Trajectory Tracking and Iterative Learning on an Unmanned Aerial Vehicle using Parametrized Model Predictive Control

Carmelo Sferrazza, Michael Muehlebach, and Raffaello D'Andrea

Abstract— **A parametrization of state and input trajectories is used to approximate an infinite-horizon optimal control problem encountered in model predictive control. The resulting algorithm is discussed with respect to trajectory tracking, including the problem of generating feasible trajectories. In order to account for unmodeled repeatable disturbances an iterative learning scheme is applied, and as a result, the tracking performance can be improved over consecutive trials. The algorithm is applied to an unmanned aerial vehicle and shown to be computationally efficient, running onboard at a sampling rate of 100 Hz during the experiments.**

I. INTRODUCTION

Model Predictive Control (MPC) is an established control strategy for addressing challenging control problems, often including input and state constraints, see for example [1] and references therein. It is based on the following procedure: at each time step, an optimal control problem is solved and the first portion of the resulting input trajectory is applied to the system. This yields an implicit feedback law providing robustness against disturbances and modeling errors.

Due to the fact that an optimal control problem has to be solved at each time step, MPC is computationally demanding, particularly when applied to systems with fast dynamics. A common approach to reduce the computational complexity is to discretize the dynamics and truncate the prediction horizon. This leads naturally to a trade-off between computation and prediction horizon. However, the truncation of the prediction horizon might require the introduction of a terminal cost and terminal state constraints to preserve stability, see [1], [2].

In contrast, the MPC approach presented in [3] retains an infinite prediction horizon by parametrizing input and state trajectories with decaying basis functions. Provided that the resulting trajectories fulfill the constraints for all times, this leads to inherent closed-loop stability and recursive feasibility guarantees. Moreover, by choosing suitable basis functions, the resulting MPC formulation tends to have fewer optimization variables and therefore exhibits small execution times, as reported in [4].

This article extends the approach presented in [3] to trajectory tracking and discusses the application to an unmanned aerial vehicle. In addition, an iterative learning scheme is incorporated in order to reject repeatable disturbances.

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

The authors are with the Institute for Dynamic Systems and Control, ETH Zurich. Email correspondence to Carmelo Sferrazza csferrazza@ethz.ch.

A. Related Work

Due to the increase in computational power and the availability of dedicated optimization routines, see for example [5], [6], MPC has been applied to a wide range of dynamic systems. The vast majority of MPC controllers found in the literature are based on a discrete-time finite-horizon formulation, an overview of which is given in [1]. In [7] and [8, Ch. 3, Ch. 6], an alternative formulation is proposed, where the finite differences of the control inputs (in discrete time) or the time derivative of the inputs (in continuous time) are parametrized with so-called Laguerre or Kauz basis functions. Although similar basis functions are used herein, their approach is different due to the fact that a finite prediction horizon is retained, and that the control inputs are not parametrized directly. Moreover, the state variable is eliminated, whereas we encode the dynamics as an equality constraint that may or may not be eliminated, potentially leading to a sparser optimization problem.

In the following, we point out some applications of MPC to real-world systems, which we find relevant for our work. This includes the control of unmanned aerial vehicles with MPC, and trajectory tracking or iterative learning in combination with MPC.

In [9], the application of MPC to a thrust-vectored flight control experiment with a ducted fan actuation is presented. Input and state constraints are included and the region of attraction of the MPC controller is shown to be larger than that of the corresponding linear quadratic regulator.

Different strategies have been presented to tackle the problem of trajectory tracking with MPC. In [10], a successive linearization approach has been applied to the discrete-time infinite-horizon MPC strategy. This method has been shown to be more efficient than a nonlinear MPC formulation, while maintaining a comparable performance, when applied to a mobile robot. Guarantees for recursive feasibility have been presented in [11] for constrained trajectory tracking problems through the use of time-varying terminal regions. In [12], the tracking control problem of underactuated vehicles is addressed by allowing an asymptotic tracking error. This provides a means to compute the terminal set and the terminal control law that guarantee asymptotic convergence of the position of the vehicle to a tube centered around the desired path. A direct multiple-shooting approach is introduced in [13] for solving optimal control problems encountered in MPC. The approach is evaluated in simulation, by performing extreme maneuvers with a flying vehicle. In [14], a MPC framework is implemented and used for trajectory tracking of a quadrotor, where the dynamics are modeled as a set of piecewise affine systems given by different operating points. An application of MPC to the trajectory tracking of a formation of flying vehicles is presented in [15].

Unlike most of the approaches summarized above, we do not discretize the dynamics, but parametrize input and state trajectories with exponentially decaying basis functions. As a result, we retain an infinite prediction horizon, and obtain an optimization problem with relatively few variables. The numerical effectiveness of our approach will be demonstrated by the fact that the resulting MPC controller achieves a sampling time of 100 Hz on an embedded computer (Gumstix DuoVero COM).

Trajectory tracking can often be improved by incorporating learning approaches. Iterative Learning Control (ILC), see [16], provides a way to iteratively improve the system model available to the controller over multiple trials. An overview of ILC can be found in [17]. In [18], a time-varying Kalman filter is used to estimate the repeatable disturbances along a trajectory, which might stem from unmodeled system dynamics and uncertainties in the physical parameters. We will use a similar approach to identify the repeatable disturbances, and incorporate them into our MPC formulation.

