Data-Enabled Predictive Control for Quadcopters

Working Paper

Author(s):
Elokda, Ezzat; Coulson, Jeremy; Beuchat, Paul N.; Lygeros, John; Dörfler, Florian

Publication date:
2019

Permanent link:
https://doi.org/10.3929/ethz-b-000415427

Funding acknowledgement:
787845 - Optimal control at large (EC)

Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

Data-Enabled Predictive Control for Quadcopters

Ezzat Elokda | Jeremy Coulson* | Paul N. Beuchat | John Lygeros | Florian Dörfler

Automatic Control Lab (IfA), ETH Zürich, Zürich, Switzerland

Correspondence
*Physikstrasse 3, 8092 Zürich, Switzerland.
Email: jcoulson@control.ee.ethz.ch

Present Address
Physikstrasse 3, 8092 Zürich, Switzerland

Summary

We study the application of a data-enabled predictive control (DeePC) algorithm for position control of real-world nano-quadcoptors. The DeePC algorithm is a finite-horizon, optimal control method that uses input/output measurements from the system to predict future trajectories without the need for system identification or state estimation. The algorithm predicts future trajectories of the quadcopter by linearly combining previously measured trajectories (motion primitives). We illustrate the necessity of a regularized variant of the DeePC algorithm to handle the nonlinear nature of the real-world quadcopter dynamics with noisy measurements. Simulation-based analysis is used to gain insights into the effects of regularization, and experimental results validate that these insights carry over to the real-world quadcopter. Moreover, we demonstrate the reliability of the DeePC algorithm by collecting a new set of input/output measurements for every real-world experiment performed. The performance of the DeePC algorithm is compared to Model Predictive Control (MPC) based on a first-principles model of the quadcopter. The results are demonstrated with a video of successful trajectory tracking of the real-world quadcopter.

KEYWORDS:
Data-driven control, Predictive control, Quadcopters

1 | INTRODUCTION

The analysis and design of control systems is traditionally addressed using a model-based control approach where a model for the system is first identified from data, and the control policy is then designed based on the identified model. The system identification step is often the most time-consuming and challenging part of model-based control approaches [1,2]. System identification often requires expert knowledge and partial system models [3], and unless the control objective is taken into account during the identification process, the obtained model may not be useful for control [4]. These observations as well as the advancements in sensing and computation technologies have motivated a tendency toward data-driven control methods yielding many successes [5,6,7,8]. Such methods bypass the traditional model-based control approach, and design control inputs directly from data. These so-called direct data-driven methods for control design benefit from ease of implementation on complex systems where system identification is too time-consuming and cumbersome. Among these data-driven methods are learning-based and adaptive Model Predictive Control (MPC) approaches, where the unknown system dynamics are substituted with a learned model which maps inputs to output predictions [9,10,11,12]. However, such methods still require learning an input/output model and often involve (stochastic) function approximation by means of neural networks or Gaussian processes, which come with their own tuning challenges and can be inconsistent across applications [13].

One algorithm that does not require any function learning or system identification is the so-called DeePC algorithm[14]. Instead, this algorithm *directly* uses previously measured input/output data to predict future trajectories. The previously measured input/output data from the system act as *motion primitives* that serve as a basis for the subspace of possible system trajectories. The DeePC algorithm builds on the seminal work on linear time invariant (LTI) systems by Willems et al., specifically what is known as the *fundamental lemma* in behavioural systems theory[15]. This result was used by Markovsky et al. for the first time for control purposes allowing for the synthesis of data-driven open loop control for LTI systems[16]. The DeePC algorithm extended this method to closed-loop control and was implemented in a receding horizon optimal control setup. This algorithm was shown to be equivalent to MPC for deterministic LTI systems[14], and was later extended giving guarantees on recursive feasibility and closed-loop stability[17]. Additionally, numerical case studies have illustrated that the algorithm performs robustly on some stochastic and nonlinear systems and often outperforms system identification followed by conventional MPC[18,19]. The focus of this paper is on implementing the DeePC algorithm[14] for the first time on a real-world system. In particular, we seek to analyze how the algorithm can be applied for real-time control of a quadcopter whose dynamics are nonlinear and the measurements are corrupted by noise.

Several other data-driven control methods have been proposed that make use of input/output data in similar ways as DeePC. One method uses the fundamental lemma to synthesize stabilizing output feedback controllers solving the linear quadratic regulation problem using only input/output data[20]. Other methods use previously measured input/output trajectories as motion primitives to compute minimum energy inputs[21], or produce new control inputs for LTI systems[22]. All of these methods, including the DeePC algorithm, rely on the linearity property. The problem becomes challenging when the system is nonlinear and the measurements are noisy. In this paper, we investigate how a robustified, regularized variant of the DeePC algorithm can tackle these challenges, on a real-world implementation of a quadcopter.

Contributions: The DeePC algorithm is implemented for the first time on a real-world system bridging the gap from theory to application. Through this, we gain key insights into choices of the algorithm's hyperparameters, providing tuning guidelines. We demonstrate that the DeePC algorithm is computationally tractable and suitable for real-time control. A video of the DeePC algorithm performing figure 8 trajectory tracking on the real-world quadcopter is provided here: https://polybox.ethz.ch/index.php/s/I0KKwIsudwaLj3n.

Outline: The real-world quadcopter system, problem statement, and DeePC algorithm are introduced in Section 2. The main contributions appear in Section 3, where we present simulation analysis and experimental results, as well as a video of successful trajectory tracking of the quadcopter. We conclude in Section 4 stating some future directions of research.

2 | SETTING

We first present the quadcopter system in Section 2.1, providing details about its input/output channels, and the first-principles modelling that is used for simulation-based analysis. We then formally state in Section 2.2 the quadcopter control goal as a general finite-horizon, discrete-time, optimal control problem. Section 2.3 recalls the DeePC algorithm, showing how it can be used to address both linear time invariant (LTI) and nonlinear stochastic control problems in a data-driven way.

2.1 | Quadcopter

For the purpose of simulation, we use a nonlinear, continuous-time quadcopter model. Full details of the model derivation are provided in other works[23,24]. Here we highlight the key definitions, equations, and control architecture. The model presented is also the starting point for the model-based control methods that are used for comparison in the experimental results in Section 3.4.

We define the model in terms of an *inertial* frame of reference, denoted (I), and a *body* frame of reference attached to the quadcopter, denoted (B), with the origin of frame (B) fixed at the quadcopter's center-of-gravity. The position of the body frame with respect to the inertial frame is denoted by $\vec{p} = \left(p_x, p_y, p_z\right)$. We use Euler angles to describe the orientation of the body frame relative to the inertial frame, and following the ZYX intrinsic Euler angle convention, we denote the roll, pitch, and yaw angles by $\vec{\psi} = (\gamma, \beta, \alpha)$ respectively. The angular rates about the body frame axes are denoted by $\vec{\omega} = \left(\omega_x, \omega_y, \omega_z\right)$. Thus, the model has 12 states, $\left(\vec{p}, \dot{\vec{p}}, \vec{\psi}, \vec{\omega}\right)$, and the inputs to the model are the thrust force from each propeller, denoted f_i, $i = 1, \dots, 4$. The parameters required for the quadcopter model are the mass m, the mass moment of inertia $\boldsymbol{J}$, the body frame coordinates for the center-of-thrust of each propeller (d_{x_i}, d_{y_i}), and the constant of proportionality d_{τ_i} that approximates a linear relation between the torque due to propeller drag and the thrust force f_i. Figure 1 visualizes this definition of the quadcopter. The nonlinear,

continuous-time equations of motion are readily derived as,

$$\ddot{\vec{p}} = \frac{1}{m}\sum_{i=1}^{4} f_i \begin{pmatrix} \cos(\alpha)\sin(\beta)\sin(\gamma) + \sin(\alpha)\cos(\gamma) \\ \sin(\alpha)\sin(\beta)\cos(\gamma) - \cos(\alpha)\sin(\gamma) \\ \cos(\beta)\cos(\gamma) \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ a_g \end{pmatrix}, \tag{1a}$$

$$\dot{\vec{\omega}} = \boldsymbol{J}^{-1}\left(\begin{pmatrix} \sum_{i=1}^{4} f_i d_{y_i} \\ \sum_{i=1}^{4} -f_i d_{x_i} \\ \sum_{i=1}^{4} f_i d_{\tau_i} \end{pmatrix} - \vec{\omega} \times \boldsymbol{J}\vec{\omega} \right), \tag{1b}$$

where a_g is the acceleration due to gravity. An important feature of these equations is that the equilibrium inputs are the same at all positions $\vec{p}$.

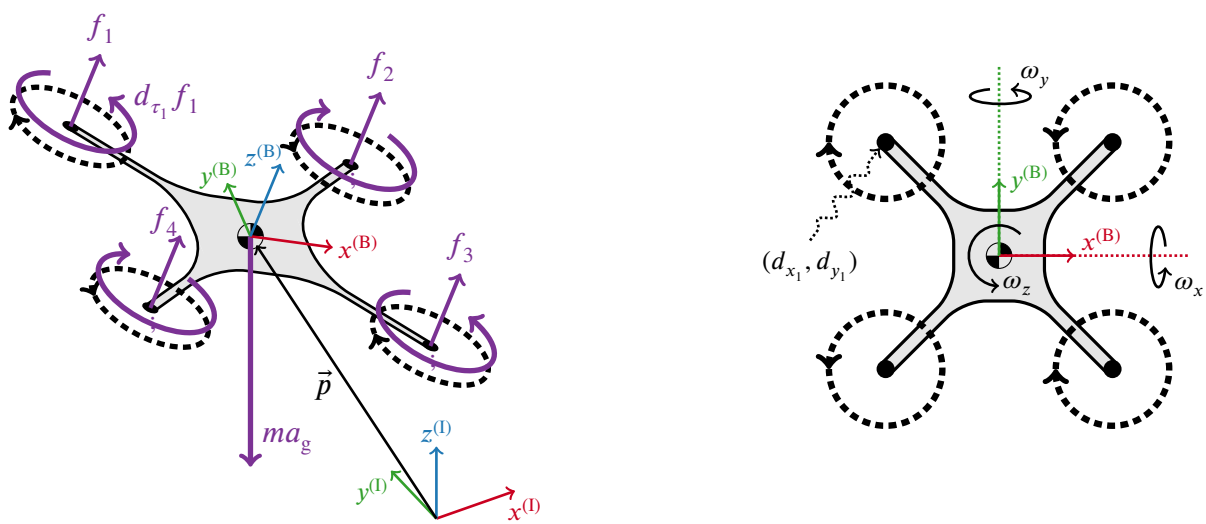


FIGURE 1 Perspective view (left) and top view (right) of the quadcopter model used for simulation; the annotations are defined in Section 2.1. The (red,green,blue) arrows represent the *inertial* and *body* frames of reference, the dashed black circles indicate the direction of rotation of the propellers, and the purple arrows show the forces and torques acting on the quadcopter model.

Most off-the-shelf quadcopters are equipped with an on-board controller that allows the user to specify references instead of directly specifying the thrust force for each propeller, we refer to this as the *inner controller*. Often the manufacturer does not provide details of the inner controller and does not allow the user to bypass it. We consider a quadcopter with an inner controller that uses the data from the onboard inertial measurement unit (IMU) to track user provided references for the angular rate about the $x^{(B)}$ and $y^{(B)}$ axes of the body frame and maintains a constant yaw angle. We leave the inner controller as implemented by the manufacturer, and consider the following three inputs to the system:

- the body rate references about the $x^{(B)}$ and $y^{(B)}$ axes, denoted by $\omega_{\text{ref},x}$ and $\omega_{\text{ref},y}$ respectively, and
- the total thrust force from the propellers combined, denoted by f_{tot}.

