Learning-based parametrized model predictive control for trajectory tracking

Carmelo Sferrazza | Michael Muehlebach | Raffaello D’Andrea

Institute for Dynamic Systems and Control, ETH Zurich, Zurich, Switzerland

Correspondence
Carmelo Sferrazza, Institute for Dynamic Systems and Control, ETH Zurich, Sonneggstrasse 3, 8092 Zurich, Switzerland.
Email: csferrazza@ethz.ch

Summary
This article is concerned with the tracking of nonequilibrium motions with model predictive control (MPC). It proposes to parametrize input and state trajectories of a dynamic system with basis functions to alleviate the computational burden in MPC. As a result of the parametrization, an optimization problem with fewer variables is obtained, and the memory requirements for storing the reference trajectories are reduced. The article also discusses the generation of feasible reference trajectories that account for the system’s dynamics, as well as input and state constraints. In order to cope with repeatable disturbances, which may stem from unmodeled dynamics for example, an iterative learning procedure is included. The approach relies on a Kalman filter that identifies the repeatable disturbances based on previous trials. These are then included in the system’s model available to the model predictive controller, which compensates them in subsequent trials. The proposed approach is evaluated on a quadcopter, whose task is to balance a pole, while flying a predefined trajectory.

KEYWORDS
iterative learning, model predictive control, trajectory generation, trajectory tracking

1 | INTRODUCTION

Model predictive control (MPC) is an effective and popular control strategy for systems that have input and state constraints. There are numerous examples where MPC has been successfully applied in practice; see, for example, the works of Richalet et al.,1 Borrelli et al.,2 Geyer et al.,3 and Papaioiou et al.4 The underlying idea is to repeatedly solve an optimal control problem that takes input and state constraints into account, and generates optimal input and state trajectories that guide the system from its current state to a desired state (often the origin). In order to be robust against modeling errors and disturbances, only a small portion of the input trajectory is applied to the system, before resolving the optimal control problem, subject to the actual position as an initial condition.5

As a consequence, the underlying optimal control problem often needs to be solved very quickly in order to achieve a sufficiently small sampling time. Even if the optimization is warm-started, the resulting computational load is often substantial and represents a bottleneck of MPC. To reduce the computational load, the optimal control problem is often simplified, for example by using a coarser system model with fewer dimensions, and/or by reducing the prediction horizon. However, both the simplified system model and the smaller prediction horizon may degrade the resulting closed-loop performance and may even lead to instability.
The parametrization of the input and state trajectories with basis functions is an approach that avoids the discretization and the truncation of the time horizon. For the regulator problem, it has been shown in the work of Muehlebach and D'Andrea\(^6\) that the resulting MPC controller asymptotically stabilizes the origin and is recursively feasible (without the need for a terminal constraint and a terminal cost). This article discusses the extension of such a parametrized MPC strategy from the regulator problem to the tracking problem. In fact, this step is non-trivial, particularly with regard to practical implementation. In addition, a strategy to identify and account for repeatable disturbances is presented, in case the same trajectory is executed multiple times. The implementation of the resulting learning-based MPC control strategy on a flying inverted pendulum system (see Figure 1) is discussed.

Related work: In the work of Dunbar et al.\(^7\) a testbed for a ducted-fan actuated flying vehicle is controlled with MPC. The authors discuss the regulator problem and it is shown that the region of attraction of the MPC controller is larger than that of the related linear quadratic regulator.

Various works focus on reducing the computational complexity of MPC, see for example move-blocking strategies\(^8\) or tailored MPC solvers.\(^9,10\) The works of Wang,\(^11\) Rossiter and Wang,\(^12\) and Khan and Rossiter\(^13\) propose parametrizing the input trajectories with Laguerre or Kauz basis functions. The potential benefits of the parametrization in terms of performance and extension of the feasibility regions are discussed.

The article by Faulwasser and Findeisen\(^14\) proposes an extension of MPC from the regulation problem to trajectory tracking. A nonlinear and continuous-time MPC formulation is presented, including time-varying terminal set constraints that guarantee recursive feasibility and closed-loop stability. Stephens et al.\(^15\) discuss the design and implementation of an MPC controller to track trajectories of an industrial machine tool servo drive. The authors suggest extending the reference trajectory, for example via first-order hold, and are able to reduce the input horizon (they distinguish between the input and the prediction horizons), such that they can apply explicit MPC. In the work of Neunert et al.,\(^16\) a general nonlinear control strategy is presented, which uses time-varying feedforward and feedback terms. The resulting cascaded control structure includes a low-level controller that can be executed at very high rates, while MPC is used at a higher-level for updating the parameters of the low-level controller. The research of Kamel et al.\(^17\) compares a linear and a non-linear MPC controller for trajectory tracking with a hexacopter. The authors conclude that the nonlinear MPC controller outperforms the linear one for the experiments conducted. In the work of Mueller and D’Andrea,\(^18\) a diminishing horizon MPC controller is introduced to track interception trajectories with a quadcopter. The trajectories generated at each sampling step are obtained by optimizing a quadratic program that is tailored to the quadcopter’s dynamics and is based on conservative bounds on the quadcopter’s jerk. Compared to the approach proposed in this article, this results in a comparable computational effort. However, the algorithm does not provide convergence guarantees and does not adapt to the environment.

In order to cope with modeling errors a learning-based predictive controller is presented in the work by Bouffard et al.\(^19\) The dynamics are assumed to be affine and time-invariant and an extended Kalman filter is used to learn, respectively update, the dynamics. As a result, the authors demonstrate offset-free tracking and discuss the improvements of a step response compared to the nominal MPC controller. In the work of Wang et al.,\(^20\) iterative learning control is included.
FIGURE 2  At each trial, the generated trajectory is followed by means of a trajectory tracking controller. The state and input rollouts are used to estimate the repeatable disturbances after the trial completion, and are then taken into account in the next generation of the trajectories to improve the tracking performance. Note that in this diagram $x_{\text{des}}(t)$ and $v(t)$ denote the state and disturbance trajectories in continuous time for the entire trajectory duration. Similarly $x[k]$ and $u[k]$ denote the state and input rollouts in discrete time for the entire trajectory duration.

in an MPC framework to control multi-phase batch processes. Two different strategies are proposed to design an updating law using MPC, which can intrinsically deal with constraints. A reference-free approach is presented in the work of Rosolia and Borrelli, where a nonlinear MPC controller improves the performance over repetitive tasks when the reference trajectory is not known. The design of terminal constraints at each iteration is proposed to guarantee performance improvement and stability requirements.

However, the approaches cited mostly rely on a discretized formulation of the dynamics and/or a piece-wise constant control input. The approach followed in this article is different: It aims to reduce the number of degrees of freedom and alleviate the computational burden of MPC by parametrizing input and state trajectories with basis functions. In addition to requiring less memory (see Section 3) and reducing computation, inherent (without the addition of terminal constraints) closed-loop stability and recursive feasibility results can be demonstrated when regulating linear time-invariant systems. This parametrization is originally presented in the work of Muehlebach and D’Andrea, where simulation results show the regulation of a linear time-invariant system to equilibrium, without addressing the problem of constraint satisfaction. In the work of Muehlebach et al, a strategy to guarantee constraint satisfaction is implemented and the resulting algorithm is shown to run at 100 Hz on an embedded platform, regulating the system to equilibrium. In the work of Sferrazza et al, a heuristic strategy has been proposed to apply this idea to the trajectory tracking problem. In this article, non-equilibrium motions are successfully tracked with a general framework that also accounts for the generation of feasible trajectories. This approach is demonstrated in experiments, where a quadrotor balances an inverted pendulum. In order to cope with the system’s nonlinearities and unmodeled dynamics, an iterative learning approach that identifies the repeatable disturbances and augments the model accordingly is applied. The computation related to the learning can be done offline, thus not altering the complexity of the MPC algorithm during the tracking of a trajectory. The trajectory generation problem is formulated here as a single optimization over the entire trajectory duration. This differs from the work of Sferrazza et al, where a greedy approach was used to simplify the optimization. The main differences between the two approaches are emphasized throughout this article.

Outline: Section 2 introduces the different building blocks of the proposed control strategy and shows how they are connected: In Section 2.1, the MPC controller that tracks the generated reference trajectories is presented. Section 2.2 describes how the reference trajectories are generated, and the identification of repeatable disturbances is explained in Section 2.3. The benefits in terms of memory usage are elaborated in Section 3. Section 4 presents experimental results, where a quadrotor balances an inverted pendulum while flying a three-dimensional figure-eight trajectory. The article concludes in Section 5 with remarks.

2 | METHOD

The following section proposes a parametrized MPC approach to generate and track non-equilibrium motions of a dynamic system. It is assumed that the system can be roughly approximated by linear time-invariant dynamics. A scheme
of the proposed framework is shown in Figure 2 and consists of the following building blocks: The trajectory generation procedure takes a desired trajectory, \( x_{\text{des}}(t) \), and, based on an estimate of the repeatable disturbances \( v(t) \), computes feasible reference trajectories for the state and input. These trajectories are parametrized by linear combinations of basis functions and include steady-state offsets (as indicated by the variables \( \eta_{x,\text{ref}}, \eta_{u,\text{ref}}, \eta_v, x_{ss}, u_{ss}, \) and \( v_{ss} \)). The trajectory tracking procedure implements an (online) MPC controller for tracking the reference trajectories. The actual state and input trajectories are recorded, and after execution of the trajectory, the disturbance estimation procedure updates the estimate \( v(t) \) of the repeatable disturbances. In the first iteration, the disturbance estimate \( v(t) \) is typically initialized with zero, unless some prior knowledge is available.

