller is proposed in [12]. The quadratic programming solver is running on an onboard computer including an Intel Atom processor. The authors of [13] apply robust MPC to UAVs using a multi-parametric approach. The control law is computed explicitly and evaluated onboard on a computer with an Intel Atom processor.

While the common approach is to discretize the dynamics and truncate the time horizon, [14] introduces a different paradigm. The derivative of the control input is described by polynomial basis functions over a finite prediction horizon.

Similarly, this paper proposes a continuous time approximation of the control and state trajectory, but over an infinite horizon. The quadratic programming solver routine is running purely onboard, on a STM32 Cortex M4 processor (168 MHz) with 265 KB RAM. The full model of the system including 12 states and 4 inputs is incorporated into the MPC design.

B. Outline

The continuous time approximate MPC approach is introduced in Section II. Section III discusses two first order methods used for the online optimization. In Section IV, the aerial vehicle used for experimental evaluation is described and results from simulation and practical experiments are presented in Section V. Concluding remarks are made in Section VI.

*This work was supported by ETH-Grant 0-20125-15.

The authors are with the Institute for Dynamic Systems and Control, ETH Zurich. Email correspondence to hofermat@ethz.ch.
{hofermat,michaemu,rdandrea}@ethz.ch.

For ease of notation, vectors are expressed as n-tuples, with stacking clear from the context. Moreover, the two norm is denoted by $\|\cdot\|_2$, and $I_m \in \mathbb{R}^{m\times m}$ denotes the identity matrix.

II. Continuous Time Model Predictive Control

In this section, a continuous-time approximation to the following infinite horizon optimal control problem is derived:

$$\begin{aligned} J_\infty := \min_{u(t)} & \int_0^\infty \frac{1}{2}x(t)^\mathsf{T} Q x(t) + \frac{1}{2}u(t)^\mathsf{T} R u(t)\,\mathrm{d}t \\ \text{s.t.} \quad & \dot{x}(t) = Ax(t) + Bu(t), \quad x(0) = x_0 \\ & u_{\min} \le u(t) \le u_{\max}, \quad \forall t \in [0,\infty) \\ & x \in L^2([0,\infty),\mathbb{R}^n),\ u \in L^2([0,\infty),\mathbb{R}^m), \end{aligned} \tag{1}$$

where $u_{\min} \in \mathbb{R}^m$ and $u_{\max} \in \mathbb{R}^m$ define lower and upper bound constraints on $u(t)$, Q is positive semidefinite ($Q \succeq 0$), and R is positive definite ($R \succ 0$). Note that we deliberately consider box-constraints only, since they lead to a significant simplification and speed-up of the optimization routines used in Sec. III.

The simplification of (1) consists of three steps: 1) representing input and state trajectories as linear combinations of basis functions, 2) simplifying the dynamics using a variational formulation (Galerkin method), 3) simplifying the constraints using constraint sampling. These three steps will be discussed in the remainder of this section.

A. Laguerre Functions

We represent state and input trajectories as linear combinations of Laguerre functions, see e.g. [14]. That is,

$$\tilde{x}(t) = (I_n \otimes \tau(t)^\mathsf{T})\eta_x, \quad \tilde{u}(t) = (I_m \otimes \tau(t)^\mathsf{T})\eta_u, \tag{2}$$

with $\tau(t) := (l_1(t), l_2(t), \ldots, l_s(t))^\mathsf{T}$, for all $t \in [0,\infty)$, and

$$l_i(t) := \sqrt{2\lambda}\exp(-\lambda t)\sum_{k=0}^{i-1}\binom{n}{k}\frac{(-1)^k}{k!}(2\lambda t)^k. \tag{3}$$

The exponential decay of the basis functions is described by the parameter $\lambda > 0$, which will be used as a tuning parameter in a later stage. The basis functions are square integrable and orthogonal, in the sense that $\int_0^\infty \tau(t)\tau(t)^\mathsf{T}\mathrm{d}t = I_s$. Moreover, they fulfill the first-order differential equation

$$\dot{\tau}(t) = M_\lambda \tau(t), \quad \forall t \in [0,\infty), \tag{4}$$

where $M_\lambda \in \mathbb{R}^{s\times s}$ (see e.g. [14]). These properties will be used in the following.