The *outer controller* adjusts these three inputs to ensure that the quadcopter tracks a position reference provided by the user, based on feedback of position and orientation measurements, $\vec{p}$, γ, and β, provided by an external motion capture system[25,26]. Our aim is to design a data-driven outer controller for this 3 input, 5 output off-the-shelf quadcopter system (see Figure 2 for a schematic of the architecture).

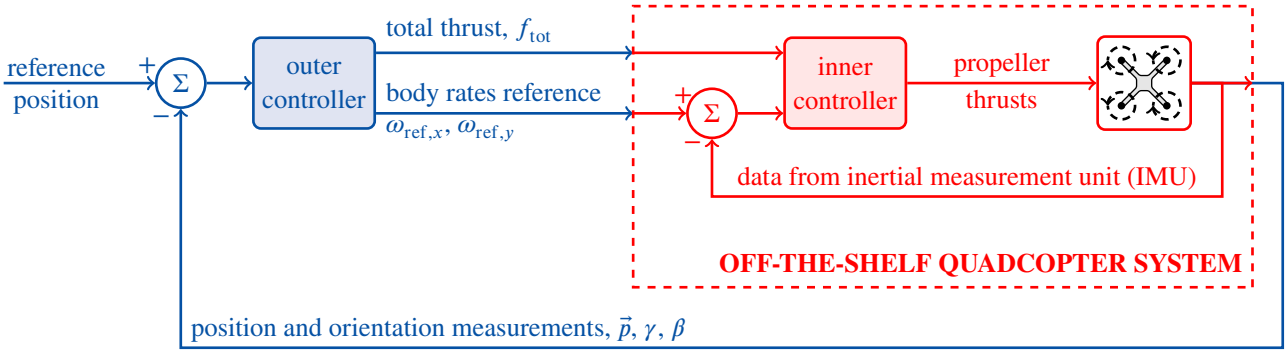


FIGURE 2 Block diagram of the cascaded control architecture used for the simulations and experiments. In an off-the-shelf quadcopter system, the *inner controller* is typically already implemented. Here we focus on the synthesis of the *outer controller*.

2.2 | Problem statement

Let us consider a discretized version of the quadcopter dynamics (1), which we denote by

$$\begin{aligned} x(t+1) &= f(x(t), u(t)), \\ y(t) &= h(x(t), u(t)), \end{aligned} \tag{2}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, and $y(t) \in \mathbb{R}^p$ are respectively the state, control input, and output at time $t \in \mathbb{Z}_{\geq 0}$. Note that even though the continuous-time dynamics (1) are known, an analytic expression does not exist for the nonlinear discretized dynamics described by mappings $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ in (2). We purposefully abstract notation above to highlight the fact that the problem statement is not unique to a quadcopter, but can be applied to any system with nonlinear dynamics. For the quadcopter, we have that, $u(t) = (f_{\text{tot}}, \omega_{\text{ref},x}, \omega_{\text{ref},y}) \in \mathbb{R}^3$, and $y(t) = (p_x, p_y, p_z, \gamma, \beta) \in \mathbb{R}^5$.

We consider the problem of constrained finite-horizon optimal control. Given the current time $t \in \mathbb{Z}_{\geq 0}$, a time horizon $T_{\text{f}} \in \mathbb{Z}_{\geq 0}$, input and output constraint sets $\mathcal{U} \subseteq \mathbb{R}^m$, $\mathcal{Y} \subseteq \mathbb{R}^p$, the goal is to design a sequence of admissable control inputs $\{u(t+i)\}_{i=0}^{T_{\text{f}}-1} \subset \mathcal{U}$ such that when applied to system (2), the resulting outputs $\{y(t+i)\}_{i=0}^{T_{\text{f}}-1} \subset \mathbb{R}^p$ lie in the constraint set $\mathcal{Y}$ and minimize the stage costs given by cost function $c : \mathbb{R}^p \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$. More formally, we wish to solve the following optimization problem:

$$\begin{aligned} \underset{u,y}{\text{minimize}} \quad & \sum_{i=0}^{T_{\text{f}}-1} c(y(t+i), u(t+i)) \\ \text{subject to} \quad & x(t+i+1) = f(x(t+i), u(t+i)), \ \forall i \in \{0, \dots, T_{\text{f}}-1\} \\ & y(t+i) = h(x(t+i), u(t+i)), \ \forall i \in \{0, \dots, T_{\text{f}}-1\} \\ & u(t+i) \in \mathcal{U}, \ y(t+i) \in \mathcal{Y}, \ \forall i \in \{0, \dots, T_{\text{f}}-1\} \\ & x(t) = \hat{x}(t), \end{aligned} \tag{3}$$

where $\hat{x}(t)$ is an estimate of the state at time t, typically computed by "filtering" the sequence of past inputs and outputs. Problem (3) is solved in a receding horizon fashion and is widely known as output MPC. The cost function c can be designed by the user to attain various control objectives (e.g., regulation, or trajectory tracking).

Without knowledge of system (2), solving problem (3) is no longer possible as we are unable to predict forward trajectories of the system, and estimate the current state $x(t)$. To resolve these issues, we approach the problem in a data-driven manner. In particular, we use the data-enabled predictive control (DeePC) algorithm[18], which replaces the constraints requiring system knowledge by raw input/output data to solve an optimization problem similar to (3), and, under assumptions to be recalled next, directly equivalent to (3).

2.3 | Data-Enabled Predictive Control (DeePC)

DeePC for deterministic LTI systems

The DeePC algorithm has been shown to be an equivalent data-driven method for solving (3) when the unknown system (2) is a deterministic LTI minimal realization, i.e., when the dynamics in (2) are of the form

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{4}$$

where A, B, C, D are matrices of appropriate dimensions. Note that (4) being a minimal realization implies controllability and observability properties of the system. Several modifications have also been proposed for robustifying the algorithm against stochastic disturbances[18]. We first introduce the necessary preliminaries, then recall the DeePC algorithm as applied to LTI systems of the form (4), followed by the robustifying regularizations that allows the algorithm's adaptation for the nonlinear quadcoptor system (2) with noisy measurements.

Let the Hankel operator which maps a sequence of signals $u = \{u(i)\}_{i=1}^{T} \subset \mathbb{R}^m$ to a Hankel matrix with $L \in \mathbb{Z}_{>0}$ block rows be denoted by

$$\mathscr{H}_L(u) := \begin{pmatrix} u(1) & u(2) & \dots & u(T-L+1) \\ u(2) & u(3) & \dots & u(T-L+2) \\ \vdots & \vdots & \ddots & \vdots \\ u(L) & u(L+1) & \dots & u(T) \end{pmatrix}.$$

Definition 1. (Persistency of Excitation): Let $L \in \mathbb{Z}_{>0}$. The sequence of signals $u = \{u(i)\}_{i=1}^{T} \subset \mathbb{R}^m$ is called *persistently exciting of order* L if the Hankel matrix $\mathscr{H}_L(u)$ has full row rank.

Note that the property of being persistently exciting of order L requires the length of the sequence of signals be large enough; in particular, the length must be such that $T \geq (m+1)L - 1$. Intuitively, a persistently exciting sequence of signals must be *sufficiently long* and *sufficiently rich* to excite all aspects of the dynamics (4). The DeePC algorithm relies on the following fundamental result.

Theorem 1. (Theorem 1[15]): Let $T_\mathrm{d}, L \in \mathbb{Z}_{>0}$. Let $(u_\mathrm{d}, y_\mathrm{d}) = \{(u_\mathrm{d}(i), y_\mathrm{d}(i))\}_{i=1}^{T_\mathrm{d}}$ be a trajectory of (4) of length T_d such that $\{u_\mathrm{d}(i)\}_{i=1}^{T_\mathrm{d}}$ is persistently exciting of order $L+n$. Then $(u, y) = \{(u(i), y(i))\}_{i=1}^{L}$ is a trajectory of (4) if and only if there exists $g \in \mathbb{R}^{T_\mathrm{d}-L+1}$ such that

$$\begin{pmatrix} \mathscr{H}_L(u_\mathrm{d}) \\ \mathscr{H}_L(y_\mathrm{d}) \end{pmatrix} g = \begin{pmatrix} u \\ y \end{pmatrix}.$$

The result above states that the subspace spanned by the columns of the Hankel matrix $\begin{pmatrix} \mathscr{H}_L(u_\mathrm{d}) \\ \mathscr{H}_L(y_\mathrm{d}) \end{pmatrix}$ corresponds exactly to the subspace of possible trajectories of (4). Hence, the Hankel matrix may serve as a non-parametric model for (4), one that is simply constructed from raw time-series data and does not require any learning.

In what follows, we will see how the above theorem allows us to perform implicit state estimation as well as predict forward trajectories of the unknown system allowing us to solve an optimization problem equivalent to (3) when the system is of the form (4).

Data collection: Let $T_\mathrm{d}, T_\mathrm{ini} \in \mathbb{Z}_{>0}$ be the length of data collection and the time horizon used for initial condition estimation, respectively. Suppose $(u_\mathrm{d}, y_\mathrm{d}) = \{(u_\mathrm{d}(i), y_\mathrm{d}(i))\}_{i=1}^{T_\mathrm{d}}$ is a sequence of input/output measurements collected from (4) during an offline procedure. Suppose further that the input $\{u_\mathrm{d}(i)\}_{i=1}^{T_\mathrm{d}}$ is persistently exciting of order $T_\mathrm{ini} + T_\mathrm{f} + n$. We partition the input/output measurements into Hankel matrices

$$\begin{pmatrix} U_\mathrm{p} \\ U_\mathrm{f} \end{pmatrix} := \mathscr{H}_{T_\mathrm{ini}+T_\mathrm{f}}(u_\mathrm{d}), \quad \begin{pmatrix} Y_\mathrm{p} \\ Y_\mathrm{f} \end{pmatrix} := \mathscr{H}_{T_\mathrm{ini}+T_\mathrm{f}}(y_\mathrm{d}), \tag{5}$$

where U_p consists of the first T_ini block rows of $\mathscr{H}_{T_\mathrm{ini}+T_\mathrm{f}}(u_\mathrm{d})$ and U_f consists of the last T_f block rows of $\mathscr{H}_{T_\mathrm{ini}+T_\mathrm{f}}(u_\mathrm{d})$ (similarly for Y_p and Y_f). The data in U_p and Y_p will be used in conjunction with *past* data to perform implicit initial condition estimation, and the data in U_f and Y_f will be used to predict *future* trajectories.

