2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

Implementation of a Parametrized Infinite-Horizon Model Predictive Control Scheme with Stability Guarantees*

Michael Muehlebach, Carmelo Sferrazza, and Raffaello D'Andrea[1]

Abstract— **This article discusses the implementation of an infinite-horizon model predictive control approach that is based on representing input and state trajectories by a linear combination of basis functions. An iterative constraint sampling strategy is presented for guaranteeing constraint satisfaction over all times. It will be shown that the proposed method converges. In addition, we will discuss the implementation of the resulting (online) model predictive control algorithm on an unmanned aerial vehicle and provide experimental results. The computational efficiency of the algorithm is highlighted by the fact that a sampling rate of 100 Hz was achieved on an embedded platform.**

I. Introduction

Model predictive control (MPC) is one of few control strategies that take a system's model including input and state constraints explicitly into account. At each time step, an optimal control problem is solved, and the first part of the resulting input is applied to the system. By doing so, an implicit feedback law is obtained that yields robustness against modeling uncertainties and disturbances. Consequently, MPC is a well-established control strategy and has been successfully applied in industry, see for example [1], [2], [3], and [4].

Although the computational resources available in embedded systems has increased in the past years, computation still represents a bottleneck when applying online MPC, where the aforementioned optimal control problem is solved online, for controlling systems with relatively fast dynamics and a large number of states and inputs. A widely used approach to render MPC computationally tractable is to discretize the dynamics and truncate the time horizon, resulting in a trade-off between computational complexity and prediction horizon. However, truncating the prediction horizon leads necessarily to issues with closed-loop stability and recursive feasibility, [5]. We therefore propose an alternative approach. As suggested in [6], we represent input and state trajectories as a linear combination of basis functions, thereby avoiding the truncation of the time horizon. As a result, we obtain straightforward stability and recursive feasibility guarantees that are inherent to the predictive control formulation. The trade-off is now shifted from computational complexity versus prediction horizon to computational complexity versus the number of basis functions used to describe input and state trajectories. It is conjectured that already few basis functions provide a relatively good approximation to the underlying optimal control problem, leading to a compact problem formulation (with few variables) that can be solved efficiently. The conjecture is supported by the results from [7], where a variant of the proposed MPC approach was successfully implemented on resource-constrained hardware (sampling time: 50 Hz, processor: STM32F4 with 168 MHz).

A major difficulty of the proposed approach is to enforce input and state constraints over an infinite time horizon. In previous work, [7], constraint sampling, that is, enforcing input and state constraints only at certain time instants, was shown to work reliably in practice, although the recursive feasibility and closed-loop stability guarantees may be lost. In the following, we will tackle this problem, and present an iterative strategy guaranteeing constraint satisfaction over an infinite time horizon for a certain class of basis functions. We will show convergence of the proposed strategy. In addition, we will discuss a method for efficiently checking constraint satisfaction by exploiting the convex-hull property of Bézier curves (to be made precise in the following). The proposed MPC algorithm is applied for controlling an unmanned aerial vehicle (UAV) that is modeled using 12 states and 9 inputs. The computational efficiency of the approach is highlighted by the fact that the control algorithm runs at 100 Hz.

Related work: An overview about the discrete-time finite-horizon MPC approach can be found in [8]. In [9], a continuous-time MPC formulation is presented that relies on a parametrization of input trajectories using Laguerre functions. Unlike the approach presented in [6], the finite prediction horizon is retained, and consequently, closed-loop stability and recursive feasibility need to be imposed using a combination of terminal state constraints and terminal cost (similar to the discrete-time approach). Input and state constraints are only enforced at certain time instances and as a result, constraint satisfaction is not guaranteed for all times.

MPC has been successfully applied to various types of UAVs. For example, in [10] and [11], a nonlinear receding horizon control approach is presented, and used to stabilize a vehicle powered by a ducted fan. The authors of [12] propose to parametrize input trajectories using polynomials in order to solve a nonlinear optimal control problem. The resulting predictive control algorithm is shown to work reliably by presenting simulations of a parafoil and a glider. Other applications include [13], where a nonlinear MPC scheme is applied to the control of autonomous helicopters, and [14], where a nonlinear MPC approach is used to stabilize a ducted-fan UAV.

The problem of imposing semi-infinite constraints (in our

*This work was supported by the ETH-Grant ETH-48 15-1.

[1]The authors are with the Institute for Dynamic Systems and Control, ETH Zurich, Switzerland. The contact author is Michael Muehlebach michaemu@ethz.ch.

case due to the fact that we require input and state constraints to be fulfilled for all times) has been extensively studied in the literature. For instance in the context of robust optimization, the authors of [15] and [16] propose to use a stochastic constraint sampling approach. They provide bounds on the probability that constraint violations occur, when solving the problem with sampled constraints. Alternative approaches include relaxation techniques, [17], that might even be exact, see for instance also [18]. Similar relaxation techniques are applied in [19] to tackle continuous linear programs. Practical applications include [20], where sums-of-squares programming was applied for planning collision-free UAV trajectories, and [21], where semi-infinite constraints arising in motion planning for humanoid robots were approximated in terms of B-spline control points. Moreover, in [22], a recursive strategy based on cubic Hermite splines is proposed for generating trajectories on manifolds (leading to a semi-infinite equality constraint), as encountered in robot locomotion and object manipulation.

In contrast to earlier work, we retain the infinite prediction horizon by parametrizing input and state trajectories with basis functions. In order to ensure constraint satisfaction over all times, we propose an iterative approach that is shown to converge. As a consequence, closed-loop stability and recursive feasibility are guaranteed, see [6]. Unlike stochastic constraint sampling strategies, our approach is deterministic, and leads to a sequence of quadratic programs that are to be solved. As such, it does not require the solution of semidefinite programs that typically arise with relaxation techniques, and that are often harder to solve.

The proposed model predictive control approach is applied to the control of an UAV[1], where good disturbance-rejection properties can be observed in practice.