In [19] and [20], learning approaches are included in an MPC framework. In [21], deep learning is combined with MPC for guided policy search, where MPC is used to generate data at training time. The method is evaluated with simulations of a quadrotor's flight. In [22], an underlying control sequence is included as a (deficient) reference to be improved for the predictive tracking control. At each iteration, the input sequence is corrected by performing a learning update. A similar strategy has been proposed in [23], where the MPC performance is improved by feeding back the control errors from previous iterations, based on the concept of repetitive control.

Applications of ILC in combination with MPC have been shown on a pH plant, see [24], and in [25], where learning-based MPC has been applied on a quadrotor that has been trained to catch a ball.

In our approach, we augment our system's model with repeatable disturbances, which we parametrize using the same basis functions as for representing the input and the states. As a consequence, these disturbances can be naturally incorporated in our MPC formulation. In that way, the system's model is updated over consecutive trials, improving the accuracy of the predictions used for MPC.

B. Outline

The parametrized MPC problem is presented in Section II. Input and state trajectories are approximated through a linear combination of Laguerre functions, and a constraint sampling strategy is proposed. Section III discusses the problem of generating a trajectory that is parametrized by the given basis functions. The online trajectory tracking is explained in Section IV. In Section V, an iterative learning scheme is incorporated. Simulation and experimental results obtained from flights with an unmanned aerial vehicle are shown in Section VI. Concluding remarks are made in Section VII.

II. PARAMETRIZED MPC

In order to approximate the infinite-horizon optimal control problem that will be used in our MPC approach, we will represent state and input trajectories as linear combinations of Laguerre functions, that is,

$$\tilde{x}(t) := (I_n \otimes \tau(t))^{\mathsf{T}} \eta_x, \quad \tilde{u}(t) := (I_m \otimes \tau(t))^{\mathsf{T}} \eta_u, \tag{1}$$

with $\tau(t) := (\tau_1(t), \tau_2(t), \ldots, \tau_s(t))$, for all $t \in [0, \infty)$, and

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

where $\eta_x \in \mathbb{R}^{ns}$, $\eta_u \in \mathbb{R}^{ms}$ are the parameter vectors, λ is the exponential decay, n and m describe the state respectively the input dimension, and $\otimes$ denotes the Kronecker product. This approach has been previously presented in [3], where additional motivation and examples are included. For ease of notation, vectors are expressed as n-tuples, with dimension and stacking clear from the context, i.e. $\tau(t) = (\tau_1(t), \tau_2(t), \ldots, \tau_s(t)) \in \mathbb{R}^s$.

It can be shown that the basis functions $\tau(t)$ satisfy the following properties,

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

where

$$M_\lambda = \begin{bmatrix} -\lambda & 0 & \cdots & 0 \\ -2\lambda & -\lambda & & \vdots \\ \vdots & & \ddots & 0 \\ -2\lambda & \cdots & -2\lambda & -\lambda \end{bmatrix} \in \mathbb{R}^{s \times s}, \tag{4}$$

which will be used in a later stage.

As discussed in [3], by representing input and state trajectories with $\tilde{x}$ and $\tilde{u}$ according to (1), the constrained infinite-horizon linear-quadratic optimal regulator problem can be approximated as

$$\inf_{\eta_x, \eta_u} \frac{1}{2} \left\{ \eta_x^{\mathsf{T}} (Q \otimes I_s) \eta_x + \eta_u^{\mathsf{T}} (R \otimes I_s) \eta_u \right\} \tag{5}$$

$$\text{s.t.} \quad A_x \eta_x + B_u \eta_u = 0, \tag{6}$$

$$(I_n \otimes \tau(0))^{\mathsf{T}} \eta_x = x_0, \tag{7}$$

$$F \eta_u \in \mathcal{U} := [u_{\min}, u_{\max}]^s, \tag{8}$$

where $u_{\min} \in \mathbb{R}^m$ and $u_{\max} \in \mathbb{R}^m$ are the lower and upper bounds on the control input $u(t)$, $Q \in \mathbb{R}^{n \times n}$ is positive definite ($Q \succ 0$), $R \in \mathbb{R}^{m \times m}$ is positive definite ($R \succ 0$),

$$F := \begin{bmatrix} \mathrm{I}_m \otimes \tau(t_1)^{\mathsf{T}} \\ \mathrm{I}_m \otimes \tau(t_2)^{\mathsf{T}} \\ \vdots \\ \mathrm{I}_m \otimes \tau(t_s)^{\mathsf{T}} \end{bmatrix} \in \mathbb{R}^{ms \times ms}, \tag{9}$$

and

$$A_x := A \otimes I_s - I_n \otimes M_\lambda^{\mathsf{T}}, \quad B_u := B \otimes I_s, \tag{10}$$

where $A \in \mathbb{R}^{n\times n}$ is the system matrix and $B \in \mathbb{R}^{n\times m}$ is the matrix through which the inputs enter the system. The suboptimality of this approximation can be quantified, as discussed in [26].

The cost function in (5) matches the quadratic cost

$$\int_0^\infty \frac{1}{2}\left\{\tilde{x}(t)^\mathsf{T} Q\tilde{x}(t) + \tilde{u}(t)^\mathsf{T} R\tilde{u}(t)\right\} \mathrm{d}t. \tag{11}$$

In addition, as shown in [3], the constraints (6) and (7) imply that the system's dynamics are fulfilled exactly, that is,

$$\dot{\tilde{x}}(t) = A\tilde{x}(t) + B\tilde{u}(t), \quad \tilde{x}(0) = x_0, \quad \forall t \in [0,\infty). \tag{12}$$

The constraint (8) represents a box constraint on the inputs, where the input constraints are relaxed, and are only enforced at the specific time instants t_i, $i = 1,\ldots,s$. The choice of these time instants is discussed in [4]. The above formulation can be generalized to the case of arbitrary linear constraints on the states and the inputs. However, restricting the input (and possibly state) constraints to be box constraints enables an efficient implementation of the optimization routine for solving the optimization problem (5), as highlighted in [4].