Data-driven control and estimation: Let $(u_\mathrm{ini}, y_\mathrm{ini}) = \{(u_\mathrm{ini}(t-i), y_\mathrm{ini}(t-i))\}_{i=T_\mathrm{ini}}^{1}$ be the T_ini most recent past input/output

measurements from the system. By Theorem 1, $(u, y) = \{u(t+i), y(t+i)\}_{i=0}^{T_\mathrm{f}-1}$ is a possible future trajectory of (4) if and only if there exists $g \in \mathbb{R}^{T_\mathrm{d}-T_\mathrm{ini}-T_\mathrm{f}+1}$ satisfying

$$\begin{pmatrix} U_\mathrm{p} \\ Y_\mathrm{p} \\ U_\mathrm{f} \\ Y_\mathrm{f} \end{pmatrix} g = \begin{pmatrix} u_\mathrm{ini} \\ y_\mathrm{ini} \\ u \\ y \end{pmatrix}. \tag{6}$$

Every column of the Hankel matrix is a trajectory of the system (motion primitive), and any new trajectory (right-hand side of (6)) can be synthesized by a linear combination of these motion primitives. Hence, given an input sequence u to be applied to the system, one can solve the first three block equations of (6) for g, and the corresponding output sequence is given by $y = Y_\mathrm{f} g$. The top two block equations in (6) are used to implicitly fix the initial condition from which the future trajectory departs. To uniquely fix the initial condition from which the future trajectory departs, one must set $T_\mathrm{ini} \geq \ell$, where ℓ is the lag of the system (i.e., the number of past measurements required to uniquely identify the current state of the system through back-propogation of the dynamics (4)). This in turn implies that the predicted trajectory given by $y = Y_\mathrm{f} g$ is unique[16]. Note that the lag ℓ of the system is a priori unknown, but is upper bounded by n. Hence, knowing an upper bound on the state dimension n of the system is sufficient to obtain unique predictions.

The Hankel matrix in (6) simultaneously performs state estimation and prediction, and can thus be used as a predictive model for system (4). Substituting (6) for the unknown dynamics (4) in the optimization problem (3) gives rise to the following data-driven optimization problem allowing for the computation of optimal control inputs without knowledge of a system model:

$$\begin{aligned} \underset{u,y,g}{\text{minimize}} \quad & \sum_{i=0}^{T_\mathrm{f}-1} c(y(t+i), u(t+i)) \\ \text{subject to} \quad & \begin{pmatrix} U_\mathrm{p} \\ Y_\mathrm{p} \\ U_\mathrm{f} \\ Y_\mathrm{f} \end{pmatrix} g = \begin{pmatrix} u_\mathrm{ini} \\ y_\mathrm{ini} \\ u \\ y \end{pmatrix} \\ & u \in \mathcal{U}^{T_\mathrm{f}}, \; y \in \mathcal{Y}^{T_\mathrm{f}}, \end{aligned} \tag{7}$$

where $\mathcal{U}^{T_\mathrm{f}}$ is the T_f-fold cartesian product of $\mathcal{U}$ (similarly for $\mathcal{Y}^{T_\mathrm{f}}$). The optimization problem (7) was shown to be equivalent to the MPC problem given in (3) when the unknown system is of the form (4)[18].

Regularized DeePC for nonlinear noisy systems

The goal of this paper is to implement the above DeePC optimization problem to control a real-world quadcopter described above in Section 2.1. As the quadcopter dynamics do not satisfy the deterministic LTI assumption necessary to show the equivalence of the MPC optimization problem (3) and the DeePC optimization problem (7), regularizations are needed. Indeed, when the input/output data used for the Hankel matrix in (7) is obtained from a nonlinear system or is corrupted by process or measurement noise (as is the case with any real-world application) the subspace spanned by the columns of the Hankel matrix no longer coincides with the subspace of possible trajectories of the system. In fact, in any real-world problem setting the Hankel matrix used for predictions in (7) will generally be full rank. Hence, the Hankel matrix constraint will imply that any trajectory is possible leading to poor closed-loop performance of the DeePC algorithm. Furthermore, the online measurements y_ini used to set the initial condition from which the predicted trajectory departs are corrupted by measurement noise, and thus may cause poor predictions. Including a 2-norm penalty on the difference between the estimated initial condition $Y_\mathrm{p} g$ and the measured initial condition y_ini coincides roughly with a least-square estimate of the true initial condition.

Regularization has been proposed as one method to deal with these difficulties and extend the DeePC algorithm to nonlinear noisy systems[14]. We present a variation of these regularizations in the following *regularized DeePC* optimization problem

$$\begin{aligned} \underset{u,y,g}{\text{minimize}} \quad & \sum_{i=0}^{T_\mathrm{f}-1} c(y(t+i), u(t+i)) + \lambda_s \|Y_\mathrm{p} g - y_\mathrm{ini}\|_2^2 + r(g) \\ \text{subject to} \quad & \begin{pmatrix} U_\mathrm{p} \\ U_\mathrm{f} \\ Y_\mathrm{f} \end{pmatrix} g = \begin{pmatrix} u_\mathrm{ini} \\ u \\ y \end{pmatrix} \\ & u \in \mathcal{U}^{T_\mathrm{f}}, \; y \in \mathcal{Y}^{T_\mathrm{f}}, \end{aligned} \tag{8}$$

where $\lambda_s \geq 0$, and $r\colon \mathbb{R}^{T_\mathrm{d}-T_\mathrm{ini}-T_\mathrm{f}+1} \to \mathbb{R}_{\geq 0}$ is a function used to regularize g. Algorithm 1 below summarizes the DeePC procedure where (8) is implemented in a receding horizon fashion.

Algorithm 1 Regularized DeePC

Input: T_d, T_ini, T_f, cost function c, λ_s, constraint sets $\mathcal{U}$ and $\mathcal{Y}$, regularization function r, data sequence $\{(u_\mathrm{d}(i), y_\mathrm{d}(i))\}_{i=1}^{T_\mathrm{d}}$, the T_ini most recent past input/output measurements $(u_\mathrm{ini}, y_\mathrm{ini})$.

1. Set $g^\star$ equal to the solution of (8).
2. Compute the optimal input sequence $u^\star = U_\mathrm{f} g^\star$.
3. Apply input $(u(t), \dots, u(t+s)) = (u_0^\star, \dots, u_s^\star)$ for some $s \leq T_\mathrm{f} - 1$.
4. Set t to $t+s$ and update u_ini and y_ini to the T_ini most recent past input/output measurements.
5. Return to 1.

It has been shown that when $r(g) = \lambda_g \|g\|_q$, where $\lambda_g \geq 0$ and $q \in \mathbb{Z}_{>0} \cup \{+\infty\}$, problem (8) coincides with a distributionally robust problem formulation. Using such a q-norm regularization for the decision variable g induces robustness to all systems (nonlinear or stochastic) that could have produced the data in the Hankel matrices (5) within an s-norm induced Wasserstein ball around the data samples used, where $\frac{1}{q} + \frac{1}{s} = 1$ [18].

The computational complexity of (8) can be characterized by the number of decision variables and constraints. There are $\big((m+p)T_\mathrm{f}\big) + \big(T_\mathrm{d} - T_\mathrm{ini} - T_\mathrm{f} + 1\big)$ decision variables, $mT_\mathrm{ini} + (m+p)T_\mathrm{f}$ equality constraints, and $2(m+p)T_\mathrm{f}$ input/output constraints, when $\mathcal{U}^{T_\mathrm{f}}$ and $\mathcal{Y}^{T_\mathrm{f}}$ are box constraint sets. As is expected of a finite-horizon optimal control method, the computational complexity grows with the time horizon T_f. Furthermore, T_ini and T_d also affect the computational complexity. The former is related to the observability of the unknown system (2), the latter to the system's dimensionality.

3 | RESULTS

In this section, we present the results and insights gained by applying DeePC Algorithm 1 described in Section 2.3 for trajectory tracking of the quadcopter system described in Section 2.1. The challenges posed by this application are:

1. The nonlinear and stochastic nature of the quadcopter system requires that the regularization function in (8) and the other hyperparameters offered by the DeePC Algorithm 1 be chosen appropriately for the application at hand. This is addressed by the simulation-based analysis in Section 3.2.
2. It is not clear that simulation-based parameter selection can be directly transferred to real-world experiments, mainly due to unmodelled dynamics, delays in actuation, communication or sensing, and noise. This is addressed by the experimental results in Section 3.3.

The real-world results were collected from laboratory experiments conducted using a motion capture system to provide measurements of the position and orientation of the quadcopter at a frequency of 25Hz. Thus, the sampling time in the discrete-time dynamics (2) is 40ms. The laboratory setup was developed as part of a previous work [27]. To provide the reader with an idea for the scale of the setup, the Crazyflie 2.0 [28] quadcopter weighs 28 grams and a 12 cubic meter flying space was available. Further details on the setup are given in Section 3.3 where the experimental results are presented. The simulation environment uses the model presented in Section 2.1 and the model parameters identified in a previous work [29]. These model parameters do not match the specific Crazyflie 2.0 used for the experiments, partially due to additional hardware required for detection by the motion capture system.

3.1 | Data collection

As described in Section 2.3, the input signal used in the Hankel matrices appearing in (7) must be persistently exciting of sufficient order. This data can be collected by injecting a random input sequence, or by performing a manual flight experiment where a human performs the function of the outer controller. For repeatability of results, we chose the former. Two possible choices of random input signals to be applied during the data collection phase are a pseudorandom binary sequence (PRBS) designed for multiple inputs[30], or a white noise signal. Both types of perturbations were tested in simulations and showed a negligible difference in the performance of the DeePC algorithm. The results in this paper are presented using a PRBS input signal during the data collection phase because it generally provides better performance for classical system identification techniques (see, e.g.,[31] Chapter 13). The input signals applied for data collection consist of the PRBS excitation signal added to an existing controller that maintains the quadcopter around the hover state. The data collected was used to populate the Hankel matrices in (5).

3.2 | Simulation-based analysis and insights

The aim of our controller is to track a steady state reference $(u_r, y_r) \in \mathbb{R}^m \times \mathbb{R}^p$. We therefore consider as the cost function c the quadratic tracking error between the prediction and the given steady state reference, i.e.,

$$c(y, u) = (y - y_r)^T Q(y - y_r) + (u - u_r)R(u - u_r), \tag{9}$$

where $Q \succeq 0$, $R \succ 0$. The values chosen are given in Appendix A. The time horizon was chosen as $T_\mathrm{f} = 25$ which corresponds to 1 second in real time. Furthermore, we choose the regularization function in (8) as the following:

$$r(g) = \lambda_g \, \|g - g_r\|_q, \qquad \text{with} \qquad g_r = \begin{pmatrix} U_\mathrm{p} \\ Y_\mathrm{p} \\ U_\mathrm{f} \\ Y_\mathrm{f} \end{pmatrix}^\dagger \begin{pmatrix} \mathbb{1}_{T_\mathrm{ini}} \otimes u_r \\ \mathbb{1}_{T_\mathrm{ini}} \otimes y_r \\ \mathbb{1}_{T_\mathrm{f}} \otimes u_r \\ \mathbb{1}_{T_\mathrm{f}} \otimes y_r \end{pmatrix}. \tag{10}$$

where $\lambda_g \geq 0$, $q \in \mathbb{Z}_{>0} \cup \{+\infty\}$, the vector $\mathbb{1}_{T_\mathrm{ini}} \otimes u_r$ denotes the stacked column vector consisting of T_ini copies of u_r (similarly for $\mathbb{1}_{T_\mathrm{ini}} \otimes y_r$), and $\dagger$ denotes the pseudoinverse. The vector g_r in the above can be thought of as a "steady-state trajectory mapper" which linearly combines columns of the Hankel matrix to match the given steady-state reference trajectory. Among the possibly infinite number of vectors g that match the steady state, this is the one with the smallest 2-norm. In the case when there is no g that matches the steady state, g_r matches it as closely as possible in the 2-norm sense. However, this case is unlikely in practice since the Hankel matrix is generally full rank as discussed above. Penalizing the difference between g and g_r ensures that the stage cost in (8) is zero when the quadcopter is at the steady-state reference (u_r, y_r).