B. Approximation of the Dynamics

We use a variational formulation of the dynamics. Due to the fundamental lemma of the calculus of variations the dynamics $\dot{x}(t) = Ax(t) + Bu(t)$ for all $t \in [0,\infty)$ together with the initial condition $x(0) = x_0$, are equivalent (almost everywhere) to

$$\begin{aligned} \int_0^\infty & \delta p(t)^\mathsf{T}\left(Ax(t) + Bu(t) - \dot{x}(t)\right)\mathrm{d}t \\ & - \delta p(0)^\mathsf{T}(x(0) - x_0) = 0, \end{aligned} \tag{5}$$

$\forall \delta p \in L^2([0,\infty),\mathbb{R}^n)$, where $L^2([0,\infty),\mathbb{R}^n)$ denotes the space of square integrable functions, mapping from $[0,\infty)$ to $\mathbb{R}^n$. By restricting the variations δp to be linear combinations of basis functions, i.e. $\delta\tilde{p}(t) = (I_n \otimes \tau(t)^\mathsf{T})\eta_p$, with $\eta_p \in \mathbb{R}^{ns}$, and inserting the parametrized input and state trajectories, (5) is simplified to

$$\begin{aligned} \delta\eta_p^\mathsf{T}\Bigg[\int_0^\infty & (I_n \otimes \tau(t))\left(A\tilde{x}(t) + B\tilde{u}(t) - \dot{\tilde{x}}(t)\right)\mathrm{d}t \\ & -(I_n \otimes \tau(0))(\tilde{x}(0) - x_0)\Bigg] = 0, \quad \forall \delta\eta_p \in \mathbb{R}^{ns}. \end{aligned} \tag{6}$$

Combined with (2), this results in a linear relationship between the parameters η_u, η_x, and the initial condition x_0, i.e.

$$A_x\eta_x + B_u\eta_u = Fx_0, \tag{7}$$

where

$$\begin{aligned} A_x &:= A \otimes I_s - I_n \otimes M_\lambda, \\ B_u &:= I_{ns}, \quad F := I_n \otimes \tau(0). \end{aligned} \tag{8}$$

C. Constraint Sampling

The input constraints are relaxed from being enforced over the full interval $[0,\infty)$ to only certain time instances t_i, called the constraint sampling instances. Accordingly, the input constraints are reduced to

$$u_{\min} \le \left[\mathrm{I}_m \otimes \tau(t_i)^\mathsf{T}\right]\eta_u \le u_{\max}, \quad i = 1,2,\ldots,s. \tag{9}$$

To simplify notation, the s constraints are stacked together in a matrix, which leads to

$$T\eta_u \in \mathcal{U} := [u_{\min}, u_{\max}]^s, \tag{10}$$

with

$$T = \begin{bmatrix} \mathrm{I}_m \otimes \tau(t_1)^\mathsf{T} \\ \mathrm{I}_m \otimes \tau(t_2)^\mathsf{T} \\ \vdots \\ \mathrm{I}_m \otimes \tau(t_s)^\mathsf{T} \end{bmatrix} \in \mathbb{R}^{ms\times ms}. \tag{11}$$

To apply the fast gradient method and the generalized fast dual gradient method, see Sec. III, the projection onto the constraint set $\mathcal{U}$ should be particularly simple and computationally efficient. More precisely, the following projections need to be evaluated

$$\mathrm{prox}_{T^{-1}\mathcal{U}}^{X\otimes I_s}(\eta_u) := \underset{\eta_u^* \in T^{-1}\mathcal{U}}{\mathrm{argmin}}\ \frac{1}{2}(\eta_u^* - \eta_u)^\mathsf{T}(X \otimes I_s)(\eta_u^* - \eta_u), \tag{12}$$

where $X = R$ in the case of the generalized fast dual gradient method, and $X = I_m$ in the case of the fast gradient method. Using the change of variables $\hat{\eta}_u^* = T\eta_u^*$, the projection can be reformulated as

$$\mathrm{prox}_{T^{-1}\mathcal{U}}^{X\otimes I_s}(\eta_u) = T^{-1}\mathrm{prox}_{\mathcal{U}}^{T^{-\mathsf{T}}(X\otimes I_s)T^\mathsf{T}}(T\eta_u). \tag{13}$$