If the resulting optimal input trajectory violates the constraint $u_{\text{min}} \leq u(t) \leq u_{\text{max}}$ for all times $t \in [0,\infty)$, for instance in between the time instants t_i, $i = 1,\ldots,s$, the theoretic stability and recursive feasibility guarantees are no longer valid in general. In practice however, this sampling strategy often results in constraint satisfaction for all times, and enables to solve (5) efficiently with the Generalized Fast Dual Gradient (GFDG) method, see [4].

III. TRAJECTORY GENERATION

We propose to generate feasible trajectories for all states and inputs of the given system by solving an optimization problem similar to (5). This process includes a transformation of the desired trajectories from the physical space, which we denote by $\bar{x}_{\text{des}}(t)$, to the parameter space.

In particular, we propose solving an optimization problem for finding feasible trajectories (compatible with the dynamics and fulfilling the constraints) that are the closest possible fit to $\bar{x}_{\text{des}}(t)$ (in the weighted L^2-sense), while keeping the control effort as small as possible.

The Laguerre functions that are used in our parametrized MPC approach have limited polynomial order, and as such they would not be suitable to perform a single fit over the entire trajectory horizon. In order to tackle this problem, we split the entire trajectory into N smaller intervals of length T. In order to not lose the predictive power of the MPC approach, we solve the trajectory generation problem over a prediction window containing a number N_{pred} of these intervals, covering a time horizon of length $N_{\text{pred}}T$. As a result, this approach improves the accuracy of the fit. In practice, a trade-off between the prediction horizon $N_{\text{pred}}T$ and the accuracy of the fit has to be found.

Moreover, the basis functions are decaying to zero and as such, they are unable to capture steady-state deviations. Provided that these steady-state offsets correspond to equilibrium points of the dynamics, they are included in the trajectory generation problem. More precisely, for each interval j we may introduce the offsets $x_{\text{b},j}$, that are chosen such that $\bar{x}_{\text{des}}((j + N_{\text{pred}})T) - x_{\text{b},j} = 0$, and shift the desired trajectory by these offsets, leading to the shifted trajectories $\bar{x}_j(t) := \bar{x}_{\text{des}}(t + jT) - x_{\text{b},j}$. These are defined over the intervals $j = 0,\ldots,N-1$, which are used as a starting point for the trajectory generation problem. The procedure is illustrated in Figure 1.

We generate the feasible trajectories, represented by $\eta_{\text{ref},j} := (\eta_{x,\text{ref},j}, \eta_{u,\text{ref},j})$, by minimizing, for the j-th interval,

$$\int_0^{N_{\text{pred}}T} \frac{1}{2}\left\{\Delta_{x,j}(t)^\mathsf{T}\bar{Q}\Delta_{x,j}(t) + \tilde{u}_j(t)^\mathsf{T}\bar{R}\tilde{u}_j(t)\right\}\mathrm{d}t \tag{13}$$

with respect to $\eta_{\text{ref},j}$, where $\bar{Q} \succeq 0$ and $\bar{R} \succ 0$ are suitable tuning matrices,

$$\Delta_{x,j}(t) := \bar{x}_j(t) - (I_n \otimes \tau(t))^\mathsf{T}\eta_{x,\text{ref},j}, \tag{14}$$

and

$$\tilde{u}_j(t) := (I_m \otimes \tau(t))^\mathsf{T}\eta_{u,\text{ref},j}. \tag{15}$$

Using the properties of the Kronecker product, and eliminating the terms not depending on $\eta_{\text{ref},j}$, (13) can be rearranged as,

$$\frac{1}{2}\left\{\eta_{x,\text{ref},j}^\mathsf{T}(\bar{Q}\otimes J_s)\eta_{x,\text{ref},j} + \eta_{u,\text{ref},j}^\mathsf{T}(\bar{R}\otimes J_s)\eta_{u,\text{ref},j}\right\} - \int_0^{N_{\text{pred}}T} \bar{x}_j(t)^\mathsf{T}\bar{Q}(I_n\otimes\tau(t))^\mathsf{T}\eta_{x,\text{ref},j}\mathrm{d}t, \tag{16}$$

where

$$J_s := \int_0^{N_{\text{pred}}T} \tau(t)\tau(t)^\mathsf{T}\mathrm{d}t. \tag{17}$$

The matrix J_s can be computed efficiently considering that

$$M_\lambda J_s = \int_0^{N_{\text{pred}}T} M_\lambda\tau(t)\tau(t)^\mathsf{T}\mathrm{d}t = \int_0^{N_{\text{pred}}T} \dot{\tau}(t)\tau(t)^\mathsf{T}\mathrm{d}t,$$

and integrating by parts results in

$$\begin{aligned} M_\lambda J_s &= \left[\tau(N_{\text{pred}}T)\tau(N_{\text{pred}}T)^\mathsf{T} - \tau(0)\tau(0)^\mathsf{T}\right] \\ &\quad - \int_0^{N_{\text{pred}}T} \tau(t)\dot{\tau}(t)^\mathsf{T}\ \mathrm{d}t \\ &= \left[\tau(N_{\text{pred}}T)\tau(N_{\text{pred}}T)^\mathsf{T} - \tau(0)\tau(0)^\mathsf{T}\right] - J_s M_\lambda^\mathsf{T}. \end{aligned} \tag{18}$$

This leads to the Lyapunov equation,

$$M_\lambda J_s + J_s M_\lambda^\mathsf{T} - \left[\tau(N_{\text{pred}}T)\tau(N_{\text{pred}}T)^\mathsf{T} - \tau(0)\tau(0)^\mathsf{T}\right] = 0$$

that gives J_s as a unique solution. Existence and uniqueness of the solution J_s of the above Lyapunov equation is guaranteed, since the triangular matrix M_λ is negative definite, see [27, p. 114].