Outline: The parametrized MPC formulation is introduced in Sec. II. Sec. III presents an iterative constraint sampling procedure ensuring constraint satisfaction over all times. It is shown that the algorithm converges. Sec. IV illustrates an efficient algorithm for checking constraint satisfaction. Experimental results are presented in Sec. V and the article concludes with remarks in Sec. VI.

II. Problem Formulation

We seek to approximate the following constrained infinite-horizon optimal control problem

$$\begin{aligned}\inf\ \frac{1}{2}\int_0^\infty x(t)^\mathsf{T} Qx(t) + u(t)^\mathsf{T} Ru(t)\ \mathrm{d}t \quad \text{s.t.} \qquad (1)\\ \dot{x}(t) = Ax(t) + Bu(t),\\ l_\mathrm{l} \leq C(x(t), u(t)) \leq l_\mathrm{u}, \quad \forall t \in [0,\infty),\\ x(0) = x_0, x \in L_n^2, u \in L_m^2,\end{aligned}$$

where x denotes the state trajectory, u the input trajectory, L_n^2 and L_m^2 refer to the set of square integrable functions mapping from $[0,\infty)$ to $\mathbb{R}^n$, respectively $\mathbb{R}^m$, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ describe the system dynamics, x_0 refers to the initial condition, and $C \in \mathbb{R}^{n_c\times(n+m)}$, $l_\mathrm{l} \in \mathbb{R}^{n_c}$, and $l_\mathrm{u} \in \mathbb{R}^{n_c}$ describe the linear constraints. The above inequalities have to be understood component wise, and it is assumed that $l_\mathrm{l} \leq 0$ and $l_\mathrm{u} \geq 0$. The integers n, m, and n_c refer to the number of states, inputs, and constraints, whereas the positive definite matrices $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$ describe the running cost. Throughout the article we denote vectors as n-tuples with dimension and stacking clear from the context, for example $(x(t), u(t)) \in \mathbb{R}^{n+m}$.

Unlike the "standard" MPC approach, which is based on discretizing the dynamics and truncating the time horizon, input and state trajectories are represented as a linear combination of basis functions. In the following, we choose the basis functions to be spanned by exponentially decaying polynomials. By means of the Gram-Schmidt process we choose the basis functions to be orthonormal with respect to the L^2-scalar product over the interval $[0,\infty)$, which leads to

$$\tau_i(t) := \sqrt{2\lambda}\mathcal{L}_{i-1}(2\lambda t)e^{-\lambda t}, \qquad (2)$$

$i = 1, 2, \ldots, s$, where the constant $\lambda > 0$ refers to the exponential decay and $\mathcal{L}_i$ denotes the ith Laguerre polynomial, [23, p. 775]. It can be shown that the basis functions fulfill the following identity

$$\dot{\tau}(t) = M_\lambda \tau(t), \quad \forall t \in [0,\infty), \qquad (3)$$

which will be used in the following. The vector τ contains the elements τ_i, $i = 1, 2, \ldots, s$, and the matrix $M_\lambda \in \mathbb{R}^{s\times s}$ is defined as

$$M_\lambda := \begin{pmatrix} -\lambda & 0 & 0 & \ldots \\ -2\lambda & -\lambda & 0 & \ldots \\ -2\lambda & -2\lambda & -\lambda & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \qquad (4)$$

Thus, we represent input and state trajectories as

$$\tilde{x}(t) := (I_n \otimes \tau(t))^\mathsf{T}\eta_x, \quad \tilde{u}(t) := (I_m \otimes \tau(t))^\mathsf{T}\eta_u, \quad (5)$$

where $\otimes$ denotes the Kronecker product and $I_q \in \mathbb{R}^{q\times q}$ the identity matrix (for any integer $q > 0$). The vectors $\eta_x \in \mathbb{R}^{ns}$ and $\eta_u \in \mathbb{R}^{ms}$ are the parameter vectors that are to be determined by the optimization.

According to [6], the optimal control problem (1) can be approximated by

$$J := \inf\ \frac{1}{2}\eta_z^\mathsf{T} H\eta_z \quad \text{s.t} \qquad (6a)$$

$$\mathcal{A}\eta_z = Dx_0, \qquad (6b)$$

$$l_\mathrm{l} \leq (C \otimes \tau(t))^\mathsf{T}\eta_z \leq l_\mathrm{u}, \quad \forall t \in [0,\infty), \qquad (6c)$$

where the vector $\eta_z \in \mathbb{R}^{s(n+m)}$ contains the parameter vectors η_x and η_u, $\eta_z := (\eta_x, \eta_u)$, and the matrices $\mathcal{A}$, H, and D are given by

$$H := \mathrm{diag}(Q \otimes I_n, R \otimes I_m),\ D := \begin{pmatrix} 0_{ns\times n} \\ I_n \end{pmatrix}, \qquad (7)$$

$$\mathcal{A} := \begin{pmatrix} I_n \otimes M_\lambda^\mathsf{T} - A \otimes I_s & -B \otimes I_s \\ I_n \otimes \tau(0)^\mathsf{T} & 0 \end{pmatrix}. \qquad (8)$$

As pointed out in [6], the constraint (6b) enforces the dynamics exactly, that is, given parameter vectors η_x and

[1]See `https://youtu.be/NYY9q-vs4Nw`.

η_u fulfilling (6b), the corresponding trajectories $\tilde{x}$ and $\tilde{u}$ fulfill the equations of motion exactly. Likewise, (6c) is equivalent to an exact fulfillment of the constraints, i.e. $l_\text{l} \leq C(\tilde{x}(t), \tilde{u}(t)) \leq l_\text{u}$ for all times $t \in [0, \infty)$, and therefore, trajectories satisfying (6b) and (6c) are feasible candidates for (1). As a result, the optimizer of (6) (if it exists) achieves the cost J on the real system, which is an upper bound on the optimal cost in (1). The constraints (6b) and (6c) describe a closed convex set in the parameter space η_z, and consequently, the optimization (6) is convex. Provided that a feasible η_z exists, the infimum is attained and the corresponding optimizer is unique due to the strong convexity of the Hessian H.