Note that the matrix $T^{-\mathsf{T}}(X \otimes I_s)T^\mathsf{T}$ can be simplified to $X \otimes \hat{T}$, where $\hat{T} \in \mathbb{R}^{s\times s}$ has the following components

$$\hat{T}_{ij} = \tau(t_i)^\mathsf{T}\tau(t_j). \tag{14}$$

This yields the equivalence

$$\mathrm{prox}_{T^{-1}\mathcal{U}}^{X\otimes I_s}(\eta_u) = T^{-1}\mathrm{prox}_{\mathcal{U}}^{X\otimes \hat{T}}(T\eta_u). \tag{15}$$

For an efficient evaluation of (15), $X \otimes \hat{T}$ must be diagonal, since in that case (15) simplifies to

$$T^{-1}\min(\max(T\eta_u, \mathbf{1}_s \otimes u_{\min}), \mathbf{1}_s \otimes u_{\max}), \tag{16}$$

where $\mathbf{1}_s = (1, 1, \ldots, 1) \in \mathbb{R}^s$. For the fast gradient method, this means that $\hat{T}$ must be diagonal. For the generalized fast dual gradient method this means that $\hat{T}$ and $X = R$ must be diagonal. Therefore, we require the constraint sampling instants to fulfill

$$\tau(t_i)^\mathsf{T}\tau(t_j) = 0, \quad \forall i \neq j. \tag{17}$$

It turns out, that for the basis functions given in (2), such time instants exist and are uniquely defined, once t_1 is set. In order to guarantee a feasible control input at $t = 0$, we choose $t_1 = 0$ and calculate the remaining constraint sampling instants in order to fulfill (17). For $s = 4$ and $\lambda = 5\,\mathrm{s}^{-1}$ this yields $t_1 = 0$, $t_2 = 0.094\,\mathrm{s}$, $t_3 = 0.331\,\mathrm{s}$, $t_4 = 0.776\,\mathrm{s}$.

D. Non-Condensed Formulation

Summarizing, (1) is therefore simplified to

$$\begin{aligned} \min_{\eta_x,\eta_u} \quad & \frac{1}{2}\begin{bmatrix}\eta_x^\mathsf{T} & \eta_u^\mathsf{T}\end{bmatrix} \underbrace{\begin{bmatrix} Q\otimes I_s & 0 \\ 0 & R\otimes I_s\end{bmatrix}}_{:=H_{\mathrm{ncf}}} \begin{bmatrix}\eta_x \\ \eta_u\end{bmatrix} \\ \text{s.t.} \quad & E\begin{bmatrix}\eta_x \\ \eta_u\end{bmatrix} = Fx_0 \quad T\eta_u \in \mathcal{U}, \end{aligned} \tag{18}$$

with $E := \begin{bmatrix} A_x & B_u \end{bmatrix}$. This problem formulation, where the parameters of the state trajectory and the parameters of the input trajectory are both kept as optimization variables, will be solved using the generalized fast dual gradient method, see Sec. III. To simplify the required projections, i.e. efficiently evaluate (15) using (16), we restrict the matrix R to be diagonal.

E. Condensed Formulation

In order to apply the fast gradient method, the equality constraints are eliminated using $\eta_x = A_x^{-1}(Fx_0 - B_u\eta_u)$. Rearranging the expressions results in a quadratic objective function in η_u, that is

$$\begin{aligned} \min_{\eta_u} \quad & \frac{1}{2}\eta_u^\mathsf{T} H_{\mathrm{cf}}\eta_u + x_0^\mathsf{T} G\eta_u \\ \text{s.t.} \quad & T\eta_u \in \mathcal{U}, \end{aligned} \tag{19}$$

with

$$\begin{aligned} H_{\mathrm{cf}} &:= B_u^\mathsf{T} A_x^{-\mathsf{T}}(Q\otimes I_s)A_x^{-1}B_u + R\otimes I_s, \\ G &:= -F^\mathsf{T} A_x^{-\mathsf{T}}(Q\otimes I_s)A_x^{-1}B_u. \end{aligned} \tag{20}$$

III. Quadratic Programming Solver Methods

The Parametric Model Predictive Control (PMPC) approach presented in Sec. II results either in a quadratic optimization problem with linear equality and box constraints in the non-condensed formulation or a quadratic optimization problem with box constraints only, in the condensed formulation. The execution time of the optimization routine is essential for ensuring real-time capability.

