Annual Reviews in Control 53 (2022) 138–146

Contents lists available at ScienceDirect

Annual Reviews in Control

journal homepage: www.elsevier.com/locate/arcontrol

Tutorial article

Safe-by-design planner–tracker synthesis with a hierarchy of system models

Katherine S. Schweidel [a,*], He Yin [a], Stanley W. Smith [b], Murat Arcak [b]

[a] Department of Mechanical Engineering, University of California, Berkeley, United States of America
[b] Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, United States of America

ARTICLE INFO

Keywords:
Motion planning
Hierarchical control
Sum-of-squares programming
Model Predictive Control

ABSTRACT

We present a safe-by-design trajectory planning and tracking framework for nonlinear dynamical systems using a hierarchy of system models. The planning layer uses a low-fidelity model to plan a feasible trajectory satisfying the planning constraints, and the tracking layer utilizes the high-fidelity model to design a controller that restricts the error states between the low- and high-fidelity models to a bounded set. The simplicity of the low-fidelity model enables the planning to be performed online (e.g. using Model Predictive Control) and the tracking controller and error bound are derived offline (e.g. using sum-of-squares programming). This error bound is then used by the planner to ensure safety for the combined planner–tracker system To provide freedom in the choice of the low-fidelity model, we allow the tracking error to depend on both the states and inputs of the planner. The goal of this article is to provide a tutorial review of this hierarchical framework and to illustrate it with examples, including a design for vehicle obstacle avoidance.

1. Introduction

Modern engineering systems such as autonomous vehicles and unmanned aerial vehicles (UAVs) must operate subject to complex safety and performance requirements in changing environments. Designing controllers that meet such requirements in real-time may be computationally intractable, e.g., due to large system dimension or nonlinearities in a high-fidelity dynamical model of the system. The planner–tracker framework (Herbert etal, 2017; Kousik, Vaskov, Bu, Johnson-Roberson, & Vasudevan, 2020; Rosolia & Ames, 2021; Singh, Chen, Herbert, Tomlin, & Pavone, 2020; Singh, Majumdar, Slotine, & Pavone, 2017; Smith, Yin, & Arcak, 2019; Tedrake, Manchester, Tobenkin, & Roberts, 2010) addresses this challenge with a layered architecture where a lower-fidelity "planning" model is employed for online planning and a "tracking" controller, synthesized offline, keeps the tracking error between the high-fidelity ("tracking") model and the planning model within a bounded set. System safety is then guaranteed if the planner constraints, when augmented by the tracking error bound, lie within the safety constraints.

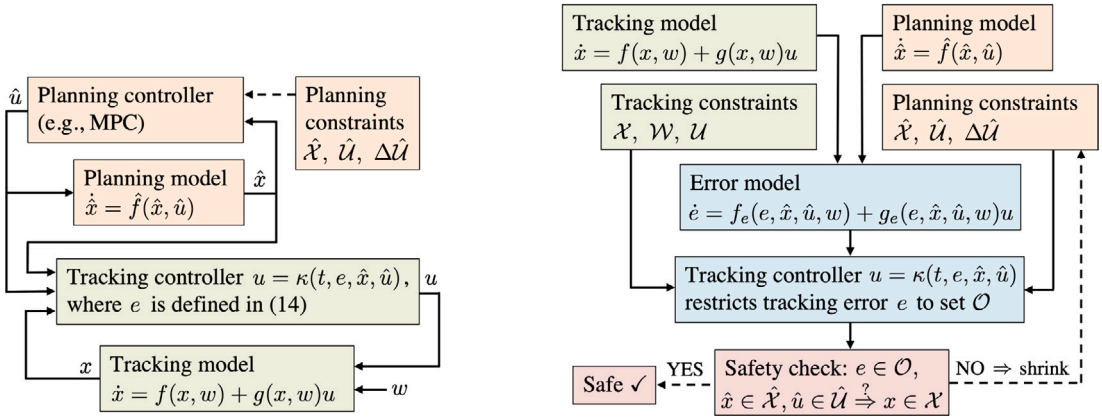
There is a choice to be made when defining the tracking error between the planner and tracker systems. In Singh et al. (2020), Smith et al. (2019), Yin, Bujarbaruah, Arcak, and Packard (2020), the tracking error depends on only the planner/tracker *states*. In Meyer, Yin, Brodtkorb, Arcak, and Sørensen (2020), which studies a ship control problem, the tracking error is generalized to also depend on the planner *input*. This is achieved by accounting for the jumps in the error variable that are induced by jumps in the zero-order hold input between time-steps. Including the planner input in the error definition allows for a lower-order planning model whose states mimic the tracker *position* states while the planner inputs correspond to the tracker *velocity* states. Smith et al. (2019) and Meyer et al. (2020) further make a connection between the layered planner–tracker architecture and the notion of abstractions introduced in Girard and Pappas (2009). In doing so, they also eliminate the restrictive geometric conditions in Girard and Pappas (2009), also implicit in Singh et al. (2020), which require that the set where the tracking error vanishes be invariant. Removing this requirement and allowing the tracking error to depend on planner inputs greatly expand the applicability of the planner–tracker framework.