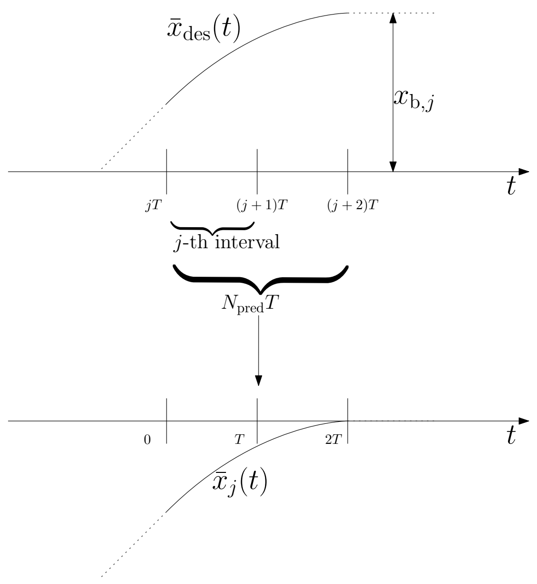

Fig. 1. Offset and prediction window for a single state, with $N_{\text{pred}} = 2$. The curve is shifted by the offset $x_{\text{b},j}$, such that at the end of the prediction window the state will be at zero. The shifted curve is used for solving the trajectory generation problem in (19) and the resulting parameters $\eta_{\text{ref},j}$ are stored for the j-th interval. The procedure is repeated for all the intervals $j = 0, \dots, N-1$.

Therefore, the complete problem at each interval j reduces to

$$\inf_{\substack{\eta_{x,\text{ref},j}\\ \eta_{u,\text{ref},j}}} \frac{1}{2}\left\{\eta_{x,\text{ref},j}^{\mathsf{T}}(\bar{Q}\otimes J_s)\eta_{x,\text{ref},j} + \eta_{u,\text{ref},j}^{\mathsf{T}}(\bar{R}\otimes J_s)\eta_{u,\text{ref},j}\right\}$$

$$-\int_0^{N_{\text{pred}}T} \bar{x}_j(t)^{\mathsf{T}}\bar{Q}(I_n\otimes\tau(t))^{\mathsf{T}}\eta_{x,\text{ref},j}\mathrm{d}t \quad (19)$$

$$\text{s.t.}\quad A_x\eta_{x,\text{ref},j} + B_u\eta_{u,\text{ref},j} = 0, \quad (20)$$

$$(I_n\otimes\tau(0)^{\mathsf{T}})\eta_{x,\text{ref},j} + x_{\text{b},j} = (I_n\otimes\tau(T)^{\mathsf{T}})\eta_{x,\text{ref},j-1} + x_{\text{b},j-1}, \quad (21)$$

$$(I_m\otimes\tau(0)^{\mathsf{T}})\eta_{u,\text{ref},j} = (I_m\otimes\tau(T)^{\mathsf{T}})\eta_{u,\text{ref},j-1}, \quad (22)$$

$$F\eta_{u,\text{ref},j}\in\mathcal{U}. \quad (23)$$

In case parts of the desired state trajectory are not provided, the corresponding value in $\bar{Q}$ should be set to zero (or very close to zero) for the corresponding state. In this way, a higher flexibility will be left to the algorithm in order to find a feasible trajectory.

The constraints (21) and (22) are introduced in order to preserve the continuity of the states and the inputs, such that the trajectory defined by $\eta_{\text{ref,j}}$ starts exactly where the trajectory defined by $\eta_{\text{ref,j-1}}$ ends. In a similar way, both constraints are replaced for the interval $j = 0$ by

$$(I_n\otimes\tau(0)^{\mathsf{T}})\eta_{x,\text{ref},0} = \bar{x}_{\text{des}}(0) - x_{\text{b},0}. \quad (24)$$

The results of the problem on two of the states of the system that we will present in Section VI are shown in Figure 2.

IV. ONLINE TRAJECTORY TRACKING

Once a trajectory has been generated offline, it can be tracked online by the MPC controller. At each time step, a counter is used to detect the current interval and load the appropriate $\eta_{\text{ref},j}$ and $x_{\text{b},j}$. For the time instants within the intervals, the parameters have to be shifted in time by pre-multiplying a delay matrix to the current $\eta_{\text{ref},j}$. Considering a single state x, and given the sampling time T_s and the time instant kT_s, $k = 0, \dots, K-1$, we have, according to (3),

$$\tau(kT_s) = e^{M_\lambda kT_s}\tau(0). \quad (25)$$

Within the j-th interval, this leads to

$$x(kT_s) = \tau(kT_s)^{\mathsf{T}}\eta_{x,\text{ref},j} = \tau(0)^{\mathsf{T}}e^{M_\lambda^{\mathsf{T}}kT_s}\eta_{x,\text{ref},j} = \tau(0)^{\mathsf{T}}\eta_{x,\text{ref,d},j}. \quad (26)$$

