Sensitivity-Aware Model Predictive Control for Robots with Parametric Uncertainty

Tommaso Belvedere, Marco Cognetti, Giuseppe Oriolo, Paolo Robuffo Giordano

***Abstract*—This paper introduces a computationally efficient robust Model Predictive Control (MPC) scheme for controlling nonlinear systems affected by parametric uncertainties in their models. The approach leverages the recent notion of *closed-loop state sensitivity* and the associated ellipsoidal tubes of perturbed trajectories for taking into account online time-varying restrictions on state and input constraints. This makes the MPC controller "aware" of potential additional requirements needed to cope with parametric uncertainty, thus significantly improving the tracking performance and success rates during navigation in constrained environments. One key contribution lies in the introduction of a computationally efficient robust MPC formulation with a *comparable computational complexity* to a standard MPC (i.e., an MPC not explicitly dealing with parametric uncertainty). An extensive simulation campaign is presented to demonstrate the effectiveness of the proposed approach in handling parametric uncertainties and enhancing task performance, safety, and overall robustness. Furthermore, we also provide an experimental validation that shows the feasibility of the approach in real-world conditions and corroborates the statistical findings of the simulation campaign. The versatility and efficiency of the proposed method make it therefore a valuable tool for real-time control of robots subject to non-negligible uncertainty in their models.**

***Index Terms*—Optimization and Optimal Control, Model Predictive Control, Robust/Adaptive Control of Robotic Systems, Aerial Systems: Mechanics and Control.**

I. Introduction

ADVANCEMENTS in sensing, planning, and control have empowered robots with the ability to accomplish highly complex tasks such as, e.g., navigation in cluttered environments or manipulation and physical interaction with the environment and human users. However, many challenges still need to be solved for an effective deployment of robots in real-world scenarios, one of which being that of *robustness* against the (unavoidable) uncertainties of the robot/environment models.In fact, any advanced planning and control algorithm relies on a suitable model of the robot/environment for generating the motion plan or actions for realizing a task of interest. However, any model is only an approximation of the real world and, thus, planning/control schemes must exhibit some degree of robustness against model uncertainties.

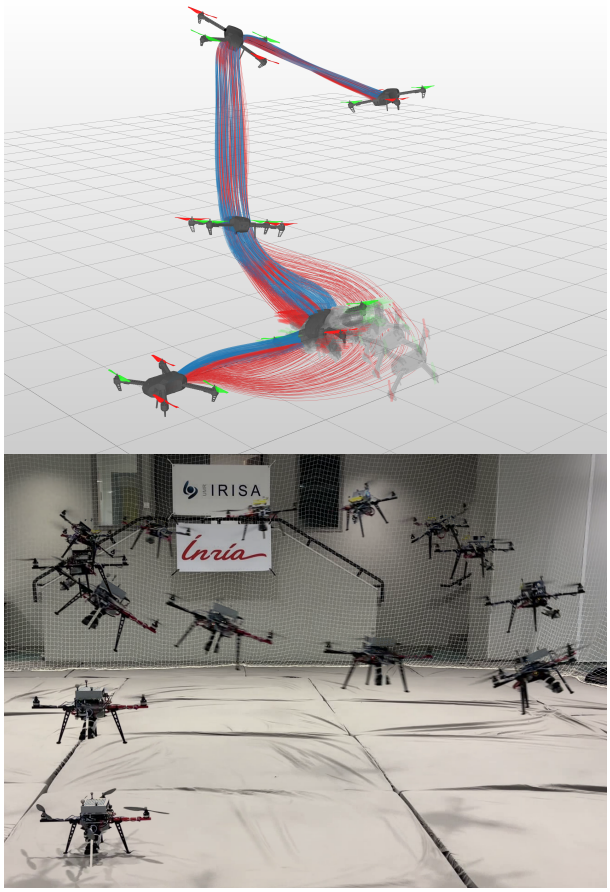

Fig. 1. Top: Stroboscopic view of the comparison between our proposed method (blue) and a regular MPC controller (red) while performing trajectory tracking. The spread of trajectories is obtained by varying the robot's physical parameters and letting the drone track the nominal trajectory. It is clear how our proposed method is able to remain much closer to its nominal behavior. Bottom: Stroboscopic view of the real quadrotor tracking an aggressive trajectory with the proposed method.

Uncertainties in a robot/environment can have many sources and can be modeled at different levels of detail. For instance, a typical choice in many robotics applications is to consider a generic additive "process noise" (or "actuation noise"), often taken as Gaussian random variables. This additive term can capture the presence of some uncertainty in the robot's

T. Belvedere and P. Robuffo Giordano are with CNRS, Univ Rennes, Inria, IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France. E-mail: {tommaso.belvedere,prg}@irisa.fr

M. Cognetti is with the LAAS-CNRS, Université de Toulouse, CNRS, UPS, Toulouse, France. E-mail: marco.cognetti@laas.fr

G. Oriolo is with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, via Ariosto 25, 00185 Roma, Italy. E-mail: oriolo@diag.uniroma1.it

This work was partially carried out while Tommaso Belvedere was a a Ph.D. student at the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma.

This work was supported by the project ANR-20-CE33-0003 "CAMP" and by the Chaire de Professeur Junior Grant ANR-22-CPJ1-0064-01.

Experiments presented in this paper were carried out thanks to a platform of the Robotex 2.0 French research infrastructure.

This article has supplementary downloadable material available at ...

Digital Object Identifier

motion/actuation, but its genericity does not allow, in general, to obtain precise predictions of how the actual uncertainties will affect the robot's motions or actions. However, the simplicity of this choice (and tractability from a mathematical point of view) make additive noise terms virtually the most popular modeling choice for developing control, planning, and estimation schemes.

In many cases of interest, a model of the robot/environment can be considered available with the main source of uncertainty lumped in the inaccurate knowledge of some model parameters. In these cases, the uncertainty is not *generic* but it has a very specific structure which, if exploited, can lead to better predictions of its effects on the robot motion. In light of these considerations, some recent works [1]–[6] have introduced and exploited the notion of *closed-loop state sensitivity matrix*: this quantity locally captures how deviations in the model parameters (w.r.t. their nominal values) affect the evolution of the robot/environment states in *closed-loop*, i.e., by also taking into account the strengths/weaknesses of the particular control action chosen for executing the task. A norm of the state sensitivity can, for instance, be minimized for generating reference trajectories that result by construction minimally sensitive to parametric uncertainties, thus increasing the intrinsic robustness of their tracking in closed-loop. This is particularly relevant whenever the task requirements allow for some redundancy in the reference trajectory that can be optimized w.r.t. the sensitivity metrics, e.g., in case of collision-free navigation in cluttered environments or regulation tasks. The notion of state sensitivity has been later extended to also consider the effects of uncertainties on the control inputs [2], by defining and minimizing the so-called *input sensitivity*: this allows, for instance, to better cope with actuation limits that might otherwise be violated when the robot deviates from its nominal trajectory [5]. Finally, the sensitivity matrix has also been leveraged to obtain time-varying bounds (tubes) on the state and/or input evolution assuming a (known) range of variation for the parameters [7]. This makes it possible to plan robust trajectories that, at least locally, ensure the feasibility of the resulting motion (against state [7] or input [8] constraints) also in the presence of model parametric uncertainties.

Even if successful in generating robust trajectories, the above-mentioned works have only considered the case of *offline* planning. However, the ability to re-plan *online* can clearly offer an additional and substantial improvement of the robustness against uncertainties. Therefore, the main goal of this work is to study how to exploit the closed-loop state sensitivity (and related/derived quantities) within an *online* replanning scheme formulated as a Model Predictive Control (MPC) problem. Being an online local planner, MPC is a widely adopted control approach in robotics, with many examples of its application to quadrotor control [9], [10], autonomous racing [11], wheeled mobile robot navigation and control [12]–[14], and legged locomotion [15]–[18]. Its predictive nature, which leverages a *nominal* model of the robot/environment, and the possibility to explicitly handle constraints and optimization of performance indexes over the future evolution of the system, has made it an attractive choice for solving complex control problems. For instance, one of the main key advantages of MPC over traditional robot controllers is the ability to account for input constraints [10] so as to ensure the feasibility of the planned motion also in the presence of limited actuation.

Nonetheless, a practical MPC deployment can face significant challenges related to feasibility and robustness to uncertainties, in particular when hard constraints play an active role in motion generation. Standard MPC, in fact, generates an *open-loop trajectory* with no information regarding the ability to counteract the effects of possible future disturbances. However, the ability to correctly predict the future behavior of the robot by also accounting for the impact of feedback actions is crucial. Whenever the robot deviates from the predicted trajectory, the feedback controller will act against the disturbance to steer the robot back on the planned motion. Although this positive effect can decrease the impact of the uncertainties, it also requires, in general, an *increased control effort* that needs to be accurately taken into account if input constraints are present. Planning a feasible robust motion must then ensure that the predicted motion possesses enough control authority to satisfy constraints amid potential disturbances.

Related works: The issue of robustness against uncertainties in MPC has motivated many research efforts over the years to propose a variety of "robust MPC" schemes. The goal of robust MPC is to guarantee the feasibility of the predicted trajectory for all possible disturbance sequences [19, Ch. 3]. When dealing with uncertain systems, the set of all possible trajectories can be seen as a *bundle*. Robust MPC methods try to control the bundle of trajectories to guarantee feasibility. Since MPC acts online as a controller, the time complexity of one iteration of any MPC algorithm must be compatible with the real-time requirements of the controlled robot. This requirement has naturally shaped MPC methods to search for the best trade-off between optimality, robustness, and complexity.

A way to directly control the bundle of trajectories, referred to as the scenario-tree approach [20], is to discretize the disturbance set and to control the evolution of all the possible realizations of the system. This approach has, however, limited practical applicability because of its computational complexity. A more practical approach is to find an outer approximation of the bundle of perturbed trajectories in the form of time-varying *tubes*. The so-called Tube MPC, first developed for linear systems [21] and then extended to nonlinear ones [22], [23], aims at finding a tractable way to compute such tubes, possibly *online*. The tube computation typically involves a propagation of the disturbances over the closed-loop dynamics. In most cases, the computation of the tubes' cross-section relies on an *ancillary* control law providing disturbance rejection through feedback. Such controller can be fixed a-priori [24]–[26] or it can be parameterized and determined online as part of the algorithm in order to, for instance, minimize the tubes' cross-section [23]. Typically, an ellipsoidal approximation of the tubes' cross-section around the nominal trajectory is used. In [23], the tubes' cross-section is obtained, along with the parameterized feedback gains, by solving a Semi-Definite Program point-wise for each predicted sampling instant. Similarly, [24] defines a point-wise optimization problem to find

the direction and length of the ellipsoid axes for each state dimension. In [25], an incremental Lyapunov function with a corresponding incrementally stabilizing feedback is precomputed offline and used by the MPC to construct the tubes online. Moreover, this feedback action is then added to the input computed by the MPC and applied to the system in closed-loop. In [26] and [27], the zero-order Robust Optimization (zoRO) [28], which uses ellipsoidal uncertainty sets to robustify the constraints while neglecting their sensitivities in the optimization, is used in an MPC setting with a precomputed static feedback. The Tube MPC in [29] uses a boundary layer sliding mode controller for the ancillary control law. Notably, it optimizes both the state trajectory and the tubes at, however, a likely high computational cost as pointed out in [28]. Also in the Stochastic Nonlinear MPC setting, a prestabilizing infinite horizon LQR controller is added to the control action to propagate uncertainties [30]. As an alternative to the use of an ancillary controller altogether, [31] uses min-max differential inequalities to obtain Linear Matrix Inequality constraints which are added to the MPC optimization problem to make the (ellipsoidal) tubes forward invariant.

The previously mentioned methods address the robust MPC problem by proposing solutions on a spectrum with varying degrees of complexity. For example, some methods trade performance for simplicity by fixing the ancillary controller with a static linear feedback [24]–[27], [30]. This helps in simplifying the uncertainty propagation but can decrease the performance if the system is not operating close to the linearization point. More complex methods can instead provide strong robustness theoretical guarantees, but they also typically have limited practical applicability due to their computation time complexity, which can exceed that of a standard non-robust MPC by several orders, and their reliance on disturbance bounds that can be challenging to certify [25], [31]. Motivated by these considerations and the challenges posed by this problem, the main goal of this work is to propose a *computationally efficient* and *tractable* Robust MPC scheme that falls in the middle of this spectrum.

To this end, we make use of the results of sensitivity analysis of optimization problems [32], [33] to compute *online* the equivalent gains of the MPC action, thus freeing from the need for an (often) precomputed and unconstrained ancillary controller, which does not necessarily capture well the effect of the actual feedback action provided by the MPC scheme. Crucially, this allows our proposed robust MPC scheme to be aware of how the presence of hard constraints affects the feedback action itself, making the uncertainty prediction more accurate and potentially less conservative. Sensitivity analysis has often been used in the MPC context, for instance in differentiating MPC policies for learning purposes [34], or to provide high-frequency feedback to apply during the sampling interval in cases where the possibly complex MPC controller cannot run sufficiently fast [15], [35]–[37]. Compared to these works in which the MPC gains are only used for reactive control, we are instead interested in approximating the MPC feedback action over the whole prediction horizon for propagating the closed-loop sensitivity. This requires the development of an efficient way of computing the predicted MPC gains, enabling the proposed method to be suitable for real-time use. It is worth noting that numerical optimal control methods based on the iterative Linear Quadratic Regulator (iLQR) naturally provide linear feedback gains as a byproduct of the Riccati recursion performed to solve the Optimal Control Problem (OCP); see, e.g., [38] for the unconstrained case and [15], [39] for the extension to equality and inequality constrained problems. However, this requires using solvers based on the iLQR method which, in the MPC setting, are usually limited to treating inequalities as soft constraints.

Summary of contributions: We propose a method to robustify MPC control against parametric uncertainties in the robot model by making use of *sensitivity-based tubes*. Since the closed-loop sensitivity represents an accurate and easy-to-obtain quantity to analyze the effects of the uncertainties on generic nonlinear systems, the proposed MPC scheme can be applied to complex robotic systems without having to rely on hand-crafted robustness conditions or on unnecessary simplifications of the robot model. Our main contributions are:

- The first application of the closed-loop sensitivity framework to an *online* setting based on an *optimization-based* controller, differently from all previous works on this topic that instead leveraged a *closed-form* expression for the control action;
- The formulation of a robust MPC scheme offering a tractable and convenient approach for adding a robustness layer to any standard, and possibly already available, MPC controller. In this way, the safety and the performance of the robot in executing an assigned task is increased with a very minor computational overhead;
- In Sect. III-B (and in the proof in App. A), we present a way of computing the MPC gains when using an off-the-shelf Quadratic Programming (QP) solver and a standard direct multiple shooting transcriptions of the OCP, making the proposed method applicable to general constrained MPC problems not necessarily relying on specialized solvers;
- An extensive simulation campaign to showcase how the introduction of input tubes improves the performance and feasibility of the system under disturbances compared to a standard MPC;
- An experimental validation on a quadrotor UAV with all the computations performed on the onboard hardware, thus demonstrating the real-world applicability of the proposed method and further validating the numerical results.

The rest of the paper is structured as follows. In Sect. II, we introduce the notion of Closed-loop State Sensitivity and review the MPC formulation. In Sect. III, we describe the proposed robust MPC algorithm, and, in Sect. IV, we showcase its application to two case studies involving a quadrotor affected by model uncertainties, providing statistical simulation results over multiple scenarios and experimental results to demonstrate the real-world performance of the method. Finally, we draw several conclusions and discuss possible future directions in Sect. V. For reference, Tab. I reports the notation used in the following.

TABLE I
LIST OF SYMBOLS USED IN THE PAPER

Symbol	Description
a	Scalar
$\boldsymbol{a}$	Vector
$\boldsymbol{A}$	Matrix
A_{ij}	i-th row, j-th column element of $\boldsymbol{A}$
$\boldsymbol{a}(t)$	Time-varying vector
$\boldsymbol{a}_k = \boldsymbol{a}(t_k)$	Vector $\boldsymbol{a}$ evaluated at time t_k
$\boldsymbol{I}_n$	Identity matrix of dimension n
$\mathbf{1}_n$	n-dimensional vector of ones
$\mathbf{0}_n$	n-dimensional vector of zeros
$\boldsymbol{A}^\dagger$	Moore-Penrose inverse of $\boldsymbol{A}$
$\mathbb{I}_j^k = \{i \in \mathbb{N} \mid j \leq i \leq k\}$	Set of indices from j to k
$\|\cdot\|_{\boldsymbol{W}}$	2-norm weighted by the matrix $\boldsymbol{W}$
$\mathbb{S}^3$	Unit 3-sphere
$\boldsymbol{q} = [q_w, q_x, q_y, q_z]^T$	Quaternion
$\boldsymbol{q}_\mathbf{v} = [q_x, q_y, q_z]^T$	Vector part of the quaternion
$\otimes$	Quaternion product's operator
$\|\boldsymbol{q}_a - \boldsymbol{q}_b\| = \|(\boldsymbol{q}_a \otimes \boldsymbol{q}_b^{-1})_\mathbf{v}\|$	Error quaternion norm
$\boldsymbol{a} \times \boldsymbol{b} = [\boldsymbol{a}]_\times \boldsymbol{b}$	Skew-symmetric cross product
$\boldsymbol{a} \leq \boldsymbol{b}$	Element-wise inequality between the two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$

II. PRELIMINARIES

In this section, we review the notion of closed-loop state sensitivity (Sect. II-A) and its use for evaluating the tubes of perturbed trajectories (Sect. II-B). We then briefly recall the MPC formulation used in the rest of the paper (Sect. II-C).