The following subsections describe each individual block in detail. In the remainder of this article, vectors are expressed as tuples, with dimension and stacking clear from the context.

2.1 | Trajectory tracking

In the following subsection, the trajectory tracking procedure is described. As indicated in Figure 2, its aim is to track feasible reference trajectories, given a disturbance estimate. All trajectories are parametrized with basis functions, that is,

\[
\begin{align*}
\ddot{x}(t) &= (I_n \otimes \tau(t)) \eta_x + x_{ss} \\
\ddot{x}_{\text{ref}}(t) &= (I_n \otimes \tau(t)) \eta_{x,\text{ref}} + x_{ss} \\
\ddot{u}(t) &= (I_m \otimes \tau(t))^\top \eta_u + u_{ss} \\
\ddot{u}_{\text{ref}}(t) &= (I_m \otimes \tau(t))^\top \eta_{u,\text{ref}} + u_{ss} \\
\ddot{v}(t) &= (I_{n_v} \otimes \tau(t))^\top \eta_v + v_{ss},
\end{align*}
\]

for all \( t \in [0, \infty) \), where \( \ddot{x} \) and \( \ddot{u} \) denote the planned state and input trajectories, and \( \ddot{x}_{\text{ref}}, \ddot{u}_{\text{ref}}, \) and \( \ddot{v} \) are the reference and disturbance trajectories that are obtained from the trajectory generation routine and are fixed during trajectory tracking. The parameter vectors of the planned state and input trajectories, which are subject to the (online) optimization, are denoted by \( \eta_x \in \mathbb{R}^{m_x} \) and \( \eta_u \in \mathbb{R}^{m_u} \), the parameters of the reference state, the reference input, and the disturbance trajectory are denoted by \( \eta_{x,\text{ref}} \in \mathbb{R}^{m_x}, \eta_{u,\text{ref}} \in \mathbb{R}^{m_u} \), and \( \eta_v \in \mathbb{R}^{n_v} \), and the steady-state offsets on the state, the input, and the disturbances are denoted by \( x_{ss} \in \mathbb{R}^n, u_{ss} \in \mathbb{R}^m, \) and \( v_{ss} \in \mathbb{R}^{n_v} \). The integers \( n, m, \) and \( n_v \) describe the dimension of the state, the input, and the disturbances, and \( \otimes \) refers to the Kronecker product. In contrast to the work of Sferrazza et al.,\(^{23}\) steady-state offsets are also included in the inputs and the disturbances. In addition, these offsets are decision variables in the optimization problem introduced in Section 2.2. This makes it possible to capture large steady-state disturbances and gives the trajectory generation algorithm the flexibility to trade off the placement of these offsets during the optimization. The basis functions \( \tau(t) := (\tau_1(t), \tau_2(t), \ldots, \tau_s(t)) \in \mathbb{R}^s \), where \( s \) refers to the number of basis functions, are assumed to satisfy the following standing assumptions:

H1 The basis functions \( \tau_1, \tau_2, \ldots, \tau_s \) are linearly independent.

H2 The basis functions satisfy the following first order differential equation \( \dot{\tau}(t) = M \tau(t) \) for all \( t \in [0, \infty) \), where the matrix \( M \in \mathbb{R}^{s \times s} \) is Hurwitz.

Assumption H1 is required to guarantee that the resulting optimization problems have unique solutions. The motivation of Assumption H2 is twofold: First, if both H1 and H2 are fulfilled, linear time-invariant dynamics \( \ddot{x}(t) = A \ddot{x}(t) + B \ddot{u}(t) + G \ddot{v}(t) \) are fulfilled if and only if the corresponding parameter vectors satisfy

\[
(I_n \otimes M^\top) \eta_x = (A \otimes I_s) \eta_x + (B \otimes I_s) \eta_u + (G \otimes I_s) \eta_v,
\]

where \( A, B, \) and \( G \) are the system’s matrices. A proof of this statement can be found in the work of Muehlebach and D’Andrea.\(^6\) Second, if Assumption H2 is fulfilled, it implies that the basis functions are able to capture arbitrary time
shifts. More precisely, given the trajectory \( \tilde{x}(t) \) parametrized by the basis functions, the shifted version of this trajectory, \( \tilde{x}(t + \Delta t) \) can be parametrized with the same basis functions,

\[
\tilde{x}(t + \Delta t) = (I_n \otimes \tau(t))^{T} (I_n \otimes \exp(M^T \Delta t)) \eta_u,
\]

for all \( t \in [0, \infty) \), where \( \Delta t \in [0, \infty) \) is any time shift. This is particularly important for fast online execution, where the optimization at the next time step can be warm-started with the solution obtained at the current time. The time-shift property also implies closed-loop stability and recursive feasibility of the resulting MPC algorithm in the regulation case (that is for \( \eta_{x, \text{ref}} = 0, \eta_{u, \text{ref}} = 0 \)).

Examples for basis functions that fulfill Assumptions H1 and H2 are exponentially decaying polynomials or exponentially decaying harmonics. In the remainder, exponentially decaying polynomials of the form

\[
\tau(t) \in \exp(-\lambda t) \text{span}(1, t, t^2, \ldots, t^{p-1}),
\]

will be used, which are obtained by choosing \( M \) as

\[
M = \begin{pmatrix}
-\lambda & 0 & \cdots & 0 \\
-2\lambda & -\lambda & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
-2\lambda & -2\lambda & \cdots & -\lambda
\end{pmatrix}.
\]

The steady-state offsets \( x_{ss}, u_{ss}, v_{ss} \), obtained from the trajectory generation procedure describe an equilibrium, that is,

\[
0 = Ax_{ss} + Bu_{ss} + Gv_{ss}.
\]

The aim of the trajectory tracking procedure is to repeatedly plan state and input trajectories that start from the current state of the system and follow the reference trajectories \( \tilde{x}_{\text{ref}}, \tilde{u}_{\text{ref}} \) as closely as possible. The reference trajectories \( \tilde{x}_{\text{ref}} \) and \( \tilde{u}_{\text{ref}} \) will ultimately converge to \( x_{ss} \) and \( u_{ss} \) (since \( \lim_{t \to \infty} \tau(t) = 0 \)), hence the name steady-state offsets. This leads to the following optimization problem

\[
\min_{\eta_u, \eta_v} \frac{1}{2} \int_0^\infty \{ (\dot{x}(t) - \tilde{x}_{\text{ref}}(t))^T Q (\dot{x}(t) - \tilde{x}_{\text{ref}}(t)) + (\dot{u}(t) - \tilde{u}_{\text{ref}}(t))^T R (\dot{u}(t) - \tilde{u}_{\text{ref}}(t)) \} \, dt
\]

\[
\text{s.t. } \dot{x}(t) = Ax(t) + Bu(t) + Gv(t), \dot{x}(0) = x_0
\]

\[
\min_{\eta_u, \eta_v} b_{\min} \leq C_x \dot{x}(t) + C_u \dot{u}(t) \leq b_{\max}, \forall t \in [0, \infty),
\]

where \( Q \geq 0 \) and \( R > 0 \) are matrices of appropriate dimension. The problem is repeatedly solved subject to the current state \( x_0 \) as an initial condition (as common in MPC). Note that for simplicity, only a linear constraint on the state and input trajectories is included in the optimization. Slightly more general linear constraints including derivatives or integrals of the state and input trajectories can be dealt with in an analogous way.

Feedback control is achieved by repeatedly solving (8) and applying the first portion of the input \( \tilde{u}(t), t \in [0, T_s] \) to the system, where \( T_s \) denotes the sampling time. The reference trajectories \( \tilde{x}_{\text{ref}} \) and \( \tilde{u}_{\text{ref}} \), as well as the disturbances \( \tilde{v} \), are adapted according to the time that has elapsed. Thus, at time \( t = T_s \), the reference and disturbance trajectories in (8) are modified to \( \tilde{x}_{\text{ref}}(t + T_s), \tilde{u}_{\text{ref}}(t + T_s) \), and \( \tilde{v}(t + T_s) \). According to (6), this amounts to a simple multiplication of the parameter vectors \( \eta_{x,\text{ref}}, \eta_{u,\text{ref}}, \) and \( \eta_{v} \) with the matrix

\[
I \otimes \exp(M^T T_s),
\]

where the dimension of the identity matrix \( I \) is chosen accordingly.
For orthonormal basis functions,\(^{1}\) the above optimization problem reduces to\(^{6}\)
\[
\min_{\eta, \eta_u} \frac{1}{2}(\eta_k - \eta_{k,\text{ref}})^\top (Q \otimes I)(\eta_k - \eta_{k,\text{ref}}) + \frac{1}{2}(\eta_u - \eta_{u,\text{ref}})^\top (R \otimes I)(\eta_u - \eta_{u,\text{ref}})
\]
\[
\text{s.t.} \ (I_n \otimes M^\top)\eta_k = (A \otimes I_n)\eta_k + (B \otimes I_n)\eta_u + (G \otimes I_n)\eta_v,
\]
\[
(I_n \otimes \tau(0))^\top \eta_k + x_{ss} = x_0
\]
\[
b_{\min} \leq F_{\xi} \eta_k + F_u \eta_u + D_x x_{ss} + D_u u_{ss} \leq b_{\max}.
\]

The linear equality constraint (12) imposes the dynamics and (13) the initial condition. The linear inequality constraint (14) describes the input and state constraints. These are obtained by sampling the constraint (9). The values of the matrices \(F_n, F_u, D_k,\) and \(D_u\) may change slightly for different applications, and are given in Appendix C for the system considered in Section 4.