Under these design choices, the regularized DeePC optimization problem (8) offers several hyperparameters given by:

- T_d, the total number of data points used to construct the Hankel matrices in (5),
- T_ini, the time horizon used for initial condition estimation,
- λ_s, the weight on the softened initial condition constraint,
- λ_g, the weight on the regularization of g,
- q, the norm used to regularize g in (10), and
- p, the number of outputs used to construct the Hankel matrices in (5).

Although p may seem fixed by the output measurements available, in the case of quadcopter control, it is reasonable to consider whether to use all measurements for position control, i.e., set $p = 5$, or use only the position measurements, i.e., set $p = 3$. Note that if one were to approach the control problem through system identification followed by MPC, a number of hyperparameters would also need to be selected. To investigate the effect of the hyperparameters for DeePC, we perform a grid search over the ranges

$$T_\mathrm{ini} \in \{2, \dots, 10\}, \quad \lambda_s \in [10^5, 10^{10}], \quad \lambda_g \in [10^0, 10^8], \quad q = \{1, 2\}, \quad p = \{3, 5\}, \tag{11}$$

and a range of T_d values that satisfy the minimum data length prescribed by the persistency of excitation requirement from Definition 1. Note that the prediction horizon T_f, and the cost matrices Q and R are not parameters unique to the regularized

DeePC optimization problem (8), but are also parameters for MPC. For the sake of clarity we do not consider them as hyperparameters in the simulation-based analysis. Moreover, fixing $T_{\mathrm{f}} = 25$, and Q and R as in Appendix A, was sufficient for achieving good closed-loop performance, and allows for a focus on the hyperparameters unique to the DeePC. For each combination of hyperparameters the following procedure is carried out in simulation. The same procedure is used for the real-world experiments presented in Section 3.3.

Procedure 1. (Procedure for collecting results in simulation and real-world experiments): For simulation, the system used was a model of the off-the-shelf quadcopter system with dynamics (1) and architecture as in Figure 2, where measurements were affected by zero-mean Gaussian noise with covariance matrix Σ_y as in Appendix A. For the real-world experiments, the system used was the Crazyflie 2.0.

1. The quadcoptor is brought to hover at $y = (0, 0, 1)$ with a stabilizing controller. The system is excited by adding a PRBS signal to the output of the stabilizing controller, as per Section 3.1, for the input/output data collection step of the DeePC algorithm.

2. The regularized DeePC optimization problem (8) is setup with the input/output data collected in step 1.

3. The DeePC controller is turned on and the quadcoptor is commanded to track a diagonal step up from $y(0) = (-0.5, -0.5, 0.5)$ to $y_r = (0.5, 0.5, 1.5)$.

4. The resulting closed-loop *tracking error* is measured as $\sum_{t=0}^{T_e-1} \|y(t) - y_r\|_2^2$, where $t = 0$ is the time index at the start of the step trajectory and $T_e = 250$ is the chosen experiment length, which corresponds to 10 seconds in real time.

Sensitivity to T_{d} and T_{ini}

As discussed in Section 2.3, for LTI systems the DeePC algorithm requires a minimum number of data points to satisfy the persistency of excitation property. Since we apply the DeePC algorithm to a nonlinear system subject to measurement noise, it becomes unclear as to how many data points are needed in order to construct the Hankel matrices in (5). Figure 3 shows the sensitivity analysis of T_{d} and T_{ini} on the tracking error. Figure 3 (left) shows the influence of T_{d} on the tracking error, where for each value of T_{d} considered we show the smallest tracking error achieved over all combinations of the other hyperparameters in the grid given by (11) with $T_{\mathrm{ini}} = 6$. Similarly, Figure 3 (right) shows the influence of T_{ini} on the tracking error, where for each value of T_{ini} considered we show the smallest tracking error achieved over all combinations of the other hyperparameters in the grid given by (11) with $T_{\mathrm{d}} = 331$.

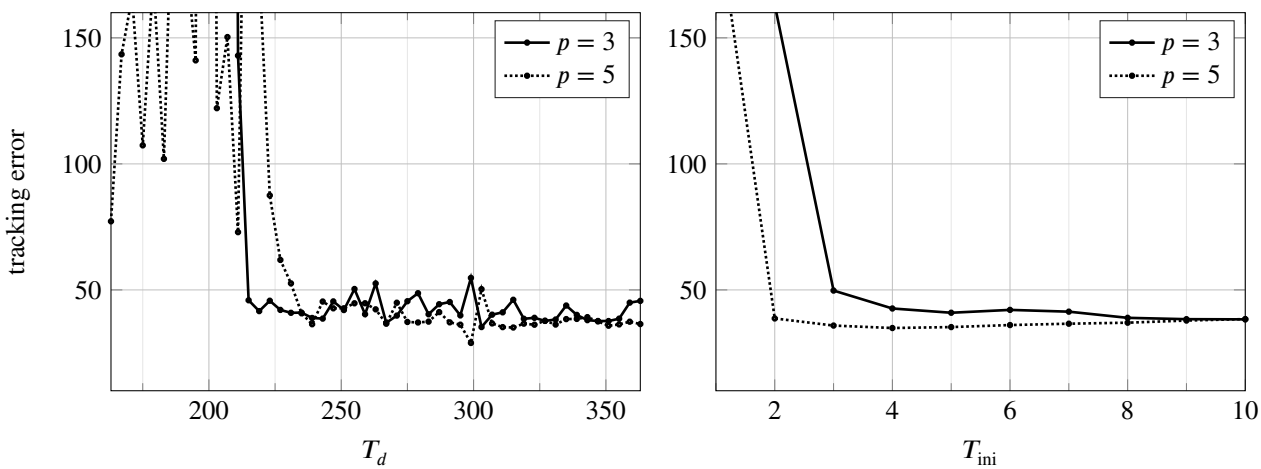


FIGURE 3 Influence of T_{d} (left) and T_{ini} (right) on the tracking error. For each point plotted, the tracking error is the minimum achieved over all other hyperparameter combinations considered, with $T_{\mathrm{ini}} = 6$ for the left-hand plot, and $T_{\mathrm{d}} = 331$ for the right-hand plot. Evaluating the expression in (12), the Hankel matrix becomes square at $T_{\mathrm{d}} = 223$ for $p = 3$ and at $T_{\mathrm{d}} = 287$ for $p = 5$.

The key insight from the grid search result in Figure 3 (left) is the distinct improvement in the tracking error of the regularized DeePC algorithm when the number of data points is chosen such that the Hankel matrix appearing in the DeePC optimization problem (8) has at least as many columns as rows. Since the Hankel matrix is generally full rank when the data is obtained from a nonlinear noisy system (and thus its columns span $\mathbb{R}^{(m+p)(T_{\text{ini}}+T_{\text{f}})}$), having a square Hankel matrix ensures that the subspace spanned by its columns contains the actual subspace of possible trajectories of the system. When the Hankel matrix is slim (i.e., has less columns than rows), this property may not hold; the subspace spanned by the columns of a slim Hankel matrix may not contain the subspace of possible trajectories of the system. This insight is summarized as the following inequality which states that T_{d} should be chosen to be larger than both the minimum amount needed for persistency of excitation in the LTI case and the minimum amount such that the Hankel matrix in (8) is square

$$T_{\text{d}} \geq \max\left\{(m+1)(T_{\text{ini}}+T_{\text{f}}+n)-1, (m+p+1)(T_{\text{ini}}+T_{\text{f}})-1\right\}. \tag{12}$$

Here $n = 8$ is the number of states corresponding to a minimal realization of (1) linearized about hover. Note that the minimum number of data points such that the Hankel matrix in (8) is square is directly affected by the number of outputs p. Hence, a larger p requires more data points to satisfy the lower bound in (12) and thus results in more decision variables in problem (8).

A similar trend is observed in Figure 3 (right) for T_{ini} where good tracking performance is achieved for values larger than $T_{\text{ini}} = 2$ for $p = 5$, and $T_{\text{ini}} = 3$ for $p = 3$. This suggests that more past measurements are needed to estimate the initial condition of the unknown system when $p = 3$. We observed, however, that setting $T_{\text{ini}} = 6$ gives steadier flight of the quadcopter. Under noisy measurements, increasing T_{ini} leads to better initial condition estimates. For the remaining results (simulation and experimental), Procedure 1 was conducted with the number of data points $T_{\text{d}} = 331$ and with $T_{\text{ini}} = 6$. This resulted in good tracking error performance for both $p = 3$ and $p = 5$, while keeping the size of the DeePC optimization problem (8) small enough to be computationally tractable in real-time.

Sensitivity to λ_s, λ_g, q, and p

Figure 4 shows the results from the grid search as a heat map over (λ_s, λ_g) with fixed values of $q = 2$ and $p = 3$ for the purpose of visualization, and fixed value of $T_{\text{d}} = 331$ and $T_{\text{ini}} = 6$ for the reasons described above. The figure provides the insight that there is a threshold for λ_s (approximately $\lambda_s \geq 10^7$) beyond which small tracking error can be achieved. The intuitive explanation for this insight is that a large enough penalization on the softened initial condition constraint ensures that the future predicted trajectory departs from an initial condition close to the actual initial condition.

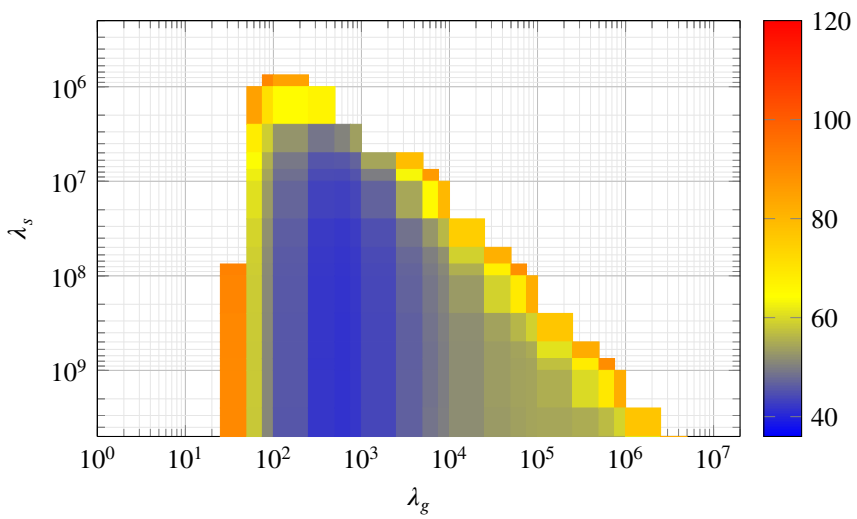


FIGURE 4 Influence of λ_g and λ_s on the tracking error. All other parameters are fixed to the values described in the text. The coloured shading is restricted to the interval (36, 120) to sufficiently display the shape of the region shown. The cost increases steeply in regions where the cost is greater than 120, thus the plot is clipped for values greater than 120 for the sake of clarity.