However, the constraint (6c) is not polyhedral, and therefore, (6) cannot be solved by standard optimization routines (like quadratic programming solvers). We propose to solve the optimization (up to a desired tolerance) by sampling the constraint (6c) in an iterative manner. At each iteration, we will check whether the current optimizer fulfills (6c) and refine the constraint sampling points if needed. We will show in the remainder that the satisfaction of (6c) can be checked efficiently, which, when combined with warm starts, leads to an efficient procedure for solving (6).

III. An iterative constraint sampling procedure

When imposing the inequality constraints (6c) only at a fixed and finite number of time instances t_i, $i = 1, 2, \ldots, N$, the optimization (6) reduces to the quadratic program

$$J(I) := \inf \ \frac{1}{2}\eta_z^\mathsf{T} H \eta_z \quad \text{s.t.} \tag{9a}$$

$$\mathcal{A}\eta_z = Dx_0 \tag{9b}$$

$$l_\text{l} \leq (C \otimes \tau(t))^\mathsf{T} \eta_z \leq l_\text{u}, \quad \forall t \in I, \tag{9c}$$

where $I := \{t_1, t_2, \ldots, t_N\}$. We propose to solve (6) in an iterative manner as summarized in Alg. 1. The algorithm solves (9) for a given I; if the solution η_z satisfies (6c)[2] then η_z is likewise the optimizer of (6) and the algorithm terminates. If not, at least one time instant at which the constraint is violated, is added to the set I. The time instants at which the constraint (6c) was not active, that is, their corresponding Lagrange multipliers were 0, are removed from the set I, and (9) is solved again. Removing these inactive time instants ensures that a finite number of constraint sampling points is used.

The following observation can be made: A function $[0, \infty) \to \mathbb{R}$ of the type $\tau(t)^\mathsf{T}\eta$, where η are constant coefficients, can have at most $s-1$ stationary points, since the basis functions consist of exponentially decaying polynomials of order of $s-1$ or smaller. As a result, the optimization (6) will have at most $(s-1)n_c$ active inequality constraints. If we knew the corresponding time instances (at which the inequality constraints are active) in advance, we could impose the constraint (6c) only at those instances, and would thereby not change the optimal solution, nor the optimal cost.

[2]The problem of checking whether η_z fulfills (6c) is discussed in Sec. IV.

Algorithm 1 Iterative constraint sampling

```
Initialize: initial guess for the constraint sampling points:
    $I_0 = \{t_1, t_2, \ldots, t_N\}$; maximum number of iterations:
    MAXITER;
 1: $k = 0$
 2: for $k <$MAXITER do
 3:     solve (9) for $I_k \to \eta_z^k$
 4:     if infeasible then
 5:         abort
 6:     else if $\eta_z^k$ fulfills (6c) then
 7:         algorithm converged
 8:         return $\eta_z^k$
 9:     end if
10:     find at least one constraint violation instant $\to t_c$
11:     remove inactive time instants in $I_k \to \bar{I}_k$
12:     $I_{k+1} = \bar{I}_k \cup \{t_c\}$, $k = k + 1$
13: end for
```

In other words, if the collection of these active time instances is denoted by I^*, it follows that $J(I^*) = J$.

In addition, a similar argument can be used to conclude that the optimization (9) can have at most $2(s-1)n_c$ active constraints. Thus, the index set I is guaranteed not to exceed a size of at most $2(s-1)n_c + 1$ elements (we may add one additional constraint violation instant in Alg. 1). This is because, if all the stationary points of (6c) (stationary with respect to t in a row-wise sense) violate the constraint (6c), there are at most $2(s-1)$ intersections with the corresponding lower and upper bounds, which limits the maximum number of active constraints to $2(s-1)n_c$.

As we will show in the following, the algorithm is guaranteed to converge. Hence, the sets I^k are approximating the set I^* better and better as k increases.

Proposition 3.1: Provided that (6) is feasible, the sequence of optimizers η_z^k in Algorithm 1 converges to the optimizer of (6). Similarly, the cost $J(I_k)$ converges to J for $k \to \infty$.

Proof: By assumption, the cost J is finite and the corresponding optimizer η_z is unique. From the fact that the optimizer η_z is a feasible candidate to the optimization with cost $J(I_k)$ (where the constraint (6c) is only imposed at the constraint sampling instances contained in I_k) it follows that $J(I_k) \leq J$, that the infimum in (9) is attained, and that the corresponding minimizer η_z^k is unique for all k.

We define the set $\bar{I}_k$ as the set of the active constraints contained in I_k. When removing the inactive constraints, the optimizer and the cost remain unchanged, and thus it holds that $J(\bar{I}_k) = J(I_k)$.

We claim that $J(I_{k+1}) > J(I_k) = J(\bar{I}_k)$ (provided that $J(I_k) \neq J$). Proof by contradiction: If $J(I_{k+1}) < J(I_k)$ this would imply that the optimizer η_z^{k+1} with cost $J(I_{k+1})$, which is a feasible candidate for the optimization with cost $J(\bar{I}_k) = J(I_k)$, achieves a lower cost, leading to a contradiction. If $J(I_{k+1}) = J(I_k)$ this implies by the same argument and the uniqueness of the optimizer η_z^k, that $\eta_z^k = \eta_z^{k+1}$. However, η_z^k violates the constraints at least at one time instant for which no violation occurs with η_z^{k+1}, leading again to a contradiction.