Due to the usage of an embedded platform with limited computational resources (see Section IV), second order methods tend to be less suitable because of the larger computational effort at each iteration. Therefore, the focus lies on two first order methods, which are discussed in the following.

A. Fast Gradient Method

A well-known first order method is the fast gradient method (FGM), [10]. The optimization problem is required to be in condensed form as defined in (19), meaning that the equality constraints are used to eliminate the optimization variables corresponding to the state trajectory. The benefit of a reduced problem size comes at the cost of an increased condition number of the Hessian. The major memory requirement is caused by the Hessian with dimensions ms times ms.

B. Generalized Fast Dual Gradient Method

A more elaborate and less known first order routine is the generalized fast dual gradient method (GFDGM), [1]. The starting point is the optimization problem in non-condensed form (18). Dual variables are introduced for imposing the equality constraints. The key property of the method is the computation of a tight upper bound on the negative dual objective function. This bound allows specific information about the curvature to be incorporated and not only a worst case upper bound across dimensions. As a result, the convergence speed in the dual variables is increased. A learning rate comparable to the one of the FGM is introduced for further speed-up. The resulting algorithm is shown by Alg. 1.

In accordance with [1, eq.(30)], step 5 of Algorithm 1 can be implemented efficiently as $w_i = \max\{\min\{u_i,\ u_{\max}\},\ u_{\min}\}$, if the primal Hessian is diagonal. The matrices $L_\lambda^{-1} = (EH^{-1}E^\mathsf{T})^{-1}$ and $H^{-1}E^\mathsf{T}$ are computed offline and stored as parameters. The largest

Algorithm 1 Generalized Fast Dual Gradient Method [1]

Initialize: $z_0 = \lambda_0 \in \mathbb{R}^{ns}$, $y_0 \in \mathbb{R}^{(n+m)s}$,
$L_\lambda^{-1} = (EH_{\mathrm{ncf}}^{-1}E^\mathsf{T})^{-1}$

1: **while** $||z_{i+1} - z_i||_2 > \mathrm{tol}$ **do**
2: $\quad p_i = -H_{\mathrm{ncf}}^{-1}E^\mathsf{T} z_i$
3: $\quad u_i = \begin{bmatrix}0 & T\end{bmatrix} p_i$
4: $\quad v_i = \begin{bmatrix}I_{ns} & 0\end{bmatrix} p_i$
5: $\quad w_i = T^{-1}\mathrm{prox}_{\mathcal{U}}^{T^{-\mathsf{T}}H_{\mathrm{ncf}}T^{-1}}(u_i)$
6: $\quad y_i = \begin{bmatrix}v_i \\ w_i\end{bmatrix}$
7: $\quad \lambda_{i+1} = z_i + L_\lambda^{-1}(Ey_i - Fx_0)$
8: $\quad t_{i+1} = \frac{1+\sqrt{1+4(t_i)^2}}{2}$
9: $\quad z_{i+1} = \lambda_{i+1} + \left(\frac{t_i - 1}{t_{i+1}}\right)(\lambda_{i+1} - \lambda_i)$
10: **end while**