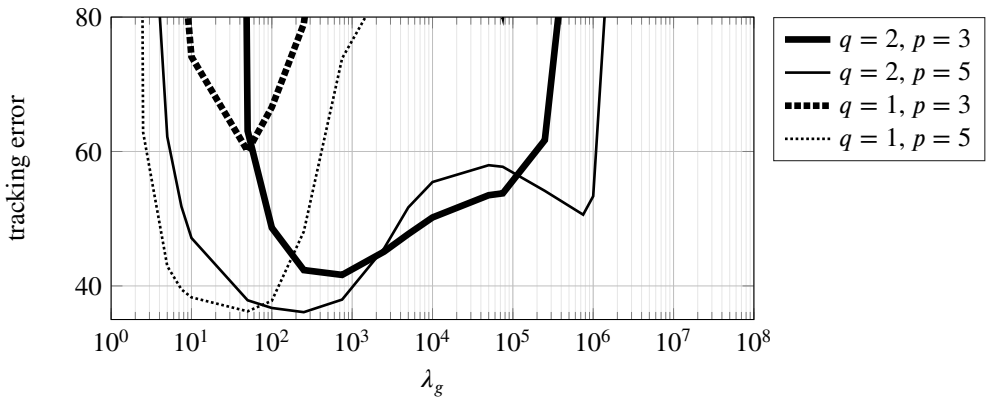


FIGURE 5 Influence of λ_g, q, and p on the tracking error with the fixed value of $\lambda_s = 7.5 \times 10^8$. Hence for the combination $q = 2$, $p = 3$ (solid thick line) this is the respective slice of Figure 4. The main observation is that the choice $q = 2$, i.e., a 2-norm regularization on decision variable g, provides a wider range of λ_g for which acceptable tracking error is achieved.

Figure 4 also exposes a range for λ_g in which small tracking error is achieved. To investigate this further we consider the grid search results for all combinations of $q \in \{1, 2\}$ and $p \in \{3, 5\}$. Figure 5 shows the results from the grid search over λ_g for a fixed value of $\lambda_s = 7.5 \times 10^8$ and for all four combinations of q and p, e.g., the line for $q = 2$, $p = 3$, is the slice of Figure 4 at the fixed value of λ_s. In all cases a small tracking error is achieved for a range of λ_g, although the combination $q = 1$, $p = 3$ performs relatively poorly. This range of λ_g with acceptable tracking error is wider for $q = 2$ than for $q = 1$, which suggests that for the setup under consideration, 2-norm regularization is less sensitive to hyperparameter selection than 1-norm regularization. This observation is supported by observing the heat maps for all four combinations $q \in \{1, 2\}$ and $p \in \{3, 5\}$ as provided in Appendix B. Based on these insights, for the remainder of the results we fix the values $\lambda_s = 7.5 \times 10^8$ and $q = 2$ and now investigate in more detail the influence of λ_g and the choice of output measurements $p \in \{3, 5\}$.

To provide some intuition for how λ_g influences the optimal solution of the regularized DeePC optimization problem (8) we now take a closer look at the closed loop trajectories resulting from $\lambda_g \in \{0, 500\}$. Figure 6 (a,b) shows the p_z coordinate of the

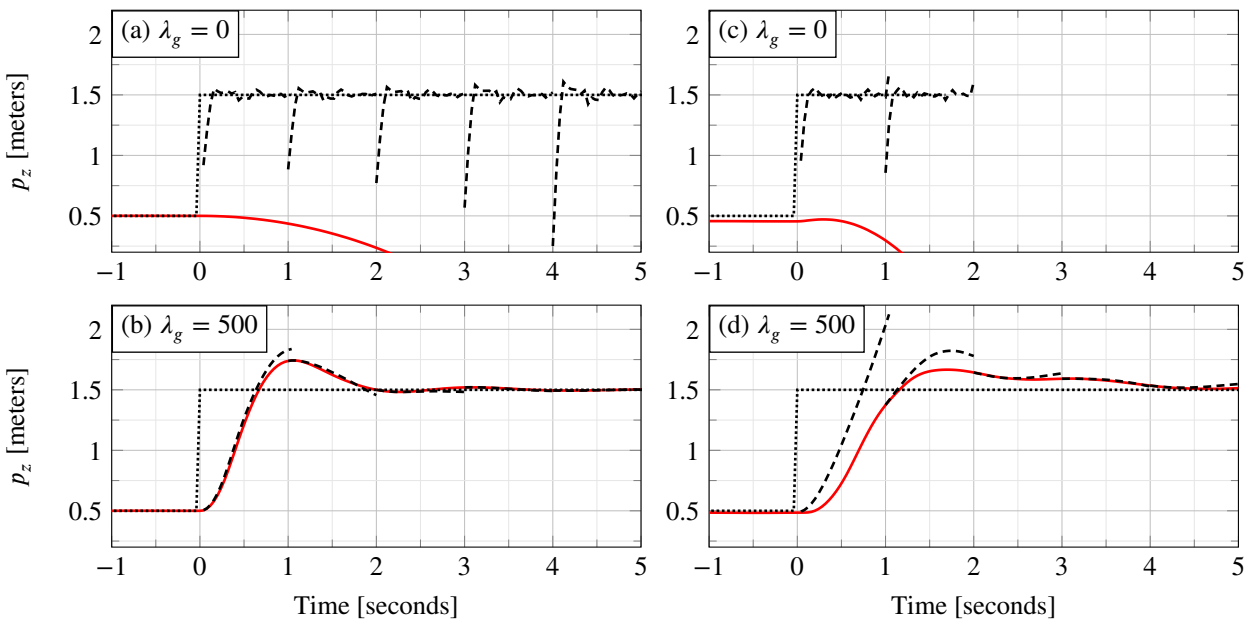


FIGURE 6 Actual trajectories (solid) versus predicted trajectories from optimization problem (8) (dashed). (a,b) are simulated results and (c,d) are experimental results. The top plots (a,c) are for $\lambda_g = 0$, and the bottom plots (b,d) are for $\lambda_g = 500$.

simulated closed loop trajectory over time (solid line), the reference y_r (dotted line), and the trajectory predicted by problem (8) at representative time instants (dashed line).

In the case of no regularization (Figure 6 (a), $\lambda_g = 0$), the predictions do not correspond to the physics of the model and the actual position diverges, i.e., the quadcopter crashes. Since the data used in the Hankel matrix in (8) is obtained from a nonlinear system and is corrupted by measurement noise, then the subspace spanned by the columns of the Hankel matrix is all of $\mathbb{R}^{(m+p)(T_{\text{ini}}+T_{\text{f}})}$. Hence, without regularization on the decision variable g, the Hankel matrix predicts that every trajectory is possible. The value $\lambda_g = 500$ is selected from the grid search result where the DeePC algorithm achieved the smallest tracking error (see Figure 5). We see in Figure 6 (b) that desirable reference tracking is achieved and that more physical predictions are computed by the regularized optimization problem (8).

An important distinction between the λ_g hyperparameter and the T_{d}, T_{ini} hyperparameters discussed above, is that the λ_g regularization cannot be arbitrarily increased, shown also in Figure 5. The reason is that at a certain level the regularization term $r(g)$ in (8) dominates the tracking error term, leading to poor tracking performance and eventually instability of the system. However, the range of λ_g resulting in small tracking error is large (e.g., $\lambda_g \in [100, 10000]$ for $q = 2, p = 3$ in Figure 5) indicating robustness to the choice of λ_g.

3.3 | Real-world DeePC implementation

We now investigate how the insights gained through the simulation analysis of Section 3.2 transfer to laboratory experiments on a real-world quadcopter, with the details of the experimental setup provided at the start of Section 3. The experiments are performed as per Procedure 1 (see Section 3.2) and through the results we investigate: (a) whether the insights from the simulation-based analysis are validated in experiments; (b) whether the hyperparameter values identified from the simulation-based analysis can be directly transferred to the laboratory environment; and (c) the reliability of the tracking performance achieved.

Figure 7 provides a schematic of the laboratory setup used to collect the experimental results. The motion capture system consists of multiple cameras placed around the flying space and connected to a dedicated computer. The software running on the motion capture computer provides accurate measurements[26] of the position and orientation of the Crazyflie 2.0[28] quadcopter, i.e., measurements of $(\vec{p}, \gamma, \beta)$. These measurements are available to an offboard laptop where the outer controller from Figure 2 is implemented. The control decisions of the outer controller, i.e., $(f_{\text{tot}}, \omega_{\text{ref},x}, \omega_{\text{ref},y})$, are sent via the Crazyradio link to the Crazyflie 2.0 where the firmware provided with the quadcopter runs an onboard controller to track these.

Figure 6 (c,d) shows the p_x coordinate of the closed loop trajectory, reference, and DeePC predictions when implemented on the quadcopter using the same hyperparameter values as Figure 6 (a,b) respectively. The main feature of Figure 6 is that the simulation and experimental results show qualitatively similar closed-loop trajectories (solid lines) and predictions computed by the DeePC optimization problem (8) (dashed lines). This provides experimental validation of the insight that regularization is required to predict physically reasonable trajectories when applying the DeePC to a real system. Moreover, a direct transfer of the hyperparameters selected via simulation to the experiments was possible, and we observed that tracking performance was not significantly improved by adjusting the regularization parameter λ_g. Appendix C provides a similar comparison for

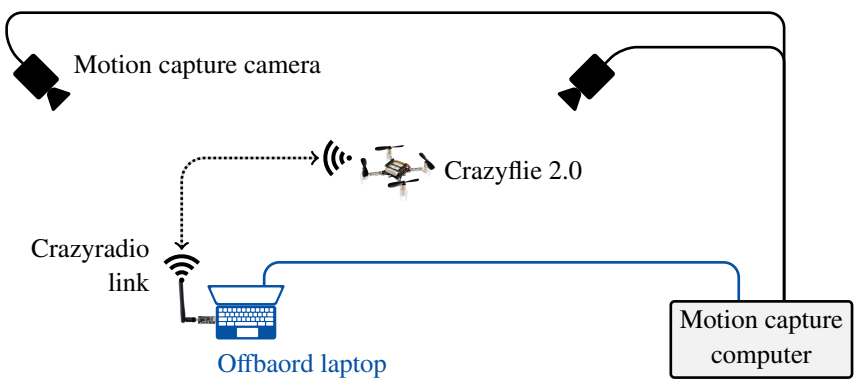


FIGURE 7 Schematic showing the laboratory setup used to collect the experimental results described in Sections 3.3 and 3.4.

hyperparameter values above and below $\lambda_g = 500$, indicating that the real-world implementation also achieves the best tracking performance at approximately $\lambda_g = 500$.

To investigate the reliability of the performance observed in Figure 6 (d), and also to investigate the influence of hyperparameter p, Procedure 1 was repeated in 28 experiments for each of $p = 3$ and $p = 5$. To capture different operating conditions, 14 trials were performed with a fully charged battery and 14 with a partially depleted battery. Figures 8 and 9 and Table 1 summarize the results. Figure 8 shows the position time series data (solid grey) of all 28 trajectories for $p = 3$ (a,b,c) and for $p = 5$ (d,e,f), with the average at each time point (dashed) shown to assist with visualization. Figure 9 shows that same data as a top view.

Quantitatively, Table 1 shows that $p = 3$ achieves a lower tracking error compared to $p = 5$, in terms of mean, median, and standard deviation. This is likely due to the orientation measurements having higher noise than the position measurements. This can be addressed by performing a weighted penalization of $Y_{\mathrm{p}}g - y_{\mathrm{ini}}$ using the covariance matrix of the measurement noise. Qualitatively, Figures 8 and 9 suggest that there is less variation in the closed loop trajectories with $p = 3$ than with $p = 5$. From the online computation persective, Table 1 shows that optimization problem (8) is solved sufficiently fast for both $p = 3$ and $p = 5$ considering that output measurements are provided for real-time implementation at 25Hz.