A. Closed-loop Sensitivity

Consider a discrete-time[1] dynamical system described by

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{p}) \tag{1}$$

under the action of a controller $\boldsymbol{\mu} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_u}$

$$\boldsymbol{u}_k = \boldsymbol{\mu}(\boldsymbol{x}_k, \boldsymbol{p}_c), \tag{2}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ and $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ denote the state and input vectors, $\boldsymbol{p} \in \mathbb{R}^{n_p}$ the (uncertain) model parameters, and $\boldsymbol{p}_c \in \mathbb{R}^{n_p}$ the *nominal* parameters used by the (possibly model-based) controller. The combination of (1) and (2) determines the closed-loop system

$$\begin{cases} \boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{p}) \\ \boldsymbol{u}_k = \boldsymbol{\mu}(\boldsymbol{x}_k, \boldsymbol{p}_c) \end{cases} \tag{3}$$

and the map $\boldsymbol{\phi}_k(\boldsymbol{x}_0) := \boldsymbol{\phi}(\boldsymbol{x}_0, t_k) = \boldsymbol{x}_k$ denotes the solution of (3) at time t_k starting from an initial state $\boldsymbol{x}_0$ at t_0. Given a known initial state $\boldsymbol{x}_0$, the closed-loop trajectory of the system will depend on the actual value of the parameters $\boldsymbol{p}$. We denote with $\bar{\boldsymbol{x}}$ the closed-loop trajectory obtained in the *nominal* case, i.e., when $\boldsymbol{p} = \boldsymbol{p}_c$, and with $\bar{\boldsymbol{u}}$ the associated nominal input trajectory. In general, however, $\boldsymbol{p} \neq \boldsymbol{p}_c$ because of parametric uncertainty and, therefore, the actual state/input trajectories $(\boldsymbol{x}, \boldsymbol{u})$ will deviate from the nominal ones $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$.

Following [1], [2], we use the notion of closed-loop sensitivity to describe how changes in the parameters affect the system trajectories under the action of the controller. Define the *state sensitivity* $\boldsymbol{\Pi} \in \mathbb{R}^{n_x \times n_p}$ as

$$\boldsymbol{\Pi}_k = \left.\frac{d\boldsymbol{\phi}_k(\boldsymbol{x}_0)}{d\boldsymbol{p}}\right|_{\boldsymbol{p}=\boldsymbol{p}_c}. \tag{4}$$

This quantity is evaluated around the nominal trajectory (i.e., the one obtained when $\boldsymbol{p} = \boldsymbol{p}_c$) and represents a first-order approximation of the effects of a parameter deviation on the closed-loop state trajectory. For instance, a system that is structurally less affected by a particular parameter, or for which the controller is already able to compensate for parameter inaccuracies, will present a smaller (in some norm) closed-loop state sensitivity.

A closed-form expression for (4) is, in general, not available but, as shown in [1], it is possible to obtain the following update rule

$$\begin{aligned} \boldsymbol{\Pi}_{k+1} &= \frac{d\boldsymbol{\phi}_{k+1}(\boldsymbol{x}_0)}{d\boldsymbol{p}} = \frac{d\boldsymbol{f}(\boldsymbol{\phi}_k(\boldsymbol{x}_0), \boldsymbol{u}_k, \boldsymbol{p})}{d\boldsymbol{p}} \\ &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\frac{d\boldsymbol{\phi}_k(\boldsymbol{x}_0)}{d\boldsymbol{p}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\frac{d\boldsymbol{u}_k}{d\boldsymbol{p}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}} \\ &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{\Pi}_k + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\frac{d\boldsymbol{u}_k}{d\boldsymbol{p}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}, \end{aligned} \tag{5}$$

i.e., a discrete-time prediction model for the state sensitivity $\boldsymbol{\Pi}_k$ with initialization $\boldsymbol{\Pi}_0 = \mathbf{0}$ since the initial state is assumed known and by definition $\boldsymbol{\phi}_0(\boldsymbol{x}_0) = \boldsymbol{x}_0$.

From (5), we also define matrix $\boldsymbol{\Theta} \in \mathbb{R}^{n_u \times n_p}$

$$\boldsymbol{\Theta}_k = \frac{d\boldsymbol{u}_k}{d\boldsymbol{p}} = \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{x}}\boldsymbol{\Pi}_k.$$

This matrix, denoted as *input sensitivity*, represents the indirect effect of a parameter change on the control action by means of feedback, which can be seen as a measure of how the input in closed-loop will deviate from its nominal trajectory $\bar{\boldsymbol{u}}$.

Finally, denoting with

$$\boldsymbol{A}_k = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, \boldsymbol{p}_c} \quad \boldsymbol{B}_k = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, \boldsymbol{p}_c}, \tag{6}$$

$$\boldsymbol{M}_k = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}\right|_{\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, \boldsymbol{p}_c} \quad \boldsymbol{F}_{k|k} = \left.\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{x}}\right|_{\bar{\boldsymbol{x}}_k, \boldsymbol{p}_c}, \tag{7}$$

the closed-loop sensitivity prediction model (5) can be compactly rewritten as

$$\boldsymbol{\Pi}_{k+1} = \left(\boldsymbol{A}_k + \boldsymbol{B}_k\boldsymbol{F}_{k|k}\right)\boldsymbol{\Pi}_k + \boldsymbol{M}_k, \tag{8}$$

$$\boldsymbol{\Theta}_k = \boldsymbol{F}_{k|k}\boldsymbol{\Pi}_k. \tag{9}$$

An interesting interpretation can be given to matrix $\boldsymbol{F}_{k|k}$. Suppose that the system deviates from the nominal state $\bar{\boldsymbol{x}}_k$ to a perturbed state $\boldsymbol{x}_{\epsilon,k}$. Then, $\boldsymbol{F}_{k|k}$ represents the state feedback gains of the first-order Taylor approximation of the control law (2) since, for a small perturbation, the control action $\boldsymbol{u}_{\epsilon,k} = \boldsymbol{\mu}(\boldsymbol{x}_{\epsilon,k}, \boldsymbol{p}_c)$ can be expanded as

$$\boldsymbol{u}_{\epsilon,k} \simeq \bar{\boldsymbol{u}}_k + \boldsymbol{F}_{k|k}(\boldsymbol{x}_{\epsilon,k} - \bar{\boldsymbol{x}}_k).$$

[1]Since we are interested in designing a control system working in discrete-time with a sampling time δ_t, we consider a discretized model obtained from the continuous time dynamics $\dot{\boldsymbol{x}} = \boldsymbol{f}_c(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p})$ by integrating it numerically over the time interval $[t_k, t_{k+1})$ with $t_{k+1} = t_k + \delta_t$ with constant inputs.

With this in mind, we will refer to $\boldsymbol{F}_{k|k}$ as the *(feedback) gains* of the controller.

Remark 1. *The notation $\boldsymbol{F}_{i|j}$ is introduced to express the dependence of the input at time t_i on the state at a time $t_j \leq t_i$. This will become relevant in the MPC context (Sect. III-B) where the predicted (future) input depends directly on the initial condition.*

Additionally, it is worth mentioning that, from (8) and (9), one can also define the sensitivity of any function of the state and inputs to characterize its local behavior to changes in the parameters. Letting $\boldsymbol{\gamma}(\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{R}^{n_\gamma}$ be a generic function, its closed-loop sensitivity to parameters is simply:

$$\boldsymbol{\Gamma}_k = \left.\frac{d\boldsymbol{\gamma}}{d\boldsymbol{p}}\right|_{\boldsymbol{p}=\boldsymbol{p}_c} = \frac{\partial \boldsymbol{\gamma}}{\partial \boldsymbol{x}}\boldsymbol{\Pi}_k + \frac{\partial \boldsymbol{\gamma}}{\partial \boldsymbol{u}}\boldsymbol{\Theta}_k. \tag{10}$$

B. Tubes of perturbed trajectories

Exploiting the state sensitivity $\boldsymbol{\Pi}$, it is possible to obtain an ellipsoidal approximation of the bundle of the perturbed trajectories for the states and the inputs (or for any function of these quantities), as discussed in [7]. We here recall the main steps.

Assume a (known) maximum parameter deviation $\Delta\boldsymbol{p}_{\max}$ for each component of the vector $\boldsymbol{p}$, that is, the parameters are supposed to belong to the set

$$\mathcal{P} = \{\boldsymbol{p} \in \mathbb{R}^{n_p} : -\Delta\boldsymbol{p}_{\max} \leq \boldsymbol{p} - \boldsymbol{p}_c \leq \Delta\boldsymbol{p}_{\max}\} \tag{11}$$

centered at the nominal value $\boldsymbol{p}_c$, and let $\boldsymbol{W} = \mathrm{diag}(\Delta p_{\max,i}^2)$. To estimate the effect of a parameter deviation on the closed-loop trajectory, we define a mapping from the parameter space to the state space through ellipsoids. Let the parameter ellipsoid with matrix $\boldsymbol{W} \in \mathbb{R}^{n_p \times n_p}$ centered at $\boldsymbol{p}_c$ be

$$\mathcal{E}_{\boldsymbol{p}} = \left\{\boldsymbol{p} \in \mathbb{R}^{n_p} \,:\, \Delta\boldsymbol{p}^T \boldsymbol{W}^{-1} \Delta\boldsymbol{p} \leq 1\right\} \tag{12}$$

with $\Delta\boldsymbol{p} = \boldsymbol{p} - \boldsymbol{p}_c$. Equivalently, let $\boldsymbol{\sigma}_{\boldsymbol{p}} = \boldsymbol{W}^{-\frac{1}{2}}\Delta\boldsymbol{p}$ be the scaled parameter deviation and

$$\mathcal{S}_{\boldsymbol{p}} = \left\{\boldsymbol{\sigma}_{\boldsymbol{p}} \in \mathbb{R}^{n_p} \,:\, \boldsymbol{\sigma}_{\boldsymbol{p}}^T \boldsymbol{\sigma}_{\boldsymbol{p}} \leq 1\right\} \tag{13}$$

the scaled parameter sphere. From a first-order approximation around the nominal trajectory $\bar{\boldsymbol{x}}$, one has

$$\Delta\boldsymbol{x}_k = \boldsymbol{x}_k - \bar{\boldsymbol{x}}_k \simeq \boldsymbol{\Pi}_k \Delta\boldsymbol{p} = \boldsymbol{\Pi}_k \boldsymbol{W}^{\frac{1}{2}} \boldsymbol{\sigma}_{\boldsymbol{p}} \tag{14}$$

which, assuming that $\boldsymbol{p} \in \mathcal{E}_{\boldsymbol{p}} \subset \mathcal{P}$, can be exploited to obtain the corresponding uncertainty ellipsoid in the state space at a given time t_k. This is obtained by pseudo-inverting (14) for mapping *feasible* state deviations[2] to parameter deviations

$$\boldsymbol{\sigma}_{\boldsymbol{p}} = \left(\boldsymbol{\Pi}_k \boldsymbol{W}^{\frac{1}{2}}\right)^{\dagger} \Delta\boldsymbol{x}_k, \tag{15}$$

from which, applying (15) to (13) and noting that $(\boldsymbol{\Pi}_k \boldsymbol{W}^{\frac{1}{2}})^{\dagger T} (\boldsymbol{\Pi}_k \boldsymbol{W}^{\frac{1}{2}})^{\dagger} = \left(\boldsymbol{\Pi}_k \boldsymbol{W} \boldsymbol{\Pi}_k^T\right)^{\dagger}$, one obtains the sought uncertainty ellipsoid in state space:

$$\Delta\boldsymbol{x}_k^T \left(\boldsymbol{\Pi}_k \boldsymbol{W} \boldsymbol{\Pi}_k^T\right)^{\dagger} \Delta\boldsymbol{x}_k \leq 1. \tag{16}$$

[2]We use the term feasible to highlight the fact that in case $n_x > n_p$ — thus (14) is overdetermined — only state deviations which are in the range space of $\boldsymbol{\Pi}_k$ will provide an exact solution. This is however always the case in our treatment since we assume that all state deviations satisfy (14).

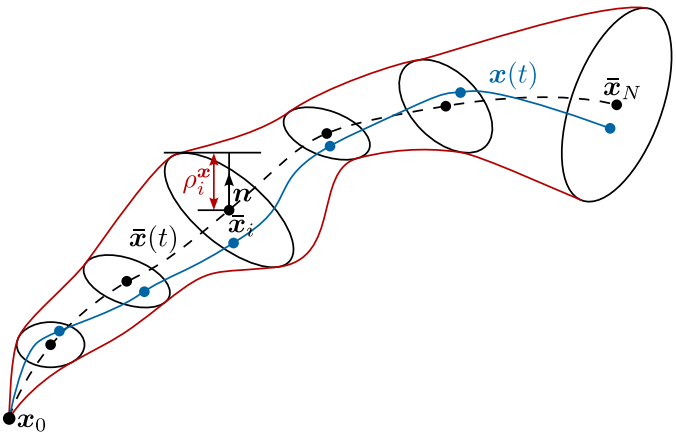

Fig. 2. Qualitative visualization of a perturbed state trajectory $\boldsymbol{x}(t)$ (blue) and the ellipsoidal tubes (red) centered around the nominal trajectory $\bar{\boldsymbol{x}}(t)$ (dashed black). Note how, in general, the tube radius does not necessarily monotonically increase thanks to the presence of the feedback action, and despite the cumulative effect of uncertainties over time. This effect can for instance be appreciated in the results reported in [6] for what concerns state and input tubes.

Moreover, the same reasoning applies to the input deviation $\Delta\boldsymbol{u}_k = \boldsymbol{u}_k - \bar{\boldsymbol{u}}_k$ resulting in the input space ellipsoid

$$\Delta\boldsymbol{u}_k^T \left(\boldsymbol{\Theta}_k \boldsymbol{W} \boldsymbol{\Theta}_k^T\right)^{\dagger} \Delta\boldsymbol{u}_k \leq 1. \tag{17}$$

Note that both the state and the input ellipsoids are time-varying, as the respective sensitivities follow the evolution given by (8) and (9). The evolution of these ellipsoids over time is what we will refer to as *state* and *input tubes* in the following. Under assumptions (12)–(14), the closed-loop trajectories will evolve inside the tubes centered around the nominal trajectory (see Fig. 2 for an illustrative example), that is, the state/input at time t_k will belong to their corresponding ellipsoid at t_k[3]. One can then make use of these tubes for several purposes, e.g., for shaping the closed-loop trajectory so that the state/input ellipsoids do not violate any state or input constraint.

In many cases of interest, one needs to determine the maximum deviation along a *particular direction of interest* $\boldsymbol{n}$ in the state/input space (for instance, for evaluating the deviation of a particular component of the state/input). Given an ellipsoid with matrix $\boldsymbol{K}$, the *ellipsoid radius* ρ along a direction $\boldsymbol{n}$ can be obtained as [40]

$$\rho_{\boldsymbol{n}} = \sqrt{\boldsymbol{n}^T \boldsymbol{K} \boldsymbol{n}}. \tag{18}$$

Considering a set of directions of interest, one can then compute the radius along each direction for obtaining a vector of radii $\boldsymbol{\rho}$ which can be used as a measure of the worst-case deviation of the system along these directions. For illustration let us consider w.l.o.g. the case of input tubes. Once the evolution of the input sensitivity $\boldsymbol{\Theta}$ has been obtained, one can compute the input ellipsoid matrix $\boldsymbol{K}_k^{\boldsymbol{u}} = \boldsymbol{\Theta}_k \boldsymbol{W} \boldsymbol{\Theta}_k^T$. Then,

[3]While this result formally holds for the approximate mapping (14), Monte Carlo simulations in [7] have shown how the resulting tubes capture well the ensemble of perturbed trajectories also for complex robots such as an hexarotor. Furthermore, the validity of the state/input tubes has also been confirmed in real experimental conditions.

using (18) with $\boldsymbol{n}$ spanning the canonical basis of the input space $\mathbb{R}^{n_u}$, it is possible to obtain

$$\rho^{\boldsymbol{u}}_{j,k} = \sqrt{K^{\boldsymbol{u}}_{jj,k}}, \quad \forall j \in \mathbb{I}_1^{n_u}, \quad \forall k \in \mathbb{I}_0^{N-1}. \tag{19}$$

The quantities $\rho^{\boldsymbol{u}}_{j,k}$, collected in vector $\boldsymbol{\rho}^{\boldsymbol{u}}_k$, are the radii of the input ellipsoid along each axis j and represent, therefore, the worst-case deviations of each input component j w.r.t. its nominal value due to parametric uncertainty. Note that, at the start of the trajectory, $\boldsymbol{\rho}^{\boldsymbol{u}}_0 = \mathbf{0}$ by construction since $\boldsymbol{\Pi}_0 = \mathbf{0}$ and, therefore, $\boldsymbol{\Theta}_0 = \mathbf{0}$.

The same reasoning can clearly be applied to the states to find the associated radii from the state sensitivity (8) and, using (10), to any function of the state and inputs to find its worst-case deviation along the nominal trajectory.

C. Model Predictive Control

We briefly recall the classical MPC formulation and a few details on its solution. The reader already familiar with these notions can skip until (22) at the end of this section, noting that we consider Optimal Control Problems (OCPs) in the form (20) that can be approximately solved as the QP (22).