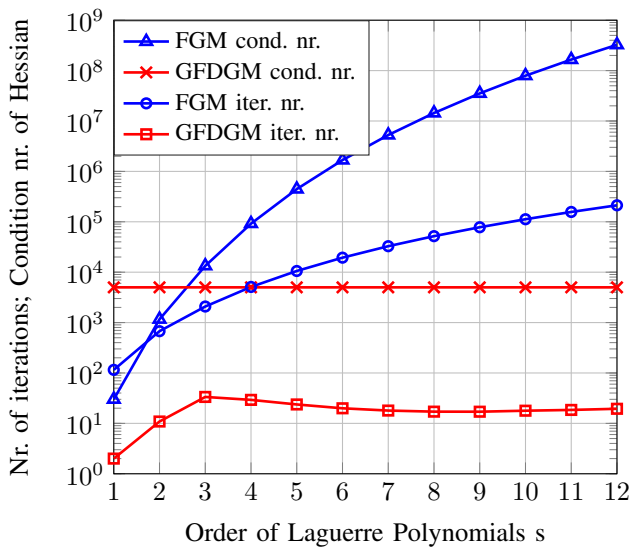

Fig. 1: The blue line indicated by (△) shows the condition number of the optimization problem in condensed formulation as required for the FGM for increasing polynomial orders s. The condition number resulting from the non-condensed formulation required by the GFDGM is depicted by the red line (×). The resulting iteration numbers for the FGM and GFDGM are shown by the blue line (○), respectively the red line (□).

matrices have dimensions $(n+m)s$ times ns, namely E^{T} and $H^{-1}E^{\mathsf{T}}$.

C. Comparison of the two Methods

The performance of the two first order methods are assessed using MATLAB for the optimization problem given by (18), resp. (19). The condition number of the Hessian and the number of iterations required by each method are computed for different polynomial orders s. The dependency on the initial state is addressed by averaging the results for each order over 1000 randomly chosen initial states, x_0. The relative tolerance of the method is set to 10^{-8}.

As can be seen in Fig. 1, the condition number of the FGM grows rapidly for increasing polynomial orders (scales with the 5th order). The GFDGM shows equal condition number for all polynomial orders, since the condition number for the problem in non-condensed form is purely determined by the ratio of the largest over the smallest entry of the tuning matrices Q and R.

The FGM also shows a strong dependency on the polynomial order in terms of the number of iterations (scales cubically), as shown in Fig. 1. The GFDGM, on the other hand, exhibits a peak at low orders, but converges to a certain value for increasing orders. Overall the GFDGM requires significantly less iterations compared to the FGM.

The computational effort for both methods is not only determined by the number of iterations, but also through the complexity of a single iteration. The cost of one iteration is incorporated by computing the number of floating point operations (FLOPs) for both methods required in the hardware implementation. The analysis is carried out for a polynomial order of 4, as used in Section V.

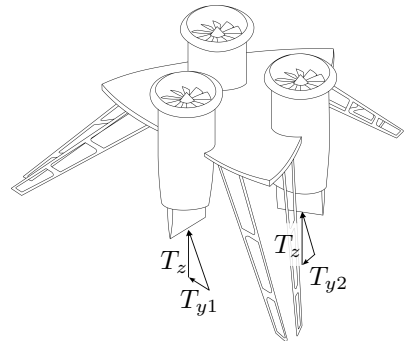

Fig. 2: Simplified drawing of the vehicle. Three electric ducted fans are attached to a connecting lightweight platform. A support frame allows for takeoff and landing from the ground. The electric ducted fans are numbered counter-clockwise from 1 to 3. The control inputs are shown for fan 1 and 2. The same convention is used for the third fan.

Table I summarizes the number of iterations, the number of FLOPs for one iteration and the total number of FLOPs for both methods. The more sophisticated GFDGM requires approximately 175 times less iterations, but 10 times more FLOPs for a single iteration compared to the FGM. In total, this leads to around 17 times less FLOPs for the GFDGM. For smaller polynomial orders ($s \in \{1,2,3\}$), the GFDGM becomes even more efficient in terms of FLOPs per iteration compared to the FGM.

	Iter.	FLOPs single iter.	Total FLOPs
FGM	5'096	2'164	$11.03 \cdot 10^6$
GFDGM	29	19'764	$0.57 \cdot 10^6$

TABLE I: Number of iterations and number of FLOPs for the FGM and the GFDGM for a polynomial order of 4.

The analysis carried out in this section shows the clear advantages of the GFDGM. Therefore, the GFDGM is used as online solver routine.

IV. The Flying Platform

This section briefly introduces the Flying Platform, an unmanned aerial vehicle used for the experimental testing of the previously described control algorithm.