In this tutorial we introduce a broad framework which encompasses those earlier results while further generalizing the error definition compared to Meyer et al. (2020). In addition, the framework described here is not restricted to a particular planner. Indeed, unlike the computationally heavy symbolic design method used for planning in Meyer et al. (2020), the numerical example presented here uses the popular choice of Model Predictive Control (MPC), which is appropriate for real-time implementation.

Although MPC is often used for both planning and control, under mismatch of planning model and the plant, the MPC optimization problem must be robustified. Feasibility and stability properties of robust MPC have been studied in Kothare, Balakrishnan, and

* Corresponding author.
E-mail addresses: kschweidel@berkeley.edu (K.S. Schweidel), he_yin@berkeley.edu (H. Yin), swsmth@berkeley.edu (S.W. Smith), arcak@berkeley.edu (M. Arcak).

https://doi.org/10.1016/j.arcontrol.2022.04.004
Received 22 December 2021; Received in revised form 4 April 2022; Accepted 6 April 2022
Available online 11 May 2022



(a) Online implementation. The planning controller uses the state $\hat{x}$ and the constraint sets $\hat{\mathcal{X}}$, $\hat{\mathcal{U}}$, and $\Delta\hat{\mathcal{X}}$ to generate a reference input $\hat{u}$. The tracking controller converts this into a control u which is guaranteed to keep the tracker state x within state constraints $\mathcal{X}$. This is accomplished by keeping the tracking error e within a set $\mathcal{O}$ as described in Figure 1(b).

(b) Offline synthesis. The tracking and planning models are combined to obtain a model for the error system. A controller κ is derived to keep the error in the set $\mathcal{O}$ (e.g., using SOS). If the safety condition is met, the tracking system is guaranteed to satisfy the state constraints $\mathcal{X}$. Otherwise, the sets $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ are shrunk and the process is repeated.

Fig. 1. Online implementation and offline synthesis of the planner–tracker control scheme.

Morari (1996), Mayne, Rawlings, Rao, and Scokaert (2000) and subsequent publications. For linear systems, Tube MPC (Bujarbaruah, Rosolia, Stürz, Zhang, & Borrelli, 2020; Chisci, Rossiter, & Zappa, 2001; Fleming, Kouvaritakis, & Cannon, 2014; Goulart, Kerrigan, & Maciejowski, 2006; Langson, Chryssochoos, Rakovic, & Mayne, 2004; Muñoz-Carpintero, Cannon, & Kouvaritakis, 2013; Raković & Cheng, 2013; Raković, Kouvaritakis, Cannon, Panos, & Findeisen, 2012; Raković, Levine, & Açikmese, 2016) is a widely used approach that solves a computationally efficient convex optimization problem for robust control synthesis. Although Tube MPC design with feasibility and stability properties are proposed for nonlinear systems in Allgöwer and Zheng (2012), Cannon, Buerger, Kouvaritakis, and Rakovic (2011), Köhler, Müller, and Allgöwer (2018), Köhler, Soloperto, Müller, and Allgöwer (2021), Koller, Berkenkamp, Turchetta, and Krause (2018), Yu, Maier, Chen, and Allgöwer (2013), control synthesis can become either too conservative, or computationally demanding.

Another related work is (Majumdar & Tedrake, 2017), in which multiple tracking controllers and error-bound funnels are computed for a library of nominal trajectories. By contrast, the planner–tracker framework presented here generates a single tracking controller and can accommodate any trajectory from the planner, not just one that belongs to a pre-specified library.