At each control instant t_k, the MPC solves a constrained trajectory optimization problem starting from the current state $\boldsymbol{x}_k$ over a prediction horizon $[t_k, t_{k+N}]$ for finding a sequence of optimal inputs $\boldsymbol{u}^* = (\boldsymbol{u}_0^*, \dots, \boldsymbol{u}_{N-1}^*)$ that generates the optimal trajectory. The first input $\boldsymbol{u}_0^*$ of this optimal sequence is selected and commanded to the system until the next feedback measurement $\hat{\boldsymbol{x}}_{k+1}$ at time $t_{k+1} = t_k + \delta_t$ becomes available for repeating the procedure.

At each instant, the MPC solves the following OCP

$$\begin{cases} \min\limits_{\boldsymbol{u}} & \sum\limits_{i=0}^{N-1} \ell(\boldsymbol{x}_i, \boldsymbol{u}_i) + \ell_N(\boldsymbol{x}_N) & \text{(20a)} \\ \text{s.t.} & \boldsymbol{x}_0 = \hat{\boldsymbol{x}}_k & \text{(20b)} \\ & \boldsymbol{x}_{i+1} = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{p}_c) \qquad \forall i \in \mathbb{I}_0^{N-1} & \text{(20c)} \\ & \boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{u}_i) \le \mathbf{0} \qquad \forall i \in \mathbb{I}_0^{N-1} & \text{(20d)} \end{cases}$$

where $\ell(\boldsymbol{x}_i, \boldsymbol{u}_i)$ and $\ell_N(\boldsymbol{x}_N)$ in (20a) denote the running and final costs, respectively. The problem is subject to the initial state constraint (20b), the dynamics constraint (20c) (evaluated at the nominal parameters $\boldsymbol{p}_c$), and possibly to some nonlinear path constraints[4] $\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{u}_i) \le \mathbf{0}$ (20d).

Remark 2. *In the MPC formulation, note how the subscript i refers to the* predicted *time of the trajectory over the control horizon with $t_{k+i} \in [t_k,\ t_{k+N}]$ (for $i = 0, \dots, N$), that is always reset to zero at each MPC iteration. On the other hand, the subscript k on the time t_k and the current state $\hat{\boldsymbol{x}}_k$ refer to the* actual *time of the closed-loop system evolving with the dynamics* (3)*.*

In order to solve the trajectory optimization problem we employ a direct multiple shooting transcription [41] and choose $\boldsymbol{w} = (\boldsymbol{x}, \boldsymbol{u}) = (\boldsymbol{x}_0, \dots, \boldsymbol{x}_N, \boldsymbol{u}_0, \dots, \boldsymbol{u}_{N-1})$, consisting of the state and the input trajectories, as the decision variables. The resulting problem is in general a non-convex Nonlinear Program (NLP) due to the nonlinear dynamics and the possibly non-convex inequality constraints. In an MPC context, the time-budget to solve the optimization problem is at most the sampling time δ_t, and reducing the control delay is of paramount importance. For this reason, we employ the Real-Time Iteration scheme (RTI) [42], which consists in performing one quasi-Newton step for solving (20) at each control cycle, appropriately linearizing the optimization problem around the current solution guess $\bar{\boldsymbol{w}} = (\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$ obtained from the solution computed at the previous time instant, and solving a Quadratic Program (QP). The cost function is assumed to be in the usual least-squares form, for which we define the compact notation $\sum_0^{N-1} \ell(\boldsymbol{x}_i, \boldsymbol{u}_i) + \ell_N(\boldsymbol{x}_N) = \frac{1}{2} \left\| \boldsymbol{R}(\boldsymbol{w}) \right\|^2$ through the residual vector $\boldsymbol{R}(\boldsymbol{w})$. This allows using the Gauss-Newton Hessian approximation [43] which represents a simple to compute and multiplier-free alternative to the evaluation of the exact Hessian of the problem. To this end, define the decision variables vector for the QP as $\Delta\boldsymbol{w} = \boldsymbol{w} - \bar{\boldsymbol{w}}$. The MPC controller then solves the following QP:

$$\begin{cases} \min\limits_{\Delta\boldsymbol{w}} & \frac{1}{2}\Delta\boldsymbol{w}^T \boldsymbol{H} \Delta\boldsymbol{w} + \boldsymbol{c}^T \Delta\boldsymbol{w} \\ \text{s.t.} & \Delta\boldsymbol{x}_0 + \bar{\boldsymbol{x}}_0 - \hat{\boldsymbol{x}}_k = \mathbf{0} \\ & \Delta\boldsymbol{x}_{i+1} - \boldsymbol{A}_i \Delta\boldsymbol{x}_i - \boldsymbol{B}_i \Delta\boldsymbol{u}_i + \boldsymbol{f}_i = \mathbf{0} \quad \forall i \in \mathbb{I}_0^{N-1} \\ & \boldsymbol{C}_i \Delta\boldsymbol{x}_i + \boldsymbol{D}_i \Delta\boldsymbol{u}_i + \boldsymbol{g}_i \le \mathbf{0} \quad \forall i \in \mathbb{I}_0^{N-1} \end{cases}$$

where

$$\begin{aligned} &\boldsymbol{H} = \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right|_{\bar{\boldsymbol{w}}}^T \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right|_{\bar{\boldsymbol{w}}} && \boldsymbol{c} = \left.\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\right|_{\bar{\boldsymbol{w}}}^T \boldsymbol{R}(\bar{\boldsymbol{w}}) \\ &\boldsymbol{A}_i,\ \boldsymbol{B}_i \text{ from (6)} && \boldsymbol{f}_i = \bar{\boldsymbol{x}}_{i+1} - \boldsymbol{f}(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{u}}_i, \boldsymbol{p}_c) \\ &\boldsymbol{C}_i = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\right|_{\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{u}}_i} \boldsymbol{D}_i = \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}}\right|_{\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{u}}_i} && \boldsymbol{g}_i = \boldsymbol{g}(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{u}}_i). \end{aligned} \tag{21}$$

To write the QP problem in a compact form, we stack and group the equality and inequality constraints into matrices $\boldsymbol{E}$ and $\boldsymbol{G}$, respectively, and we define $\boldsymbol{\eta}(\hat{\boldsymbol{x}}_k) = (\bar{\boldsymbol{x}}_0 - \hat{\boldsymbol{x}}_k, \boldsymbol{f}_0, \dots, \boldsymbol{f}_{N-1})$ and $\boldsymbol{g} = (\boldsymbol{g}_0, \dots, \boldsymbol{g}_{N-1})$. In this way, everything can be expressed with respect to the decision variables vector $\Delta\boldsymbol{w}$:

$$\begin{cases} \min\limits_{\Delta\boldsymbol{w}} & \frac{1}{2}\Delta\boldsymbol{w}^T \boldsymbol{H} \Delta\boldsymbol{w} + \boldsymbol{c}^T \Delta\boldsymbol{w} \\ \text{s.t.} & \boldsymbol{E}\Delta\boldsymbol{w} + \boldsymbol{\eta}(\hat{\boldsymbol{x}}_k) = \mathbf{0} \\ & \boldsymbol{G}\Delta\boldsymbol{w} + \boldsymbol{g} \le \mathbf{0}. \end{cases} \tag{22}$$

In the previous equations, $\boldsymbol{E}$ and $\boldsymbol{G}$ are sparse matrices with block diagonal non-zero entries (see, e.g., [44]). This structure can be exploited by sparse QP solvers to efficiently compute the solution to the problem.

By solving this problem at each control instant, the MPC algorithm computes an optimal state and input trajectory $\boldsymbol{w}^* = \bar{\boldsymbol{w}} + \Delta\boldsymbol{w}^*$ from the current state at time t_k to the end state at time t_{k+N}. Let us partition the output of the MPC controller as

$$\boldsymbol{w}^*(\hat{\boldsymbol{x}}_k, \boldsymbol{p}_c) = \begin{bmatrix} \boldsymbol{x}_i^* & \forall i \in \mathbb{I}_0^N \\ \boldsymbol{u}_0^* & \\ \boldsymbol{u}_i^* & \forall i \in \mathbb{I}_1^{N-1} \end{bmatrix}, \tag{23}$$

[4]Note that, although we use this particular formulation to carry out the analysis throughout the paper, one can easily recast different kinds of constraints (such as double-sided constraints) in this form or adapt the derivations to account for, e.g., additional equality constraints.

which renders explicit the control action $\boldsymbol{u}_0^*$ to be applied to the system, the predicted optimal input trajectory $\boldsymbol{u}_i^*$, and the associated state trajectory $\boldsymbol{x}_i^*$, respectively. Note that the MPC solution is essentially an open-loop trajectory to be applied to the system with no information on the behavior of the controller in case of disturbances that would make it deviate from the plan. The goal of the next section is to show how the main results of parametric optimization can be exploited to approximate the control policy in the vicinity of the predicted trajectory, and how this information can be leveraged to improve the robustness of the MPC controller against model uncertainties.

III. The Proposed Approach

The safety and performance of the closed-loop system (robot + MPC controller) are subject to two main conditions: (i) the ability of the MPC to find a feasible solution and remain feasible during motion, and (ii) the satisfaction of the constraints in closed-loop, i.e., whether or not predicting satisfaction of a constraint will result in its actual satisfaction during motion. As previously discussed, a discrepancy between the real parameters $\boldsymbol{p}$ of the robot and the nominal parameters $\boldsymbol{p}_c$ used by the MPC controller can result in a divergence between the actual closed-loop trajectory and the one generated in the MPC prediction phase. In general, the MPC will cope with this discrepancy by searching for an alternative (but still feasible) optimal solution and satisfaction of the state/input constraints plays a crucial role in the determination of the feasibility of this MPC "corrective" action.

The idea behind our approach is to use the closed-loop sensitivity (and related quantities) to generate an appropriate time-varying restriction of the constraints over the prediction horizon that can improve feasibility also in the presence of model uncertainties which could degrade the prediction ability of the MPC. This naturally introduces a possible trade-off between robustness and performance that will be discussed in the results of Sect. IV.

A. Formulation

The proposed MPC scheme, denoted as *Sensitivity-aware Tube MPC* (ST-MPC), consists of two main steps: (i) The sensitivity of the solution guess is evaluated using (8) and (9), from which the sensitivity-based tube radii are obtained as in (19) to find the worst-case predicted constraint deviation; (ii) When the feedback for the current state $\hat{\boldsymbol{x}}_k$ is available, the following optimization problem

$$\begin{cases} \min\limits_{\boldsymbol{u}} & \sum\limits_{i=0}^{N-1} \ell(\boldsymbol{x}_i, \boldsymbol{u}_i) + \ell_N(\boldsymbol{x}_N) \\ \text{s.t.} & \boldsymbol{x}_0 = \hat{\boldsymbol{x}}_k \\ & \boldsymbol{x}_{i+1} = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{p}_c) & \forall i \in \mathbb{I}_0^{N-1} \\ & \boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{u}_i) + \boldsymbol{\rho}_i^{\boldsymbol{h}} \leq \boldsymbol{0} & \forall i \in \mathbb{I}_0^{N-1} \\ & \boldsymbol{u}_{\min} + \boldsymbol{\rho}_i^{\boldsymbol{u}} \leq \boldsymbol{u}_i \leq \boldsymbol{u}_{\max} - \boldsymbol{\rho}_i^{\boldsymbol{u}} & \forall i \in \mathbb{I}_0^{N-1} \end{cases} \tag{24}$$

is solved by approximating it as a QP using the RTI method described in Sect. II-C. Note how, in (24), the constraints are reduced by the tube radii $\boldsymbol{\rho}_i^{\boldsymbol{h}}$ (being the tube radii for the task constraint function $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u})$) and $\boldsymbol{\rho}_i^{\boldsymbol{u}}$ that act as back-off terms to account for the effects of parametric uncertainty. These are obtained by following the procedure described in Sect. II-B for the input case, and by simply repeating the same procedure on the sensitivity of any function $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u})$ of interest for the case of generic task constraints. Once the optimal solution is found, the input $\boldsymbol{u}_0^*$ is sent to the system and the algorithm is repeated.

In order to compute the tube radii $\boldsymbol{\rho} = \left(\boldsymbol{\rho}^{\boldsymbol{h}}, \boldsymbol{\rho}^{\boldsymbol{u}}\right)$ from the state and the input sensitivities $\boldsymbol{\Pi}$ and $\boldsymbol{\Theta}$, one must first determine the feedback gains $\boldsymbol{F}_{k|k}$ of the MPC controller that describe how the solution of (24) varies w.r.t. changes in $\hat{\boldsymbol{x}}_k$. This step is not straightforward and the following Sect. III-B details the proposed procedure for obtaining the term $\boldsymbol{F}_{k|k}$. Finally, Sect. III-C provides a detailed description of the final algorithm.

B. Computing the MPC feedback gains

Consider the QP problem (22) and denote its optimal solution as $\boldsymbol{y}^* = (\Delta \boldsymbol{w}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$, collecting the primal and dual variables. The MPC feedback gains are obtained by differentiating the optimal solution $\boldsymbol{y}^*$ and extracting the components relative to the first input $\boldsymbol{u}_0^*$. In fact, note how the whole optimization problem can be seen as a function of the initial state $\hat{\boldsymbol{x}}_k$ and of the nominal model parameters $\boldsymbol{p}_c$. Under mild regularity assumptions [32], [35], [45], one can compute the sensitivities as

$$\frac{d\boldsymbol{w}^*(\hat{\boldsymbol{x}}_k, \boldsymbol{p}_c)}{d\hat{\boldsymbol{x}}_k} = \begin{bmatrix} \frac{d\boldsymbol{x}_i^*}{d\hat{\boldsymbol{x}}_k} \\ \frac{d\boldsymbol{u}_0^*}{d\hat{\boldsymbol{x}}_k} \\ \frac{d\boldsymbol{u}_i^*}{d\hat{\boldsymbol{x}}_k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{P}_{k+i|k} & \forall i \in \mathbb{I}_0^N \\ \boldsymbol{F}_{k|k} & \\ \boldsymbol{F}_{k+i|k} & \forall i \in \mathbb{I}_1^{N-1} \end{bmatrix} \tag{25}$$

by applying the implicit function theorem (IFT) to the first-order optimality conditions of the problem, and noting that by definition $\frac{d\boldsymbol{w}^*}{d\hat{\boldsymbol{x}}_k} = \frac{d\Delta\boldsymbol{w}^*}{d\hat{\boldsymbol{x}}_k}$. More in detail, assume that the Linear Independence Constraint Qualification (LICQ), the Second-Order Sufficient Conditions (SOSC), and Strict Complementarity (SC) hold at the solution $\boldsymbol{y}^*$ (we refer to [46] for definitions, and to [36] for a discussion on the role of these conditions and their possible relaxations). The optimal solution of (22) is characterized by the Karush-Khun-Tucker (KKT) optimality conditions:

$$\boldsymbol{\mathcal{R}}(\boldsymbol{y}, \hat{\boldsymbol{x}}_k)|_{\boldsymbol{y}^*} = \begin{bmatrix} \boldsymbol{H}\Delta\boldsymbol{w}^* + \boldsymbol{c} + \boldsymbol{E}^T\boldsymbol{\lambda}^* + \boldsymbol{G}_{\mathcal{A}}^T\boldsymbol{\mu}_{\mathcal{A}}^* \\ \boldsymbol{E}\Delta\boldsymbol{w}^* + \boldsymbol{\eta}(\hat{\boldsymbol{x}}_k) \\ \boldsymbol{G}_{\mathcal{A}}\Delta\boldsymbol{w}^* + \boldsymbol{g}_{\mathcal{A}} \end{bmatrix} = \boldsymbol{0}, \tag{26}$$

where subscript $\mathcal{A}$ denotes the set of active inequality constraints at the solution $\boldsymbol{y}^*$. Under the conditions stated above, we can apply the IFT to (26) and differentiate with respect to $\hat{\boldsymbol{x}}_k$ to obtain

$$\begin{aligned} \frac{d}{d\hat{\boldsymbol{x}}_k}\, \boldsymbol{\mathcal{R}}(\boldsymbol{y}(\hat{\boldsymbol{x}}_k), \hat{\boldsymbol{x}}_k)|_{\boldsymbol{y}^*} &= \boldsymbol{0}, \\ \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{y}} \frac{d\boldsymbol{y}}{d\hat{\boldsymbol{x}}_k} + \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \hat{\boldsymbol{x}}_k} &= \boldsymbol{0}, \end{aligned} \tag{27}$$

so that

$$\frac{d\boldsymbol{y}}{d\hat{\boldsymbol{x}}_k} = -\left(\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{y}}\right)^{-1} \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \hat{\boldsymbol{x}}_k}. \tag{28}$$

Note that $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{y}}$, being the Jacobian of the KKT conditions (26), is nothing else than the KKT matrix of the QP, i.e.,

$$\boldsymbol{\mathcal{K}}^* = \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{y}} = \begin{bmatrix} \boldsymbol{H} & \boldsymbol{E}^T & \boldsymbol{G}_{\mathcal{A}}^T \\ \boldsymbol{E} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{G}_{\mathcal{A}} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}. \tag{29}$$

A crucial point here is that this quantity is *already available* from the construction of the QP problem (22), having linearized around the solution guess $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$, by simply removing the rows of $\boldsymbol{G}$ relative to the inactive constraints. We also note that, in the general case, the inversion of the (large) matrix $\boldsymbol{\mathcal{K}}^*$ could have a non-negligible computational cost. However, this is not the case in our context thanks to its structured sparsity, which then allows to efficiently invert matrix $\boldsymbol{\mathcal{K}}^*$ and solve (27) by relying on efficient factorization and solution algorithms for sparse linear systems. Recalling (25), it is then trivial to extract the quantities

$$\boldsymbol{F}_{k|k},\ \boldsymbol{F}_{k+i|k},\ \text{and}\ \boldsymbol{P}_{k+i|k} \tag{30}$$

from (28), which correspond to the sensitivities of the control input $\boldsymbol{u}_0$, of the predicted inputs, and of the predicted states w.r.t. $\hat{\boldsymbol{x}}_k$.