Thus, $J(I_k)$ is a strictly increasing sequence, bounded above by J, and therefore converges. It remains to show that $J(I_k)$ converges to J and that the sequence η_z^k converges to η_z. Due to the strong convexity of the objective function, the two-norm $|\eta_z^k - \eta_z^{k-1}|_2^2$ can be bounded by a constant times $J(I_k) - J(I_{k-1})$. This is because η_z^k is a feasible candidate to the optimization corresponding to the cost $J(\bar{I}_{k-1}) = J(I_{k-1})$ and so is $1/2(\eta_z^k + \eta_z^{k-1})$. However, the feasible candidate $1/2(\eta_z^k+\eta_z^{k-1})$ attains certainly a cost greater than $J(\bar{I}_{k-1}) = J(I_{k-1})$, leading to

$$J(I_{k-1}) \leq \frac{1}{8}(\eta_z^k + \eta_z^{k-1})^{\mathsf{T}} H (\eta_z^k + \eta_z^{k-1}). \tag{10}$$

From the strong convexity of the objective function it can be concluded that

$$J(I_{k-1}) \leq \frac{1}{2}(J(I_k) + J(I_{k-1})) - \frac{\sigma}{8}|\eta_z^k - \eta_z^{k-1}|_2^2, \tag{11}$$

where σ corresponds to smallest eigenvalue of H. This implies

$$|\eta_z^k - \eta_z^{k-1}|_2^2 \leq 4\sigma^{-1}(J(I_k) - J(I_{k-1})), \tag{12}$$

and therefore the sequence η_z^k converges. As a result, $\lim_{k\to\infty} \eta_z^k$ exists and is guaranteed to fulfill the constraints. It is therefore a feasible candidate for (6), implying that $\lim_{k\to\infty} J(I_k) \geq J$, which, combined with $J(I_k) \leq J$, leads to $\lim_{k\to\infty} J(I_k) = J$. Thus, $\lim_{k\to\infty} \eta_z^k$ is in fact the optimizer of (6) and by uniqueness, it follows that $\lim_{k\to\infty} \eta_z^k = \eta_z$. ■

In practice, the optimization (9) in Alg. 1 can be warm-started with the optimizer η_z^k of the previous step, and can therefore be solved efficiently. Moreover, the solution tolerance can be successively increased over the iterations in the hope that the constraint sampling instances I^* are well-approximated by I^k within the first few iterations.

It remains to discuss an efficient implementation for checking whether the constraint (6c) is violated for a given parameter vector η_z and retrieving at least one corresponding constraint violation instant.

IV. A RECURSIVE STRATEGY FOR CHECKING CONSTRAINT SATISFACTION

This section illustrates a strategy for checking the satisfaction of the constraint (6c).

The basis functions are spanned by exponentially decaying polynomials. For polynomials of degree $s-1 \leq 4$, there are closed-form expressions for the stationary points of the mapping

$$t \to (C \otimes \tau(t))^{\mathsf{T}} \eta_z, \tag{13}$$

as a function of the parameter vector η_z (the exponential decay does not affect the stationary points). Consequently, one could check whether the constraint (6c) is satisfied by evaluating (13) at its stationary points. Although such an approach tends to be very efficient, it does not generalize to polynomials of higher order, and therefore, we will focus on an alternative strategy. We will need the following two ingredients:

1) Due to the exponential decay of the basis functions it can be concluded that it is enough to check the constraint (6c) over a finite time interval;
2) The trajectories in (13) are approximated by Bézier curves to enable quick but conservative constraint satisfaction checks. If the check is not conclusive, the approximation of (13) is refined, and the check is repeated.

Both elements, the reduction of (6c) to a finite time interval and the approximation of (13) by Bézier curves, are discussed below.

A. Reduction to a finite time interval

According to [24, p. 280] exponentially decaying polynomials have the following property

$$\sup_{t\in[0,\infty)} |p_s(t)e^{-t/2}| = \max_{t\in[0,4(s-1)]} |p_s(t)e^{-t/2}|, \tag{14}$$

which holds for any polynomial $p_s : [0,\infty) \to \mathbb{R}$ that has a maximum degree of $s-1$. By means of an appropriate time scaling the above relation can be used to conclude

$$\sup_{t\in[0,\infty)} |(C\otimes\tau(t))^{\mathsf{T}}\eta_z| = \max_{t\in[0,2(s-1)/\lambda]} |(C\otimes\tau(t))^{\mathsf{T}}\eta_z| \tag{15}$$

for any parameter vector η_z, where the absolute value $|\cdot|$ and the supremum are applied component wise. Hence, it is enough to check the constraint (6c) over the compact interval $[0, 2(s-1)/\lambda]$ instead of the unbounded interval $[0,\infty)$.

B. An iterative procedure for checking the constaints

In the following, we will construct upper and lower bounds on (13) using Bézier curves.[3] We will then exploit the convex hull property of Bézier curves to come up with an efficient, but conservative test for checking (6c). If the test is not decisive, the approximation of (13) is refined using de Casteljau's algorithm, [26, p. 151]. The procedure is repeated until a decisive answer is obtained.

In order to construct upper and lower bounds on (13) we will approximate the exponential and polynomial parts of the basis functions separately. Furthermore, time is re-scaled for mapping the interval $[0, 2(s-1)/\lambda]$ to $[0,1]$ (Bézier curves are commonly defined on $[0,1]$). As a consequence, (6c) is restated as

$$l_{\mathrm{l}} \leq \left(C \otimes \hat{\tau}\left(\frac{2(s-1)t}{\lambda}\right)\right)^{\mathsf{T}} \eta_z \; e^{-2(s-1)t} \leq l_{\mathrm{u}}, \tag{16}$$

for all $t \in [0,1]$, where the basis functions τ are divided into the polynomial part, denoted by $\hat{\tau}$, and the exponential decay. The polynomial part is rewritten as a Bézier curve of degree $s-1$, that is,

$$\left(C \otimes \hat{\tau}\left(\frac{2(s-1)t}{\lambda}\right)\right)^{\mathsf{T}} \eta_z = \sum_{k=0}^{s-1} \beta_k(\eta_z) B_{k,s-1}(t), \tag{17}$$

[3]For the sake of completeness the key properties of Bézier curves are summarized in the online appendix [25].

for all $t \in [0,1]$, where $\beta_k(\eta_z) \in \mathbb{R}^{n_c}$ are the control points and $B_{k,s-1}$ are the Bernstein polynomials of order $s-1$, [26, p. 144], $k = 0,1,\ldots,s-1$. The control points are linearly dependent on the parameter vector η_z, and are therefore determined by multiplying η_z with a sparse matrix that can be computed offline (see for example [26, Ch. 7.3] for a conversion from Bernstein to monomial form).