The remainder of the paper is organized as follows. Section 2 introduces the high-fidelity tracking model and the low-fidelity planning model and defines a simple tracking error that depends only on the planner/tracker *states*. We build intuition with this simple error model and present the method for constructing a tracking controller and an error bound using sum-of-squares (SOS) programming. Section 3 generalizes the tracking error definition to additionally depend on the planner *input* and extends the results in Section 2 to handle this generalized error. In Section 4, we demonstrate the method on a vehicle obstacle avoidance example, and we provide concluding remarks in Section 5.

Notation

$\mathbb{S}^n$ denotes the set of n-by-n symmetric matrices. $\mathbb{S}^n_+$ and $\mathbb{S}^n_{++}$ denote the sets of n-by-n symmetric positive semi-definite and positive definite matrices, respectively. For $\xi \in \mathbb{R}^n$, $\mathbb{R}[\xi]$ represents the set of polynomials in ξ with real coefficients, and $\mathbb{R}^m[\xi]$ and $\mathbb{R}^{m\times p}[\xi]$ denote all vector and matrix valued polynomial functions. The subset $\Sigma[\xi] := \{p = p_1^2 + p_2^2 + \cdots + p_M^2 : p_1, \ldots, p_M \in \mathbb{R}[\xi]\}$ of $\mathbb{R}[\xi]$ is the set of sum-of-squares polynomials in ξ. Unless defined otherwise, notation x^j denotes a variable x used in the j'th iteration of an iterative algorithm. The symbol "$\preceq$" represents component-wise inequality.

2. Problem setup

In this section we describe the hierarchical approach to safe-by-design trajectory planning and control that consists of two layers: a planning layer, which uses a low-fidelity "planning" model, and a tracking layer, with a high-fidelity "tracking" model. The planning model might be a model with a lower state dimension than the tracking model or a linearization of the tracking model to reduce the computational burden of planning. By analyzing the dynamics of the error between these two systems' states, we will show how we can bound this error by synthesizing an appropriate tracking controller. In this article, the controller and corresponding error bound are designed via SOS programming.

The online implementation and offline synthesis of the planner–tracker control scheme are summarized in Fig. 1. We begin with a high-fidelity tracking model and a low-fidelity planning model, each with state and input constraints. Defining an appropriate error variable, e, between the two models, and using the error dynamics and the planner/tracker constraints, we design a tracking controller and derive a tracking error bound. This bound takes the form of a set $\mathcal{O}$ such that $e(t) \in \mathcal{O}$. If the planner constraints, when augmented by $\mathcal{O}$, still satisfy the tracking constraints, then the tracking system is safe: it will satisfy all constraints with the synthesized controller. Otherwise, the planner constraints are shrunk and the process is repeated.

2.1. High-fidelity tracking model

The high-fidelity model is of the form:

$$\dot{x}(t) = f(x(t), w(t)) + g(x(t), w(t)) \cdot u(t), \tag{1}$$

with state $x(t) \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, disturbance $w(t) \in \mathcal{W} \subseteq \mathbb{R}^{n_w}$, bounded control $u(t) \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$, and $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$. The sets $\mathcal{X}$ and $\mathcal{U}$ are the constraint sets imposed on the states and control inputs in the high-fidelity model, respectively.

2.2. Low-fidelity planning model

The low-fidelity model, which is a simplified version of (1), is of the form:

$$\dot{\hat{x}}(t) = \hat{f}(\hat{x}(t), \hat{u}(t)), \tag{2}$$

where $\hat{x}(t) \in \hat{\mathcal{X}} \subseteq \mathbb{R}^{\hat{n}_x}$, $\hat{u}(t) \in \hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{n}_u}$, and $\hat{f} : \mathbb{R}^{\hat{n}_x \times \hat{n}_u} \to \mathbb{R}^{\hat{n}_x}$. The sets $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ are constraint sets enforced by the planning layer. The control input for the low-fidelity model, computed via the planning algorithm

of choice, is assumed to be a zero-order hold signal with sampling time $T_s > 0$. This means:

$$\hat{u}(t) = \hat{u}(\tau_k),\ \forall t \in [\tau_k, \tau_{k+1}),\ \text{with}\ \tau_k = k \cdot T_s, \tag{3a}$$

$$\hat{u}(\tau_{k+1}) = \hat{u}(\tau_k) + \Delta\hat{u}(\tau_{k+1}), \tag{3b}$$

where $\Delta\hat{u}(t)$ is the change in the control input between sampling periods (also referred to as the "input jump"), restricted to a set $\Delta\hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{n}_u}$.