The optimization therefore simplifies to solving the quadratic program (11), subject to the optimization variables \(\eta_k \in \mathbb{R}^{n_x}\) and \(\eta_u \in \mathbb{R}^{n_u}\). Due to the parametrization with the basis functions, the quadratic program (11) typically has few optimization variables and can be solved efficiently. As highlighted before, by exploiting the time-shift property of the basis functions (see (6)), the problem can be warm-started with the solution from the previous time step.

### 2.2 Trajectory generation

The goal of the trajectory generation is to generate feasible input, state, and disturbance trajectories, such that the state trajectory follows the desired trajectory (given to the algorithm) as closely as possible. We assume that the desired trajectory might not necessarily prescribe the motion for all states. For example, the desired trajectory could only contain a position reference that should be followed, while finding the appropriate reference trajectories for the velocities is left to the algorithm. The feasible trajectories satisfy the system’s dynamics, as well as the input and state constraints.

The feasible trajectory is generated by solving an optimization problem, where the desired trajectory is given in the time domain. The resulting trajectories are parametrized by \(\eta_{k,\text{ref}}, \eta_{u,\text{ref}},\) and \(\eta_v,\) and the steady-state offsets \(x_{ss}, u_{ss},\) and \(v_{ss},\) as indicated in Figure 2.

Due to the finite number of basis functions, only trajectories with a finite length can be approximated well. In order to overcome this limitation, the trajectory’s duration is split into \(N\) smaller intervals of length \(T\), and different linear combinations of basis functions are used to approximate the trajectory in each of these intervals, as shown in Figure 3. Therefore, the trajectory tracking problem (11) is modified such that, depending on the time that has elapsed, the reference trajectory in the corresponding interval is tracked, that is, \(\eta_{k,\text{ref}}, \eta_{u,\text{ref}}, \eta_v, x_{ss},\) and \(u_{ss}\) are replaced by the corresponding \(\eta_{k,\text{ref},j}, \eta_{u,\text{ref},j}, \eta_v, j, x_{ss,j},\) and \(u_{ss,j},\) for \(j = 0, 1, \ldots, N - 1.\) A practical advantage of this approach is the possibility of aborting the trajectory at any time during execution, leading the system to a safe equilibrium position (represented by the current interval’s steady-state offsets). In that case, provided that the constraints are satisfied for all times (a sampling strategy is applied in this article), closed-loop stability and recursive feasibility are guaranteed.\(^{6,22}\) The approach is particularly suited to smooth desired trajectories, where the approximations at two consecutive intervals generally overlap for a certain time (see Figure 3). However, this might not be the case for trajectories with sharp changes at the time of switching between two intervals.

In order to find a feasible approximation that is closest to the desired trajectory in the \(L^2\)-sense, while penalizing the input and fitting the estimated disturbances, the set of reference parameters and steady-state offsets is chosen to minimize
\[
\sum_{j=0}^{N-1}\left\{ \int_0^T \frac{1}{2} \Delta_{x,j}(t)^\top Q \Delta_{x,j}(t) + \Delta_{u,j}(t)^\top R \Delta_{u,j}(t) + \Delta_{v,j}(t)^\top \Delta_{v,j}(t) \right\} dt.
\]

\(^{1}\) Without loss of generality, linear independent basis functions can always be made orthonormal. Choosing \(M\) according to (7) yields orthonormal basis functions.
Influence of multiple intervals in the trajectory generation procedure. Because of the limited representation power of the basis functions, it is not possible to fit arbitrary long trajectories, see (A). To overcome this problem, it is proposed to split the trajectory over multiple intervals, with different parametrizations for each of these, see (B). Merging the different intervals’ trajectories results in a more accurate fit, as shown in (C) [Colour figure can be viewed at wileyonlinelibrary.com]
where \( r \) controls the smoothness of the input trajectories, \( \bar{Q} \geq 0, \bar{R} > 0 \) and \( \bar{V} \geq 0 \) are suitable symmetric tuning matrices, and,

\[
\Delta x_j(t) := x_j(t) - [(I_n \otimes r(t))\eta_{x,ref_j} + x_{ss,j}],
\]

(16)

\[
\tilde{u}_{ref_j}(t) := (I_m \otimes r(t))\eta_{u,ref_j} + u_{ss,j},
\]

(17)

\[
\Delta v_j(t) := v_j(t) - [(I_n \otimes r(t))\eta_{v,j} + v_{ss,j}],
\]

(18)

for \( t \in [0,T] \). The desired state trajectory at the intervals \( j = 0, 1, \ldots, N - 1 \) is represented by \( x_j(t) := x_{\text{des}}(t + jT) \), whereas \( v_j(t) := v(t + jT) \) represents the disturbances divided over the different intervals. Defining the optimization variables through the following auxiliary quantities,

\[
z_{x,ref} := (\eta_{x,ref,0}, x_{ss,0}, \eta_{x,ref,1}, x_{ss,1}, \ldots)
\]

(19)

\[
z_{u,ref} := (\eta_{u,ref,0}, u_{ss,0}, \eta_{u,ref,1}, u_{ss,1}, \ldots)
\]

(20)

\[
z_v := (\eta_{v,0}, v_{ss,0}, \eta_{v,1}, v_{ss,1}, \ldots)
\]

(21)

and eliminating the terms that do not depend on these variables, simplifies (15) to

\[
\frac{1}{2} [z_{x,ref}^T \bar{Q} z_{x,ref} + 2f_x^T z_{x,ref} + z_{u,ref}^T \bar{R} z_{u,ref} + z_v^T \bar{V} z_v + 2f_v^T z_v],
\]

(22)

with \( \bar{Q}, \bar{R}, \bar{V}, f_x, \) and \( f_v \) defined in Appendix A.

In addition, the dynamics need to be fulfilled for the generated trajectories, and the steady-state offsets are required to be equilibria. This is encoded, for \( j = 0, \ldots, N - 1 \), as

\[
(I_n \otimes M^T)\eta_{x,ref_j} = (A \otimes I_j)\eta_{x,ref,j} + (B \otimes I_j)\eta_{u,ref,j} + (G \otimes I_j)\eta_{v,j}
\]

(23)

\[
Ax_{ss,j} + Bu_{ss,j} + Gv_{ss,j} = 0.
\]

(24)

In order to ensure that the state, the input, and the disturbances are continuous over subsequent intervals, the following constraints are added, for \( j = 1, \ldots, N - 1 \),

\[
(I_n \otimes r(0)^\top)\eta_{x,ref,j} + x_{ss,j} - (I_n \otimes r(T)^\top)\eta_{x,ref,j-1} - x_{ss,j-1} = 0
\]

(25)

\[
(I_m \otimes r(0)^\top)\eta_{u,ref,j} + u_{ss,j} - (I_m \otimes r(T)^\top)\eta_{u,ref,j-1} - u_{ss,j-1} = 0
\]

(26)

\[
(I_n \otimes r(0)^\top)\eta_{v,j} + v_{ss,j} - (I_n \otimes r(T)^\top)\eta_{v,j-1} - v_{ss,j-1} = 0.
\]

(27)

The constraints (25), (26), (27) prevent discontinuities in the actuation when the trajectory tracking controller switches reference parameters among consecutive intervals. In case the disturbances stem from the system’s nonlinearities, (27) encodes the natural fact that the disturbances are continuous. Without loss of generality, the trajectories are enforced to start at zero (other cases can be dealt with in a similar way, up to a translation) with the following constraints,

\[
(I_n \otimes r(0)^\top)\eta_{x,ref,0} + x_{ss,0} = 0
\]

(28)

\[
(I_m \otimes r(0)^\top)\eta_{u,ref,0} + u_{ss,0} = 0
\]

(29)

\[
(I_n \otimes r(0)^\top)\eta_{v,0} + v_{ss,0} = 0.
\]

(30)
Finally, the state and input constraints are imposed, yielding for \( j = 0, \ldots, N - 1, \)
\[
b_{\text{min}} \leq F_k \eta_{k, \text{ref}, j} + D_c x_{ss, j} + F_u \eta_{u, \text{ref}, j} + D_u u_{ss, j} \leq b_{\text{max}}.
\]
(31)

Defining,
\[
z := (z_{x, \text{ref}}, z_{u, \text{ref}, z}).
\]
(32)

the complete optimization problem that is solved to generate the reference set of parameters and steady state offsets is rearranged as,
\[
\inf_{z} \frac{1}{2} z^{\top} H z + f^{\top} z
\]
\[\text{s.t. } A_{\text{eq}} z = 0 \]
\[z_{\text{min}} \leq A_{\text{in}} z \leq z_{\text{max}},\]
(34)

with \( H, f \) defined in Appendix A, and where \( A_{\text{eq}}, A_{\text{in}}, z_{\text{min}}, \) and \( z_{\text{max}} \) are defined through appropriate concatenations of the constraints (23) to (31). As mentioned in the previous subsection, in contrast to the work of Sferrazza et al.,\textsuperscript{23} the steady-state offsets are included among the decision variables.