According to [25, Prop. 6.1], the exponential decay can be approximated over the time interval $[0,1]$ as

$$e^{-2(s-1)t} \approx \sum_{k=0}^{n_e} e^{-2k(s-1)/n_e} B_{k,n_e}(t), \qquad (18)$$

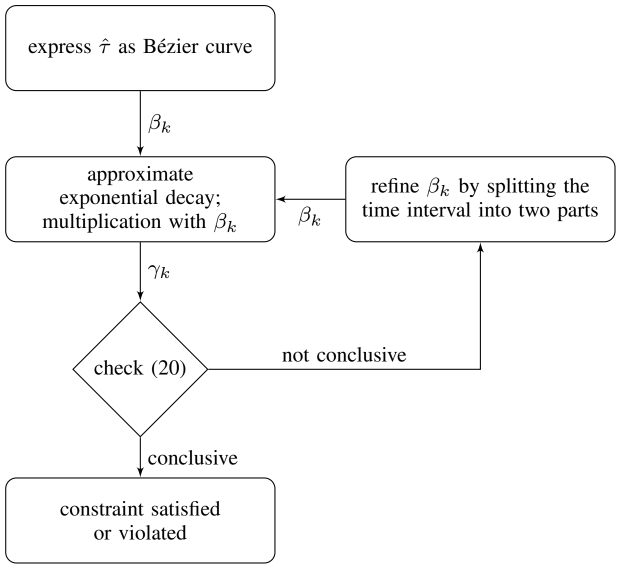
where n_e is a fixed positive integer, typically greater or equal to $s-1$, that controls the approximation quality. The above approximation extends naturally to the case where the exponential decay is approximated over a compact subset of $[0,1]$, which will become important in a later stage. By multiplying the two Bézier curves, that is, (17) and (18), the constraint (16) is approximated as

$$l_\mathrm{l} \leq \sum_{k=0}^{s-1+n_e} \gamma_k(\eta_z) B_{k,s-1+n_e}(t) \leq l_\mathrm{u}, \quad \forall t \in [0,1]. \qquad (19)$$

Combining the fact that the exponential decay is a convex function in t with [25, Prop. 6.1] guarantees that the Bézier curve (18) is an upper bound on the exponential decay, and as a consequence, (19) implies (16) (but certainly not vice versa). In fact, the approximation (19) is exact at the points $t = 0$ and $t = 1$. The control points $\gamma_k(\eta_z) \in \mathbb{R}^{n_c}$ that parametrize the Bézier curve in (19) are linearly dependent on η_z, and can therefore be computed easily. An explicit formula for calculating the product of two Bézier curves can be found in the appendix. The convex hull property of Bézier curves enables efficient checks for deciding whether (6c) is fulfilled. More precisely, if

$$l_\mathrm{l} \leq \gamma_k(\eta_z) \leq l_\mathrm{u}, \quad k = 0,1,\ldots,s-1+n_e, \qquad (20)$$

holds, it can be concluded that (6c) is satisfied. If the above condition is violated for all control points γ_k, it can be concluded that (6c) is certainly not satisfied. The same conclusion can be drawn, if the above constraint is violated by γ_0 or γ_{s-1+n_e}, as these two control points are guaranteed to lie on the trajectory. Hence, γ_0, respectively γ_{s-1+n_e} would then yield potential constraint violation instants. If the check is not decisive, the approximation of (16) by the Bézier curve (19) is refined by splitting the interval $[0,1]$ into two parts (that may have different lengths). De Casteljau's algorithm is used for calculating the two Bézier curves corresponding to the polynomial part in (16). Likewise, the approximation of the exponential decay is refined, in order to obtain conditions similar to (19) for the two sub intervals of $[0,1]$. This refinement step is repeated until a decisive answer is found. The approach is summarized by the flow chart shown in Fig. 1.



express $\hat{\tau}$ as Bézier curve
β_k
approximate exponential decay; multiplication with β_k
refine β_k by splitting the time interval into two parts
β_k
γ_k
check (20)
not conclusive
conclusive
constraint satisfied or violated

Fig. 1. The flow chart illustrates the iterative procedure for checking (6c).

C. Numerical example

The proposed approach is illustrated on a numerical example as shown in Fig. 2. It is checked whether the curve $f : [0,\infty) \to \mathbb{R}$, consisting of a linear combination of the first 5 basis functions, satisfies the constraint $-1.8 \leq f(t) \leq 8$ for all times $t \in [0,\infty)$. After rescaling time, the check can be reduced to the interval $[0,1]$, see (16). The top figure shows the original curve f in black and its approximation by a Bézier curve (in blue) according to (19). Indeed, the Bézier curve represents an upper bound, whenever f is positive, and a lower bound, whenever f is negative. The corresponding control points γ_k, $k = 0,1,\ldots,9$, are indicated by the blue stars, and their corresponding convex hull is shown in blue (dashed). The condition (20) is then checked, and found to be violated for some of the control points (but not for γ_0 and γ_9). As a result, the approximation of f over the interval $[0,1]$ is refined by splitting the interval into $[0,0.3]$ and $[0.3,1]$. The corresponding Bézier curves are shown in blue and red (second figure), and are constructed using de Casteljau's algorithm and by refining the exponential decay. The corresponding control points are again indicated by stars (blue for the interval $[0,0.3]$ and red for the interval $[0.3,1]$). Although the two Bézier curves yield a much better approximation of f, some of the inner control points are still violating the constraint, and therefore, the approximations are refined once more. This leads to the last figure, where the algorithm terminates, guaranteeing constraint satisfaction. The required computation amounts to two matrix multiplications per refinement step (one as a consequence of de Casteljau's algorithm and one due to the multiplication with the exponential decay) and evaluating (20), which is linear in the number of control points.