A video of the quadcoptor successfully tracking step trajectories and a figure 8 using the DeePC algorithm can be found here: https://polybox.ethz.ch/index.php/s/I0KKwIsudwaLj3n.

TABLE 1 Real-world experimental results comparison for $p \in \{3, 5\}$. Solve time values reported use solver OSQP[32] on a 64bit Ubunto 16.04 LTS, Intel i7-8550U, 1.8GHz, 4 Cores, 16GB memory machine.

	Tracking Error[1]			**Solve time [ms]**		
p	**mean**	**median**	**std. dev.**	**mean**	**median**	**std. dev.**
3	75	69	21	4.14	3.92	1.49
5	93	86	23	6.66	5.70	4.78

1 Computed as described in the Procedure 1.

3.4 | Real-world comparison with model-based control

The results in Section 3.3 show that DeePC Algorithm 1 achieves good performance for the step reference tracking task specified in Procedure 1 without ever constructing an explicit model of the quadcopter system being controlled. We now present a model-based point of comparison that is developed for linear systems. We take a first-principles approach that considers the linearization of the quadcopter dynamics (1) about the hover equilibrium point, and we assume that the inner controller tracks the body rates reference signal without dynamics or delays. We use a sampling time of 0.04 seconds, i.e., 25Hz, to convert the continuous-time linear model to discrete-time. The resulting linear system model can be readily derived[33]. Hence we consider a model based-controller with eight states and three inputs, $\left(\vec{p}, \dot{\vec{p}}, \gamma, \beta\right)$ and $\left(f_{\mathrm{tot}}, \omega_{\mathrm{ref},x}, \omega_{\mathrm{ref},y}\right)$ respectively.

The model-based control method we implement is output MPC, as described in Section 2.2. Optimization problem (3) is solved in a receding horizon fashion with the dynamics function f replaced by the linear-time invariant system model described above, the cost function c given by (9), and all parameters $\{T_f, Q, R, \mathcal{U}, \mathcal{Y}, u_r\}$ set to the same values as used for the DeePC as given in Appendix A. The state estimate, $\hat{x}(t)$, is constructed by directly taking the measurements for $(\vec{p}, \gamma, \beta)$, and $\dot{\vec{p}}$ is estimated as the discrete time derivative of subsequent $\vec{p}$ measurements. Figure 10 compares a trajectory of this first-principles MPC approach with that of the DeePC. Figure 10 (a) shows the time series of the vertical position p_z, and Figure 10 (b) shows the trajectory in the (p_x, p_y)-plane

Figure 10 (a) shows that DeePC and MPC achieve qualitatively similar tracking performance for the vertical position p_z. Both have a similar rise time and settling time, with the most distinct feature being that the DeePC controller overshoots the reference but then settles to a smaller steady state offset. For MPC, this offset is present because there is a model mismatch between the steady state input, u_r, and that needed to maintain the real-world quadcopter at steady state. As the DeePC controller is provided with the same u_r, this indicates that the structure of the DeePC controller is able, to some extent, to correct for a mismatch of the

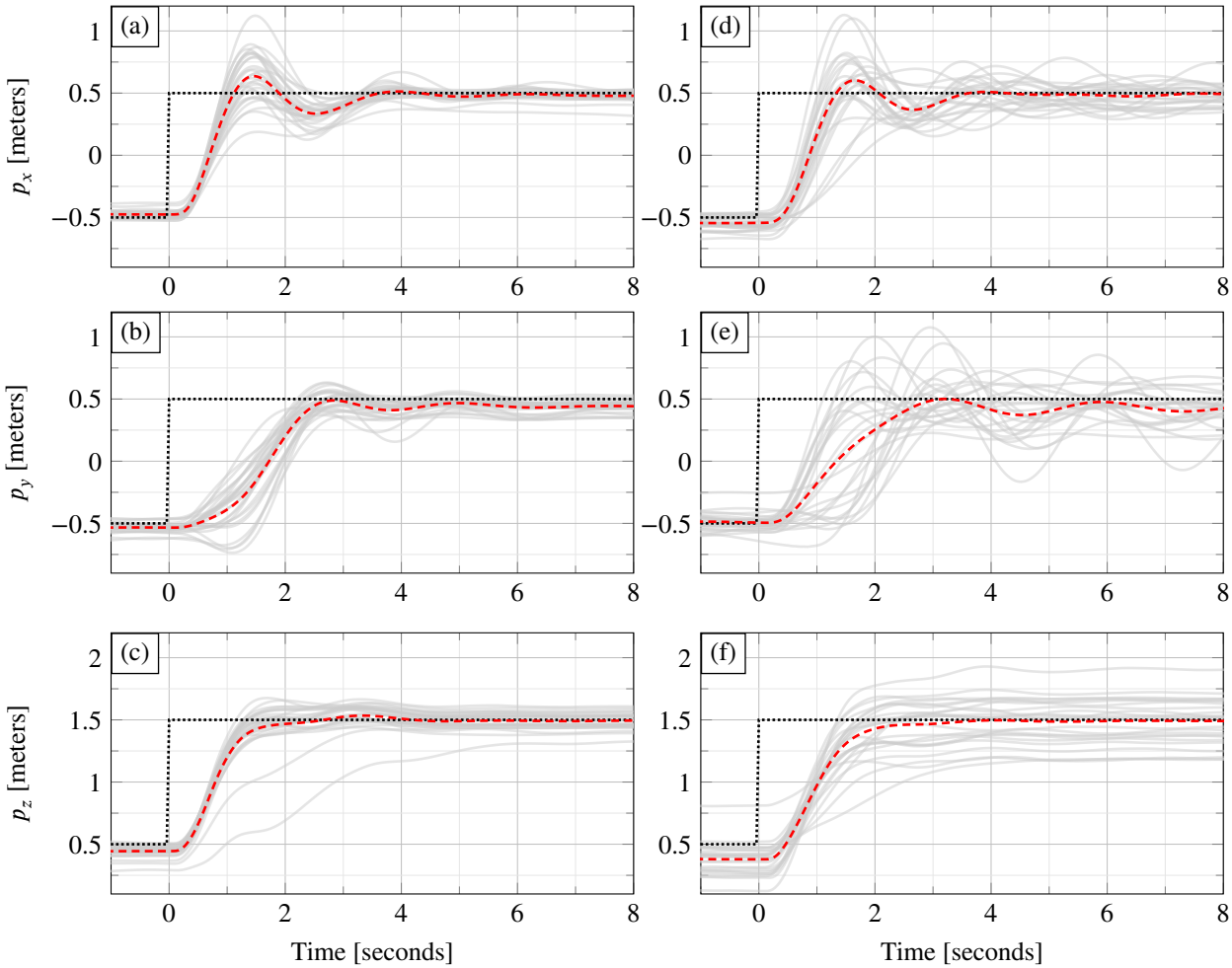


FIGURE 8 Real-world quadcopter trajectories (solid grey) for 28 experiments, each with the same change in reference signal (dotted black). Plots (a,b,c) are for $p = 3$ and plots (d,e,f) are for $p = 5$. The dashed lines show the average of the 28 experiments at each time point.

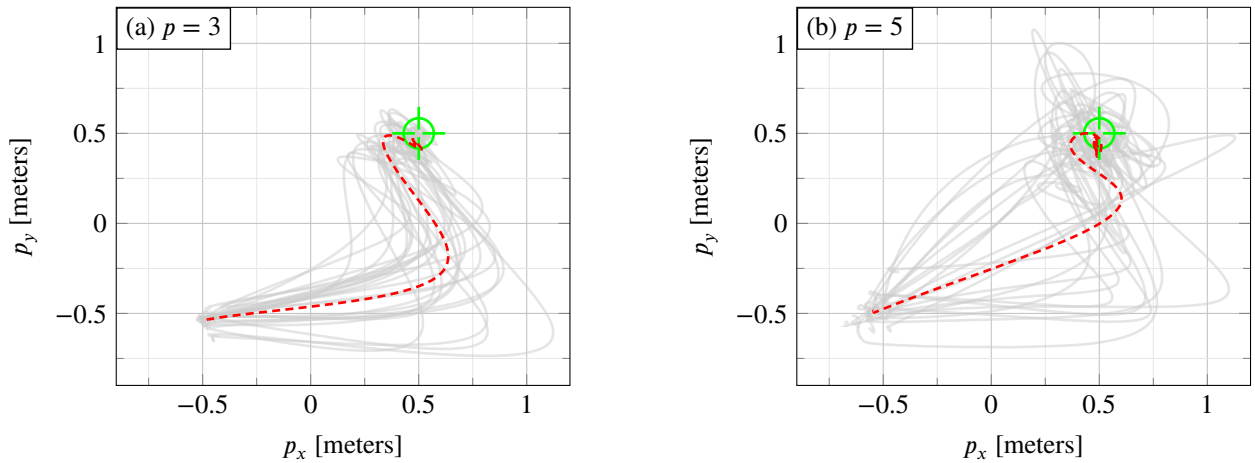


FIGURE 9 The same data as shown in Figure 8 shown as a top view on the (p_x, p_z)-plane. Plot (a) is for $p = 3$ and plot (b) is for $p = 5$. The dashed lines show the average at each time point of the 28 real-world trajectories (solid grey).

steady state input u_r provided. Figure 10 (b) shows a clear disparity between the tracking performance in the horizontal (p_x, p_y)-plane. Where the MPC follows an almost straight line trajectory from the starting point to the target, the DeePC controller by contrast has quite different tracking behaviour for the p_x and p_y directions, a trend also observed in Figure 9 and in our simulation-based tests. This leaves open an interesting direction for further investigation to understand why the DeePC controller produces a faster rise time for the p_x direction compared to the p_y direction.

Overall, for the quadcopter application we see that DeePC performs similarly to MPC where a first-principles model is available. This indicates the potential for DeePC to tackle applications where a first-principles model is either not available or identifying all the necessary model parameters is not conceivable.

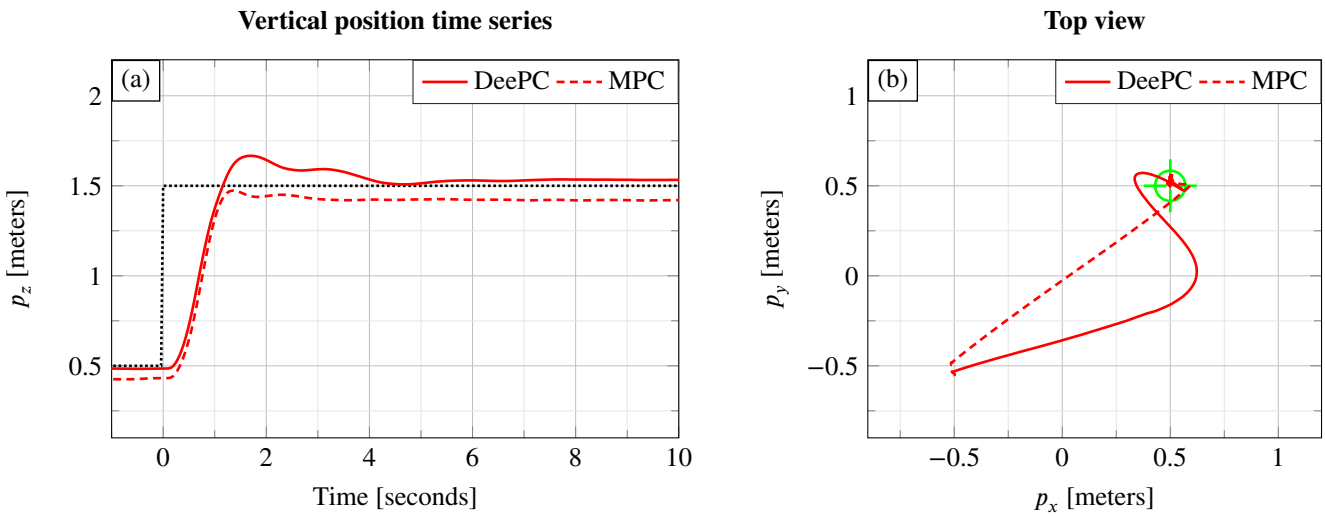


FIGURE 10 Experimental comparison of DeePC and MPC.