Note that the planning model does not depend directly on the tracker state x, in contrast to reduced order methods where planning is done on a lower dimensional state that is a projection of the true higher dimensional state (Löhning, Reble, Hasenauer, Yu, & Allgöwer, 2014).

Remark 1. The planner–tracker synthesis framework is applicable to any planning algorithm that is able to bound $\hat{x}(t), \hat{u}(t)$, and $\Delta\hat{u}(t)$. For example, this framework has been applied to different planning algorithms, using Nonlinear MPC in Smith et al. (2019), Yin et al. (2020), signal temporal logic (STL) in Pant, Yin, Arcak, and Seshia (2021), and discrete abstraction in Meyer et al. (2020).

2.3. Error dynamics

The goal is to design a controller for the high-fidelity tracking model (1) to track a reference trajectory planned using the low-fidelity planning model (2). In order to do so, we proceed by deriving the evolution of the error between (1) and (2). Since $\hat{n}_x \leq n_x$ in general, we define a C^1 map $\pi : \mathbb{R}^{\hat{n}_x} \to \mathbb{R}^{n_x}$, called the *comparison* map, and we define the tracking error as:

$$e(t) = x(t) - \pi(\hat{x}(t)). \tag{4}$$

If the planning model is simply a linearization, we may select π to be the identity map, but our primary interest is in the case where $\hat{x}$ is of lower dimension and π lifts it to the dimension of x. We will first describe the method with this simple error definition to build intuition before generalizing the error definition in Section 3, where π is allowed to also depend on $\hat{u}$.

Differentiating (4) with respect to time (dropping time arguments to improve readability), and eliminating the variable x, we obtain:

$$\begin{aligned}\dot{e} &= \dot{x} - \frac{\partial \pi}{\partial \hat{x}} \cdot \dot{\hat{x}} \\ &= f(x,w) + g(x,w)\cdot u - \frac{\partial \pi}{\partial \hat{x}} \cdot \hat{f}(\hat{x},\hat{u})\Big|_{x=e+\pi(\hat{x})}, \\ &= f_e(e,\hat{x},\hat{u},w) + g_e(e,\hat{x},w)\cdot u,\end{aligned} \tag{5}$$

where we have defined:

$$\begin{aligned} f_e(e,\hat{x},\hat{u},w) &= f(\pi(\hat{x})+e,w) - \frac{\partial \pi}{\partial \hat{x}} \cdot \hat{f}(\hat{x},\hat{u}), \\ g_e(e,\hat{x},w) &= g(\pi(\hat{x})+e,w). \end{aligned} \tag{6}$$

In this section, we consider controllers of the form

$$u(t) = \kappa(e(t), \hat{x}(t), \hat{u}(t)),\ \kappa \in \mathcal{K}_{\mathcal{U}} \tag{7}$$

where the set $\mathcal{K}_{\mathcal{U}} := \{\kappa : \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \to \mathcal{U}\}$ defines a set of admissible error-state feedback control laws. Plugging in this controller, the closed-loop dynamics become

$$\dot{e} = f_e(e,\hat{x},\hat{u},w) + g_e(e,\hat{x},w)\cdot\kappa(e,\hat{x},\hat{u}). \tag{8}$$

Our goal is to design a control law κ and an associated error bound $\mathcal{O}$ that is ideally as small as possible.

Definition 1 (*Tracking Error Bound*)**.** Given the closed loop error dynamics (8), we say that a set $\mathcal{O}$ is a *tracking error bound (TEB)* from an initial set $\mathcal{I}$ if

$$e(0) \in \mathcal{I},\ \hat{x}(t) \in \hat{\mathcal{X}},\ \hat{u}(t) \in \hat{\mathcal{U}},\ w(t) \in \mathcal{W}\ \forall\ t \geq 0$$

$$\Rightarrow\ e(t) \in \mathcal{O}\ \forall\ t \geq 0. \tag{9}$$

The initial set $\mathcal{I}$ will be constructed together with the TEB $\mathcal{O}$. In Theorem 1 in Section 2.4, $\mathcal{I}$ and $\mathcal{O}$ will be identical, while in Theorem 2 in Section 3.4, $\mathcal{I} \subseteq \mathcal{O}$.