A. Hardware

The Flying Platform consists of three electric ducted fans which are attached to a light-weight frame. A flap is attached below the exit nozzle of each fan, which allows for thrust vectoring in a single direction. A PX4FMU[1] is used as flight controller. It includes a 32 bit STM32F405 Cortex M4 processor, with a 168 MHz clock rate, 256 KB RAM, 2 MB flash memory, and a floating point unit. The sampling time of the controller is set to 50 Hz.

[1] www.pixhawk.org

Position and attitude information is provided by a Vicon motion capture system[2]. Linear and angular velocities are obtained by offboard state estimation techniques and the onboard measurements from the inertial measurement unit. Offboard state information can be sent to the vehicle via wireless communication.

B. Software

All matrices defining the optimization problem described in Section II are computed offboard. The matrices are loaded as parameters for the online solver routine, which is implemented using the arm matrix class. This class provides basic functionality for the required matrix algebra. The onboard code is compiled with the GNU GCC ARM Embedded compiler using the optimization option `-O3`.

C. Linear Time Invariant Model

The nonlinear model is derived from first principles and linearized around hover. The linearization is carried out in a yaw fixed coordinate system leading to a model, which is valid for all yaw orientations. The states are position, attitude, linear and angular velocity resulting in a total number of 12 states. The four thrust forces $(T_{y1}, T_{y2}, T_{y3}, T_z)$ depicted in Fig. 2 are chosen as control inputs. The horizontal thrust component of each fan is denoted by T_{yi}, the vertical component, T_z, is equal for all fans. The full mechanical model is incorporated into the MPC scheme. The numerical values of the continuous-time state space representation of the linearized dynamics are given by

$$A = \begin{bmatrix} 0 & \mathrm{I}_3 & 0 & 0 \\ 0 & 0 & A_{23} & 0 \\ 0 & 0 & 0 & \mathrm{I}_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} & & 0 & \\ -0.041 & -0.145 & 0.160 & 0 \\ 0.195 & -0.078 & -0.050 & 0 \\ 0 & 0 & 0 & 0.396 \\ & & 0 & \\ 1.640 & -0.691 & -0.655 & 0 \\ 0.161 & 1.322 & -1.540 & 0 \\ -0.984 & -0.883 & -0.950 & 0 \end{bmatrix}.$$

$$A_{23} = \begin{bmatrix} 0 & 9.81 & 0 \\ -9.81 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The force inputs are then translated into the corresponding servo commands (actuating the flaps) and turbine duty cycles. The maximum thrust $T_{\max}$ produced by each fan is limited, as well as the maximum pivoting angle of the flaps, $\psi_{\max}$. Therefore, the force inputs are subject to saturation limits, which can be expressed as

$$\|T_{yi} + T_z\|_2 \in [0, T_{\max}], \tag{21}$$

$$c \cdot \arctan\left(\frac{T_{yi}}{T_z}\right) \in [-\psi_{\max}, \psi_{\max}], \tag{22}$$

where c denotes a proportional constant between thrust angle and servo angle and is identified from measurements. Both constraints depend on T_{yi} as well as T_z. However, by assuming $T_{yi} \ll T_z$ and replacing T_z with the hover thrust $T_{z,0}$, (21) and (22) can be approximated as

$$T_z \in [0,\ T_{\max}], \tag{23}$$

$$T_{yi} \in \left[-T_{z,0} \tan\left(\frac{\psi_{\max}}{c}\right),\ T_{z,0} \tan\left(\frac{\psi_{\max}}{c}\right)\right]. \tag{24}$$

The input constraints (23) and (24) are box constraints matching the structure of the optimization problem (1) as introduced in Section II.