4 | CONCLUSION

We demonstrated that the regularized DeePC algorithm is suitable for real-time control of a real-world quadcoptor, thereby bridging the gap between theory and practice. In the process, we performed a sensitivity analysis on the hyperparameters of the DeePC algorithm in simulation, gaining key insights on their effect. These simulation takeaways generalized well to the real-world quadcoptor system, where minimal hyperparameter refining was performed. Through the real-world implementation, it was demonstrated that the DeePC algorithm is computationally tractable and adequately solvable in real-time, with solve times far beneath the real-time requirement. Future work includes applying the DeePC on other real-worlds systems from which no first-principles model can be derived.

Financial disclosure

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme grant agreement OCAL, No. 787845

Conflict of interest

The authors declare no potential conflict of interest.

APPENDIX

A PARAMETERS FOR IMPLEMENTATION OF THE DeePC ALGORITHM

The following lists the hyperparameters offered by the DeePC algorithm, and the design choices required to specify the quadcopter tracking goal. The value specified in this list is used for all results unless otherwise indicated in the text.

- $T_d = 331$, the total number of data points used to construct the Hankel matrices in (5),
- $T_{\text{ini}} = 6$, the number of initial inputs and outputs,
- $\lambda_s = 7.5 \times 10^8$, the weight on the softened initial condition constraint,
- $\lambda_g = 500$, the weight on the regularization of g,
- $q = 2$, the type of norm used to regularize g,
- $p = 3$, the number of outputs used to construct the Hankel matrices in (5),
- $T_f = 25$, the time horizon, (corresponds to $1s$ in continuous time),
- $Q = \begin{pmatrix} 40 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 40 \end{pmatrix}$, the quadratic tracking error cost matrix,
- $R = \begin{pmatrix} 160 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{pmatrix}$, the quadratic control effort cost matrix,
- $\mathcal{U}$, the control inputs constraints set, given by: $f_{\text{tot}} \in [0.1597, 0.4791]$, $\omega_{\text{ref},x}, \omega_{\text{ref},y} \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$,
- $\mathcal{Y}$, the outputs constraints set, given by: $p_x, p_y \in [-4, 4]$, $p_z \in [0.1, 4]$, $\gamma, \beta \in \left[-\frac{\pi}{6}, \frac{\pi}{6}\right]$ when $p = 5$. Note that the constraints on the quadcoptor orientation, γ, β, are omitted when $p = 3$,
- $u_r = (0.2747, 0, 0)$, the steady state hovering control inputs,
- $\Sigma_y = \begin{pmatrix} 1 \times 10^{-8} & 5 \times 10^{-9} & 0 & 0 & 0 \\ 5 \times 10^{-9} & 1 \times 10^{-8} & 0 & 0 & 0 \\ 0 & 0 & 1 \times 10^{-8} & 0 & 0 \\ 0 & 0 & 0 & 1.22 \times 10^{-5} & 0 \\ 0 & 0 & 0 & 0 & 1.22 \times 10^{-5} \end{pmatrix}$, the covariance matrix of measurement noise in simulation when $p = 5$. Note that when $p = 3$ the covariance matrix is the top left 3×3 block of Σ_y.

B FURTHER RESULTS FOR THE GRID SEARCH ANALYSIS

For completeness, we include here the results for the grid search analysis, described in Section 3.2, for all hyperparameters considered. Figure B1 bottom left is the same as shown in Section 3.2, and the other plots in Figure B1 are for the remaining combinations of $q = \{1, 2\}$ and $p = \{3, 5\}$.

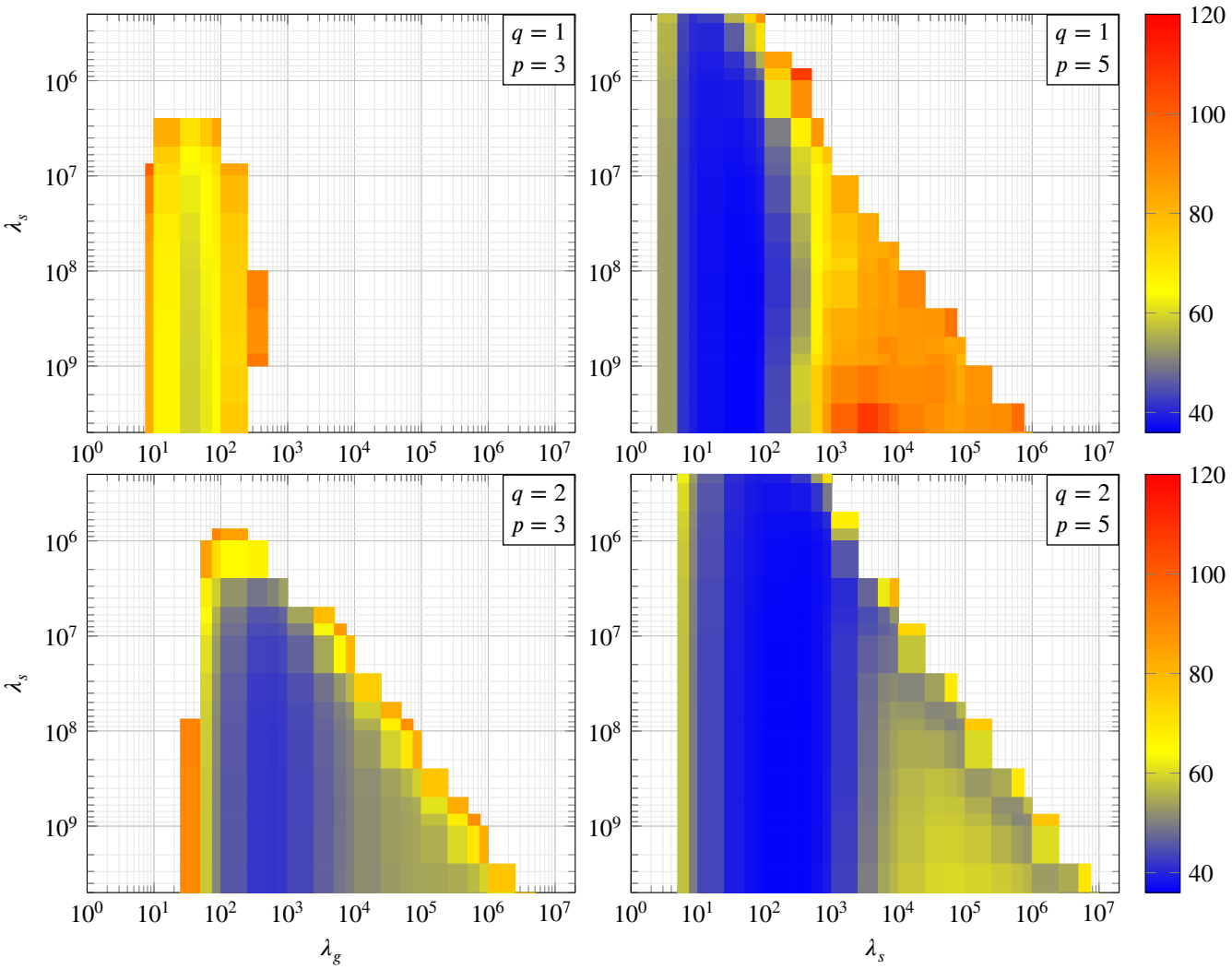


FIGURE B1 Influence of λ_g and λ_s on the tracking error for the four combinations of 1-norm or 2-norm regularization ($q \in \{1, 2\}$ respectively) on the decision variable g, and $p = \{3, 5\}$ output parameters measured, as labelled on the axes. All other parameters are fixed to the values described in the Section 3.2. The coloured shading is restricted to the interval $(36, 120)$ to sufficiently display the shape of each plot. All plots increase steeply for values greater than 120, and the plots are clipped for values greater than 120.

C COMPARING SENSITIVITY TO λ_G IN SIMULATION AND EXPERIMENT

Figure C2 shows results similar to Figure 6 for comparing the closed loop trajectories (solid lines) and the predictions computed by the DeePC optimization problem (8) (dashed lines). This shows the same trend that the performance observed in simulation-based analysis, Figure C2 (a,b,c), is qualitatively similar to that observed in the real-world experiments, Figure C2 (d,e,f).

Qualitatively, the best λ_g chosen in simulation also performs best in reality and results in a similar closed loop trajectory. The small value of λ_g results in a faster but more oscillatory response, and the large value of λ_g results in a sluggish response. This figure demonstrates that, despite unmodelled dynamics in simulation, the real-world system behaves similarly to the simulation model when applying DeePC Algorithm 1. Consequently, simulation-based hyperparameter selection was adapted on the real system with minimal adjustments required.