We note, however, that the evaluation of the sensitivity (8) over the prediction horizon of the MPC requires the availability of the controller gains $\boldsymbol{F}_{k|k}$ over the *whole future horizon*, that is, availability of $\boldsymbol{F}_{k+i|k+i}$, $\forall i \in \mathbb{I}_1^{N-1}$. These quantities represent the sensitivity of the future inputs $\boldsymbol{u}_{k+i}$ with respect to a variation in the future state $\hat{\boldsymbol{x}}_{k+i}$ and they are unfortunately not directly available from (30) that only provides the sensitivity of the full MPC solution with respect to the *current state* $\hat{\boldsymbol{x}}_k$. At first glance, the computation of $\boldsymbol{F}_{k+i|k+i}$, $i \in \mathbb{I}_1^{N-1}$ would require to solve (26–30) for each $i \in \mathbb{I}_1^{N-1}$, which would quickly become computationally unfeasible in the typically short time available. To circumvent this problem, we instead employ an approximation of the future gains exploiting the information on the future inputs provided by the parametric sensitivity of the MPC. In fact, once the full MPC sensitivity (30) has been obtained, one can utilize the sensitivity of the future states and inputs to approximate the future gains relative to the current prediction horizon as

$$\boldsymbol{F}_{k+i|k+i} \simeq \tilde{\boldsymbol{F}}_{k+i|k+i} = \frac{d\boldsymbol{u}_i}{d\boldsymbol{x}_i} = \boldsymbol{F}_{k+i|k}\boldsymbol{P}_{k+i|k}^{-1} \quad \forall i \in \mathbb{I}_1^{N-1}, \tag{31}$$

which represents the predicted gains at the optimal solution, i.e., how a change in the predicted state $\boldsymbol{x}_i$ would affect the input $\boldsymbol{u}_i$ at that time. These gains play a similar role to the Riccati gains of a finite-horizon LQR and approximate well the MPC receding horizon mechanism for short periods of time, while larger approximation errors can occur at the end of the horizon. Since the sensitivities computed w.r.t. the initial state $\hat{\boldsymbol{x}}_k$ are updated at each iteration, the approximation error is in practice minimal and, thus, the gains $\tilde{\boldsymbol{F}}_{k+i|k+i}$ in (31) provide an excellent replacement for $\boldsymbol{F}_{k+i|k+i}$ with the advantage of being computationally very efficient and suitable for real-time use.

A detailed proof of Eq. (31) is provided in App. A. However, a sketch is proposed here for the reader's convenience. The IFT can be applied to the KKT conditions (26) using as explicit variable any element $\boldsymbol{x}_i$ of the state trajectory, from which it would be straightforward to compute the sensitivity $\tilde{\boldsymbol{F}}_{k+i|k+i}$ with respect to that state similarly to (28). However, this requires inverting, for each i, a large matrix obtained by differentiating (26) with respect to the remaining implicit variables, which would make it computationally expensive. Instead, by noting that this matrix can be written as a rank-n_x correction to matrix $\boldsymbol{\mathcal{K}}^*$ (29), one can utilize the Woodbury matrix identity [47] for obtaining an efficient formula to compute $\tilde{\boldsymbol{F}}_{k+i|k+i}$. Moreover, after some simplifications related to the structure of the problem, it is possible to extract (31), making explicit the dependence on the previously computed sensitivities.

Remark 3 (Active constraints selection)**.** *The MPC sensitivity* (25) *depends on the (strongly) active constraint set $\mathcal{A}$ which characterizes which inequality constraints are affecting, locally, the solution of the optimization problem $\boldsymbol{y}^*$. In fact, the procedure essentially treats active inequality constraints as equality constraints that have to be satisfied even when the solution is perturbed. Ultimately, we are interested in computing the MPC gains, that is, how the current and future inputs might deviate from the optimal solution due to the perturbations of the initial state. To do so, it is in general possible to trivially select all the active constraints at the solution by looking at the multipliers $\boldsymbol{\mu}^*$. Being the future input constraints over the prediction horizon restricted by the tubes, these can, in general, be active even if the input $\boldsymbol{u}_{k+i}$ is not saturated to $\boldsymbol{u}_{\max}$ (or $\boldsymbol{u}_{\min}$) due to a tube radius $\boldsymbol{\rho}_{k+i}^{\boldsymbol{u}} > 0$. On the other hand, as the MPC controller works with a receding horizon approach, the actual future input $\boldsymbol{u}_{k+i}$ (i.e., the first input of the solution of the problem at time t_{k+i}) will not experience this restriction and it will have to satisfy only the nominal input constraint. If all the future active constraints were selected in the MPC sensitivity* (25) *calculation, this would result in gains that underestimate the system's ability to respond to disturbances, producing larger and more conservative tubes. To avoid this, we select the active constraint set $\mathcal{A}$ by neglecting the input constraints which are only active with $\boldsymbol{\rho}^{\boldsymbol{u}} > 0$. All the other constraints, e.g., state constraints to perform obstacle avoidance, are instead taken into account.*

Remark 4 (KKT regularization)**.** *In some instances, although the optimization problem is feasible, the KKT matrix might be ill-conditioned or singular due to the LICQ condition not being satisfied at the solution. Most solvers rely on regularization techniques to still be able to produce a solution. In our case, this is necessary for computing the MPC sensitivity. Therefore, we employ a proximal regularization of the Lagrange multipliers of the active inequality constraints [48], [49]. Recall the KKT matrix $\boldsymbol{\mathcal{K}}^*$* (29) *and note that by construction the block*

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{E}^T \\ \boldsymbol{E} & \boldsymbol{0} \end{bmatrix}$$

is assumed full rank. In order to regularize the matrix we apply a proximal regularization, obtaining the augmented Lagrangian $\mathcal{L}_A(\Delta\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}_\mathcal{A}) = \mathcal{L}(\Delta\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}_\mathcal{A}) - \frac{\varrho}{2}\|\boldsymbol{\mu}_\mathcal{A} - \boldsymbol{\mu}^*_\mathcal{A}\|^2_{\boldsymbol{\Lambda}^{-1}}$*, where* $\boldsymbol{\Lambda} = \mathrm{diag}(\boldsymbol{\mu}^*_\mathcal{A})$*,* $\varrho > 0$*, which results in a damping term in the bottom right block as*

$$\boldsymbol{\mathcal{K}}^*_{reg} = \begin{bmatrix} \boldsymbol{H} & \boldsymbol{E}^T & \boldsymbol{G}^T_\mathcal{A} \\ \boldsymbol{E} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{G}_\mathcal{A} & \mathbf{0} & -\varrho\boldsymbol{\Lambda}^{-1} \end{bmatrix}. \tag{32}$$

In practice, this can be interpreted as a normalized constraint relaxation that minimizes the Lagrange multipliers variation [50], ensuring that the KKT matrix is non-singular so that the MPC gains (31) *can be computed.*

Algorithm 1: ST-MPC

1 $\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^* \leftarrow$ `Initialize`$(\hat{\boldsymbol{x}}_0)$
2 $\boldsymbol{\rho} \leftarrow$ `ComputeTubes`$(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$
3 $\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}} \leftarrow \boldsymbol{x}^*, \boldsymbol{u}^*$
4 `PrepareQP`$(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \boldsymbol{\rho})$
5 **while** *running* **do**
6 $\quad \boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^* \leftarrow$ `SolveQP`$(\hat{\boldsymbol{x}}_k)$
7 $\quad$ `SendCommand`$(\boldsymbol{u}^*_0)$
8 $\quad \boldsymbol{\rho} \leftarrow$ `ComputeTubes`$(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$
9 $\quad \bar{\boldsymbol{x}}, \bar{\boldsymbol{u}} \leftarrow$ `ShiftSolution`$(\boldsymbol{x}^*, \boldsymbol{u}^*)$
10 $\quad$ `PrepareQP`$(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \boldsymbol{\rho})$
11 **end**

C. The ST-MPC algorithm

With reference to Algorithm 1, we can now describe in detail the various steps of the full ST-MPC scheme. First, we perform an offline initialization of the controller and prepare the QP for its next iteration (lines 1-4). While online (lines 6-10), we divide the solution of the problem into two phases: (i) a *feedback* phase (lines 6-7), in which the QP is solved when the current state becomes available, and (ii) a *preparation* phase (lines 8-10), in which the tubes are computed and the QP for the next iteration at t_{k+1} is constructed on the basis of the current solution guess $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$. Note that, while the feedback phase needs to be performed when the current state $\hat{\boldsymbol{x}}_k$ becomes available, the preparation phase for the next iteration of the scheme at t_{k+1} does not require knowledge of next state $\hat{\boldsymbol{x}}_{k+1}$, and it can thus be performed just after having sent the input command to the system (line 7), reducing the control delay of the scheme [43].

In more detail, Alg. 1 can be described line-by-line as follows:

- **L1** At the initial state $\hat{\boldsymbol{x}}_0$, start from a trivial initial guess $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$ and $\boldsymbol{\rho} \equiv \mathbf{0}$, and solve the problem to find a feasible solution without considering the constraint restrictions from the tubes;
- **L2** Compute the sensitivity of the MPC (28) and the future gains using (31). Propagate the state and input sensitivities $\boldsymbol{\Pi}$ and $\boldsymbol{\Theta}$ via (8–9), and compute the evolution of the tube radii $\boldsymbol{\rho}$ over the predicted trajectory using (19);
- **L3-4** Prepare the QP for the next feedback phase, evaluating all the derivatives and residuals (21), but this time by including the tubes;
- **L6-7** Given the current state $\hat{\boldsymbol{x}}_k$, solve optimization problem (24) by solving the associated RTI-QP (22) and send the control action $\boldsymbol{u}^*_0$ to the robot;
- **L8** Compute the tube radii $\boldsymbol{\rho}$ as in **L2**;
- **L9-10** Shift the solution one step in time and prepare the QP for the next feedback phase.

IV. Application to Quadrotor Motion Control

In order to demonstrate the effectiveness of the proposed ST-MPC algorithm, we present here a series of tests involving the control of a Quadrotor Unmanned Aerial Vehicle (UAV) subject to parametric uncertainty. Indeed, quadrotors are a very popular robotic platform capable of agile and/or aggressive maneuvers but, at the same time, they are also typically subject to some unavoidable degree of uncertainty in their dynamical model because of, e.g., complex aerodynamics (resulting in uncertain drag/thrust coefficients), or uncertain location of the center of mass (CoM). Moreover, the full 3D quadrotor dynamics is highly nonlinear, which then contributes to showcasing the applicability of our framework to non-trivial systems.

The formulation (24) is quite general and includes both input saturation constraints $\boldsymbol{u}_{\min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{\max}$ and more generic input/state constraints $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}) \leq \mathbf{0}$ with their corresponding tubes. However, in the following case studies we only consider tubes on the *input constraints* since, in many instances, input constraints are the main limiting factor for finding a feasible solution that also satisfies other task constraints. Indeed, in many applications, presence of task constraints alone does not threaten the recursive feasibility of the MPC which, given "infinite" control authority, is virtually able to always find a feasible solution. Also when only actuation limits are present, for instance when MPC is used as a reference tracking controller (as in the first of the following case studies), input constraints are still the ones ultimately determining the performance of the system during the most aggressive maneuvers (minimization of the tracking error typically tends to increase the control effort which is limited by the input constraints).

Note also that, even if we opted for directly controlling the speed of each rotor within the MPC controller (requiring high-frequency feedback), our framework could easily accommodate the presence of any low-level controller, treating the MPC more as a high-level planner. This would be obtained by including the controller dynamics in the sensitivity computation as already done in, e.g., [8].

The rest of this section is structured as follows: in Sect. IV-A, we present the employed model of a 3D Quadrotor UAV with shifted CoM. Then, in Sect. IV-B, we describe the two test scenarios: (i) tracking of aggressive trajectories; and (ii) navigation through a narrow aperture, and we discuss the associated ST-MPC formulation. For each scenario, we performed a comparative statistical analysis (Sect. IV-D1 and Sect. IV-D2). In the first scenario, where only input constraints are present, we highlight how the proposed ST-MPC enhances

the accuracy by reducing unwanted input saturations. In the second scenario, we (purposely) introduce a strict positional constraint that significantly limits the robot's motion. This constraint can cause failures due to input saturations if uncertainties are not properly accounted for. By using ST-MPC we then show how our approach enhances the controller feasibility, thereby increasing the overall success rate. In Sect. IV-E, we report the results of several experiments obtained while tracking aggressive trajectories with the proposed method. Finally, in Sect. IV-F, we study the computational complexity of the algorithm when applied to this problem and discuss some implementation details.

A. Quadrotor Model

We consider a 3D quadrotor model with displaced center of mass (CoM) [51]. Let $\boldsymbol{\mathcal{F}}_{\mathcal{W}} = \{\boldsymbol{O}_{\mathcal{W}}, \boldsymbol{x}_{\mathcal{W}}, \boldsymbol{y}_{\mathcal{W}}, \boldsymbol{z}_{\mathcal{W}}\}$ denote the world inertial frame, and $\boldsymbol{\mathcal{F}}_{\mathcal{B}} = \{\boldsymbol{O}_{\mathcal{B}}, \boldsymbol{x}_{\mathcal{B}}, \boldsymbol{y}_{\mathcal{B}}, \boldsymbol{z}_{\mathcal{B}}\}$ represent the body frame attached to the quadrotor geometric center. We define: $\boldsymbol{r} = (x, y, z) \in \mathbb{R}^3$ as the drone position in $\boldsymbol{\mathcal{F}}_{\mathcal{W}}$, $\boldsymbol{v} = (v_x, v_y, v_z) \in \mathbb{R}^3$ as its linear velocity in $\boldsymbol{\mathcal{F}}_{\mathcal{W}}$, $\boldsymbol{q} = (q_w, q_x, q_y, q_z) \in \mathbb{S}^3$ as the unit-norm quaternion representing the orientation of $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$ relative to $\boldsymbol{\mathcal{F}}_{\mathcal{W}}$, and $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$ as the angular velocity of $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$ with respect to $\boldsymbol{\mathcal{F}}_{\mathcal{W}}$, expressed in $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$. The quadrotor state vector is then $\boldsymbol{x} = (\boldsymbol{r}, \boldsymbol{v}, \boldsymbol{q}, \boldsymbol{\omega}) \in \mathbb{R}^6 \times \mathbb{S}^3 \times \mathbb{R}^3$.

Let w_i be the angular speed of the i-th propeller and define the quadrotor control input as $\boldsymbol{u} = (w_1^2, \dots, w_4^2)$. An allocation matrix $\boldsymbol{T}$ is used to relate the inputs $\boldsymbol{u}$ to the thrust/torques $(f, \boldsymbol{\tau})$ in body frame:

$$\begin{bmatrix} f \\ \boldsymbol{\tau} \end{bmatrix} = \boldsymbol{T}(l, k_f, k_m)\boldsymbol{u}, \tag{33}$$

where $\boldsymbol{T}$ is a function of the length of the quadrotor arms l (specifically of the propellers' positions in the body frame) and of the thrust and the drag aerodynamic coefficients of the propellers k_f and k_m [51], [52]. We assume that a payload of mass m_p is attached to the quadrotor with mass m_r at a position $\boldsymbol{d}_p = (0, 0, d_z)$ in $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$, with the total mass then being $m = m_r + m_p$. We denote with $\mathbf{J} = \mathrm{diag}\big(I_{xx} + m_p d_z^2, I_{yy} + m_p d_z^2, I_{zz}\big) \in \mathbb{R}^{3\times3}$ the quadrotor body frame inertia matrix about the geometric center $\boldsymbol{O}_{\mathcal{B}}$, and with I_{xx}, I_{yy}, I_{zz} the principal moments of inertia along the x, y, z axes of the robot body frame, respectively. We finally consider that the quadrotor center of mass $\boldsymbol{G}_{\mathcal{B}}$ is displaced w.r.t. the geometric center $\boldsymbol{O}_{\mathcal{B}}$ by a displacement denoted as $\boldsymbol{g}_C = (g_x, g_y, g_z)$ in $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$. With these settings, in the subsequent derivations, the set of parameters that are supposed to be *uncertain* is $\boldsymbol{p} = (g_x,\, g_y,\, g_z,\, m_p) \in \mathbb{R}^4$. This is motivated by a series of empirical results and previous works on this topic, such as [2], [3], [5], [8], which highlighted how uncertainties in these parameters are, in practice, the most important ones in affecting the closed-loop performance of a drone, as opposed to, for instance, the thrust coefficients[5]

[5]Note that the sensitivity w.r.t. the considered parameters is at least one order of magnitude higher than the other ones, also taking into account the typical expected parameter deviation.