V. Experimental results

The proposed MPC approach is evaluated in real-world experiments with the Flying Platform, an unmanned aerial

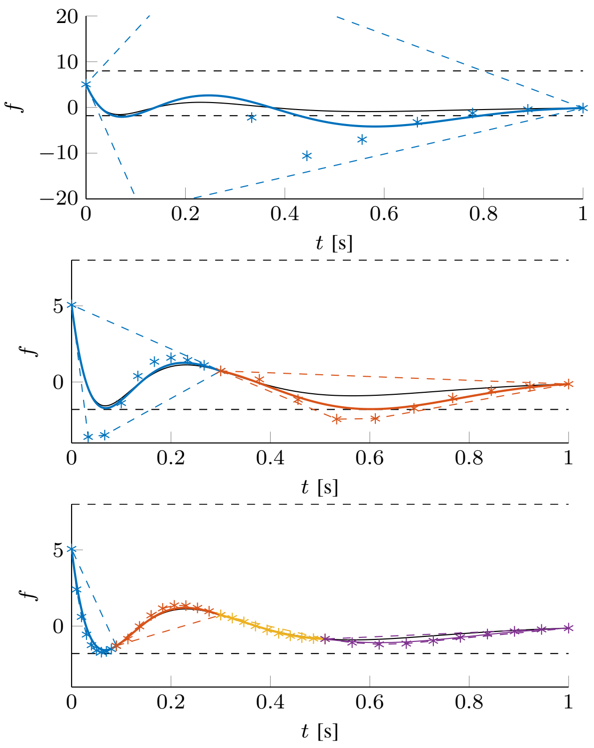


Fig. 2. The recursive constraint refinement illustrated on a simple example with three steps. Shown is the original curve (black, solid), its Bézier approximation according to (19) (solid, colored), the upper and lower bounds (black, dashed), the control points γ_k (stars, colored), and their convex hull (dashed).

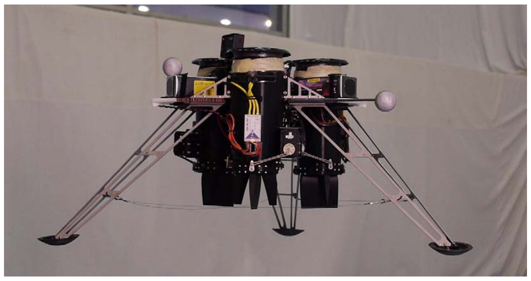


Fig. 3. The Flying Platform hovers in the Flying Machine Arena, [27]. Two flaps for redirecting the airflow are mounted below each electric ducted fan.

vehicle that serves as a testbed for electric ducted fan actuation. Fig. 3 shows the Flying Platform during flight in the Flying Machine Arena, [27].[4]

A. The Flying Platform

1) Hardware: The Flying Platform is powered by three electric ducted fans, providing each 40 N of thrust. It has a total mass of roughly 8 kg, and an overall dimension of about 1 m. The airflow through each ducted fan is redirected by two control flaps in order to vector the thrust. The three ducted fans are all rotating in the same direction and therefore thrust vectoring is essential for stabilizing the vehicle about hover. The Flying Platform is controlled by a combination of thrust vectoring and differential thrust, leading to a total of nine inputs (two control flaps per fan, one thrust command per fan, three fans).

Position and attitude information is provided by a motion capture system, and linear velocity estimates are obtained by an offboard state-estimation. Both, the position and attitude information, as well as the linear velocity estimate is sent to the vehicle via wireless communication. The angular velocities are measured with an onboard gyroscope.

The PX4 flight management unit[5] provides a low-level interface to the actuators and the onboard gyroscope. The predictive control algorithm is run on a Gumstix duo-vero computer-on-module[6] that interfaces the PX4 via serial communication. The Gumstix is equipped with an ARM Cortex A9 dual core processor with a clock frequency of 1 GHz and with 1 GB random-access memory. The Linux-based operating system Ubuntu is run on the Gumstix.

2) State-space model: A nonlinear model based on first principles is linearized about hover. The linearization is carried out in a yaw-fixed coordinate system leading to a model that is valid for all yaw orientations. The states are position, attitude, linear and angular velocity, resulting in a total of 12 states. The input to the system is given by the three thrust vectors corresponding to each actuation unit, that is, $u(t) = (T_1(t), T_2(t), T_3(t)) \in \mathbb{R}^9$, where $T_i(t) \in \mathbb{R}^3$, $i = 1, 2, 3$. The dependence on t is occasionally omitted to simplify notation. As a result, a standard state-space description of the dynamics is obtained, where the values of the system matrices A and B are obtained from a grey-box system identification, see [28]. Controlling the system is challenging as the actuation is limited (see below) and the open-loop system is unstable (with an unstable pole at around 5 rad/s). The unstable pole is attributed to the fact that the ducted fans redirect crosswinds that stem, for example, from horizontal flight, [29].

The thrust commands are related by a one-to-one correspondence to the position commands of the control flaps and the thrust commands of the ducted fans. The maximum thrust produced by each fan is limited to $T_{\max} = 40\,\text{N}$, whereas the maximum pivoting angle of the flaps is required to be below $\psi_{\max} = 12°$. Linearizing the resulting nonlinear constraints about hover yields the following actuation limits

$$T_{i3} \in [0, T_{\max}], \tag{21}$$

$$T_{i1} \in \tan\left(\frac{\psi_{\max}}{c_1}\right) T_z \; [-1, 1], \tag{22}$$

$$T_{i2} \in \tan\left(\frac{\psi_{\max}}{c_2}\right) T_z \; [-1, 1], \tag{23}$$

$i = 1, 2, 3$, where c_1 and c_2 denote the proportional constants between thrust angle and flap angle and $T_z \in \mathbb{R}$ denotes the steady-state (hover) thrust of each ducted fan. The constants c_1 and c_2 are identified from measurements. The components of the thrust vector $T_i \in \mathbb{R}^3$, as generated by the ith fan and

[4]A video showing parts of the experiments with the Flying Platform can be found under `https://youtu.be/GgIwrnoNvTY`.