As we will see in the next subsection, we aim to minimize the volume of the set $\mathcal{O}$ when designing the tracking controller κ; however, we do not emphasize asymptotic behavior of the error $e(t)$ since we do not need perfect tracking of the planning model. Indeed we allow the dynamics (8) to depend on $\hat{x}$ besides $\hat{u}$ and w, and we do not require the right-hand side to vanish when $e = 0$. The benefit of this relaxed approach, as alluded to in the Introduction, is to remove restrictive geometric constraints from the selection of the map π and controller κ that would render the set $e = x - \pi(\hat{x}) = 0$ invariant and attractive.

2.4. Computing the TEB and tracking controller

The TEB $\mathcal{O}$ and the tracking controller κ can be obtained with the help of the following theorem, which gives conditions under which the error can be constrained to lie within a certain sublevel set of a storage function $V(e)$.

Theorem 1. *Given the error dynamics (5) with* $f_e : \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$, $g_e : \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$, *and* $\gamma \in \mathbb{R}$, $\hat{\mathcal{X}} \subseteq \mathbb{R}^{\hat{n}_x}$, $\hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{n}_u}$, $\mathcal{W} \subseteq \mathbb{R}^{n_w}$, *if there exists a* C^1 *function* $V : \mathbb{R}^{n_x} \to \mathbb{R}$ *and* $\kappa : \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \to \mathbb{R}^{n_u}$ *such that*

$$\begin{aligned} &\frac{\partial V(e)}{\partial e} \cdot \left(f_e(e,\hat{x},\hat{u},w) + g_e(e,\hat{x},w)\cdot\kappa(e,\hat{x},\hat{u})\right) < 0, \\ &\forall e,\hat{x},\hat{u},w,\ \text{s.t.}\ V(e) = \gamma,\ \hat{x} \in \hat{\mathcal{X}},\ \hat{u} \in \hat{\mathcal{U}},\ w \in \mathcal{W}, \end{aligned} \tag{10}$$

then the sublevel set $\Omega(V,\gamma) := \{e \in \mathbb{R}^{n_x} : V(e) \leq \gamma\}$ *is a TEB as in Definition 1 with* $\mathcal{I} = \mathcal{O} = \Omega(V,\gamma)$.

Proof. The theorem is proved by contradiction. Assume there exist a time $t_2 > 0$ and a trajectory $e(\cdot)$ such that $e(0) \in \Omega(V,\gamma)$ but $e(t_2) \notin \Omega(V,\gamma)$, i.e., $V(e(t_2)) > \gamma$. Since $V(e(0)) \leq \gamma$, by continuity of V there exists t_1 such that $0 \leq t_1 < t_2$, $V(e(t_1)) = \gamma$, and $\frac{d}{dt}V(e(t))|_{t=t_1} \geq 0$. (If all crossings of $V(e(t)) = \gamma$ satisfied $\frac{d}{dt}V(e(t)) < 0$, then V would not be continuous.) This contradicts (10). □

It is straightforward to augment the statement above to ensure a bound on the input. Adding the following constraint ensures $u = \kappa(e,\hat{x},\hat{u}) \in \mathcal{U}$:

$$\Omega(V,\gamma) \subseteq \{e \in \mathbb{R}^{n_x} : \kappa(e,\hat{x},\hat{u}) \in \mathcal{U}\}\ \forall \hat{x} \in \hat{\mathcal{X}}, \hat{u} \in \hat{\mathcal{U}}. \tag{11}$$

Furthermore, if a user-specified initial error set $\mathcal{E}_0$ is known, then adding the set constraint

$$\Omega(V,\gamma) = \mathcal{I} \supseteq \mathcal{E}_0 \tag{12}$$

will ensure that $e(t) \in \mathcal{O}$ for all $t \geq 0$. As a practical consideration, we add such a constraint when searching for V and κ.