V. Results

The same settings (s, λ, Q, R) are used to evaluate the controller in simulation and experiments. The order of the Laguerre polynomials s is set to 4, resulting in the largest problem formulation without exceeding the RAM capacity of the embedded platform.[3] The decay rate λ is chosen as $5\mathrm{s}^{-1}$ which corresponds to the averaged real-parts of the closed-loop poles resulting from an LQR design. The following tuning matrices showed good performance in simulation and experimental tests:

$$Q = \mathrm{diag}\big(\overbrace{50,\ 50,\ 10,}^{\text{position}}\ \overbrace{10,\ 10,\ 10,}^{\text{lin. velocity}}\ \overbrace{40,\ 40,\ 10,}^{\text{attitude}}\ \overbrace{10,\ 10,\ 5}^{\text{ang. velocity}}\big)$$

$$R = \mathrm{diag}\big(\underbrace{0.085,\ 0.085,\ 0.085,}_{\text{horizontal thrust}}\ \underbrace{0.01}_{\text{vertical thrust}}\big).$$

The emphasis for the state variables is put on accurate position tracking with stabilizing weights on the pitch and roll angle. The weights on the first three control inputs are chosen such that no saturation occurs when hovering. The last weight, corresponding to the thrust in vertical direction, is tuned for fast tracking of vertical set-point changes, while avoiding oscillations due to the neglected turbine dynamics.

A. Simulation Results

A set-point shift of 0.65 m in positive x-direction is used to compare the MPC and LQR controller. Figure 3 illustrates the different behavior of the two controllers. The initial response of the two controllers is similar. Both controllers react with maximum inputs for T_{y2} and T_{y3} and show a similar braking maneuver. However, after a first overshoot in the x coordinate, the MPC acts earlier than the LQR controller. Due to the lack of information about the input saturations, the LQR is overestimating the available control input and hence, reacting later. This results in additional oscillations leading to instability.

B. Experimental Results

The reliable performance of the MPC controller can be shown in practice, where it is tested for disturbance rejection (see Fig. 4).[4] The experiments were conducted in the Flying Machine Arena[5].

[2] `http://www.vicon.com/`

[3] We did not account for sparsity.

[4] The interested reader is referred to the video attachment to get an impression of the conducted experiment and a detailed illustration of the experimental setup.

[5] `http://flyingmachinearena.org/`

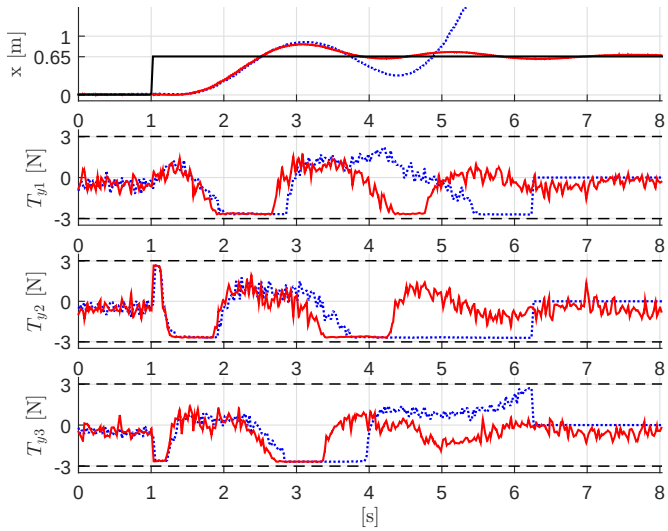

Fig. 3: Simulation results for a set-point shift of 0.65 m in positive x direction. The x coordinate and control inputs T_{y1}, T_{y2}, T_{y3} of the MPC are depicted in red. The corresponding trajectories of the LQR controller are shown in blue (dotted line).