The above formula (26) extends naturally to multiple states and inputs, by virtue of the Kronecker product. Therefore, the corresponding shifted parameters are calculated as

$$\eta_{\text{ref,d},j} = (I_{n+m}\otimes e^{kM_\lambda^{\mathsf{T}}T_s})\eta_{\text{ref},j}. \quad (27)$$

We will drop the subscript '$_\text{d}$' from now on to simplify notation. As a result, in the j-th interval, the following problem is solved online:

$$\inf_{\eta_x,\eta_u}\frac{1}{2}\left\{(\eta_x - \eta_{x,\text{ref},j})^{\mathsf{T}}(Q\otimes I_s)(\eta_x - \eta_{x,\text{ref},j}) + (\eta_u - \eta_{u,\text{ref},j})^{\mathsf{T}}(R\otimes I_s)(\eta_u - \eta_{u,\text{ref},j})\right\}$$

$$\text{s.t.}\quad A_x\eta_x + B_u\eta_u = 0, \quad (28)$$

$$(I_n\otimes\tau(0)^{\mathsf{T}})\eta_x = x_0 - x_{\text{b},j},$$

$$F\eta_u\in\mathcal{U},$$

where x_0 is an estimate of the current position. The problem can be solved with the GFDG method as proposed in Section II, using the change of variables $\eta_x' = \eta_x - \eta_{x,\text{ref},j}$, $\eta_u' = \eta_u - \eta_{u,\text{ref},j}$.

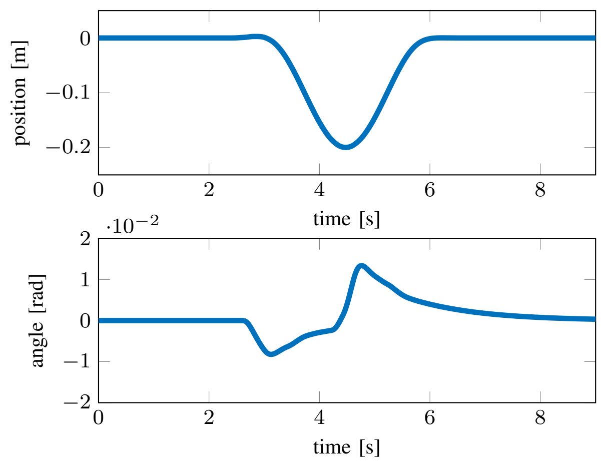

Fig. 2. Trajectory generation results. The upper plot shows a sinusoidal pulse in the x-position for the Flying Platform, an unmanned aerial vehicle to be presented in Section VI. The lower plot shows the corresponding pitch (Euler angle) over the same horizon. This state was not specified among the desired ones, so the algorithm has the freedom of finding an appropriate feasible trajectory.

V. Trajectory Tracking with Iterative Learning

In order to account for the repeatable disturbances, we implement an iterative learning scheme. It consists of a Kalman Filter, see [28, Ch. 8], for estimating these repeatable disturbances, which are then included in the proposed MPC framework. The use of a Kalman Filter is motivated by the fact that it is a well-established technique, easily tunable, and provides a means to incorporate prior information.

We model the disturbances as additive noise that enters the dynamics through the matrix $G \in \mathbb{R}^{n \times n_v}$,

$$\dot{x}(t) = Ax(t) + Bu(t) + Gv(t), \quad \forall t \in [0, \infty), \tag{29}$$

where $v(t) \in \mathbb{R}^{n_v}$ is the disturbance vector. In analogy to the system's state and input, which are modeled using $\tilde{x}$ and $\tilde{u}$, the disturbance $v(t)$ is assumed to have the form

$$\tilde{v}(t) := (I_{n_v} \otimes \tau(t))^\mathsf{T} \eta_v. \tag{30}$$

As a result, the equality constraint capturing the system dynamics in (28) is reformulated as

$$A_x \eta_x + B_u \eta_u + G_v \eta_v = 0, \quad G_v := G \otimes I_s. \tag{31}$$

Note that, as remarked earlier, trajectories $\tilde{x}(t)$, $\tilde{u}(t)$ and $\tilde{v}(t)$ satisfying (31) and the initial condition $\tilde{x}(0) = x_0$, fulfill the equations of motion (29) exactly.

We discretize the augmented system's model in (29) as (assuming a zero-order hold sampling),

$$x[k+1] = A_\text{d} x[k] + B_\text{d} u[k] + G_\text{d} v[k], \tag{32}$$

where $k = 0, 1, \ldots$, A_d, B_d and G_d are the discrete-time matrix representation of (29) with sampling time T_s, and $x[k] = x(kT_s)$. We aim at estimating the disturbance trajectory $v[k]$ based on the data available from the previous trials.

As we are interested in capturing the repeatable parts of the disturbance, we use the following model to describe the evolution of $v[k]$ over different trials,

$$v[k]^{i+1} = v[k]^i + q[k]^i, \quad q[k]^i \sim \mathcal{N}(0, Q_\text{KF}), \tag{33}$$

where $q[k]^i$ denotes the process noise that is assumed to be normally distributed with zero mean and variance Q_KF, and the superscript i refers to the trial number. Thus, the prediction step of the Kalman filter is given by

$$\hat{v}_\text{p}[k]^{i+1} = \hat{v}_\text{m}[k]^i, \quad P_\text{p}[k]^{i+1} = P_\text{m}[k]^i + Q_\text{KF}, \tag{34}$$

where $\hat{v}_\text{p}[k]^i$ and $P_\text{p}[k]^i$ indicate the expected value and the variance of the random variable $v[k]^i$ conditioned on the trajectory data up to the $(i-1)$-th trial, while $v_\text{m}[k]^i$ and $P_\text{m}[k]^i$ indicate the expected value and the variance of the same random variable conditioned on the trajectory data up to the i-th trial.