TABLE II
NOMINAL QUADROTOR MODEL PARAMETERS

Symbol	Value	
	Simulations	Experiments
m_r	1.535 kg	1.420 kg
m_p	0.2 kg	0.2 kg
$\boldsymbol{g}_C$	$(0, 0, 0)$ m	$(0, 0, 0)$ m
I_{xx}, I_{yy}	0.029 kg·m^2	0.022 kg·m^2
I_{zz}	0.055 kg·m^2	0.020 kg·m^2
k_f	$5.86{\cdot}10^{-6}$ N/(rad/s)^2	$5.9{\cdot}10^{-4}$ N/Hz^2
k_m	0.06	0.017
l	0.28 m	0.23 m
d_z	0.15 m	0.15 m

From [51], the total force $\boldsymbol{f}_{\text{tot}}$ acting on the quadrotor in $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$ includes the propeller total thrust f, gravitational effects, and an additional fictitious force due to the displaced $\boldsymbol{g}_C$:

$$\boldsymbol{f}_{\text{tot}} = f\boldsymbol{z}_W - mg\mathbf{R}(\boldsymbol{q})^T\boldsymbol{z}_{\mathcal{W}} - m[\boldsymbol{\omega}]_\times[\boldsymbol{\omega}]_\times\boldsymbol{g}_C,$$

where $\mathbf{R}(\boldsymbol{q}) \in SO(3)$ is the rotation matrix associated to the quaternion $\boldsymbol{q}$. For the total torque $\boldsymbol{\tau}_{\text{tot}}$ (expressed in $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$), one has:

$$\boldsymbol{\tau}_{\text{tot}} = \boldsymbol{\tau} - mg[\mathbf{g}_C]_\times\mathbf{R}(\boldsymbol{q})^T\boldsymbol{z}_{\mathcal{W}} - [\boldsymbol{\omega}]_\times\mathbf{J}\boldsymbol{\omega},$$

where $\boldsymbol{\tau}$ represents the propeller torque from (33). Premultiplying by the inverse spatial inertia matrix one can finally obtain the robot accelerations

$$\begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} m\boldsymbol{I_3} & -m[\mathbf{g}_C]_\times \\ m[\mathbf{g}_C]_\times & \mathbf{J} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{f}_{\text{tot}} \\ \boldsymbol{\tau}_{\text{tot}} \end{bmatrix}$$

from which the quadrotor dynamic model with a displaced center of mass can be obtained as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_c(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) = \begin{cases} \dot{\boldsymbol{r}} = \boldsymbol{v} \\ \dot{\boldsymbol{v}} = \mathbf{R}(\boldsymbol{q})\boldsymbol{\nu}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) \\ \dot{\boldsymbol{q}} = \dfrac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \\ \dot{\boldsymbol{\omega}} = \boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) \end{cases}. \tag{34}$$

The continuous-time dynamics (34) is then discretized using a fixed-step 4th order Runge-Kutta integration to obtain the discrete-time model. All nominal parameters are reported in Table II[6].

B. Test scenarios

We now describe the two case studies.

1) Tracking of aggressive trajectories: The first task consists of tracking a trajectory defined by a series of waypoints $\mathcal{W}_d \in \mathbb{R}^3$, assumed to be generated by an external planner guiding the quadrotor through the environment. The desired trajectory is obtained by linear interpolation of the waypoints with constant velocity, resulting in a position and velocity trajectory $(\boldsymbol{r}_d(t), \dot{\boldsymbol{r}}_d(t))$. As for the orientation (yaw) we consider two possibilities: (i) keeping a constant yaw angle with a desired orientation $\boldsymbol{q}_d(t) \equiv \boldsymbol{q}_0 = (1, 0, 0, 0)$ or (ii) keeping the yaw direction of the quadrotor always

[6]For the simulation, these follow the 3DR Iris drone parameters reported in the PX4 Gazebo package https://github.com/PX4/PX4-SITL_gazebo-classic

TABLE III
MPC SETTINGS FOR TRAJECTORY TRACKING

Symbol	Value	
	Simulations	Experiments
δ_t	0.02 s	0.025 s
N	25	20
$\boldsymbol{Q}_r$	80 $\boldsymbol{I}_3$	10 $\boldsymbol{I}_3$
$\boldsymbol{Q}_v$	2 $\boldsymbol{I}_3$	$\boldsymbol{I}_3$
$\boldsymbol{Q}_q$	$\mathrm{diag}(10^{-2}, 10^{-2}, 1)$	$\mathrm{diag}(0.1, 0.1, 5)$
$\boldsymbol{Q}_\omega$	10^{-2} $\boldsymbol{I}_3$	10^{-2} $\boldsymbol{I}_3$
$\boldsymbol{Q}_u$	$\boldsymbol{I}_4$	$\boldsymbol{I}_4$
ϱ	10^{-2}	10^{-2}
$\Delta\boldsymbol{p}_{\max}$	$(0.04, 0.04, 0.02, 0.1)$	$(0.03, 0.03, 0.02, 0.05)$

pointing towards the upcoming waypoint, i.e., by setting $\psi_d(t) = \mathrm{atan2}(\dot{y}_d(t), \dot{x}_d(t))$ which results in a desired orientation $\boldsymbol{q}_d(t) = (\cos(\psi_d(t)/2), 0, 0, \sin(\psi_d(t)/2))$. This latter possibility is meant to increase the actuation effort during tracking and, therefore, the chances of incurring in actuation saturation.

By construction, these trajectories have discontinuous Cartesian velocities — due to the linear interpolation between the waypoints that are not continuous — and are thus not initially dynamically feasible. As typical in these cases, we rely on the MPC scheme for generating online a *feasible* motion that tracks at best the various trajectory segments. This is obtained by designing the cost function of the MPC controller to minimize the position, velocity, and orientation tracking errors together with a regularization term on the angular velocity $\boldsymbol{\omega}$ for smoothing, and with a feedforward input equal to the nominal hovering condition $\boldsymbol{u}_h = \frac{mg}{4k_f}\mathbf{1}_4$. Letting the desired state and input be $\boldsymbol{x}_d(t) = (\boldsymbol{r}_d(t), \dot{\boldsymbol{r}}_d(t), \boldsymbol{q}_d(t), \mathbf{0}_3)$ and $\boldsymbol{u}_d(t) = \boldsymbol{u}_h$, the running cost then takes the form $\ell(\boldsymbol{x}, \boldsymbol{u}) = \ell_{\boldsymbol{x}} + \ell_{\boldsymbol{u}}$, $\ell_N(\boldsymbol{x}) = \ell_{\boldsymbol{x}}$ with

$$\ell_{\boldsymbol{x}} = \|\boldsymbol{r} - \boldsymbol{r}_d\|^2_{\boldsymbol{Q}_r} + \|\boldsymbol{v} - \dot{\boldsymbol{r}}_d\|^2_{\boldsymbol{Q}_v} + \|\boldsymbol{q} - \boldsymbol{q}_d\|^2_{\boldsymbol{Q}_q} + \|\boldsymbol{\omega}\|^2_{\boldsymbol{Q}_\omega}, \tag{35a}$$

$$\ell_{\boldsymbol{u}} = \|\boldsymbol{u} - \boldsymbol{u}_h\|^2_{\boldsymbol{Q}_u}. \tag{35b}$$

The weights used in all the experiments of this scenario as well as the standard MPC settings are reported in Tab. III.

2) Passing through a narrow aperture: In the second scenario, illustrated in Fig. 3, the robot is tasked with reaching a position goal $\boldsymbol{r}_{sp} \in \mathbb{R}^3$ placed on the opposite side of an aperture while avoiding collisions with the aperture lower and upper sides. This is encoded as a constraint on the z-position of the robot that is activated when the horizontal position (x, y) is inside the aperture.

The cost function of the MPC problem is designed to make the system generate the required motion autonomously based only on the desired final state $\boldsymbol{x}_d = (\boldsymbol{r}_{sp}, \mathbf{0}_3, \boldsymbol{q}_0, \mathbf{0}_3)$ and the hovering input $\boldsymbol{u}_h$ as available information. Again, the running cost takes the form $\ell(\boldsymbol{x}, \boldsymbol{u}) = \ell_{\boldsymbol{x}} + \ell_{\boldsymbol{u}}$, $\ell_N(\boldsymbol{x}) = 10\ \ell_{\boldsymbol{x}}$ with

$$\ell_{\boldsymbol{x}} = \|\boldsymbol{r} - \boldsymbol{r}_{sp}\|^2_{\boldsymbol{Q}_r} + \|\boldsymbol{v}\|^2_{\boldsymbol{Q}_v} + \|\boldsymbol{q} - \boldsymbol{q}_0\|^2_{\boldsymbol{Q}_q} + \|\boldsymbol{\omega}\|^2_{\boldsymbol{Q}_\omega}, \tag{36a}$$

$$\ell_{\boldsymbol{u}} = \|\boldsymbol{u} - \boldsymbol{u}_h\|^2_{\boldsymbol{Q}_u}. \tag{36b}$$

The weights used in all the experiments of this scenario as well as the common MPC settings are reported in Tab. IV.

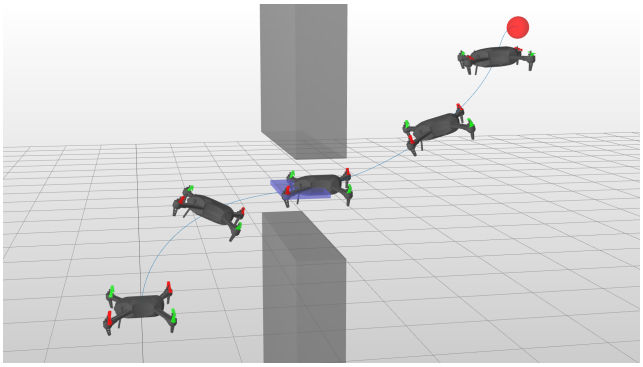

Fig. 3. Stroboscopic view of the quadrotor reaching the desired target (red) while passing through the aperture (safe region in light blue) with nominal parameters.

TABLE IV
MPC SETTINGS FOR REGULATION

Symbol	Value
δ_t	0.02 s
N	{25, 35}
$\boldsymbol{Q}_r$	$\boldsymbol{I}_3$
$\boldsymbol{Q}_v$	$\{2, 5\} \cdot 10^{-2}$ $\boldsymbol{I}_3$
$\boldsymbol{Q}_q$	10^{-2} $\boldsymbol{I}_3$
$\boldsymbol{Q}_\omega$	10^{-2} $\boldsymbol{I}_3$
$\boldsymbol{Q}_u$	0.5 $\boldsymbol{I}_4$
ϱ	10^{-2}
$\Delta\boldsymbol{p}_{\max}$	$(0.02, 0.02, 0.01, 0.1)$

In order to avoid collisions with the environment, we design task constraints that limit the position of the quadrotor when passing through the aperture. Let $A_{\text{fp}} \subset \mathbb{R}^2$ be the (x, y) projection of the aperture on the ground, h_z the z coordinate of the center of the aperture, and $a_z > 0$ the maximum deviation that the center of the robot can safely sustain from h_z. Moreover, let the task constraint function be

$$h_a(\boldsymbol{x}) = \begin{cases} z - h_z & \text{if } (x, y) \in A_{\text{fp}} \\ 0 & \text{otherwise} \end{cases}.$$

Then, the double-sided constraint

$$-a_z \leq h_a(\boldsymbol{x}_i) \leq a_z \quad \forall i \in \mathbb{I}_1^N, \tag{37}$$

expressed in a form compatible with (24)

$$\boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{u}_i) = \begin{bmatrix} h_a(\boldsymbol{x}_{i+1}) - a_z \\ -h_a(\boldsymbol{x}_{i+1}) - a_z \end{bmatrix}, \tag{38}$$

is able to encode the desired collision avoidance behavior.

C. Application of ST-MPC and comparison with alternative methods

We apply the proposed ST-MPC to the two above-mentioned test scenarios following the procedure detailed in Sect. III-C. For the second scenario, where the state constraint (37) is present, we neglect the associated tube by setting $\boldsymbol{\rho}^{\boldsymbol{h}} = \mathbf{0}$ as explained at the beginning of the section. Indeed, (37) is a stringent positional constraint that has to be active while the quadrotor is passing through the aperture. Since the tubes are fixed at each iteration of the algorithm, introducing an

additional restriction might be counterproductive and actually lead to infeasibility, as the MPC has no direct way of reducing the tube cross-section to clear the aperture. On the other hand, the input tubes radii $\boldsymbol{\rho^u}$ will still account for the constraint being active, providing a sufficient level of robustness as demonstrated by the results in Sect. IV-D2.

The controller is then obtained as the solution of the following OCP:

$$\begin{cases} \min\limits_{\boldsymbol{u}} & \sum\limits_{i=0}^{N-1} \ell(\boldsymbol{x}_i, \boldsymbol{u}_i) + \ell_N(\boldsymbol{x}_N) \\ \text{s.t.} & \boldsymbol{x}_0 = \hat{\boldsymbol{x}}_k \\ & \boldsymbol{x}_{i+1} = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{p}_c) & \forall i \in \mathbb{I}_0^{N-1} \\ & \boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{u}_i) \leq 0 & \forall i \in \mathbb{I}_0^{N-1} \\ & \boldsymbol{u}_{\min} + \boldsymbol{\rho}_i^{\boldsymbol{u}} \leq \boldsymbol{u}_i \leq \boldsymbol{u}_{\max} - \boldsymbol{\rho}_i^{\boldsymbol{u}} & \forall i \in \mathbb{I}_0^{N-1} \end{cases} \tag{39}$$

with the cost function being either (35) or (36), the quadrotor dynamic model discretized from (34), the task constraint (38) for the second scenario, and the input saturation constraints for both scenarios.

For each scenario, we performed a comparative analysis of the performance between the proposed ST-MPC and a standard (non-robust) MPC controller denoted in the following as *standard MPC*, obtained by neglecting the input tubes (i.e., by setting $\boldsymbol{\rho}_i^{\boldsymbol{u}} = \mathbf{0}$). This is meant to highlight the benefits of considering the additional constraint restrictions from the tubes on the performance and safety of the system.

Moreover, we are also interested in showing how the use of the full sensitivity analysis of the KKT conditions described in Section III-B — which accounts for the presence of active state constraints — is crucial for the success of the ST-MPC scheme. In fact, one might wonder whether the use of an *unconstrained approximation* of the feedback action of the controller, similar to applying a pre-stabilizing action like the methods using an unconstrained ancillary controller (cfr. Sect. I), could already provide enough information to compute the tubes. We then build such an approximation by neglecting all the active inequality constraints in the KKT conditions (26) and by computing the feedback gains (31) from this reduced system. In analogy to the iterative Linear Quadratic Regulator (iLQR), which yields similar gains over the prediction horizon, we then denote this variant as *ST-MPC-LQR*. We stress that this ST-MPC-LQR variant is introduced only as a mean to provide an additional baseline for comparison in the second test scenario, where accounting for the positional constraint proves to be essential to obtain the maximum performance. On the other hand, since the ST-MPC-LQR variant is essentially equivalent to ST-MPC when only input constraints are present, we ignore it for comparison in the first test scenario.

In addition to the aforementioned standard non-robust MPC controller, we also performed a comparison against zoRO [26]–[28], another state-of-the-art robust MPC algorithm. Similarly to our approach, these works propose an approximate solution to robust optimal control problems based on ellipsoidal uncertainty sets that are propagated around the initial guess, yielding fixed backoff terms in the constraints. The main difference with our work lies in the uncertainty propagation and in how the uncertainty itself is modeled. In fact, the zoRO MPC problem takes the same form as in (39) but with the equivalent of our tube radius computed (in the case of input constraints, for instance) as

$$\rho_{j,k}^{\boldsymbol{u}} = \sqrt{\boldsymbol{e}_j^T \boldsymbol{K} \boldsymbol{\Sigma}_k \boldsymbol{K}^T \boldsymbol{e}_j}$$

instead of (19), with

$$\boldsymbol{\Sigma}_{k+1} = (\boldsymbol{A}_k + \boldsymbol{B}_k \boldsymbol{K}) \boldsymbol{\Sigma}_k (\boldsymbol{A}_k + \boldsymbol{B}_k \boldsymbol{K})^T + \boldsymbol{N}_k, \quad \boldsymbol{\Sigma}_0 = \mathbf{0}, \tag{40}$$

where $\boldsymbol{K}$ is a *pre-computed* feedback gain matrix and $\boldsymbol{N}_k$ a positive-definite additive noise covariance-like matrix.

In order to perform a fair comparison and at the same time try to replicate a suitable implementation of zoRO, we have chosen as gain matrix $\boldsymbol{K}$ the equivalent MPC gain $\boldsymbol{F}_{0|0}$ of our MPC controller computed while regulating around the hovering condition (to mimic an LQR controller), and we have set $\boldsymbol{N}_k = \boldsymbol{M}_k \boldsymbol{W} \boldsymbol{M}_k^T$ to consider a structured uncertainty as close as possible to the one introduced by the parameter perturbations. It should be noted that the performance of zoRO can certainly be affected (positively or negatively) by these user-defined design parameters: this is, in our opinion, another disadvantage when compared to our proposed method for which the selection of similar quantities is automatic or guided by physical considerations (e.g., range of variation of model parameters).

D. Simulation results

We now present some statistical results to illustrate how the introduction of the input tubes improves the tracking performance, the feasibility, and thus the safety of the system. The simulations have been performed by integrating numerically the quadrotor dynamics (34) with various combinations of the parameters $\boldsymbol{p}$ and the controller running at 50 Hz.

The accompanying video provides a visualization of the various simulations.

1) Tracking of aggressive trajectories: Two trajectories, denoted as A and B in Fig. 4 and Tab. V and having an average speed of circa 1 m/s and 1.5 m/s respectively, have been tracked repeatedly by selecting 450 parameter perturbations $\Delta \boldsymbol{p} \in \mathcal{P}$ (see (11) in a grid, with $\Delta \boldsymbol{p}_{\max} = (0.04, 0.04, 0.02, 0.1)$, and by employing either the ST-MPC (with input tubes), the standard MPC (without input tubes), or the zoRO controllers. Note that some of the parameter deviations generated in $\mathcal{P}$ will lie outside the corresponding ellipsoid $\mathcal{E}_{\boldsymbol{p}}$. Therefore, the effect of their perturbation on the closed-loop trajectory could be underestimated by the input tubes (which assume a parametric uncertainty lying in $\mathcal{E}_{\boldsymbol{p}}$). This does not constitute an issue in this tracking task where only the tracking error is affected by the uncertainty, but it will have an impact in the second test scenario which also involves the feasibility of the motion (Sect. IV-D2).