In the authors’ experience, using an active-set solver,\textsuperscript{24} the optimization problem (33) can be solved more efficiently when the equality constraints are eliminated. This is done by means of QR factorization of \( A_{\text{eq}}, \) as,
\[
A_{\text{eq}} = P_A R_A Q_A^{\top} = P_A \begin{bmatrix} R_{A,1}^{\top} & 0 \\ \end{bmatrix} \begin{bmatrix} Q_{A,1}^{\top} \\ Q_{A,2}^{\top} \end{bmatrix} = P_A R_{A,1}^{\top} Q_{A,1}^{\top},
\]
(35)

where \( Q_A, Q_{A,1}, \) and \( Q_{A,2} \) are orthogonal matrices, \( R_A \) and \( R_{A,1} \) are upper triangular matrices, \( P_A \) is a permutation matrix such that the magnitude of the elements on the main diagonal of \( R_A \) is decreasing, and \( 0 \) is the zero matrix of appropriate dimensions. From the orthogonality of \( Q_A, \) it follows that by replacing,
\[
z = Q_{A,2} \hat{z},
\]
(36)

where \( \hat{z} \) is an arbitrary vector of appropriate dimension, (34) is satisfied,
\[
A_{\text{eq}} z = P_A R_{A,1}^{\top} Q_{A,1}^{\top} Q_{A,2} \hat{z} = 0.
\]
(37)

Therefore, according to (36), the optimization problem in (33) can be reformulated as,
\[
\inf_{\hat{z}} \frac{1}{2} \hat{z}^{\top} Q_{A,2}^{\top} H Q_{A,2} \hat{z} + f^{\top} Q_{A,2} \hat{z}
\]
\[\text{s.t. } z_{\text{min}} \leq A_{\text{in}} Q_{A,2} \hat{z} \leq z_{\text{max}},\]
(38)

The problem in (38) has generally considerably less optimization variables and constraints than the one in (33), and it exhibited smaller solution times in the setup used by the authors.

### 2.3 Kalman filter

The repeatable disturbances of the system are estimated with a Kalman filter, and are provided at each trial to the optimization problem described in Section 2.2. The filter aims at fusing prior and measurement information. Its state is denoted by \( v[k], \) for \( k = 0, 1, \ldots, \) where the square brackets indicate the discrete-time indexing, \( v[k] = v(kT_s), \) with \( T_s \) being the sampling time.

The process model is chosen to express the expected repeatability of the estimated disturbances over subsequent trials as,
\[ v[k]^i = v[k]^{i-1} + n[k]^i, \]  
(39)

where \( i \) is the current trial, and the process noise \( n[k]^i \) is assumed to be zero-mean Gaussian, with variance \( Q_{KF} \). Therefore, the Kalman filter’s prior update is,

\[ \hat{v}_p[k]^i = \hat{v}_m[k]^{i-1} \]  
(40)

\[ P_p[k]^i = P_m[k]^{i-1} + Q_{KF}, \]  
(41)

where \( \hat{v}_p[k]^i \) and \( P_p[k]^i \) express the current belief (that is the expected value and variance) of \( v[k]^i \), conditioned on the measurements up to the \( (i-1) \)th trial, and \( \hat{v}_m[k] \) and \( P_m[k]^i \) express the current belief (again, mean and variance) of \( v[k]^i \), given the measurements up to the \( i \)th trial. The information provided by the deviations from the system’s dynamics at each trial of the trajectory tracking are incorporated in the measurement model of the Kalman filter. Given the discretized dynamics,

\[ x[k + 1] = A_d x[k] + B_d u[k] + G_d v[k], \]  
(42)

where \( A_d, B_d, \) and \( G_d \) are the discrete-time system’s matrices, the actual state, and input trajectories, respectively, \( x[k]^i \) and \( u[k]^i \), are recorded during the \( i \)th trial, and used to compute these deviations as,

\[ G_d v[k]^i = x[k + 1]^i - A_d x[k]^i + B_d u[k]^i. \]  
(43)

In order to increase the signal-to-noise ratio, the disturbance vector is held constant for \( N_{KF} \) discrete time steps. In this way, at each trial, \( N_{KF} \) measurements subsequently contribute to update a single disturbance estimate value. Assuming that \( z[k]^i \) is corrupted by zero-mean Gaussian noise with variance \( R_{KF} \), the measurement update is performed as described in Algorithm 7. The current disturbance estimate is transformed to a continuous time signal \( v(t) \) through a zero-order hold, and is used as an input to the trajectory generation problem, as described in Section 2.2. In contrast to the work of Sferrazza et al,\(^{23}\) the parametrized approximation describing \( v(t) \) is obtained by optimizing (15), providing a trade-off between the placement of the various offsets and the fit of the reference trajectories.

**Algorithm 1.** Pseudo-code for measurement update

```
Initialize: k = 0
while k is smaller than the trajectory length do
  Initialize: \( \hat{v}_m[k]^i = \hat{v}_p[k]^i, P_m[k]^i = P_p[k]^i \)
  for \( l \in \{0, 1, \ldots, N_{KF} - 1\} \) do
    // Apply the measurements \( z[k + l]^i \) in sequential order
    \[ P_m[k]^i = ((P_m[k]^i)^{-1} + G_d^2 R_{KF}^{-1} G_d^{-1})^{-1} \]
    Update \( \hat{v}_m[k]^i \) with \( \hat{v}_m[k]^i + P_m[k]^i G_d^2 R_{KF}^{-1} (z[k + l]^i - G_d \hat{v}_m[k]^i) \)
  end
  \( \hat{v}_m[k + N_{KF} - 1]^i = \hat{v}_m[k + N_{KF} - 2]^i = \cdots = \hat{v}_m[k]^i \); // The disturbance vector is held constant
  \( P_m[k + N_{KF} - 1]^i = P_m[k + N_{KF} - 2]^i = \cdots = P_m[k]^i \); // for \( N_{KF} \) discrete time steps
  k = k + N_{KF}
end
```

### 3 MEMORY CONSIDERATIONS

The parametrization of the system’s states, inputs, and disturbances is particularly advantageous when applied to trajectory tracking. As a matter of fact, the reference trajectories need to be stored in the memory and loaded during the task execution, as is the case with a standard approach, see, for example, the work of Kühne et al.\(^{25}\) When these trajectories are represented in discrete time, a reference needs to be available for all the quantities of interest at each sampling step.
Therefore, the space requirements amount to,

\[ c \cdot (n + m + n_v) \frac{t_f}{T_s}, \]  

(44)

where \( c \) is the space needed to store a floating point number, \( t_f \) is the trajectory duration, and \( T_s \) is the sampling time. Conversely, by using the parametrization introduced in Section 2.1, a set of parameters and steady-state offsets represents the reference trajectory at each interval. Therefore, the space requirements are,

\[ c \cdot (n + m + n_v)(s + 1) \frac{t_f}{T}. \]

(45)

where \( T \) is the length of the intervals the trajectory is divided into. Space requirements are reduced through the parametrization when,

\[ T > (s + 1)T_s. \]  

(46)

Introducing a space-efficiency coefficient,

\[ \nu_s = \frac{T}{(s + 1)T_s}, \]  

(47)

the condition in (46) can be expressed as,

\[ \nu_s > 1. \]  

(48)

In the application presented in Section 4, a space-efficiency coefficient of about 8.33 is achieved, meaning that almost an order of magnitude less memory is required compared to a standard approach. Note that this also greatly affects the dimension of the trajectory generation problem. As pointed out in the book by Bentley,\(^{26}\) space efficiency often has positive effects on run time. In fact, a smaller program is likely to fit into faster levels of the memory hierarchy,\(^{27}\) which generally have limited size. These effects are even more relevant when a collection of several trajectories need to be available on embedded systems with relatively small random access memory. In this case, the accessing times stemming from the need to load the trajectory data from disk storage can be reduced by a compact representation of the reference trajectories, as described in this article.

## 4 | EXPERIMENTAL RESULTS

The proposed approach is evaluated on a quadcopter, with the task of balancing a pole while flying a predefined trajectory. This section presents the setup and the experimental results.

### 4.1 | Hardware

The experiments are conducted on a quadcopter which uses the frame, motors, and motor controllers of an AscTec Hummingbird. The quadcopter is equipped with custom electronics and a circular plate that serves as a base for the pole. The physical parameters of the experimental setup are summarized in Table 1. An onboard controller, running at 1 kHz on a PX4FMU flight controller, tracks the desired angular rates and the desired total thrust, which are commanded by the parametrized MPC controller. The latter runs on a standard desktop computer that communicates with the quadcopter’s flight controller through wireless communication and achieves a sampling time of 0.02 s. The experiments were conducted in the Flying Machine Arena,\(^{28}\) where an external motion tracking system provides measurements of the system’s state.