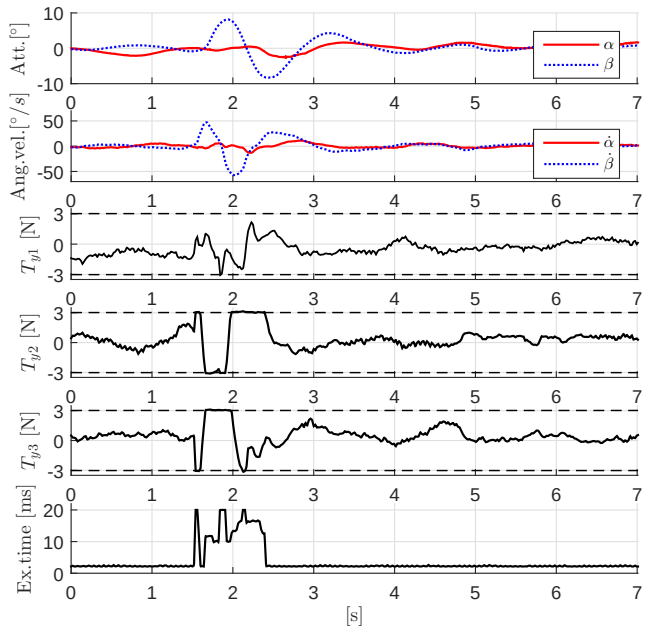

Fig. 4: Experimental evaluation of the MPC controller for disturbance rejection. The setpoint is temporarily shifted in positive y direction (0.5 m for 0.3 s) to excite the vehicle. The Euler angles α (roll, red line) and β (pitch, dotted blue line) are shown in the first plot, the angular velocities $\dot{\alpha}$ (red line) and $\dot{\beta}$ (dotted blue line) are shown in the second plot. The control inputs T_{y1}, T_{y2}, T_{y3} are depicted in the third, fourth and fifth plot and the execution time is shown in the last plot.

The vehicle tilts approximately 8 ° in the positive β direction, but can be stabilized by the MPC controller. The control inputs T_{y2} and T_{y3}, which are involved in this maneuver, reach their saturation limits. The execution time of the optimization routine is approximately 2 ms when hovering, but increases up to 20 ms when input constraints are hit. In rare cases, the solution routine is stopped prematurely.

VI. Conclusion

A parametric MPC approach has been successfully implemented on a resource constrained embedded platform. The MPC controller was evaluated in practice and yielded satisfactory flight behavior.

Acknowledgment

The authors would like to thank Marc-Andrè Corzillius, Michael Egli and Tobias Meier for their contribution to the development of the Flying Platform.

The experiments of this research were carried out in the Flying Machine Arena. The Flying Machine Arena builds upon prior contributions by numerous collaborators. A list of past and present participants is available at `http://flyingmachinearena.org/people/`.

References

[1] P. Giselsson, "Improved Fast Dual Gradient Methods for Embedded Model Predictive Control," *Proceedings of 2014 IFAC World Congress*, 2014.

[2] M. Morari and J. H. Lee, "Model predictive control: Past, present and future," *Computers & Chemical Engineering*, vol. 23, no. 4, pp. 667–682, 1999.

[3] E. F. Camacho and C. Bordons, *Model predictive control*. Springer Science & Business Media, 2013.

[4] A. Bemporad and M. Morari, "The Explicit Linear Quadratic Regulator for Constrained Systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[5] Y. Wang and S. Boyd, "Fast Model Predictive Control Using Online Optimization," *IEEE Transaction on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[6] J. Mattingley and S. Boyd, "CVXGEN: a code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[7] A. Domahidi, A. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control," in *IEEE Conference on Decision and Control*, Maui, HI, USA, Dec. 2012, pp. 668 – 674.

[8] A. Liniger, A. Domahidi, and M. Morari, "Optimization-Based Autonomous Racing of 1:43 Scale RC Cars," *Optimal Control Applications and Methods*, Oct. 2013.

[9] S. Richter, C. D. Jones, and M. Morari, "Computational Complexity Certification for Real-Time MPC With Input Constraints Based on the Fast Gradient Method," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1391–1403, 2012.

[10] Y. Nesterov, *Introductory Lectures on Convex Optimization. A Basic Course*. New York, USA: Springer, 2004.

[11] M. W. Mueller and R. D'Andrea, "A model predictive controller for quadrocopter state interception," in *European Control Conference (ECC)*, Zurich, Switzerland, Jul. 2013, pp. 1383 – 1389.

[12] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-Based Model Predictive Control on a Quadrocopter: Onboard Implementation and Experimental Results," in *IEEE International Conference on Robotics and Automation*, Minnesota, USA, May 2012, pp. 279 – 284.

[13] C. Papachristos, K. Alexis, and A. Tzes, "Dualauthority thrustvectoring of a tritiltrotor employing model predictive control," *Journal of Intelligent and Robotic Systems*, pp. 1–34, 2015.

[14] L. Wang, *Model Predictive Control System Design and Implementation Using MATLAB*, 1st ed. Springer, 2009.