During the execution of a trajectory, the actual states and inputs are recorded and are used to update the estimate of $v[k]$. We use (32) to formulate the measurement equation,

$$\underbrace{x[k+1]^i - A_\text{d} x[k]^i - B_\text{d} u[k]^i}_{:=z[k]^i} = \quad G_\text{d} v[k]^i + n[k]^i,$$

with $n[k]^i \sim \mathcal{N}(0, R_\text{KF})$. The measurement update equations can be expressed as

$$P_\text{m}[k]^i = \left((P_\text{p}[k]^i)^{-1} + G_\text{d}^\mathsf{T} R_\text{KF}^{-1} G_\text{d}\right)^{-1} \tag{35}$$

$$\hat{v}_\text{m}[k]^i = \hat{v}_\text{p}[k]^i + P_\text{m}[k]^i G_\text{d}^\mathsf{T} R_\text{KF}^{-1}(z[k]^i - G_\text{d}\hat{v}_\text{p}[k]^i). \tag{36}$$

Once the Kalman filter updates are performed, the current estimated disturbance trajectory $v_\text{m}[k]^i$ is transformed to the continuous-time domain through a zero-order hold, yielding $v(t)^i$. The latter is transformed to the parameter space by performing a curve fitting that minimizes a regularized L^2-distance. Again, the Laguerre functions are not suitable for fitting a curve over a large interval, as they only have few degrees of freedom. To tackle this problem, we use a similar approach as in the trajectory generation, and split the trajectory $v(t)^i$ in N_v pieces of length T_v. In order to simplify notation, we present this fitting procedure for one-dimensional disturbances, but it can be easily extended to the multi-dimensional case. Introducing the quantities $v_j(t)^i := v(t + jT_v)^i$, the resulting fitting procedure (for each interval j) reduces to,

$$\eta_{v,j}^i := \arg\min_\eta \int_0^{T_v} \frac{1}{2}\Big\{ \left(v_j(t)^i - \tau(t)^\mathsf{T}\eta\right)^2 + r\eta^\mathsf{T}\dot{\tau}(t)\dot{\tau}(t)^\mathsf{T}\eta \Big\} \mathrm{d}t, \tag{37}$$

where the first integrand penalizes the squared distance to the disturbance trajectory, while the second term performs a regularization in order to increase the smoothness of the resulting trajectory $\tau(t)^\mathsf{T}\eta_{v,j}^i$. The regularization is controlled with the tuning parameter $r \in \mathbb{R}$, $r \geq 0$.

The optimization problem (37) can be solved analytically leading to

$$\eta_{v,j}^i = \left(J_v + rM_\lambda J_v M_\lambda^\mathsf{T}\right)^{-1} \int_0^{T_v} v_j(t)^i \tau(t) \mathrm{d}t, \tag{38}$$

where

$$J_v := \int_0^{T_v} \tau(t)\tau(t)^\mathsf{T} \mathrm{d}t \tag{39}$$

is computed in a similar way as in (17)-(18).

The algorithm that runs at each trajectory execution is summarized by the pseudo-code given in Algorithm 1.

Algorithm 1 Pseudo-code for disturbances estimation.

```
Result: Estimate $\eta_v^i$ corresponding to the $i$-th trial
Record $x[k]^i$, $u[k]^i$ over the entire trajectory;
Update the Kalman filter state $v[k]^i$;
Split the interpolated $v(t)^i$ over $N_v$ intervals $\to v_j(t)^i$;
for each interval $j$ do
    Fit $\eta_{v,j}^i$ through $v_j(t)^i$;
end
Merge all the estimates and store a lifted vector
    $\eta_v^i = \left(\eta_{v,0}^i, \eta_{v,1}^i \cdots\right)$
```

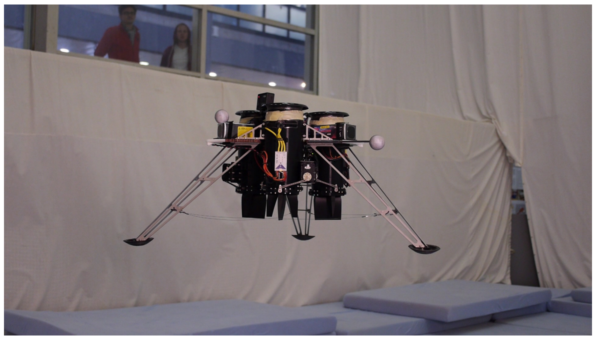

Fig. 3. The Flying Platform during flight.

After the i-th trial, the current estimate of the disturbance parameters is used to replan a feasible trajectory. This is done by solving again the generation problem in (19), replacing (20) with

$$A_x \eta_{x,\text{ref},j} + B_u \eta_{u,\text{ref},j} + G_v \eta^i_{v,j} = 0, \tag{40}$$

to include the estimated disturbances in the system's dynamics. In a similar way, in the online trajectory tracking problem the equality constraint in (28) is replaced by

$$A_x \eta_x + B_u \eta_u + G_v \eta^i_{v,j} = 0. \tag{41}$$

The vector $\eta^i_{v,j}$ is shifted in time in a similar way as described by (27).