### 4.2 | Model

The quadcopter’s state is represented by its position, that is, \( r_x, r_y, \) and \( r_z \), the corresponding linear velocities \( v_x, v_y, \) and \( v_z \), and the xyz-Euler angles, that is, \( \alpha, \beta, \) and \( \gamma \). The pole is represented by its roll and pitch angles, respectively, \( \alpha_p \) and \( \beta_p \), and the corresponding angular velocities \( \omega_{p, \alpha} \) and \( \omega_{p, \beta} \). The pole is assumed to be very thin and its rotation about its axis of


Table 1: Physical parameters

| Symbol | Value  | Description                  |
|--------|--------|------------------------------|
| $m_q$  | 0.5 Kg | mass of the quadrotor       |
| $d_q$  | 0.17 m | length of one quadrotor arm |
| $m_p$  | 0.038 Kg | mass of the pole            |
| $d_p$  | 0.584 m | half length of the pole     |
| $f_{\text{min}}$ | 0.616 N | minimum rotor thrust         |
| $f_{\text{max}}$ | 3.862 N | maximum rotor thrust         |
| $\alpha_{\text{rel,max}}$ | 30° | maximum relative roll angle  |
| $\beta_{\text{rel,max}}$ | 30° | maximum relative pitch angle |

Symmetry is not modeled. The quadcopter’s angular rates $\omega_\alpha$, $\omega_\beta$, and $\omega_\gamma$, and the difference between the total thrust $f_{\text{tot}}$ and the hovering thrust $f_{\text{tot,0}}$ are regarded as the control inputs for the MPC controller. These are then sent to the onboard controller, where they are tracked with a very high bandwidth. Summarizing, the complete state and input vectors are,

$$x = (r_x, r_y, r_z, v_x, v_y, v_z, \alpha, \beta, \gamma, \omega_\alpha, \omega_\beta, \omega_\gamma),$$

$$u = (\omega_\alpha, \omega_\beta, \omega_\gamma, f_{\text{tot}} - f_{\text{tot,0}}).$$

The system’s matrices are derived from a simple first principles model (see Appendix B for a detailed derivation), which considers the quadcopter and the pole to be rigid bodies. Friction between the pendulum and the quadcopter, drag of the pendulum and the quadcopter, as well as any other aerodynamic forces, are neglected (the propellers are assumed to provide a certain thrust aligned with the symmetry axis of the quadcopter). The nonlinear model is linearized about hover, which yields

$$A =$$

$$B =$$

where $k_1$, $k_2$, and $k_3$ are constants (see Appendix B), $m_q$ is the mass of the quadcopter, and $m_p$ is the mass of the pole. The matrix $G$, which models how the disturbances affect the system’s dynamics, is chosen as the identity matrix of dimension $n_v$.

Actuation constraints are imposed on the thrust $f_l$ of a single rotor, that is, for $l = 0, 1, 2, 3$,

$$f_{\text{min}} \leq f_l \leq f_{\text{max}}.$$
State constraints on the relative angles between the pole and the quadcopter ensure sufficient contact between the two bodies, and are expressed as,

\[ |\alpha - \alpha_p| \leq \alpha_{\text{rel, max}} \]  \hspace{1cm} (52)

\[ |\beta - \beta_p| \leq \beta_{\text{rel, max}}. \]  \hspace{1cm} (53)

These constraints can be reduced to the form in (14), as shown in Appendix C.

### 4.3 Results

The tuning matrices for the trajectory tracking are chosen as follows,

\[ Q = \text{diag} (10, 10, 10, 1, 1, 1, 1, 1, 10, 0.3, 0.3, 0.15, 0.15) \]

\[ R = \text{diag}(0.05, 0.05, 0.5, f_{\text{tot}}) \]

| Symbol | Value | Description |
|--------|-------|-------------|
| \( T_s \) | 0.02 s | sampling time |
| \( n_v \) | 13 | dimension of the disturbance vector |
| \( s \) | 5 | number of basis functions |
| \( \lambda \) | 4 s\(^{-1}\) | basis function decay |
| \( T \) | 1 s | interval length |
| \( N_{KF} \) | 10 | Kalman filter parameter |
| \( r \) | 0.1 | regularization parameter |

**FIGURE 4** Result of the trajectory generation procedure. A three-dimensional figure-eight (in solid red), with sharp edges, is approximated by a feasible trajectory (in dashed black) that is parametrized with basis functions and steady-state offsets [Colour figure can be viewed at wileyonlinelibrary.com]
The matrices for the trajectory generation are,

\[
\bar{Q} = 0.01 \text{diag} \begin{pmatrix} 10^6, 10^6, 10^6, 10^5, 10^{-4}, 10^{-4}, 1 \end{pmatrix},
\]

\[
\bar{R} = \text{diag} \begin{pmatrix} 1, 1, 1, 0.01, f_{tot} \end{pmatrix},
\]

\[
\bar{V} = 10^4 \text{diag} \begin{pmatrix} 300, 300, 100, 1, 1, 1, 1, 1, 100, 100, 1, 1 \end{pmatrix}.
\]

The desired trajectory \( x_{\text{des}}(t) \) is only provided for \( r_x, r_y, \) and \( r_z \), and is equal to zero for all the other quantities, therefore leaving the solver the flexibility to choose feasible trajectories for these states. The matrices \( Q, R, \bar{Q}, \bar{R}, \) and \( \bar{V} \) are chosen to be positive definite matrices, which implies that the trajectory tracking problem in (8) and the trajectory generation problem in (33) are strictly convex quadratic programs. This ensures convergence of both programs to a feasible solution (if a solution exists) when solved with an active set method.\(^{29}\)

A “prerun” and a “postrun” trajectory’s portions are generally added at the start and end of the trajectory, see Figure 3, by padding the desired trajectory with the initial and final state, respectively, for an appropriate time. Both the prerun and the postrun ensure a smoother transition with the hovering phase, which precedes and follows each execution. In addition, the postrun portion serves as a penalty for steering the system to the final state (note that the trajectory generation does not impose a hard constraint on the final state).

The Kalman filter’s matrices used to estimate the system’s disturbances are,

\[
Q_{\text{KF}} = 10^{-2} \text{diag} \begin{pmatrix} 1, 1, 1, 10, 10, 0.01, 0.01, 1, 1, 1, 1 \end{pmatrix},
\]

\[
R_{\text{KF}} = 10^{-4} \text{diag} \begin{pmatrix} 1, 1, 1, 0.1, 0.1, 0.1, 0.1, 100, 100, 1, 1, 1, 1 \end{pmatrix}.
\]

The Kalman filter’s state is initialized with zero mean and the following variance,

\[
P_0 = 10^{-2} I_{n_e}.
\] (54)

The remaining scalar parameters are summarized in Table 2.

Both the trajectory generation and the online trajectory tracking are solved using qpOASES,\(^{24}\) which implements an active set strategy. The trajectory generation program is solved on average in 250 ms, while the trajectory tracking controller runs at 50 Hz.

A 12 s trajectory representing a three-dimensional eight is encoded in \( x_{\text{des}}(t) \), and the parametrization obtained through the trajectory generation procedure provides a feasible approximation, as shown in Figure 4. As a result, the space-efficiency coefficient introduced in (47) is equal to,

![Figure 5](wileyonlinelibrary.com)

**FIGURE 5** This plot shows the resulting trajectory tracking cost over the 22 trials conducted. The proposed method estimates the repeatable disturbances over the entire trajectory. After an initial exploration phase, the cost converges toward a value that is about three times lower than the initial one [Colour figure can be viewed at wileyonlinelibrary.com]
This trajectory is repeatedly tracked by the parametrized MPC controller introduced in Section 2.1, and the disturbances are updated after each trial, as explained in Section 2.3. The resulting tracking performance is shown in Figures 5 and 6.\(^1\) The framework presented in this article builds upon the scenario that trajectories are repetitively tracked, and that the system's disturbances can be captured by the basis functions used for the parametrization. In fact, the iterative learning algorithm especially needs to compensate for tracking errors when the quadcopter tilts and changes direction, that is, when the model linearized about hover is not very accurate. The fact that the system improves its tracking performance in these regions over subsequent trials shows that this framework is able to account for unmodeled nonlinearities, provided that the above assumptions are met. Compared with the heuristic approach of Sferrazza et al.\(^2^3\) the use of steady-state offsets enables the proposed strategy to cope with trajectories and disturbances with larger magnitude (even at the end of an interval), as shown in the example in Appendix D. Together with the fact that a different system is used for the experiments, this facilitates the tracking of considerably more aggressive trajectories.

\(^1\)A video showing an experiment can be found at the following link: https://youtu.be/-E4znjVDCyA.
The disturbance estimate shows a higher variance toward the end of the trajectory as a consequence of the particular choice of modeling the system’s disturbance as time-varying trajectories. In fact, an incorrect disturbance estimate in an earlier part of the trajectory propagates in time affecting the later parts (it might even drive the system to a state where the current disturbance estimate is not a valid approximation). However, after multiple trials, the disturbance estimates converge, see, for example, Figure 7, which alleviates this effect.

The quadratic cost for the different trials, defined through the matrices $Q$ and $R$, is shown in Figure 8. Due to the fact that the disturbances are not modeled as a function of the state, they are subject to change when the system explores different regions in the state space, that is, during the first trials, when the algorithm has not yet converged. Therefore, the procedure might first exhibit an increase in the cost, which corresponds to this initial exploration phase, before reaching an overall lower steady-state cost. However, the aforementioned choice of modeling the disturbances as time-varying trajectories has the benefit of not increasing the complexity of the online trajectory tracking MPC controller.