[5]`https://pixhawk.org/modules/px4fmu`

[6]`https://www.gumstix.com`

the corresponding control flaps, are denoted by T_{i1}, T_{i2}, and T_{i3}. Due to the fact that $T_{i1}, T_{i2} \ll T_{i3}$, the constraints are well-approximated by (21)-(23).

B. Implementation details

The matrix Q penalizing the state deviation in the running cost in (1) is chosen to be[7]

$$Q = \mathrm{diag}(\overbrace{50, 50, 10}^{\text{position}}, \overbrace{10, 10, 10}^{\text{lin. velocity}}, \overbrace{40, 40, 10}^{\text{attitude}}, \overbrace{10, 10, 5}^{\text{ang. velocity}}). \quad (24)$$

The z-components of the thrust inputs enter the running cost as

$$\overbrace{0.1 \left(\frac{1}{3} \sum_{i=1}^{3} (T_{i3} - T_z) \right)^2}^{\text{total thrust}} + \overbrace{0.1 \sum_{i,j=1, i \neq j}^{3} \left(\frac{T_{i3} - T_{j3}}{2} \right)^2}^{\text{differential thrust}}, \quad (25)$$

whereas the horizontal thrust components are penalized with

$$0.01 \left(\sum_{i=1}^{3} 2\, T_{i1}^2 + T_{i2}^2 \right). \quad (26)$$

The predictive control algorithm runs at 100 Hz, and Alg. 1 is used for solving (6). The quadratic program (9) is solved with the generalized fast dual gradient method (GFDG), as presented in [30], which is motivated by a previous study including a comparison of different first-order optimization routines, [7]. Moreover, the projection step that is part of the GFDG method and is itself a (smaller) quadratic program, is solved using qpOASES, [31]. The optimization problems that are to be solved tend to vary only slowly over the different iterations of Alg. 1 and the different time steps, and therefore, both qpOASES and the GFDG method are warm-started. Moreover, in the implementation of Alg. 1 the total number of constraint sampling points (the maximum number of elements of I_k) is limited to N_{max}, and inactive constraints are only removed if this limit is reached. The set I_0 is chosen as the time instants t_i, satisfying

$$\tau(t_i)^{\mathsf{T}} \tau(t_j) = 0 \quad (27)$$

for all $i, j = 0, 1, \ldots, s-1$, with $i \neq j$ and $t_0 = 0$. The choice is motivated in [7]. The constraint (6c) is checked using the recursive procedure presented in Sec. IV. The different parameters characterizing the basis functions, as well as the parameters used for Alg. 1 and the constraint checking procedure are summarized in Tab. I. The choice $s = 5$ represents a reasonable trade-off between approximation quality and computational effort, as motivated by a previous study, [7].

[7]Note that all states and inputs are expressed using SI units. However, for better readability, the units have been omitted in the equations defining the running cost.

TABLE I
PARAMETERS USED FOR THE IMPLEMENTATION OF THE PREDICTIVE CONTROL ALGORITHM.

value	description
$5\,\mathrm{s}^{-1}$	exponential decay of basis functions (λ)
5	number of basis functions (s)
2	number of iterations in Alg. 1 (MAXITER)
10^{-5}	relative tolerance used for solving (9)
10	max. number of refinements in constraint check
0.3	interval division in constraint refinement
5	order of Bézier curve for the exp. decay (n_e)
135	maximum number of elements of I_k (N_{max})

C. Results

The predictive controller provides good disturbance rejection properties in practice. In hover, the average execution time is roughly 3 ms, and the root-mean-square position error is about 0.02 m. Disturbance rejection measurements are shown in Fig. 4, where a rapid recovery can be observed. The disturbance consists of an abrupt set point shift in horizontal direction for 0.3 s before moving the set point back to where it was. As can be seen in Fig. 4, the execution time rises immediately when the disturbances are applied, as the constraints become active. In such cases, Alg. 1 may require several iterations. The bulk of the computation lies in the solution of (9), which takes roughly 3 ms. Checking constraint satisfaction typically takes 1 ms. For the disturbance shown in Fig. 4, satisfaction of the constraint (6c) can be guaranteed and thereby also closed-loop stability. However, when applying larger disturbances, Alg. 1 might terminate early. In such cases, closed-loop stability might no longer be guaranteed.

VI. Conclusion

The article presents the implementation details of an online model predictive control strategy with inherent stability and recursive feasibility guarantees. An iterative algorithm is presented for enforcing the constraints over the time interval $[0, \infty)$, and convergence of the algorithm is shown. The algorithm is evaluated in practice by controlling an unmanned aerial vehicle. The flight tests indicate that the algorithm is suitable for online predictive control.

ACKNOWLEDGMENT

The authors would like to thank Marc-Andrè Corzillius, Michael Egli, Matthias Hofer, and Lukas Fröhlich for their contribution to the development of the Flying Platform.

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

References

[1] S. J. Quin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.

[2] J. Richalet, "Industrial applications of model based predictive control," *Automatica*, vol. 29, no. 5, pp. 1251–1274, 1993.