VI. EXPERIMENTAL RESULTS

A. Hardware and software

The Flying Platform is a flying machine consisting of three electrical ducted fans mounted on a lightweight frame, see [29]. Two flaps are attached to each fan in order to control the air flow and perform thrust vectoring. A PX4FMU[1] is used as a flight controller for actuation and measuring the angular velocities with the built-in gyroscope. The PX4 communicates through a serial bus with a Gumstix DuoVero Zephyr computer-on-module (COM)[2], equipped with a dual-core ARM Cortex-A9 that runs at 1 GHz. The COM disposes of 1GB RAM. Position and attitude information is provided by a Vicon[3] motion capture system. Translational velocities are estimated through off-board state estimation techniques. The Flying Platform receives the data computed off-board, that is, the actual position, attitude and (linear) velocity, through a wireless communication.

The parametrized MPC routine and the iterative learning are implemented on the Gumstix COM, that runs a Linux-based operating system.

B. Model

A first-principles model is linearized about hover. In order to be invariant to different yaw set-points, the linearization is carried out in a yaw-fixed body coordinate system. The Flying Platform model has 12 states, that is, position, translational velocities, attitude, and angular velocities, and 9 inputs (3 per fan), which are given by $T_{z,i}$ (vertical thrusts) and $T_{x,i}$, $T_{y,i}$ (horizontal components, in perpendicular directions to each flap), with $i = 1, 2, 3$. All the input saturations can be approximated as box constraints, as shown in [4].

[1]`www.pixhawk.org`
[2]`www.gumstix.org`
[3]`www.vicon.com`

TABLE I
TUNING PARAMETERS.

parameter	value	description
T	0.05 s	interval length in trajectory generation
N_{pred}	25	prediction horizon for tracking (multiples of T)
r	0.1	regularization in learning fit
T_v	1 s	interval length in learning fit

C. Results

We choose $s = 5$, and an exponential decay $\lambda = 5\text{s}^{-1}$. The following matrices are chosen for the learning, with $n_v = 12$,

$$G = I_{n_v}, \quad Q_{\text{KF}} = 10^{-1} \cdot I_{n_v}, \quad R_{\text{KF}} = 10^{-5} \cdot I_n.$$

Our choice of G doesn't specify any weights on how the disturbances enter the system. The choice of Q_{KF} leaves some freedom to our assumption of invariance of the disturbances over different trials, while the low magnitude of the values in R_{KF} emphasizes the considerable trust we give to the measurements.

The following tuning matrices are chosen for the controller presented in (28), which runs at a sampling frequency of 100 Hz,

$$\begin{aligned} Q &= \text{diag}\left(\overbrace{200, 200, 30}^{\text{position}}, \overbrace{10, 10, 10}^{\text{lin. velocity}}, \overbrace{40, 40, 10}^{\text{attitude}}, \overbrace{10, 10, 5}^{\text{ang. velocity}}\right) \\ R &= 10^{-3} \cdot \text{diag}\left(\underbrace{1.7, 0.85, 15}_{T_{x,1}, T_{y,1}, T_{z,1}}, \underbrace{1.7, 0.85, 15}_{T_{x,2}, T_{y,2}, T_{z,2}}, \underbrace{1.7, 0.85, 15}_{T_{x,3}, T_{y,3}, T_{z,3}}\right). \end{aligned}$$

The remaining tuning parameters are summarized in Table I. The desired task is to track a circle with a diameter of 20 cm in 6 s. Six consecutive trials have been performed. The resulting tracking performance is shown in Figure 4.[4]

Due to the estimation of the disturbances η_v, the predictions of the MPC are more accurate as the number of trials increases, which improves the tracking performance. The repeatable disturbances are most likely due to asymmetries in the mass distribution of the real system.

VII. CONCLUSION

A parametrized MPC approach has been extended to the trajectory tracking case, and a learning scheme has been introduced. To this extent, the inherent limitations of the Laguerre functions that are used to parametrize the state and input trajectories are addressed by introducing multiple prediction intervals. Still, accounting for large constant disturbances remains challenging due to the decaying nature

[4]A video showing some of the experiments can be found at the following link: `https://youtu.be/GgIwrnoNvTY`.

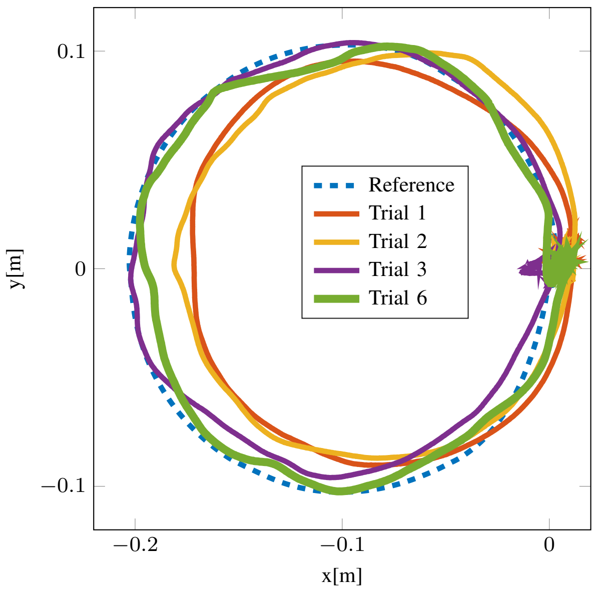

Fig. 4. Trajectory tracking of a circle (counter-clockwise direction, starting from the origin). The feasible (parametrized) trajectory matches the desired trajectory in the xy-plane, and is therefore not shown. The system improves the accuracy of the predictions used for MPC, increasing the tracking performance over subsequent trials, before reaching a steady-state after six trials.

of the basis functions and requires careful tuning. The same applies to disturbances that quickly change in time, due to the bounded derivative of the basis functions. The method has been tested on an unmanned aerial vehicle, showing satisfactory tracking performance, and achieving a sampling frequency of 100 Hz. The experiments have also shown how the system improves in performing the task over subsequent trials.