In the experiment presented, the algorithm only requires five trials to reach the steady-state cost, and it exhibits little fluctuation in the remaining executions. Although in this example the system always executed the same task under the same conditions, the fast learning rate and the fact that the learning algorithm keeps running after each trial (ie, no early stopping is required) enables the possibility of responsively adapting the disturbance estimate to new situations, for example, if the quadcopter is subject to a different load.
FIGURE 8  The above plots show how a single disturbance’s component is estimated and approximated. In (A), the first disturbance vector’s component is shown over different trials. Given the choice of the matrix $G$, this component is added to the equation of motion that describes the dynamics of the state $r_x$. The plot (A), where the last estimate is shown in blue (thick), indicates convergence of the disturbance trajectory. In (B), the last estimate is compared with the approximation (parametrized by the basis functions) obtained from the trajectory generation procedure [Colour figure can be viewed at wileyonlinelibrary.com]

5 | CONCLUSION

This article presented a method that enables the use of parametrized MPC for trajectory tracking. A systematic framework that can capture and compensate for unmodeled dynamics has been developed. Experimental results indicate that: (i) the tracking performance indeed improves when executing the same experiment, and (ii) the resulting control scheme brings advantages in terms of computation and storage compared to a traditional MPC approach.

ACKNOWLEDGMENTS

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

1. Richalet J, Rault A, Testud JL, Papon J. Model predictive heuristic control: applications to industrial processes. *Automatica*. 1978;14(5):413-428.

2. Borrelli F, Falcone P, Keviczky T, Asgari J, Hrovat D. MPC-based approach to active steering for autonomous vehicle systems. *Int J Veh Auton Syst*. 2005;3(2-4):265-291.

3. Geyer T, Papafotiou G, Morari M. Model predictive direct torque control—Part I: concept, algorithm, and analysis. *IEEE Trans Ind Electron*. 2009;56(6):1894-1905.

4. Papafotiou G, Kley J, Papadopoulos KG, Bohren P, Morari M. Model Predictive direct torque contro—Part II: implementation and experimental evaluation. *IEEE Trans Ind Electron*. 2009;56(6):1906-1915.

5. Morari M, Lee JH. Model predictive control: past, present and future. *Comput Chem Eng*. 1999;23(4):667-682.

6. Muehlebach M, D’Andrea R. Parametrized infinite-horizon model predictive control for linear time-invariant systems with input and state constraints. Paper presented at: Proceedings of the American Control Conference. 2016:2669-2674.

7. Dunbar WB, Milam MB, Franz R, Murray RM. Model predictive control of a thrust-vector flight control experiment. *Proc IFAC World Congress*. 2002;35:355-360.

8. Gondhalekar R, Imura JI. Least-restrictive move-blocking model predictive control. *Automatica*. 2010;46(7):1234-1240.

9. Domahidi A, Zgraggen AU, Zeilinger MN, Morari M, Jones CN. Efficient interior point methods for multistage problems arising in receding horizon control. Paper presented at: Proceedings of the Conference on Decision and Control. 2012:668-674.

10. Wang Y, Boyd S. Fast model predictive control using online optimization. *IEEE Trans Contr Syst Technol*. 2010;18(2):267.

11. Wang L. *Model predictive control system design and implementation using MATLAB*. Germany: Springer Science & Business Media; 2009.

12. Rossiter JA, Wang L. Exploiting Laguerre functions to improve the feasibility/performance compromise in MPC. Paper presented at: Proceedings of the Conference on Decision and Control. 2008:4737-4742.

13. Khan B, Rossiter JA. Alternative parameterisation within predictive control: a systematic selection. *Int J Control*. 2013;86(8):1397-1409.

14. Faulwasser T, Findeisen R. A model predictive control approach to trajectory tracking problems via time-varying level sets of Lyapunov functions. Paper presented at: Proceedings of the Conference on Decision and Control and European Control Conference. 2011:3381-3386.

15. Stephan MA, Manzie C, Good MC. Model predictive control for reference tracking on an industrial machine tool servo drive. *IEEE Trans Indus Inform*. 2013;9(2):808-816.

16. Neunert M, De Crousaz C, Furrer F, Kamel M, Farshidian F, Siegwart R, Buchli J. Fast nonlinear model predictive control for unified trajectory optimization and tracking. Paper presented at: Proceedings of the International Conference on Robotics and Automation. 2016:1398-1404.

17. Kamel M, Burri M, Siegwart R. Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles. *Proc IFAC World Congress*. 2017;50(1):3463-3469.

18. Mueller MW, D’Andrea R. A model predictive controller for quadrocopter state interception. Paper presented at: Proceedings of the European Control Conference. 2013:1383-1389.

19. Bouffard P, Arwani A, Tomlin C. Learning-based model predictive control on a quadrotor: onboard implementation and experimental results. Paper presented at: Proceedings of the International Conference on Robotics and Automation. 2012:279-284.

20. Wang Y, Zhou D, Gao F. Iterative learning model predictive control for multi-phase batch processes. *J Process Control*. 2008;18(6):543-557.

21. Rosolia U, Borrelli F. Learning model predictive control for iterative tasks. A data-driven control framework. *IEEE Trans Autom Control*. 2018;63(7):1883-1896.

22. Muehlebach M, Sferrazza C, D’Andrea R. Implementation of a parametrized infinite-horizon model predictive control scheme with stability guarantees. Paper presented at: Proceedings of the International Conference on Robotics and Automation. 2017:2723-2730.

23. Sferrazza C, Muehlebach M, D’Andrea R. Trajectory tracking and iterative learning on an unmanned aerial vehicle using parametrized model predictive control. Paper presented at: Proceedings of the Conference on Decision and Control. 2017:5186-5192.

24. Ferreau HJ, Kirches C, Potschka A, Bock HG, Diehl M. qpOASES: A parametric active-set algorithm for quadratic programming. *Math Program Comput*. 2014;6(4):327-363.

25. Kühne F, Gomes J, Fetter W. Mobile robot trajectory tracking using model predictive control. Paper presented at: Proceedings of the Latin-American Robotics Symposium. 2005.

26. Bentley J. *Programming Pearls*. Boston: Addison-Wesley Professional; 2016.

27. Patterson DA, Hennessy JL. Large and fast: exploiting memory hierarchy. *Computer Organization and Design: The Hardware/Software Interface*. England, UK: Newnes; 2013.
APPENDIX A. MATHEMATICAL DERIVATION OF THE PARAMETRIZED MPC MATRICES

In the following, the cost function (15) is rearranged by splitting the integral and expanding the different terms. The terms regarding the system’s state are expanded as,

$$
\int_0^T \frac{1}{2} [\Delta \dot{x}_j(t)^T \bar{Q} \Delta x_j(t)] dt = \int_0^T \frac{1}{2} \left\{ x_j(t)^T \bar{Q} x_j(t) + [(I_n \otimes \tau(t))^T \eta_{x,\text{ref},j} + x_{ss,j}]^T \bar{Q} [(I_n \otimes \tau(t))^T \eta_{x,\text{ref},j} + x_{ss,j}] \\
-2x_j(t)^T \bar{Q} [(I_n \otimes \tau(t))^T \eta_{x,\text{ref},j} + x_{ss,j}] \right\} dt
$$

$$
= \int_0^T \frac{1}{2} \left\{ x_j(t)^T \bar{Q} x_j(t) + x_{ss,j}^T \bar{Q} x_{ss,j} + \eta_{x,\text{ref},j}^T (I_n \otimes \tau(t)) \bar{Q} (I_n \otimes \tau(t))^T \eta_{x,\text{ref},j} - 2x_j(t)^T \bar{Q} x_{ss,j} \right. \\
+ \left. x_{ss,j}^T (I_n \otimes \tau(t))^T \eta_{x,\text{ref},j} + \eta_{x,\text{ref},j}^T (I_n \otimes \tau(t)) \bar{Q} x_{ss,j} - 2x_j(t)^T \bar{Q} (I_n \otimes \tau(t))^T \eta_{x,\text{ref},j} \right\} dt. \quad (A1)
$$

Neglecting the first term in the sum, which does not depend on the optimization variables, and defining $z_{x,\text{ref},j} := (\eta_{x,\text{ref},j}, x_{ss,j})$, (A1) can be rewritten as,

$$
\frac{1}{2} [z_{x,\text{ref},j}^T \tilde{Q} z_{x,\text{ref},j} + 2f_{x,j}^T z_{x,\text{ref},j}], \quad (A2)
$$

with,

$$
\tilde{Q}_j = \int_0^T \begin{bmatrix} (I_n \otimes \tau(t)) \bar{Q} (I_n \otimes \tau(t))^T & (I_n \otimes \tau(t)) \bar{Q} \\
\bar{Q} (I_n \otimes \tau(t))^T & \bar{Q} \end{bmatrix} dt
$$

$$
= \begin{bmatrix} Q_j & \int_0^T (I_n \otimes \tau(t)) \bar{Q} dt \\
\int_0^T \bar{Q} (I_n \otimes \tau(t))^T dt & \bar{Q} T \end{bmatrix}
$$

$$
f_{x,j} = -\int_0^T \begin{bmatrix} (I_n \otimes \tau(t)) \bar{Q} x_j(t) \\
\bar{Q} x_j(t) \end{bmatrix} dt, \quad (A3)
$$

where

$$
J_s := \int_0^T \tau(t) \tau(t)^T dt, \quad (A4)
$$

which can be computed as the solution to the following Lyapunov equation, as shown in the work of Sferrazza et al.,\textsuperscript{23}

$$
MJ_s + J_s M^T - [\tau(T) \tau(T)^T - \tau(0) \tau(0)^T] = 0. \quad (A5)
$$

The terms regarding the system’s disturbances are expanded in the same way, defining $z_{v,j} := (\eta_{v,j}, v_{ss,j})$, as,

$$
\frac{1}{2} [z_{v,j}^T \tilde{V}_j z_{v,j} + 2f_{v,j}^T z_{v,\text{ref},j}], \quad (A6)
$$
where

\[ V_j = \left[ \int_0^T \bar{V} \otimes J_s \quad \int_0^T (I_{n_s} \otimes \tau(t)) \bar{V} \, dt \right] \]

\[ f_{v,j} = - \int_0^T \left( (I_{n_v} \otimes \tau(t)) \bar{V}_j(t) \right) dt. \quad (A7) \]