For each trajectory, we further consider the case where the desired yaw is kept constant $\psi_d(t) = 0$ (cases "A" and "B" in Fig. 4 and Tab. V) and where the yaw has to track a desired trajectory $\psi_d(t) = \operatorname{atan2}(\dot{\boldsymbol{y}}_d(t), \dot{\boldsymbol{x}}_d(t))$ (cases "A-yaw" and "B-yaw" in Fig. 4 and Tab. V).

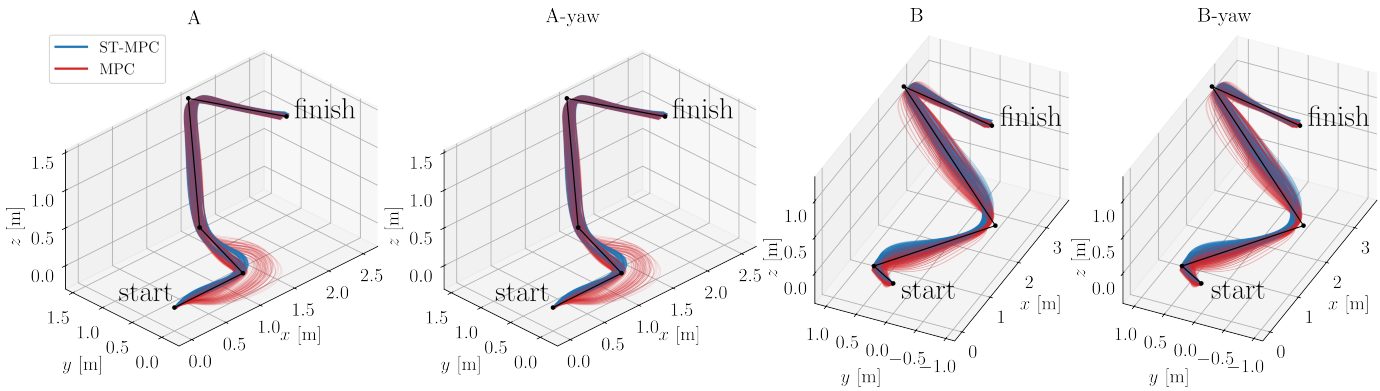

Fig. 4. Spread of trajectories obtained for different parameter perturbations while tracking the desired trajectory (with and without yaw). Note how the proposed ST-MPC method (blue) yields a tighter spread than a standard MPC controller (red). Trajectories for zoRO are not included for the sake of legibility (since they would appear in-between non-robust MPC and ST-MPC), but synthetic results can be found in Tab. V and Fig. 6.

TABLE V
POSITION ERROR AND COST FUNCTION WHILE TRACKING DISCONTINUOUS TRAJECTORIES

Case	Method	RMSE deviation [m] mean (± std. dev.)	Maximum deviation [m]	Nominal RMSE [m]	Cumul. state cost	Cumul. input cost
A	ST-MPC	**0.049 (±0.018)**	**0.112**	0.078	**4455 (±702)**	**324 (±29)**
	MPC	0.071 (±0.055)	0.697	**0.062**	5502 (±5088)	528 (±281)
	zoRO	0.064 (±0.043)	0.578	0.064	4877 (±3597)	442 (±159)
A-yaw	ST-MPC	**0.049 (±0.018)**	**0.126**	0.079	**5294 (±768)**	**337 (±30)**
	MPC	0.072 (±0.056)	0.697	**0.062**	6501 (±5446)	548 (±284)
	zoRO	0.064 (±0.044)	0.579	0.064	5849 (±3939)	459 (±160)
B	ST-MPC	**0.057 (±0.022)**	**0.241**	0.099	8904 (±1062)	**542 (±37)**
	MPC	0.079 (±0.037)	0.395	**0.070**	8051 (±2401)	1008 (±175)
	zoRO	0.070 (±0.033)	0.336	0.073	**7640 (±1826)**	803 (±114)
B-yaw	ST-MPC	**0.058 (±0.025)**	**0.226**	0.099	10004 (±1187)	**555 (±38)**
	MPC	0.080 (±0.039)	0.414	**0.070**	9211 (±2510)	1032 (±179)
	zoRO	0.071 (±0.033)	0.348	0.073	**8764 (±1911)**	819 (±114)

Figure 4 visually depicts how, in particular for the case "A", the spread of trajectories obtained with ST-MPC remains much closer around the nominal behavior, while the standard MPC produces much larger deviations during more aggressive turns. This result is due to the input saturations that affect the quadrotor because of the modeling errors: thanks to the input tubes, these are much better accounted for by ST-MPC which is able to generate a feasible trajectory more robust to model uncertainties while, on the other hand, the standard MPC lacks this inherent robustness and ultimately deviates more from the nominal trajectory. Comparing the two trajectories, for the case "A" the deviation is mostly localized in the first part of the trajectory, where input saturations are concentrated, while for the faster case "B" deviations are sustained over the entirety of the trajectory. As expected, when saturations are not involved in the motion generation, ST-MPC and MPC have a similar behavior in the proposed scenario.

Fig. 5 shows an example of the resulting input trajectory for a particular parameter deviation $\Delta \boldsymbol{p} = (0.02, -0.02, 0.0, 0.05)$. Note how, after the first instants, the standard MPC saturates three out of the four control inputs, resulting in a large deviation from the nominal trajectory, measured with the Cartesian position error $\boldsymbol{e_r} = \boldsymbol{r} - \boldsymbol{r}_{\text{nom}}$, with $\boldsymbol{r}_{\text{nom}}$ being the trajectory[7] obtained by performing the simulation with $\boldsymbol{p} = \boldsymbol{p}_c$. On the other hand, ST-MPC minimizes the deviation while still being able to utilize the full actuation capabilities. This is also due to the fact that the input tube $\boldsymbol{\rho^u}$ is by construction zero for the first input $\boldsymbol{u}_0$ of the predicted trajectory. In fact, the tubes which encode the uncertainty over the time horizon $t \in [t_k,\ t_{k+N}]$ starting from the current state $\hat{\boldsymbol{x}}_k$ are always re-computed around the current predicted trajectory $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})$: since the state sensitivity $\boldsymbol{\Pi}_0$ of the predicted trajectory is always zero at t_k, it follows that $\boldsymbol{\rho}_0^{\boldsymbol{u}} = \boldsymbol{0}$ as well. Therefore, for what concerns the first input $\boldsymbol{u}_0$ at time t_k, the input constraint always reduces to $\boldsymbol{u}_{\min} \leq \boldsymbol{u}_0 \leq \boldsymbol{u}_{\max}$.

Quantitative results can be found in Tab. V, where we also report the cumulative state and input costs over the trajectories. In all cases, the Root Mean Square Error (RMSE) deviation with respect to the nominal trajectory $\boldsymbol{r}_{\text{nom}}$ is smaller in mean and standard deviation for ST-MPC. Similarly, the maximum deviation is about 2-6 times larger for the standard MPC, confirming the better performances of ST-MPC. Finally, the reduced spread of trajectories is translated into a smaller standard deviation of the state cost. The reduction in control

[7]Note that, due to the receding-horizon nature of MPC, in general, this nominal trajectory differs from the nominal predicted trajectory $\bar{\boldsymbol{x}}$.

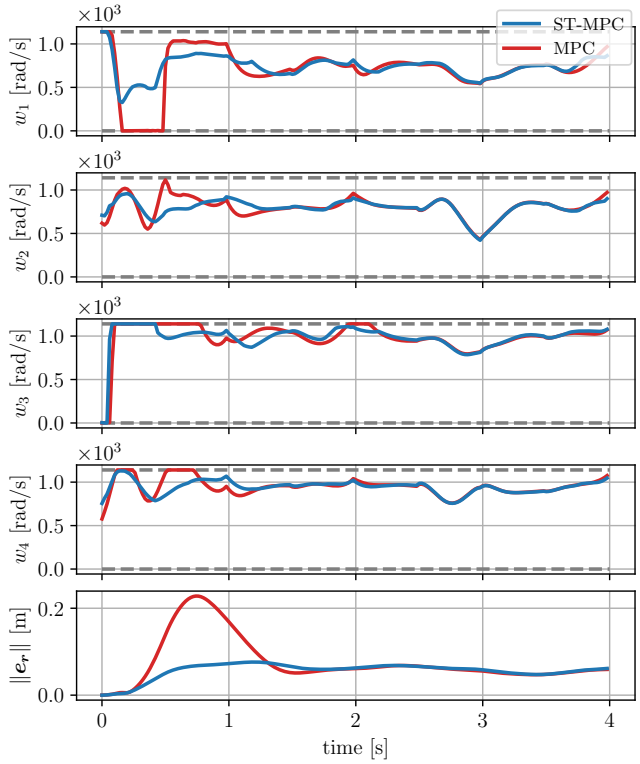

Fig. 5. Commanded propeller speeds (top) and norm of the deviation from the nominal trajectory (bottom) while tracking trajectory A with $\Delta \boldsymbol{p} = (0.02, -0.02, 0.0, 0.05)$. Dashed grey lines indicate the rotor speed limits. Note how even with the input restriction introduced by the tubes, ST-MPC fully utilizes the actuation capabilities of the drone. See the accompanying video for an animation including the predicted trajectory and tubes.

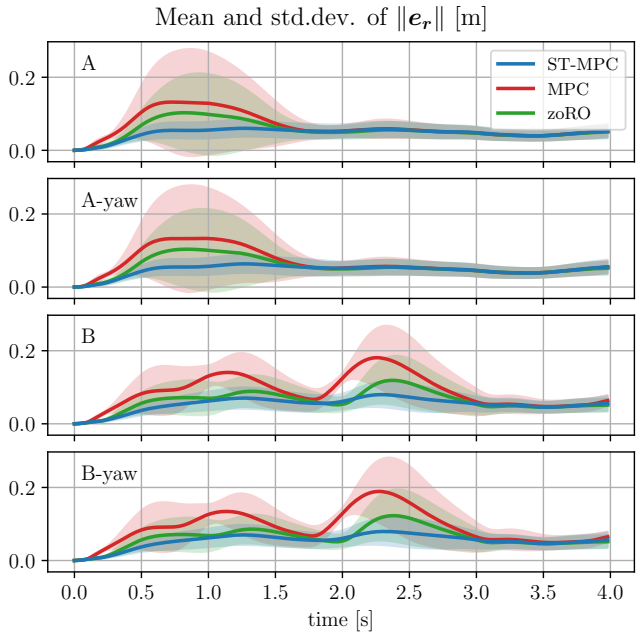

Fig. 6. Evolution of the mean and the standard deviation (shaded) of the position deviation $\|\boldsymbol{e_r}\|$ from the nominal trajectory over the 450 perturbed simulations for the four tested trajectories. Note how the proposed method achieves a smaller deviation in all cases.

saturation shown in Fig. 5 is reflected in the input cost, which is always lower for ST-MPC. The additional propeller speeds required to track a time-varying yaw angle (A-yaw, B-yaw) can indeed lead to an increase in the control effort (input cost). However, we find that this additional effort is not always accompanied by an increase in the maximum deviation, likely due to some particular correlation between the shape of the trajectory and the occurrence of the maximum deviation point in different instances for the two methods. Similar considerations can also be drawn by analyzing Fig. 6 that reports the evolution over time of the deviation from the nominal trajectory $\|\boldsymbol{e_r}\|$. Looking at Fig. 6, it is clear how the standard MPC experiences larger deviations in all cases and during the whole motion. While zoRO offers an advantage compared to the nominal MPC (as expected), it does not match the performance of our method. This is due to the two different uncertainty propagations — (8)-(9) for ST-MPC and (40) for ZoRO. The tubes computed in ZoRO can be smaller (in radius) given the same range of model uncertainty, because of the assumption of the disturbance *trajectory* lying in a high-dimensional ellipsoid (cf. [26, Remark 2]). Finally, we note from the fifth column of Tab. V how the RMSE for the tracking of the desired position $\boldsymbol{r}_d$ in the *nominal* (unperturbed) case is larger for ST-MPC: this is expected and due to the trade-off between performance and robustness introduced by the constraint restriction. Still, the overall performance in any non-nominal (real-world) case is in clear favor of ST-MPC.

2) Passing through a narrow aperture: For this scenario, we performed repeated tests with 450 parameter perturbations $\Delta \boldsymbol{p} \in \mathcal{P}$ in a grid with $\Delta \boldsymbol{p}_{\max} = (0.02, 0.02, 0.01, 0.1)$. As the parameter variations make the system deviate from the predicted trajectory, the MPC will have to progressively compensate to satisfy — at least in its prediction — the safety constraint. Since the system possesses limited control authority at any given time, the unforeseen deviation from the plan might lead it to enter states from which no feasible solution exists, even for the nominal prediction model. This emerges as the main cause of failure for the controller, which has no recursive feasibility guarantee [53]. We therefore assess the effectiveness of the proposed method using two metrics:

- The success rate, defined as the percentage of tests in which the quadrotor is able to safely reach the goal with no collision[8];
- The time t_p at which the quadrotor has completely passed the aperture, in order to measure the possible motion conservativeness introduced by adding the tube radii restriction on the input constraints. Here we consider only simulations that are successful for both the Robust MPC (ST-MPC, ST-MPC-LQR, or zoRO, alternatively) and the standard MPC[9]

To analyze the results, we distinguish between two different subsets of parameters: (i) $\mathcal{P}_{\text{ie}}$ when $\Delta \boldsymbol{p}$ is inside of the

[8]We consider as failed any test in which either the closed-loop trajectory does not satisfy the constraints or the MPC problem becomes infeasible.

[9]This choice, aimed at performing a one-to-one comparison between the methods, has been found to be slightly pejorative for the Robust MPC, due to the exclusion of many data points in which it succeeds while standard MPC does not.

TABLE VI
SUCCESS RATE AND TIME REQUIRED TO PASS THROUGH THE APERTURE

Method	Settings		Success rate [%]			Time t_p [s]
	N	Q_v	$\mathcal{P}_{\text{ie}}$	$\mathcal{P}_{\text{nie}}$ ($\mathcal{P}_{\text{nie,nbs}}$)	total	mean ($\pm$ std. dev.)
ST-MPC	25	$2 \cdot 10^{-2}$	90.5	62.9 (32.6)	75.8	0.677 ($\pm$0.032)
		$5 \cdot 10^{-2}$	98.6	83.7 (66.4)	90.7	0.731 ($\pm$0.039)
	35	$2 \cdot 10^{-2}$	99.0	91.7 (83.9)	95.1	0.722 ($\pm$0.030)
		$5 \cdot 10^{-2}$	**100**	**96.2 (92.4)**	**98.0**	0.734 ($\pm$0.034)
MPC	25	$2 \cdot 10^{-2}$	59.0	45.4 (0.0)	51.8	**0.652 ($\pm$0.044)**
		$5 \cdot 10^{-2}$	68.6	52.1 (0.0)	59.8	0.695 ($\pm$0.049)
	35	$2 \cdot 10^{-2}$	64.8	48.3 (0.0)	56.0	0.673 ($\pm$0.040)
		$5 \cdot 10^{-2}$	67.6	50.4 (0.8)	58.4	0.682 ($\pm$0.041)
ST-MPC-LQR	25	$2 \cdot 10^{-2}$	75.2	54.6 (16.8)	64.2	0.668 ($\pm$0.031)
		$5 \cdot 10^{-2}$	94.8	69.2 (35.7)	81.1	0.724 ($\pm$0.036)
	35	$2 \cdot 10^{-2}$	99.0	79.2 (59.7)	88.4	0.705 ($\pm$0.030)
		$5 \cdot 10^{-2}$	98.5	83.3 (66.7)	90.4	0.720 ($\pm$0.032)
zoRO	25	$2 \cdot 10^{-2}$	67.1	51.7 (12.8)	58.9	0.662 ($\pm$0.033)
		$5 \cdot 10^{-2}$	78.6	56.3 (8.7)	66.7	0.707 ($\pm$0.039)
	35	$2 \cdot 10^{-2}$	76.2	55.8 (14.5)	65.3	0.687 ($\pm$0.031)
		$5 \cdot 10^{-2}$	82.9	57.1 (14.2)	69.1	0.700 ($\pm$0.033)

ellipsoid $\mathcal{E}_{\boldsymbol{p}}$, and (ii) $\mathcal{P}_{\text{nie}}$ when it is outside. Moreover, we denote as $\mathcal{P}_{\text{nie,nbs}}$ the subset of $\mathcal{P}_{\text{nie}}$ in which at least one of the two controllers fails. The reason for introducing this distinction is that, in some instances, even a large parameter variation lying outside the ellipsoid could result in a motion for which the task is only slightly affected by the disturbance. These favorable configurations result in both controllers completing the task but are not necessarily indicative of the effect of the input tubes on the success rate.