ACKNOWLEDGMENT

The authors would like to thank Marc-Andrè Corzillius, Michael Egli, Tobias Meier 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] M. Morari and J. H. Lee, “Model predictive control: past, present and future,” *Computers & Chemical Engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.

[2] M. Alamir and G. Bornard, “Stability of a truncated infinite constrained receding horizon scheme: the general discrete nonlinear case,” *Automatica*, vol. 31, no. 9, pp. 1353–1356, 1995.

[3] M. Muehlebach and R. D’Andrea, “Parametrized infinite-horizon model predictive control for linear time-invariant systems with input and state constraints,” *American Control Conference*, pp. 2669–2674, 2016.

[4] M. Hofer, M. Muehlebach, and R. D’Andrea, “Application of an approximate model predictive control scheme on an unmanned aerial vehicle,” *International Conference on Robotics and Automation*, pp. 2952–2957, 2016.

[5] Y. Wang and S. Boyd, “Fast model predictive control using online optimization,” *Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2009.

[6] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, “Efficient interior point methods for multistage problems arising in receding horizon control,” *Conference on Decision and Control*, pp. 668–674, 2012.

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

[8] ——, *Model Predictive Control System Design and Implementation Using MATLAB*. Springer, 2009.

[9] W. B. Dunbar, M. B. Milam, R. Franz, and R. M. Murray, “Model predictive control of a thrust-vectored flight control experiment,” *IFAC World Congress*, vol. 35, no. 1, pp. 355–360, 2002.

[10] F. Kühne, J. M. G. da Silva Jr., and W. F. Lages, “Mobile robot trajectory tracking using model predictive control,” *Latin American Robotics Symposium*, 2005.

[11] T. Faulwasser and R. Findeisen, “A model predictive control approach to trajectory tracking problems via time-varying level sets of Lyapunov functions,” *Conference on Decision and Control and European Control Conference*, pp. 3381–3386, 2011.

[12] A. Alessandretti, A. P. Aguiar, and C. N. Jones, “Trajectory-tracking and path-following controllers for constrained underactuated vehicles using model predictive control,” *European Control Conference*, pp. 1371–1376, 2013.

[13] S. Gros, R. Quirynen, and M. Diehl, “Aircraft control based on fast non-linear MPC & multiple-shooting,” *Conference on Decision and Control*, pp. 1142–1147, 2012.

[14] K. Alexis, G. Nikolakopoulos, and A. Tzes, “On trajectory tracking model predictive control of an unmanned quadrotor helicopter subject to aerodynamic disturbances,” *Asian Journal of Control*, vol. 16, no. 1, pp. 209–224, 2014.

[15] B. Vanek, T. Péni, J. Bokor, and G. Balas, “Practical approach to real-time trajectory tracking of UAV formations,” *American Control Conference*, pp. 122–127, 2005.

[16] Z. Bien and J. X. Xu, *Iterative Learning Control: Analysis, Design, Integration and Applications*. Kluwer Academic Publishers, 1998.

[17] Y. Wang, F. Gao, and F. J. Doyle III, “Survey on iterative learning control, repetitive control, and run-to-run control,” *Journal of Process Control*, vol. 19, no. 10, pp. 1589–1600, 2009.

[18] A. Schoellig and R. D’Andrea, “Optimization-based iterative learning control for trajectory tracking,” *European Control Conference*, pp. 1505–1510, 2009.

[19] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, “Provably safe and robust learning-based model predictive control,” *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

[20] N. Amann, D. H. Owens, and E. Rogers, “Predictive optimal iterative learning control,” *International Journal of Control*, vol. 69, no. 2, pp. 203–226, 1998.

[21] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, “Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search,” *International Conference on Robotics and Automation*, pp. 528–535, 2016.

[22] E. J. Adam and A. H. Gonzalez, “Iterative learning - MPC: An alternative strategy,” in *Frontiers in Advanced Control Systems*, G. L. de Oliveira Serra, Ed. InTech, 2012, ch. 9.

[23] K. K. Tan, S. N. Huang, T. H. Lee, and A. Tay, “Disturbance compensation incorporated in predictive control system using a repetitive learning approach,” *Systems & Control Letters*, vol. 56, no. 1, pp. 75–82, 2007.

[24] J. R. Cueli and C. Bordons, “Iterative nonlinear model predictive control. Stability, robustness and applications,” *Control Engineering Practice*, vol. 16, no. 9, pp. 1023–1034, 2008.

[25] P. Bouffard, A. Aswani, and C. Tomlin, “Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results,” *International Conference on Robotics and Automation*, pp. 279–284, 2012.

[26] M. Muehlebach and R. D’Andrea, “Approximation of continuous-time infinite-horizon optimal control problems arising in model predictive control,” *Conference on Decision and Control*, pp. 1464–1470, 2016.

[27] K. J. Åström and R. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[28] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley, 2006.

[29] 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.