Note that due to Assumption H2 in Section 2.1,

\[ \dot{u}_{ref,j}(t) = (I_m \otimes \tau(t))^T \eta_{u,ref,j} \]

\[ = (I_m \otimes M \tau(t))^T \eta_{u,ref,j}, \quad (A8) \]

which provides a means to rearrange the input terms as,

\[ \int_0^T \frac{1}{2} [r\dot{u}_{ref,j}(t)^T \dot{u}_{ref,j}(t) + \ddot{u}_{ref,j}(t)^T \bar{R}\ddot{u}_{ref,j}(t)] dt = \int_0^T \frac{1}{2} \left\{ r\eta_{u,ref,j}^T (I_m \otimes M \tau(t))(I_m \otimes M \tau(t))^T \eta_{u,ref,j} \right. \]

\[ + [(I_m \otimes \tau(t))^T \eta_{a,ref,j} + u_{ss,j}]^T (I_m \otimes \tau(t))^T \eta_{a,ref,j} + u_{ss,j} \right\} \]

\[ = \int_0^T \frac{1}{2} \left\{ r\eta_{a,ref,j}^T (I_m \otimes M \tau(t)^T M^T)^T \eta_{a,ref,j} \right. \]

\[ + u_{ss,j}^T \bar{R}u_{ss,j} + \eta_{a,ref,j}^T (I_m \otimes \tau(t))^T \bar{R}(I_m \otimes \tau(t))^T \eta_{a,ref,j} \]

\[ + u_{ss,j}^T \bar{R}(I_m \otimes \tau(t))^T \eta_{a,ref,j} + \eta_{a,ref,j}^T (I_m \otimes \tau(t))^T \bar{R}u_{ss,j} \right\} dt. \quad (A10) \]

Defining \( z_{a,ref,j} := (\eta_{a,ref,j}, u_{ss,j}) \), (A10) can be rewritten as,

\[ \frac{1}{2} z_{a,ref,j}^T \bar{R}z_{a,ref,j}. \quad (A11) \]

with

\[ \bar{R}_j = \left[ \begin{array}{c} \bar{R} \otimes J_s + rI_m \otimes (M J_s M^T) \\ \int_0^T \bar{R}(I_m \otimes \tau(t))^T dt \end{array} \right]. \]

Summing (A2), (A6) and (A11) over all the intervals, the cost function (15) reduces to,

\[ \frac{1}{2} \left[ z_{x,ref}^T \bar{Q} z_{x,ref} + 2 f_{x}^T z_{x,ref} + z_{u,ref}^T \bar{R} z_{u,ref} + z_{v}^T \bar{V} z_{v} + 2 f_{v}^T z_{v} \right] = \frac{1}{2} z^THz + f^Tz. \quad (A12) \]

with \( z, z_{x,ref}, z_{u,ref}, \) and \( z_{v} \) defined as in (32), (19), (20), and (21), and,

\[ \bar{Q} = \begin{bmatrix} \bar{Q}_0 & \ddots & 0 \\ \vdots & \ddots & \ddots \\ 0 & \ddots & \bar{Q}_{N-1} \end{bmatrix}, \quad \bar{R} = \begin{bmatrix} \bar{R}_0 & \ddots & 0 \\ \vdots & \ddots & \bar{R}_{N-1} \end{bmatrix}, \quad \bar{V} = \begin{bmatrix} \bar{V}_0 & \ddots \\ \vdots & \bar{V}_{N-1} \end{bmatrix} \]

\[ H = \begin{bmatrix} \bar{Q} \\ \bar{R} \\ \bar{V} \end{bmatrix} \]

\[ f_x = (f_{x,0}, \ldots, f_{x,N-1}) \quad (A15) \]

\[ f_v = (f_{v,0}, \ldots, f_{v,N-1}) \quad (A16) \]

\[ f = (f_{x,0}, f_{v}). \quad (A17) \]
APPENDIX B. FIRST PRINCIPLES MODEL

For the purpose of modeling the system’s dynamics, the following frames are introduced:

\{I\} Inertial frame,
\{B\} Quadcopter-fixed frame, with its z-axis aligned with the thrust direction (and symmetry axis of the quadcopter),
\{C\} Pole-fixed frame, with its z-axis aligned with the symmetry axis of the pole.

The equations of motion are derived by using the principle of virtual power. This yields

\[ m_q \dot{v}_q + m_p \dot{v}_p = R_{\text{IB}} f_{\text{tot}} + (m_q + m_p) \dot{g}, \]  

(B1)

describing the position of the quadcopter and

\[ c \Theta_p \omega_{IC} - m_p c_{r_{PQ}} R_{IC}^\top I \dot{v}_p = -c \Theta_p \omega_{IC} - m_p c_{r_{PQ}} R_{IC}^\top I g, \]  

(B2)

describing the attitude of the pendulum, \( m_p \) and \( m_q \) are the masses of the quadcopter and the pole, respectively, \( v_p \) and \( v_q \) are the linear velocities of the quadcopter and the pole, \( R_{IB} \in \text{SO}(3) \) is the rotation matrix that transforms vectors from the frame \( B \) to the frame \( I \), \( R_{IC} \in \text{SO}(3) \) is the rotation matrix that transforms vectors from the frame \( C \) to the frame \( I \), \( f_{\text{tot}} \) is the total thrust generated by the four rotors, \( g \) is the gravity vector, \( \Theta_p \) is the inertia of the pole, \( r_{PQ} \) is the distance vector that goes from the pole’s center of gravity (positioned at half-length of the pole) to the quadcopter’s center of gravity and is expressed in the frame \( C \), and \( \omega_{IC} \) is the angular velocity vector of the \( C \) frame with respect to the inertial frame. Throughout this section, the preceding superscript denotes the frame in which a particular vector or tensor is expressed. Moreover, the “tilde” symbol on top of a vector denotes its corresponding skew symmetric matrix, that is, \( \tilde{a} \) is defined as \( a \times b = \tilde{a}b \) for \( a \in \mathbb{R}^3 \) and for all \( b \in \mathbb{R}^3 \).

Note that the quadrotor’s angular rates, \( \dot{\omega}_{IB} \), are considered to be control inputs for the MPC controller (with the vector’s components \( \dot{\omega}_\alpha, \dot{\omega}_\beta, \dot{\omega}_\gamma \) which are tracked with very high bandwidth by the onboard controller). The quadrotor’s angular rates govern the quadrotor’s attitude by

\[ \dot{R}_{IB} = R_{IB} \dot{\omega}_{IB}, \]  

(B3)

and enter the combined quadrotor and pole dynamics through \( R_{IB} \) in (B1).

Moreover, the fact that the quadrotor and the pole are connected yields the following constraint (expressed on acceleration level)

\[ I \ddot{v}_p - I \ddot{v}_q = -R_{IC}^\top C \dot{\omega}_{IC} C r_{PQ} - R_{IC}^\top C \dot{\omega}_{IC} C r_{PQ}. \]  

(B4)

Thus, the combination of (B1), (B2), (B3), and (B4) describes the entire quadrotor-pole dynamics.

The algorithms presented in this article rely on a linear time-invariant system model. To that extent, the nonlinear equations of motion are linearized about hover. The hover equilibrium is given by zero linear velocity and zero angular velocity of the quadcopter, the pole being at rest in the upright position, and the following constant thrust

\[ f_{\text{tot},0} = (0, 0, (m_q + m_p)g_0), \]  

(B5)

with \( g_0 = 9.81 \text{ m/s}^2 \). Linearizing (B4) about hover yields

\[ I \ddot{v}_p = I \ddot{v}_q + C_{\text{r}_{PQ}} \dot{\omega}_{IC}. \]  

(B6)

Linearizing (B1) and including the relation (B6) results in

\[ (m_q + m_p) \dot{v}_q + m_p C_{\text{r}_{PQ}} \dot{\omega}_{IC} \approx g_0 (m_q + m_p)(e_x \dot{\beta} - e_y \dot{\alpha}) + (f_{\text{tot}} - f_{\text{tot},0}), \]  

(B7)
where $e_x$, $e_y$ and $e_z$ are the standard unit vectors in $\mathbb{R}^3$, $\alpha$ and $\beta$ are the quadcopter’s roll and pitch angles. Linearizing (B2) and including (B6) yields

$$- m_p \bar{C}_p \bar{r}_{PQ} v_q + (\bar{C}_p - m_p \bar{C}_p \bar{r}_{PQ}) \bar{C}_p \omega_{IC} \approx m_p d_p g_0 (e_y \beta_p + e_x \alpha_p),$$

where $\alpha_p$ and $\beta_p$ are the pole’s roll and pitch angles, and $d_p$ is the pole’s half-length, that is, $|r_{PQ}| = d_p$.