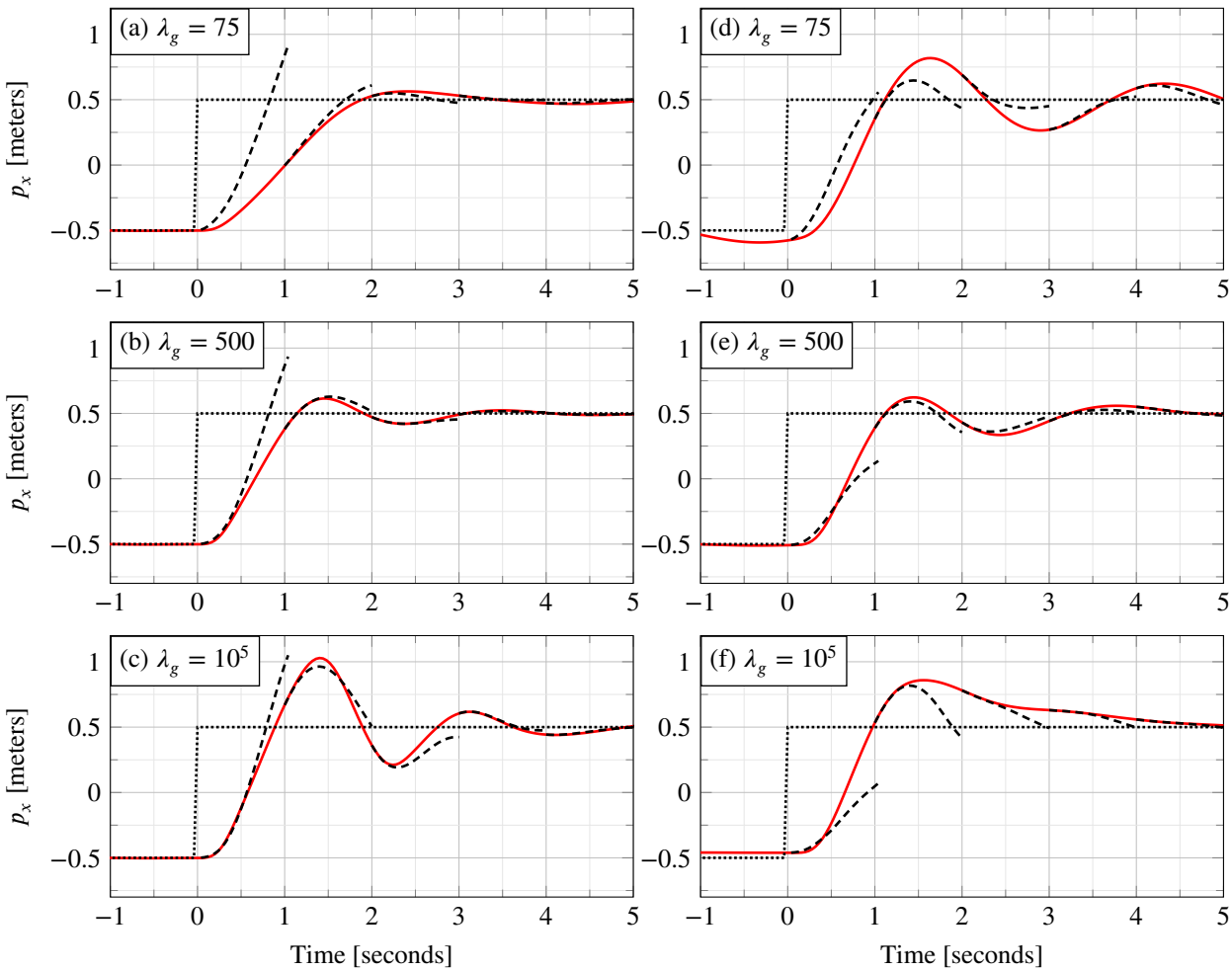


FIGURE C2 Actual trajectories (solid) versus predicted trajectories (dashed). The plots (a,b,c) are simulated results and (d,e,f) are experimental results. To highlight the transferability from simulation to real-world experiments, for each value of λ_g (indicated on the plot) all other hyperparameters have the same values. The hyperparameters are selected as those achieving the minimum tracking error in the simulation-based analysis for the particular value of λ_g.

References

1. Ogunnaike BA. A contemporary industrial perspective on process control theory and practice. *Annual Reviews in Control* 1996; 20: 1–8.

2. Hjalmarsson H. From experiment design to closed-loop control. *Automatica* 2005; 41(3): 393–438.

3. Ostafew CJ, Schoellig AP, Barfoot TD. Robust constrained learning-based NMPC enabling reliable mobile robot path tracking. *The International Journal of Robotics Research* 2016; 35(13): 1547–1563.

4. Gevers M. Towards a joint design of identification and control?. In: *Essays on Control: Perspectives in the Theory and its Applications*. 14 of *Progress in Systems and Control Theory*. Boston, MA: Birkhäuser Boston. 1993 (pp. 111-151).

5. Deisenroth MP, Fox D, Rasmussen CE. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence* 2013; 37(2): 408–423.

6. Wahlström N, Schön TB, Deisenroth MP. From pixels to torques: Policy learning with deep dynamical models. *arXiv preprint arXiv:1502.02251* 2015.

7. Berkenkamp F, Schoellig AP, Krause A. Safe controller optimization for quadrotors with Gaussian processes. In: *IEEE International Conference on Robotics and Automation (ICRA)*; 2016: 491–496.

8. Hwangbo J, Sa I, Siegwart R, Hutter M. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters* 2017; 2(4): 2096–2103.

9. Rosolia U, Borrelli F. Learning model predictive control for iterative tasks. A data-driven control framework. *IEEE Transactions on Automatic Control* 2017; 63(7): 1883–1896.

10. Koller T, Berkenkamp F, Turchetta M, Krause A. Learning-based Model Predictive Control for Safe Exploration. In: *IEEE Conference on Decision and Control (CDC)*; 2018: 6059–6066.

11. Fisac JF, Akametalu AK, Zeilinger MN, Kaynama S, Gillula J, Tomlin CJ. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control* 2018.

12. Aswani A, Gonzalez H, Sastry SS, Tomlin C. Provably safe and robust learning-based model predictive control. *Automatica* 2013; 49(5): 1216–1226.

13. Islam R, Henderson P, Gomrokchi M, Precup D. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133* 2017.

14. Coulson J, Lygeros J, Dorfler F. Data-Enabled Predictive Control: In the Shallows of the DeePC. In: *18th European Control Conference (ECC)*; 2019: 307–312.

15. Willems JC, Rapisarda P, Markovsky I, De Moor BL. A note on persistency of excitation. *Systems & Control Letters* 2005; 54(4): 325–329.

16. Markovsky I, Rapisarda P. Data-driven simulation and control. *International Journal of Control* 2008; 81(12): 1946–1959.

17. Berberich J, Köhler J, Müller MA, Allgöwer F. Data-Driven Model Predictive Control with Stability and Robustness Guarantees. *arXiv preprint arXiv:1906.04679* 2019.

18. Coulson J, Lygeros J, Dörfler F. Regularized and distributionally robust data-enabled predictive control. In: *IEEE Conference on Decision and Control (CDC)*; 2019, to appear.

19. Huang L, Coulson J, Lygeros J, Döfler F. Data-Enabled Predictive Control for Grid-Connected Power Converters. In: *IEEE Conference on Decision and Control (CDC)*; 2019, to appear.

20. De Persis C, Tesi P. On persistency of excitation and formulas for data-driven control. *arXiv preprint arXiv:1903.06842* 2019.

21. Baggio G, Katewa V, Pasqualetti F. Data-Driven Minimum-Energy Controls for Linear Systems. *IEEE Control Systems Letters* 2019; 3(3): 589–594.

22. Salvador JR, Ramirez D, Alamo T, La Pena dDM, Garcia-Marin G. Data Driven Control: An Offset Free Approach. In: *18th European Control Conference (ECC)*; 2019: 23–28.

23. Mahony R, Kumar V, Corke P. Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor. *Robotics Automation Magazine, IEEE* 2012; 19: 20–32. doi: 10.1109/MRA.2012.2206474

24. Lupashin S, Hehn M, Mueller MW, Schoellig AP, Sherback M, D'Andrea R. A platform for aerial robotics research and demonstration: The flying machine arena. *Mechatronics* 2014; 24(1): 41–54.

25. Vicon . Vicon Motion Systems Ltd.. 2019.

26. Merriaux P, Dupuis Y, Boutteau R, Vasseur P, Savatier X. A study of Vicon system positioning performance. *Sensors* 2017; 17(7).

27. Beuchat PN, Stürz YR, Lygeros J. A teaching system for hands-on quadcopter control. *IFAC-PapersOnLine* 2019; 52(9): 36–41. Symposium on Advances in Control Education (ACE)doi: 10.1016/j.ifacol.2019.08.120

28. Bitcraze . Bitcraze AB. 2019.

29. Förster J. System identification of the Crazyflie 2.0 nano quadrocopter. BSc. thesis. ETH Zürich. 2015.

30. Häggblom KE. Evaluation of Experiment Designs for MIMO Identification by Cross-Validation. *IFAC-PapersOnLine* 2016; 49(7): 308 - 313. 11th IFAC Symposium on Dynamics and Control of Process SystemsIncluding Biosystems DYCOPS-CAB 2016doi: https://doi.org/10.1016/j.ifacol.2016.07.310

31. Ljung L. *System identification : theory for the user*. Prentice Hall information and system sciences seriesUpper Saddle River, N.J: Prentice Hall. 2nd ed ed. 1999.

32. Stellato B, Banjac G, Goulart P, Bemporad A, Boyd S. OSQP: An Operator Splitting Solver for Quadratic Programs. *ArXiv e-prints* 2017.

33. Beuchat PN. *N-rotor vehicles: modelling, control, and estimation*. ETH Zürich Research Collection . 2019

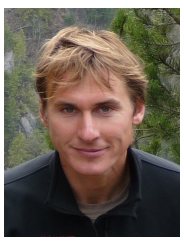
AUTHOR BIOGRAPHIES

Ezzat Elokda received the B.A Sc. in Mechatronics Engineering from the University of Waterloo, Canada in June 2014. From 2014-2018, he held control engineering positions at process automation and lifting equipment companies. Since September 2018, he has been pursuing his M.Sc. in Robotics, Systems & Control at ETH Zürich. His current research interests include optimization-based control and game theory for control.

Jeremy Coulson is a PhD student with the Automatic Control Laboratory at ETH Zürich. He received his Master of Applied Science in Mathematics & Engineering from Queen's University, Canada in August 2017. He received his B.Sc.Eng degree in Mechanical Engineering & Applied Mathematics from Queen's University in 2015. His research interests include data-driven control methods, stochastic optimization, and control of partial differential equations.

Paul Beuchat received the B.Eng. degree in mechanical engineering and B.Sc. in physics from the University of Melbourne, Australia, in 2008, and the M.Sc. degree in robotics, systems and control from ETH Zürich, Switzerland, in 2014, where he is currently working towards the Ph.D degree at the Automatic Control Laboratory. From 2009-2012 he was as a subsurface engineer for ExxonMobil. His research interests are control and optimization of large scale systems, with a focus towards developing approximate dynamic programming techniques for applications in the areas of building control, and coordinated flight.

John Lygeros completed a B.Eng. degree in electrical engineering in 1990 and an M.Sc. degree in Systems Control in 1991, both at Imperial College of Science Technology and Medicine, London, U.K.. In 1996 he obtained a Ph.D. degree from the Electrical Engineering and Computer Sciences Department, University of California, Berkeley. During the period 1996–2000 he held a series of post-doctoral researcher appointments at the Laboratory for Computer Science, M.I.T., and the Electrical Engineering and Computer Sciences Department at U.C. Berkeley. Between 2000 and 2003 he was a University Lecturer at the Department of Engineering, University of Cambridge, U.K., and a Fellow of Churchill College. Between 2003 and 2006 he was an Assistant Professor at the Department of Electrical and Computer Engineering, University of Patras, Greece. In July 2006 he joined the Automatic Control Laboratory at ETH Zurich, where he is currently serving as the Head of the Automatic Control Laboratory and the Head of the Department of Information Technology and Electrical Engineering. His research interests include modelling, analysis, and control of hierarchical, hybrid, and stochastic systems, with applications to biochemical networks, automated highway systems, air traffic management, power grids and camera networks. John Lygeros is a Fellow of the IEEE, and a member of the IET and the Technical Chamber of Greece; since 2013 he serves as the Treasurer of the International Federation of Automatic Control.



Florian Dörfler is an Associate Professor at the Automatic Control Laboratory at ETH Zürich. He received his Ph.D. degree in Mechanical Engineering from the University of California at Santa Barbara in 2013, and a Diplom degree in Engineering Cybernetics from the University of Stuttgart in 2008. From 2013 to 2014 he was an Assistant Professor at the University of California Los Angeles. His students were winners or finalists for Best Student Paper awards at the 2013/2019 European Control Conference, the 2016 American Control Conference, and the 2017 PES PowerTech Conference. His articles received the 2010 ACC Student Best Paper Award, the 2011 O. Hugo Schuck Best Paper Award, the 2012-2014 Automatica Best Paper Award, and the 2016 IEEE Circuits and Systems Guillemin-Cauer Best Paper Award. He is a recipient of the 2009 Regents Special International Fellowship, the 2011 Peter J. Frenkel Foundation Fellowship, and the 2015 UCSB ME Best PhD award.