The MPC design is based on nominal performance, i.e., the tuning of the cost function weights is performed by trial and error simulating the nominal system dynamics. Also, the choice of the control horizon N naturally affects the performance and feasibility even in the nominal case, although it is typically restricted by the real-time requirements of the platform. Nonetheless, it is interesting to analyze the increase (or lack thereof) in robustness related to changes in these settings, as they can greatly affect both the nominal and the perturbed behaviors of the system. To this end, Tab. VI reports statistics on the success rates and times t_p with different settings for the velocity weight $\boldsymbol{Q}_v = Q_v \boldsymbol{I}_3$ and the control horizon N. Four different methods are compared: (i) ST-MPC, i.e., the proposed method, (ii) a standard MPC, (iii) ST-MPC-LQR, which is obtained, as explained, by neglecting all constraints in the MPC gains computation, (iv) and zoRO, which uses an alternative tube radius and uncertainty propagation.

From the analysis of Tab. VI, we can appreciate how the introduction of the input tubes in ST-MPC is always followed by a significant improvement in robustness. In fact, the success rate increases from the 60-70% range for the standard MPC to 90-100% for our approach when the parameter deviation is in $\mathcal{E}_{\boldsymbol{p}}$ (case $\mathcal{P}_{\text{ie}}$). Moreover, when the parameter deviation falls outside the ellipsoid (cases $\mathcal{P}_{\text{nie}}$ and $\mathcal{P}_{\text{nie,nbs}}$), all methods experience a lower success rate, confirming that the satisfaction of the ellipsoid condition is important for the task success. This also shows that the approximation (14) used for deriving the tubes is, in fact, not too conservative since the effect of parameter deviations lying outside the ellipsoid (12) are (correctly) not captured by the tubes.

Increasing the control horizon N from 25 to 35 leads to a marginal increase in the success rate for both ST-MPC and the standard MPC. However, while ST-MPC reaches almost 100% success rate, the standard MPC is limited to less than 60%, indicating that the longer control horizon is not the main factor for substantially improving the recursive feasibility of the system. Similarly, the increase of the velocity weight Q_v from $2 \cdot 10^{-2}$ to $5 \cdot 10^{-2}$ does not provide significant improvements, although it would intuitively yield slower and more conservative trajectories (as confirmed by the statistics on t_p), and thus appear as an almost obvious means to improve safety. In all cases, while zoRO improves over the non-robust MPC, it is not able to match the performance of ST-MPC due to its different uncertainty propagation that does not capture as well the effects of parametric uncertainties of the model.

Concerning the performance in terms of speed and agility, we evaluate the time t_p at which the quadrotor clears the aperture, independently of the fact that the parameter variation is inside the ellipsoid or not. Although ST-MPC does indeed generate a slightly more conservative motion with a $\sim 6\%$ increase (at most) in the average t_p, this is met with an important improvement in the success rate, as already discussed. Moreover, we note how for at least one setting ($N = 25$, $Q_v = 2 \cdot 10^{-2}$) ST-MPC has a performance comparable to that of the standard MPC while still providing more safety.

Lastly, the direct comparison between ST-MPC and its simplified variant ST-MPC-LQR shows how, for $N = 25$, the latter is not able to perform as well as ST-MPC with a total success rate reduced by 10% in all instances and with a 15% reduction in the success rate for $\mathcal{P}_{\text{ie}}$ in the $Q_v = 2 \cdot 10^{-2}$ setting. This is expected since ST-MPC-LQR does not account for the active obstacle avoidance constraints (37) when computing the MPC gains, which does not allow to correctly capture the behavior of the MPC in the presence of active constraints. For $N = 35$, the performance of ST-MPC-LQR almost matches

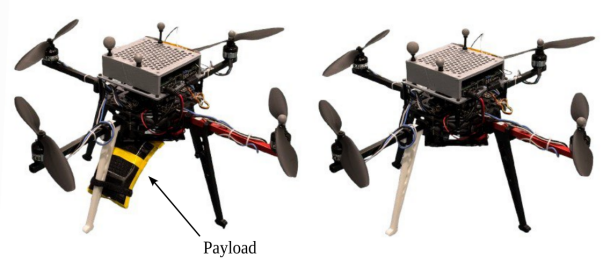

Fig. 7. The quadrotor used in the experiments, with and without the payload. During each experiment, the position of the payload was changed to perturb the physical parameters of the robot.

that of ST-MPC, which is expected thanks to the naturally larger tubes resulting from the longer horizon. Still, this comes at the cost of an increased computation time with respect to, for instance, ST-MPC with $N = 25$ and $Q_v = 5 \cdot 10^{-2}$. These results then further motivate the use of the proposed ST-MPC scheme.

In conclusion, it is clear how the proposed method can provide a substantial improvement on the safety of the system with parameter uncertainty, which can be traded for a reduction in performance (in terms of speed to execute the task), while standard ways of robustifying the MPC, such as tuning the cost function or increasing the control horizon, are not able to yield the same results.

E. Experimental validation

The proposed approach has also been validated experimentally on a quadrotor performing the aggressive trajectory tracking task. The goal was to demonstrate the practical feasibility and computational efficiency of the algorithm (which is completely run on the onboard hardware) and also assess how the numerical results would translate to real scenarios. Clips from the experiments and animations are available in the accompanying video.

The employed quadrotor, shown in Fig. 7, is based on the MikroKopter platform and runs custom firmware to allow closed-loop rotor speed control. The MPC controller is fully executed *onboard* on an Intel NUC (i7-1360P CPU) at 100 Hz and makes use of the Telekyb3 framework to interface with the hardware. An Unscented Kalman Filter implemented in Telekyb3 is used to fuse the IMU measurements and the motion capture position feedback to obtain an estimate of the state $\hat{\boldsymbol{x}}_k$. The same implementation used in simulation has been employed in the experiments, commanding rotor speeds without a lower-level controller, with only minor differences in parameters and in the hardware interface. All parameters and MPC settings are reported in Tab. II and Tab. III.

Parameter deviations are applied to the robot by physically attaching a payload of mass $m_p = 0.236$ kg to the bottom plate and landers at different places for each run in order to perturb the total mass, inertia, and position of the CoM along the 3 axes (see Fig. 7). Note that, contrarily to the simulation case, the real parameters of the robot are *unknown* and therefore there does not exist a *nominal* behavior against which each experiment can be compared. We instead tested for a given

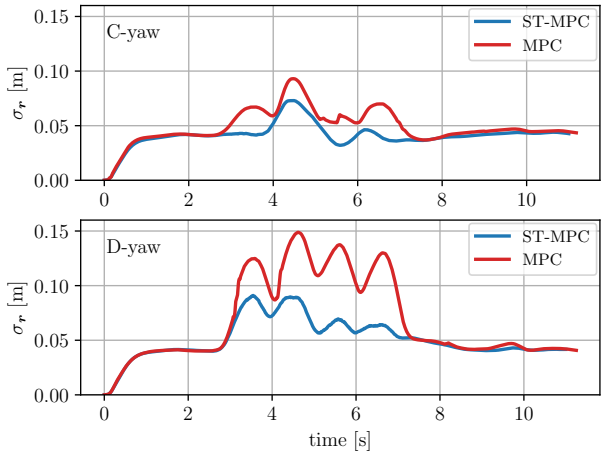

Fig. 8. Norm of the standard deviation of the quadrotor position over multiple experiments with different perturbations applied to the robot.

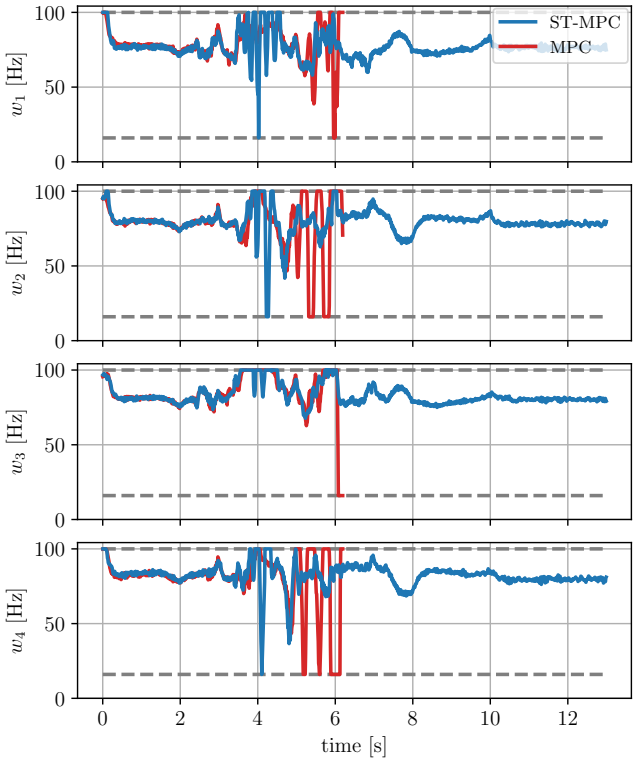

Fig. 9. Commanded propeller speeds during the experiment in which MPC resulted in a crash. Note the saturations and induced oscillations in the control signal.

perturbation both the proposed method and the standard MPC controller and analyzed the standard deviation of the robot position over time w.r.t. the average of the various runs. This allows us to compare the *repeatability* of the motion produced by ST-MPC vs. the standard MPC over the perturbed runs.

We performed 12 different experiments while tracking a trajectory with an average speed of circa 2 m/s (case C-yaw). At each time instant, we evaluate the norm of the standard deviation of the position components of the resulting trajectories, i.e., $\sigma_{\boldsymbol{r}}(t) = \sqrt{\sigma_x^2(t) + \sigma_y^2(t) + \sigma_z^2(t)}$. Results

are reported in Fig. 8 (top), highlighting how ST-MPC can provide an overall safer and more repeatable behavior with smaller deviation over different perturbations, confirming the numerical results of Sect. IV-D1.

Five additional experiments on a similar but more aggressive trajectory with an average speed of circa 2.5 m/s (case D-yaw) have also been conducted. In this case, in one instance of the perturbation, the robot has crashed when controlled with the standard MPC, which has become unstable after incurring severe input saturation, as shown in Figure 9. On the other hand, with the exact same perturbation, ST-MPC has been able to safely execute the motion. Finally, note how in this more aggressive scenario the maximum standard deviation increases to more than 15 cm for the standard MPC, while remaining under 9 cm for ST-MPC, see Fig. 8 (bottom).

While the reported experiments only consider the trajectory tracking scenario, we expect an analogous higher repeatability of ST-MPC to also apply to the success rate in performing tasks like dynamic obstacle avoidance and passing through a narrow gap, in line with the numerical results of Sect. IV-D2.

F. Implementation details and computational complexity

The proposed method has been implemented in C++ by making use of the autodiff library [54] for Automatic Differentiation (AD) and the ProxQP sparse solver [55]. To solve the linear system (27) yielding (28) we employ Eigen's linear solver to perform a sparse LU factorization. Considering the most demanding scenario (passing through the aperture) with $N = 25$, one iteration of the algorithm runs in real-time at more than 50 Hz ($\delta_t = 0.02$ s) on an Intel Core i7-13700H laptop CPU, with the MPC gains and tubes computation only requiring 3-4 ms, compared to the 1-10 ms to solve the QP during the feedback phase. Being the structure of the OCP and the number of decision variables and constraints the same, the only additional computational cost of ST-MPC compared to the standard MPC is due to the computation of the tubes. In the experiments, the controller is able to run at 100 Hz on the onboard computer. Figure 10 reports CPU times for the different phases of the MPC iteration for different control horizon lengths N. Specifically, it can be seen how the preparation phase time, including the tubes' computation for ST-MPC, grows linearly with the horizon length thanks to the use of sparsity exploiting solvers. Since the QP has the same number of constraints and decision variables as a standard MPC, the impact of the additional computations in ST-MPC is practically negligible. Therefore, if a standard MPC is amenable to real-time implementation, its ST-MPC version (with the constraint restrictions from the tubes) will also typically be implementable in real-time. This is, in our opinion, a major strength of the proposed approach: ST-MPC represents a viable alternative to MPC schemes with an added inherent robustness to model uncertainties. Moreover, since the tubes are obtained during the preparation phase, the computational cost of their evaluation is not added to the delay introduced by the MPC controller due to the feedback phase.

Finally, one could further improve the current sensitivity computation by more tightly integrating it with the QP solver

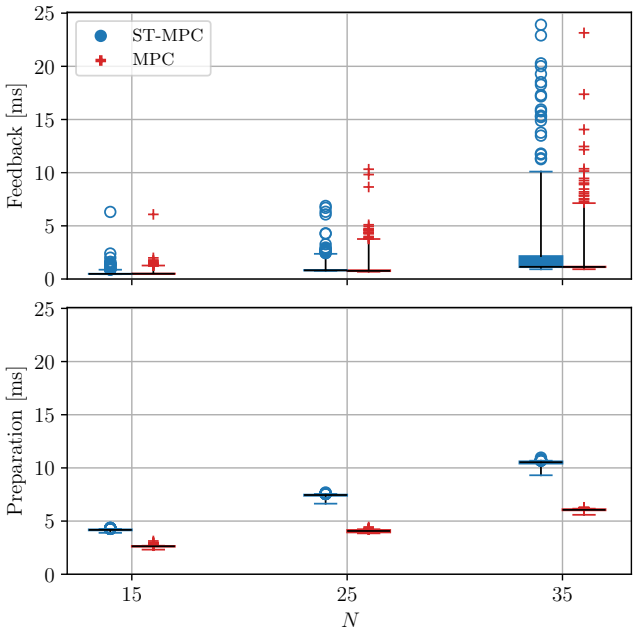

Fig. 10. CPU time of *feedback* and *preparation* phases for different control horizon lengths, represented as a box plot (whiskers include the 90th percentile).

for reusing the KKT factorization performed by the solver itself or by using differentiable QP solvers [56]. Additionally, it could be possible to exploit the structure of the optimal control problem to compute the MPC gains more efficiently by adapting, for instance, the work proposed in [39] or [57]. The sensitivity propagation can, on the other hand, reuse the derivatives of the dynamics that have been computed to prepare the QP. Compared to the current implementation based on AD, significant gains in the computation of the derivatives of the dynamics could be expected by exploiting analytical derivatives computed from libraries such as Pinocchio [58], particularly if more complex articulated robots are considered.

V. Conclusions and Future Work

We have proposed an efficient Robust Model Predictive Control algorithm, denoted ST-MPC, for robots affected by parametric uncertainties, and showed its effectiveness in improving the tracking performance and the success rate during navigation in a tight environment. By leveraging the notion of *closed-loop sensitivity* and ellipsoidal tubes for enveloping the perturbed trajectories, we were able to introduce a time-varying restriction on the input constraints to make, at each instant, the MPC controller aware of the possible additional input requirements needed to cope with parametric uncertainties. The resulting ST-MPC has the same computational complexity as a standard MPC, only adding the computation of the MPC gains and tube propagation during consecutive control instants by analyzing the sensitivity of the previous MPC solution. Since this computation is performed in the *preparation* phase, it does not introduce any additional delay.

Being, to the best of our knowledge, the first *online* application of robust trajectory optimization based on the closed-loop

sensitivity, we plan to extend the method in several possible directions. For instance, by including an online optimization of a sensitivity metric for generating trajectories that are minimally sensitive and, thus, reduce the tube radius, or by adding online parameter estimation schemes in the sensitivity calculation for updating the nominal parameter values and thus reduce the conservativeness of the motion. From a computational point of view, a tight integration with the QP solver of choice would allow to streamline the calculation of the sensitivity, strengthening even more the ability to apply our method at a very little computational cost.

APPENDIX A
EFFICIENT COMPUTATION OF THE MPC GAINS OVER THE PREDICTION HORIZON

In the computation of the sensitivity of the MPC — described in Sect. III-B — the sensitivity of the whole predicted trajectory $(\boldsymbol{x}, \boldsymbol{u})$ is computed with respect to the initial state $\hat{\boldsymbol{x}}_k$ (or, equivalently, $\boldsymbol{x}_0$). We now show how the same reasoning applies to all $\boldsymbol{x}_i$, and how the sensitivities for $i = 1, \dots, N-1$ can be computed without performing any additional KKT factorization and inversion.