The quantities in Equations (B7)-(B8) relate to the ones in (49)-(50) through the following equalities,

$$\bar{r}_{PQ} = (v_x, v_y, v_z),$$

$$\bar{\omega}_{IC} = (\omega_{p,\alpha}, \omega_{p,\beta}, \omega_{p,\gamma}),$$

where the angular velocity $\omega_{p,\gamma}$ of the pole is not included in the system’s state. For the system on which the experiments are conducted, the pole’s inertia matrix has the following form,

$$\bar{C}_p := \text{diag}(I_1, I_1, \ast),$$

where the inertia about the symmetry axis of the pole is irrelevant, as the corresponding rotations are not included in the system’s state. The $A$ and $B$ matrices in (4.2) are obtained by rearranging (B7)-(B8), where

$$c_p := m_p d_p,$$

$$\bar{I}_1 := I_1 + m_p d_p^2,$$

$$k_1 := \frac{\bar{I}_1 (m_q + m_p) g_0}{(m_q + m_p) \bar{I}_1 - c_p^2},$$

$$k_2 := \frac{c_p^2 g_0}{(m_q + m_p) \bar{I}_1 - c_p^2},$$

$$k_3 := \frac{c_p (m_q + m_p) g_0}{(m_q + m_p) \bar{I}_1 - c_p^2}.$$

The parameters used for the experiments presented in this article are summarized in Table B1.

**APPENDIX C. SYSTEM’S CONSTRAINTS**

The actuation constraints (51) on the rotors need some manipulation to be expressed in the form described by (14). Introducing the vector of ones $1_4 \in \mathbb{R}^4$, and stacking the different $f_l$, for $l = 0, 1, 2, 3$, into a vector $\bar{f} \in \mathbb{R}^4$, the constraints on the four rotors can be summarized as,

$$f_{\text{min}} 1_4 \leq \bar{f} \leq f_{\text{max}} 1_4.$$
TABLE C1  Physical parameters

| Symbol | Value | Description |
|--------|-------|-------------|
| $k_f$  | 0.016 m | proportional factor |
| $I_x$  | 0.003 Kg m$^2$ | moment of inertia around x-axis |
| $I_y$  | 0.003 Kg m$^2$ | moment of inertia around y-axis |
| $I_z$  | 0.006 Kg m$^2$ | moment of inertia around z-axis |

Defining the auxiliary vector,

$$\hat{u} := (\dot{\omega}_a, \dot{\omega}_b, \dot{\omega}_r, f_{tot}) = \Delta \hat{u} + \hat{u}_0,$$

with

$$\Delta \hat{u} := (\dot{\omega}_a, \dot{\omega}_b, \dot{\omega}_r, f_{tot} - f_{tot,0})$$

$$\hat{u}_0 := (0, 0, 0, f_{tot,0}),$$

the following relation holds,

$$\hat{u} = F \tilde{f},$$

with,

$$F := \begin{bmatrix} F_0 \\ 1_4^T \end{bmatrix},$$

$$F_0 := \Theta_q^{-1} \begin{bmatrix} 0 & d_q & 0 & -d_q \\ -d_q & 0 & d_q & 0 \\ k_f & -k_f & k_f & -k_f \end{bmatrix},$$

where $\Theta_q$ is quadcopter’s inertia matrix, $d_q$ is the length of a quadcopter’s arm, and $k_f$ is a physical constant. For the system on which the experiments are conducted, the inertia matrix has the following shape,

$$\Theta_q = \text{diag}(I_x, I_y, I_z).$$

The physical parameters introduced in this section are summarized in Table C1. With the introduced relations, (C1) becomes,

$$f_{\min} 1_4 \leq F^{-1} \hat{u} \leq f_{\max} 1_4$$

$$f_{\min} 1_4 \leq F^{-1} (\Delta \hat{u} + \hat{u}_0) \leq f_{\max} 1_4$$

$$f_{\min} 1_4 - F^{-1} \hat{u}_0 \leq F^{-1} \Delta \hat{u} \leq f_{\max} 1_4 - F^{-1} \hat{u}_0.$$  

Considering the approximation in (3), and the Assumption H2 in Section 2.1, (C11) can be rearranged as,

$$f_{\min} 1_4 - F^{-1} \hat{u}_0 \leq F^{-1} (I_m \otimes \tau(t))^{-1} \begin{bmatrix} I_{m-1} \otimes M^T & 0 \\ 0 & I_3 \end{bmatrix} \eta_u + F^{-1} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} u_{ss} \leq f_{\max} 1_4 - F^{-1} \hat{u}_0.$$
The above semi-infinite constraint is required to hold for all times $t \in [0, \infty)$, and is implemented by sampling at the time instances $t_i$, for $i = 0, \ldots, s - 1$. In the experiments described in this article, $t_0 = 0$ s, $t_1 = 0.0929$ s, $t_2 = 0.3215$ s, $t_3 = 0.7164$ s, $t_4 = 1.3692$ s. Thus, by defining

$$F_{u,i} := F^{-1}(I_m \otimes \tau(t_i))^\top \begin{bmatrix} I_{m-1} \otimes M^\top & 0 \\ 0 & I_s \end{bmatrix}, \quad \hat{F}_u := \begin{bmatrix} F_{u,0} \\ \vdots \\ F_{u,s-1} \end{bmatrix}$$

$$D_{u,i} := F^{-1} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad \hat{D}_u := \begin{bmatrix} D_{u,0} \\ \vdots \\ D_{u,s-1} \end{bmatrix}$$

$$u_{\text{min},i} := f_{\text{min}} 1_4 - F^{-1} \hat{u}_0$$

$$u_{\text{min}} := \begin{bmatrix} u_{\text{min},0} \\ \vdots \\ u_{\text{min},s-1} \end{bmatrix}$$

$$u_{\text{max},i} := f_{\text{max}} 1_4 - F^{-1} \hat{u}_0$$

$$u_{\text{max}} := \begin{bmatrix} u_{\text{max},0} \\ \vdots \\ u_{\text{max},s-1} \end{bmatrix}$$

the constraint (C12) can be approximated as

$$u_{\text{min}} \leq \hat{F}_u \eta_u + \hat{D}_u u_{ss} \leq u_{\text{max}}.$$ 

Applying the same sampling strategy to the state constraints (52)-(53), yields

$$x_{\text{min}} \leq \hat{F}_x \eta_x + \hat{D}_x x_{ss} \leq x_{\text{max}},$$

where the matrices $\hat{F}_x$ and $\hat{D}_x$, and the vectors $x_{\text{min}}$ and $x_{\text{max}}$ are defined accordingly. Consequently, the matrices $F_x$, $F_u$, $D_x$, and $D_u$, and the vectors $b_{\text{min}}$ and $b_{\text{max}}$ in (14) are obtained by appropriate stacking,

$$F_x = \begin{bmatrix} 0 \\ F_u \end{bmatrix}, \quad F_u = \begin{bmatrix} \hat{F}_u \\ 0 \end{bmatrix}, \quad D_x = \begin{bmatrix} 0 \\ D_u \end{bmatrix}, \quad D_u = \begin{bmatrix} \hat{D}_u \\ 0 \end{bmatrix}.$$ 

$$b_{\text{min}} = \begin{bmatrix} u_{\text{min}} \\ x_{\text{min}} \end{bmatrix}, \quad b_{\text{max}} = \begin{bmatrix} u_{\text{max}} \\ x_{\text{max}} \end{bmatrix}.$$ 

**APPENDIX D. EXAMPLE SIMULATION FOR COMPARISON TO PREVIOUS WORK**

In Figure D1, simulation results are provided that compare the approach presented here to previous work.\textsuperscript{23} The simulations are conducted on an unconstrained double integrator system, that is, $\ddot{x}(t) = u(t) + v$, where $v = 1$ m/s\textsuperscript{2} is a constant disturbance. A ramp of 10 m is prescribed as the desired trajectory. The same parameters chosen for the experiment presented in this article are used for this example. The plots show that using the approach of Sferrazza et al.,\textsuperscript{23} the steady-state disturbance cannot be fully captured by the basis functions for the entire length of an interval. As a consequence, the approximation shows oscillations and discontinuities at the interval boundaries. This inaccurate approximation tends to reduce the smoothness of the trajectory, compared to the desired one (see the right plots in Figure D1). In contrast, the approach presented here yields an approximation that overlaps with the estimated steady-state disturbance and results in significantly smoother trajectories, as shown in the experiments presented in this article. Note that this effect becomes considerably more relevant for the tracking of aggressive trajectories on higher dimensional systems.
**FIGURE D1** The plots on the left compare the disturbance estimation using, A, the approach presented in the work of Sferrazza et al.\textsuperscript{23} with the one introduced here. The disturbance captured by the Kalman filter after nine trials (in red) is approximated at each interval with exponentially decaying basis functions (in black). The plots on the right show the tracking performance of the state $x$ at the tenth trial for the respective approaches. The generated trajectory that accounts for the current approximation of the disturbance estimate is shown in black. The actual $x$ trajectory is shown in blue, while the desired ramp trajectory is shown in red. Note that the three trajectories overlap in (B) [Colour figure can be viewed at wileyonlinelibrary.com]