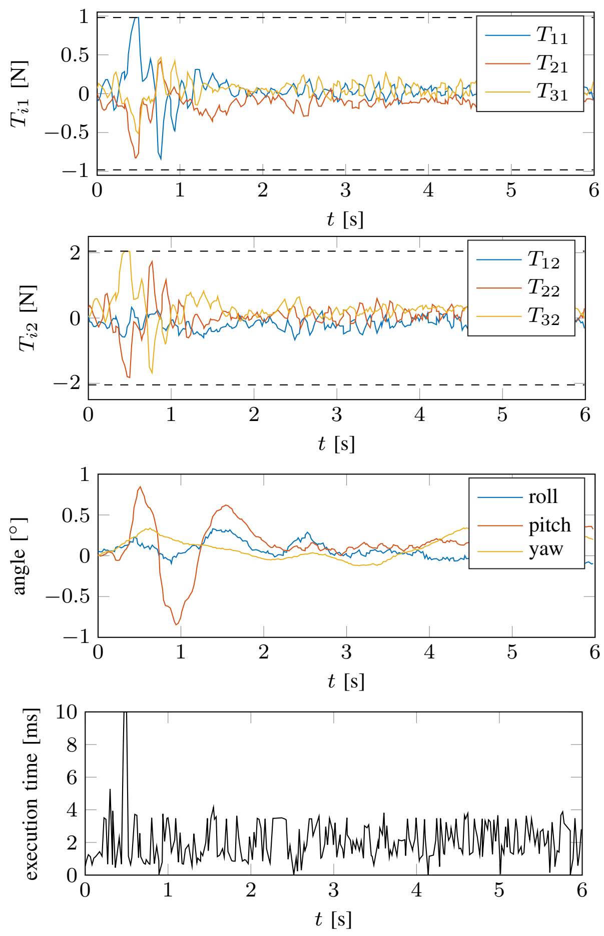


Fig. 4. Disturbance rejection of the Flying Platform: The disturbance is injected at 0.3 s and lasts until 0.6 s. The exectution time of the optimization algorithm immediately increases and the constraints represented as dashed lines are hit. The optimization had to be stopped prematurely for two time steps. Nonetheless, constraint satisfaction is always guaranteed. In steady state, the execution time amounts to roughly 3 ms.

[3] T. Geyer, G. Papafotiou, and M. Morari, "Model predictive direct torque control – part I: Concept, algorithm, and analysis," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1894–1905, 2009.

[4] G. Papafotiou, J. Kley, K. G. Papadopoulos, P. Bohren, and M. Morari, "Model predictive direct torque control – part II: Implementation and experimental evaluation," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1906–1915, 2009.

[5] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[6] M. Muehlebach and R. D'Andrea, "Parametrized infinite-horizon model predictive control for linear time-invariant systems with input and state constraints," *Proceedings of the American Control Conference*, pp. 2669–2674, 2016.

[7] M. Hofer, M. Muehlebach, and R. D'Andrea, "Application of an approximate model predicitve control scheme on an unmanned aerial vehicle," *Proceedings of the International Conference on Robotics and Automation*, pp. 2952–2957, 2016.

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

[9] L. Wang, "Continuous time model predictive control design using orthonormal functions," *International Journal of Control*, vol. 74, no. 16, pp. 1588–1600, 2001.

[10] A. Jadbabaie, J. Yu, and J. Hauser, "Stabilizing receding horizon control of nonlinear systems: A control Lyapunov function approach," *Proceedings of the American Control Conference*, pp. 1535–1539, 1999.

[11] R. Franz, M. Milam, and J. Hauser, "Applied receding horizon control of the Caltech ducted fan," *Proceedings of the American Control Conference*, pp. 3735–3740, 2002.

[12] N. Slegers, J. Kyle, and M. Costello, "Nonlinear model predictive control technique for unmanned air vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 5, pp. 1179–1188, 2006.

[13] H. J. Kim, D. H. Shim, and S. Sastry, "Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles," *Proceedings of the America Control Conference*, pp. 3576–3581, 2002.

[14] S. Bertrand, H. Piet-Lahanier, and T. Hamel, "Contractive model predictive control of an unmanned aerial vehicle model," *Proceedings of the IFAC Symposium on Automatic Control in Aerospace*, pp. 450–455, 2007.

[15] G. Calafiore and M. Campi, "Uncertain convex programs: randomized solutions and confidence levels," *Mathematical Programming*, vol. 102, no. 1, pp. 25–46, 2005.

[16] D. P. de Farias and B. V. Roy, "On constraint sampling in the linear programming approach to approximate dynamic programming," *Mathematics of Operations Research*, vol. 29, no. 3, pp. 462–478, 2004.

[17] C. Scherer, "LMI relaxations in robust control," *European Journal of Control*, vol. 12, no. 3, pp. 3–29, 2006.

[18] Y. Nesterov, "Squared functional systems and optimization problems," in *High Performance Optimization*, H. Frenk, K. Roos, T. Terlaky, and S. Zhang, Eds. Springer, 2000, ch. 17, pp. 405–440.

[19] D. Bampou and D. Kuhn, "Polynomial approximations for continuous linear programs," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 628–648, 2012.

[20] R. Deits and R. Tedrake, "Efficient mixed-integer planning for UAVs in cluttered environments," *Proceedings of the International Conference on Robotics and Automation*, pp. 42–49, 2015.

[21] S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar, "Generation of whole-body optimal dynamic multi-contact motions," *International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1104–1119, 2013.

[22] K. Hauser, "Fast interpolation and time-optimization on implicit contact submanifolds," *Proceedings of the Robotics: Science and Systems Conference*, 2013.

[23] M. Abramowitz and I. A. Stegun, Eds., *Handbook of Mathematical Functions*, 10th ed. United States Department of Commerce - National Bureau of Standards, 1972.

[24] G. Mastroianni and G. V. Milovanović, *Interpolation Processes*. Springer, 2008.

[25] M. Muehlebach, C. Sferrazza, and R. D'Andrea, "Online appendix - properties of Bézier curves," available on the first author's webpage http://www.idsc.ethz.ch/research-dandrea/people/person-detail.html?persid=156097.

[26] D. Marsh, *Applied Geometry for Computer Graphics and CAD*, 2nd ed. Springer, 2005.

[27] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The Flying Machine Arena," *Mechatronics*, vol. 24, pp. 41–54, 2014.

[28] M. Muehlebach and R. D'Andrea, "The Flying Platform - A testbed for ducted fan actuation and control design," *Mechatronics*, vol. 42, pp. 52–68, 2017.

[29] J.-M. Pfimlin, P. Binetti, P. Souères, T. Hamel, and D. Trouchet, "Modeling and attitude control analysis of a ducted-fan micro aerial vehicle," *Control Engineering Practice*, vol. 18, no. 3, pp. 209–218, 2010.

[30] P. Giselsson, "Improved fast dual gradient methods for embedded model predicitive control," *Proceedings of the IFAC World Congress*, vol. 47, no. 3, pp. 2303–2309, 2014.

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

[32] G. M. Phillips, *Interpolation and Approximation by Polynomials*. Springer, 2003.