Finding generic functions V and κ that satisfy constraints (10), (11), and (12) is a difficult problem. Below we show how SOS programming can be used to search for these functions by restricting to polynomial candidates $V \in \mathbb{R}[e]$ and $\kappa \in \mathbb{R}^{n_u}[(e,\hat{x},\hat{u})]$. Besides this restriction, we make the following assumption:

Assumption 1. The mappings $f_e \in \mathbb{R}^{n_x}[(e,\hat{x},\hat{u},w)]$ and $g_e \in \mathbb{R}^{n_x \times n_u}[(e,\hat{x},w)]$ in error dynamics (5) are polynomials. Sets $\mathcal{E}_0$, $\hat{\mathcal{X}}$, $\hat{\mathcal{U}}$, and $\mathcal{W}$ are semi-algebraic sets, i.e., there exists $p_0 \in \mathbb{R}[e]$ such that $\mathcal{E}_0 = \{e \in \mathbb{R}^{n_x} : p_0(e) \leq 0\}$; with similar definitions for $\hat{\mathcal{X}}$, $\hat{\mathcal{U}}$, and $\mathcal{W}$ with polynomials $p_{\hat{x}} \in \mathbb{R}[\hat{x}]$, $p_{\hat{u}} \in \mathbb{R}[\hat{u}]$, and $p_w \in \mathbb{R}[w]$. The control constraint set $\mathcal{U}$ is a hypercube $\mathcal{U} = \{u \in \mathbb{R}^{n_u} : \underline{u} \leq u \leq \overline{u}\}$, where $\underline{u}, \overline{u} \in \mathbb{R}^{n_u}$.

By applying the generalized S-procedure (Parrilo, 2000) to the set containment constraints (10), (11), and (12), and using the volume of $\Omega(V,\gamma)$ as the cost function to minimize, we obtain the following SOS optimization problem for finding V and κ:

$$\begin{aligned}
\min_{V,\kappa,\gamma,s,l}\ & \text{volume}(\Omega(V,\gamma)) \\
\text{s.t. } & s_0 \in \Sigma[e], s_{1\to 3} \in \Sigma[(e,\hat{x},\hat{u},w)], l \in \mathbb{R}[(e,\hat{x},\hat{u},w)] \\
& s_{4\to 9,i} \in \Sigma[(e,\hat{x},\hat{u})], i \in \{1,\dots,n_u\} && \text{(13a)} \\
& -(V(e)-\gamma) + s_0 \cdot p_0 \in \Sigma[e], && \text{(13b)} \\
& -\frac{\partial V}{\partial e}\cdot(f_e + g_e\cdot\kappa) - \epsilon e^\top e + l\cdot(V-\gamma) + s_1\cdot p_{\hat{x}} \\
& \quad + s_2\cdot p_{\hat{u}} + s_3\cdot p_w \in \Sigma[(e,\hat{x},\hat{u},w)], && \text{(13c)} \\
& \bar{u}_i - \kappa_i + s_{4,i}\cdot(V-\gamma) + s_{5,i}\cdot p_{\hat{x}} \\
& \quad + s_{6,i}\cdot p_{\hat{u}} \in \Sigma[(e,\hat{x},\hat{u})], i \in \{1,\dots,n_u\}, && \text{(13d)} \\
& \kappa_i - \underline{u}_i + s_{7,i}\cdot(V-\gamma) + s_{8,i}\cdot p_{\hat{x}} \\
& \quad + s_{9,i}\cdot p_{\hat{u}} \in \Sigma[(e,\hat{x},\hat{u})], i \in \{1,\dots,n_u\}. && \text{(13e)}
\end{aligned}$$

In the formulation above, SOS polynomials $s_{1\to 3}$ and $s_{4\to 9,i}$ are multipliers used in the generalized S-procedure, and $\epsilon > 0$ is on the order of 10^{-6}. Constraint (13a) ensures that all the polynomial multipliers are SOS. Constraint (13b) is a relaxation of (12) (for $\mathcal{I} = \mathcal{O} = \Omega(V,\gamma)$), (13c) is a relaxation of (10), and together (13d) and (13e) are a relaxation of (11) under the hypercube assumption for $\mathcal{U}$. The optimization (13) is non-convex as there are two groups of decision variables V and $(\kappa, l, s_{4,i}, s_{7,i})$ bilinear in each other. To tackle this problem, similarly to Yin, Arcak, Packard, and Seiler (2021, Algorithm 1), we decompose it into two tractable subproblems to iteratively search between the two groups of decision variables, as shown in Algorithm 1 in the Appendix.