First, note that with a simple rearrangement of variables, one can apply the implicit function theorem on the KKT system (26) with respect to the explicit variable $\boldsymbol{x}_i$. Defining $\boldsymbol{z} = (\boldsymbol{x}_0, \dots, \hat{\boldsymbol{x}}_k, \dots, \boldsymbol{x}_N, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ by replacing $\boldsymbol{x}_i$ with $\hat{\boldsymbol{x}}_k$, one can then obtain

$$\begin{aligned} \boldsymbol{\mathcal{R}}(\boldsymbol{z}(\boldsymbol{x}_i), \boldsymbol{x}_i)|_{\boldsymbol{z}^*} &= \boldsymbol{0} \\ \frac{d}{d\boldsymbol{x}_i} \boldsymbol{\mathcal{R}}(\boldsymbol{z}(\boldsymbol{x}_i), \boldsymbol{x}_i)|_{\boldsymbol{z}^*} &= \boldsymbol{0} \\ \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{z}} \frac{d\boldsymbol{z}}{d\boldsymbol{x}_i} + \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{x}_i} &= \boldsymbol{0}, \end{aligned} \tag{41}$$

from which it follows

$$\frac{d\boldsymbol{z}}{d\boldsymbol{x}_i} = -\left(\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{z}}\right)^{-1} \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{x}_i}.$$

We now show how, leveraging the availability of the inverse of the KKT matrix $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{y}}$, the sought quantities can be computed without the need of inverting the large matrix $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{z}}$ for each i. Noting that $\frac{d\boldsymbol{z}}{d\boldsymbol{x}_i}$ contains $\frac{d\boldsymbol{u}_i}{d\boldsymbol{x}_i}$, we define the selection matrix $\boldsymbol{v}_{u_i}$ such that

$$\tilde{\boldsymbol{F}}_{k+i|k+i} = \frac{d\boldsymbol{u}_i}{d\boldsymbol{x}_i} = \boldsymbol{v}_{u_i} \frac{d\boldsymbol{z}}{d\boldsymbol{x}_i}. \tag{42}$$

Since $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{x}_i}$ is the i-th n_x-dimensional column block of the KKT matrix (29), we can also define $\boldsymbol{v}_{x_i}$ such that

$$\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{x}_i} = \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{y}} \boldsymbol{v}_{x_i}.$$

The matrix $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{z}}$ can be seen as equivalent to $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{y}}$ where column-block $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{x}_i}$ has been swapped for $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \hat{\boldsymbol{x}}_k}$. Therefore, we can rewrite

$$\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{z}} = \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{y}} + \left(\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \hat{\boldsymbol{x}}_k} - \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{x}_i}\right) \boldsymbol{v}_{x_i}^T,$$

which makes it possible to compute the inverse of $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{z}}$ by only computing a rank-n_x correction via the Woodbury matrix identity [47]. Define

$$\boldsymbol{\mathcal{K}} = \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{y}}, \quad \boldsymbol{\kappa}_i = \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{x}_i}, \quad \boldsymbol{c} = \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \hat{\boldsymbol{x}}_k},$$

and $\boldsymbol{d}_i = \boldsymbol{c} - \boldsymbol{\kappa}_i$. Then

$$\begin{aligned} \left(\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{z}}\right)^{-1} &= \left(\boldsymbol{\mathcal{K}} + \boldsymbol{d}_i \boldsymbol{v}_{x_i}^T\right)^{-1} \\ &= \boldsymbol{\mathcal{K}}^{-1} - \boldsymbol{\mathcal{K}}^{-1} \boldsymbol{d}_i \left(\boldsymbol{I}_n + \boldsymbol{v}_{x_i}^T \boldsymbol{\mathcal{K}}^{-1} \boldsymbol{d}_i\right)^{-1} \boldsymbol{v}_{x_i}^T \boldsymbol{\mathcal{K}}^{-1}. \end{aligned} \tag{43}$$

We can now combine (41), (42) and (43) to find

$$\begin{aligned} \frac{d\boldsymbol{u}_i}{d\boldsymbol{x}_i} =& - \boldsymbol{v}_{u_i} \boldsymbol{\mathcal{K}}^{-1} \boldsymbol{\kappa}_i \\ &+ \boldsymbol{v}_{u_i} \boldsymbol{\mathcal{K}}^{-1} \boldsymbol{d}_i \left(\boldsymbol{I}_n + \boldsymbol{v}_{x_i}^T \boldsymbol{\mathcal{K}}^{-1} \boldsymbol{d}_i\right)^{-1} \boldsymbol{v}_{x_i}^T \boldsymbol{\mathcal{K}}^{-1} \boldsymbol{\kappa}_i. \end{aligned} \tag{44}$$

Note that, by construction

$$\begin{aligned} \boldsymbol{\mathcal{K}}^{-1} \boldsymbol{\kappa}_i &= \boldsymbol{v}_{x_i}, \\ \boldsymbol{v}_{u_i} \boldsymbol{\mathcal{K}}^{-1} \boldsymbol{\kappa}_i &= \boldsymbol{v}_{u_i} \boldsymbol{v}_{x_i} = \boldsymbol{0}, \\ \boldsymbol{v}_{x_i}^T \boldsymbol{v}_{x_i} &= \boldsymbol{I}_n. \end{aligned}$$

Then, having defined

$$\begin{aligned} \boldsymbol{F}_{k+i|k} &= \boldsymbol{v}_{u_i} \boldsymbol{\mathcal{K}}^{-1} \boldsymbol{c}, \\ \boldsymbol{P}_{k+i|k} &= \boldsymbol{v}_{x_i}^T \boldsymbol{\mathcal{K}}^{-1} \boldsymbol{c}, \end{aligned}$$

(44) becomes

$$\tilde{\boldsymbol{F}}_{k+i|k+i} = \boldsymbol{v}_{u_i} \boldsymbol{\mathcal{K}}^{-1} \boldsymbol{c} \left(\boldsymbol{v}_{x_i}^T \boldsymbol{\mathcal{K}}^{-1} \boldsymbol{c}\right)^{-1} = \boldsymbol{F}_{k+i|k} \boldsymbol{P}_{k+i|k}^{-1},$$

finally yielding the desired MPC gains. We highlight again how this formula provides an efficient mean for computing $\tilde{\boldsymbol{F}}_{k+i|k+i}$ as it involves the inversion of the *small* $n_x \times n_x$ matrix $\boldsymbol{P}_{k+i|k}$ instead of the inversion of the *large* matrix $\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{z}}$.

REFERENCES

[1] P. Robuffo Giordano, Q. Delamare, and A. Franchi, "Trajectory generation for minimum closed-loop state sensitivity," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 286–293.

[2] P. Brault, Q. Delamare, and P. Robuffo Giordano, "Robust trajectory planning with parametric uncertainties," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 095–11 101.

[3] C. Bohm, P. Brault, Q. Delamare, P. Robuffo Giordano, and S. Weiss, "COP: Control & Observability-aware Planning," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 3364–3370.

[4] S. Wasiela, P. Robuffo Giordano, J. Cortes, and T. Simeon, "A Sensitivity-Aware Motion Planner (SAMP) to Generate Intrinsically-Robust Trajectories," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 12 707–12 713.

[5] A. Srour, A. Franchi, and P. Robuffo Giordano, "Controller and Trajectory Optimization for a Quadrotor UAV with Parametric Uncertainty," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 1–7.

[6] A. Pupa, P. Robuffo Giordano, and C. Secchi, "Optimal energy tank initialization for minimum sensitivity to model uncertainties," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 8192–8199.

[7] A. Afifi, T. Belvedere, A. Pupa, P. Robuffo Giordano, and A. Franchi, "Safe and robust planning for uncertain robots: A closed-loop state sensitivity approach," *IEEE Robotics and Automation Letters*, vol. 9, no. 11, p. 9962–9969, Nov. 2024.

[8] A. Srour, S. Marcellini, T. Belvedere, M. Cognetti, A. Franchi, and P. Robuffo Giordano, "Experimental validation of sensitivity-aware trajectory planning for a quadrotor uav under parametric uncertainty," in *Int. Conf. on Unmanned Aircraft Systems (ICUAS)*. IEEE, June 2024, p. 572–578.

[9] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi, "Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs," *Journal of Intelligent & Robotic Systems*, vol. 100, pp. 1213–1247, 2020.

[10] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, p. 3357–3373, Dec. 2022.

[11] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[12] V. Vulcano, S. G. Tarantos, P. Ferrari, and G. Oriolo, "Safe robot navigation in a crowd combining nmpc and control barrier functions," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 3321–3328.

[13] S. G. Tarantos, T. Belvedere, and G. Oriolo, "Dynamics-aware navigation among moving obstacles with application to ground and flying robots," *Robotics and Autonomous Systems*, vol. 172, p. 104582, 2024.

[14] M. Kanneworff, T. Belvedere, N. Scianca, F. M. Smaldone, L. Lanari, and G. Oriolo, "Task-oriented generation of stable motions for wheeled inverted pendulum robots," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 214–220.

[15] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4730–4737.

[16] J. Carpentier and P.-B. Wieber, "Recent progress in legged robots locomotion control," *Current Robotics Reports*, vol. 2, no. 3, pp. 231–238, 2021.

[17] G. Romualdi, S. Dafarra, G. L'Erario, I. Sorrentino, S. Traversaro, and D. Pucci, "Online non-linear centroidal mpc for humanoid robot locomotion with step adjustment," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 412–10 419.

[18] T. Belvedere, N. Scianca, L. Lanari, and G. Oriolo, "Joint-Level IS-MPC: a whole-body MPC with centroidal feasibility for humanoid locomotion," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 11 240–11 247.

[19] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[20] B. Houska and M. E. Villanueva, *Robust optimization for MPC*. Springer, 2019.

[21] D. Q. Mayne, M. M. Seron, and S. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[22] D. Q. Mayne, E. C. Kerrigan, E. Van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *International journal of robust and nonlinear control*, vol. 21, no. 11, pp. 1341–1353, 2011.

[23] M. Cannon, J. Buerger, B. Kouvaritakis, and S. Rakovic, "Robust Tubes in Nonlinear Model Predictive Control," *IEEE Transactions on Automatic Control*, vol. 56, no. 8, pp. 1942–1947, Aug. 2011.

[24] G. Garimella, M. Sheckells, J. L. Moore, and M. Kobilarov, "Robust obstacle avoidance using tube nmpc." in *Robotics: Science and Systems*, 2018.

[25] J. Köhler, R. Soloperto, M. A. Müller, and F. Allgöwer, "A Computationally Efficient Robust Model Predictive Control Framework for Uncertain Nonlinear Systems," *IEEE Transactions on Automatic Control*, vol. 66, no. 2, pp. 794–801, Feb. 2021.

[26] Y. Gao, F. Messerer, J. Frey, N. v. Duijkeren, and M. Diehl, "Collision-free Motion Planning for Mobile Robots by Zero-order Robust Optimization-based MPC," in *2023 European Control Conference (ECC)*, June 2023, pp. 1–6.

[27] J. Frey, Y. Gao, F. Messerer, A. Lahr, M. Zeilinger, and M. Diehl, "Efficient zero-order robust optimization for real-time model predictive control with acados," in *2024 European Control Conference (ECC)*. IEEE, June 2024, p. 3470–3475.

[28] A. Zanelli, J. Frey, F. Messerer, and M. Diehl, "Zero-Order Robust Nonlinear Model Predictive Control with Ellipsoidal Uncertainty Sets," *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 50–57, 2021.

[29] B. T. Lopez, J.-J. E. Slotine, and J. P. How, "Dynamic Tube MPC for Nonlinear Systems," in *2019 American Control Conference (ACC)*, July 2019, pp. 1655–1662.

[30] X. Feng, S. Di Cairano, and R. Quirynen, "Inexact adjoint-based sqp algorithm for real-time stochastic nonlinear mpc," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6529–6535, 2020.

[31] M. E. Villanueva, R. Quirynen, M. Diehl, B. Chachuat, and B. Houska, "Robust mpc via min–max differential inequalities," *Automatica*, vol. 77, p. 311–321, Mar. 2017.

[32] A. V. Fiacco, *Introduction to sensitivity and stability analysis in non linear programming*. New York: Academic Press, 1983.

[33] A. Shapiro, "Sensitivity Analysis of Nonlinear Programs and Differentiability Properties of Metric Projections," *SIAM J. Control Optim.*, vol. 26, no. 3, pp. 628–645, May 1988, publisher: Society for Industrial and Applied Mathematics.

[34] M. Zanon, V. Kungurtsev, and S. Gros, "Reinforcement Learning Based on Real-Time Iteration NMPC," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5213–5218, 2020.

[35] V. M. Zavala and L. T. Biegler, "The advanced-step nmpc controller: Optimality, stability and robustness," *Automatica*, vol. 45, no. 1, p. 86–93, Jan. 2009.

[36] J. Jäschke, X. Yang, and L. T. Biegler, "Fast economic model predictive control based on nlp-sensitivities," *Journal of Process Control*, vol. 24, no. 8, p. 1260–1272, Aug. 2014.

[37] E. Dantec, M. Taïx, and N. Mansard, "First Order Approximation of Model Predictive Control Solutions for High Frequency Feedback," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4448–4455, Apr. 2022.

[38] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4906–4913.

[39] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 93–100.

[40] D. Van Hessem, C. Scherer, and O. Bosgra, "LMI-based closed-loop economic optimization of stochastic process operation under state and input constraints," in *Proceedings of the 40th IEEE Conference on Decision and Control*, vol. 5. IEEE, 2001, p. 4228–4233.

[41] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," *Fast motions in biomechanics and robotics: optimization and feedback control*, pp. 65–93, 2006.

[42] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on control and optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[43] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear mpc: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, 2020.

[44] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear mpc and moving horizon estimation," in *Nonlinear model predictive control: towards new challenging applications*, L. Magni, D. M. Raimondo, and F. Allgöwer, Eds. Springer, 2009, pp. 391–417.

[45] C. Büskens and H. Maurer, *Sensitivity Analysis and Real-Time Optimization of Parametric Nonlinear Programming Problems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 3–16.

[46] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 2006.

[47] H. V. Henderson and S. R. Searle, "On Deriving the Inverse of a Sum of Matrices," *SIAM Rev.*, vol. 23, no. 1, pp. 53–60, Jan. 1981, publisher: Society for Industrial and Applied Mathematics.

[48] J. Carpentier, R. Budhiraja, and N. Mansard, "Proximal and Sparse Resolution of Constrained Dynamic Equations," in *Robotics: Science and Systems*, July 2021.

[49] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and trends in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[50] D. P. Bertsekas, "On penalty and multiplier methods for constrained minimization," *SIAM Journal on Control and Optimization*, vol. 14, no. 2, pp. 216–235, 1976.

[51] G. Antonelli, E. Cataldi, F. Arrichiello, P. Robuffo Giordano, S. Chiaverini, and A. Franchi, "Adaptive Trajectory Tracking for Quadrotor MAVs in Presence of Parameter Uncertainties and External Disturbances," *IEEE Trans. on Control Systems Technology*, vol. 26, no. 1, pp. 248–254, 2018.

[52] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[53] A. Boccia, L. Grüne, and K. Worthmann, "Stability and feasibility of state constrained MPC without stabilizing terminal constraints," *Systems & Control Letters*, vol. 72, pp. 14–21, 2014.

[54] A. M. M. Leal, "autodiff, a modern, fast and expressive C++ library for automatic differentiation," `https://autodiff.github.io`, 2018.
[55] A. Bambade, S. El-Kazdadi, A. Taylor, and J. Carpentier, "PROX-QP: Yet another Quadratic Programming Solver for Robotics and beyond," in *RSS 2022 - Robotics: Science and Systems*, June 2022.
[56] A. Bambade, F. Schramm, A. Taylor, and J. Carpentier, "Leveraging augmented-lagrangian techniques for differentiating over infeasible quadratic programs in machine learning," in *The Twelfth International Conference on Learning Representations*, 2024.
[57] W. Jallet, E. Dantec, E. Arlaud, N. Mansard, and J. Carpentier, "Parallel and proximal constrained linear-quadratic methods for real-time nonlinear mpc," in *Robotics: Science and Systems*, 2024.
[58] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

Tommaso Belvedere (Member, IEEE) received the M.Sc. and the Ph.D. degrees in Control Engineering in 2020 and 2024, respectively, from Sapienza University of Rome, Italy. He is currently a Postdoctoral Researcher with CNRS-IRISA, Rennes, France. In 2023, he was a Visiting Ph.D. Student at the Rainbow Group, IRISA and Inria, Rennes, France. He has served as Associate Editor for ICRA and IROS conferences. His research interests include planning and control of mobile robots using optimization-based techniques, whole-body control, and uncertainty-aware methods for robust trajectory optimization.

Marco Cognetti (Senior Member, IEEE) is a CPJ (Chaire de Professeur Junior) at LAAS-CNRS and the University of Toulouse. He obtained his Ph.D. in Control Engineering from Sapienza University of Rome in 2016. During his doctoral studies, he was a visiting researcher at the Max Planck Institute for Biological Cybernetics in Tübingen (2011) and at the Robotics Institute – Carnegie Mellon University (2015). After completing his Ph.D., he worked as a postdoctoral researcher at the Robotics Lab of Sapienza University of Rome (2016–2017), followed by a posdoctoral position at CNRS, Irisa, and Inria Rennes (2017–2019). He then joined the University of Oulu as a postdoctoral researcher (2020) before serving as an Assistant Professor at Maynooth University, Ireland (2020-2022). He is currently serving as associate editor for IROS, ICRA, ICUAS, as well as he organized two special issues for RA-L and three workshops at IROS and ICRA. His research focuses on physical human-robot interaction, particularly in the context of drones. He is also working on active state estimation, shared control and autonomy for mobile robotics, whole-body motion planning, and the localization and control of multi-robot systems.

Giuseppe Oriolo (Fellow, IEEE) received his Ph.D. in Control Engineering in 1992 from Sapienza University of Rome, Italy. He is currently with the Department of Computer, Control and Management Engineering (DIAG) of the same university, where he is a Full Professor of Automatic Control and Robotics and the director of the DIAG Robotics Lab. His research interests are in the general area of planning and control of robotic systems. Prof. Oriolo has been Associate Editor of the IEEE Transactions on Robotics and Automation from 2001 to 2005 and Editor of the IEEE Transactions on Robotics from 2009 to 2013. He is a Fellow of the IEEE.

Paolo Robuffo Giordano (Senior Member, IEEE) received the M.Sc. degree in computer science engineering and the Ph.D. degree in systems engineering from the University of Rome "La Sapienza" in 2001 and 2008, respectively. In 2007 and 2008, he spent one year as a Post-Doctoral Researcher with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), and from 2008 to 2012, he was a Senior Research Scientist with the Max Planck Institute for Biological Cybernetics, Tübingen, Germany. He is currently the Senior CNRS Researcher Head of the Rainbow Group, IRISA, and Inria, Rennes, France. He received the 2018 IEEE Robotics and Automation Letters Best Paper Award, he has been Associate Editor of the IEEE Transactions on Robotics from 2012 to 2018 and Editor of the IEEE Transactions on Robotics since 2019. His research interests include modeling, estimation, planning and control for robotic systems