We note that V and γ always appear in the optimization as $V-\gamma$, so from a theoretical perspective there is no need for two separate optimization variables. However, the variable γ is practical algorithmically because in the subproblem where V is fixed, we can minimize over γ via bisection to find the smallest level set of V that forms a viable TEB

Remark 2. For simplicity, we define V to be a function of e. However, in principle, we could have defined $V(x,\hat{x})$, as is done in the incremental stability literature, e.g., Angeli (2002).

2.5. Safety check

After synthesizing V and κ, we check the following safety condition with $\mathcal{O} = \Omega(V,\gamma)$:

$$\pi(\hat{\mathcal{X}}) \oplus \mathcal{O} \subseteq \mathcal{X}. \tag{14}$$

If (14) is satisfied, then the tracker state x is guaranteed to satisfy state constraints $\mathcal{X}$ and the design is considered successful. If (14) is *not* satisfied, we shrink the planner sets $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ and repeat the process as indicated in Fig. 1.

The details of how to shrink the sets $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ are not the focus of this paper, but we refer the reader to Yin et al. (2020) for an in-depth treatment. In Yin et al. (2020), the constraint sets are parameterized by some parameter, and a bisection over this parameter is performed to find the most permissive sets that still guarantee safety.

3. Generalized tracking error definition

So far, we have used a map π that only depends on the planner state $\hat{x}$. However, as illustrated in the example below, this map may fail to provide reference signals for all the tracker states. Therefore, in Section 3.1, we move to a more general error definition that also depends on the planner input $\hat{u}$.

Example 1. As a simple illustration of why it is useful to include the planner input $\hat{u}$ in the error definition, consider the double integrator tracking model

$$x = \begin{bmatrix} s \\ v \end{bmatrix}, \quad \dot{x} = \begin{bmatrix} v \\ u \end{bmatrix}, \tag{15}$$

where s is the position, v is the velocity, and u is the acceleration input. Let the planning model be a single integrator, where the only state is the planner position ($\hat{x} = \hat{s}$) and the input is the planner velocity ($\dot{\hat{x}} = \hat{v} =: \hat{u}$). Then, letting $\pi(\hat{x},\hat{u}) = [\hat{x};\hat{u}]$, the error is

$$e = x - \pi(\hat{x},\hat{u}) = \begin{bmatrix} s-\hat{s} \\ v-\hat{v} \end{bmatrix}, \tag{16}$$

which is the deviation of the planner and tracker positions and velocities. Thus, by keeping e small, we keep the planner and tracker positions and velocities close to one another, which is desirable. On the other hand, if we had used the naïve map $\pi(\hat{x}) = [\hat{x};0]$, the error would be $[(s-\hat{s}) \quad v]^\top$, and bounding the error would mean keeping v close to zero, which is overly conservative and may not align with planning objectives.

3.1. Modified error dynamics

As motivated above, we will use a more general C^1 map $\pi : \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \to \mathbb{R}^{n_x}$ to provide better reference trajectories for the tracking model, as was done in Meyer et al. (2020) for the first time. For further generality, in this article we redefine the error state as

$$e = \phi(x,\hat{x},\hat{u})(x - \pi(\hat{x},\hat{u})), \tag{17}$$

where we add the C^1 map $\phi : \mathbb{R}^{n_x}\times\mathbb{R}^{\hat{n}_x}\times\mathbb{R}^{\hat{n}_u} \to \mathbb{R}^{n_x\times n_x}$ which provides additional flexibility, as will be demonstrated in Section 4.

Assume that for each e, $\hat{x}$, $\hat{u}$, there exists a unique x satisfying (17), and denote this inverse as

$$x = \nu(e,\hat{x},\hat{u}). \tag{18}$$

The error dynamics resulting from (17) are

$$\dot{e} = f_e(e,\hat{x},\hat{u},w) + g_e(e,\hat{x},\hat{u},w)u - h_e(e,\hat{x},\hat{u})\dot{\hat{u}}, \tag{19}$$

where

$$\begin{aligned}
f_e(e,\hat{x},\hat{u},w) :=& \left\{ \frac{\partial\phi}{\partial x} f(x,w) + \frac{\partial\phi}{\partial\hat{x}}\hat{f}(\hat{x},\hat{u}) \right\}(x-\pi(\hat{x},\hat{u})) \\
& + \phi(x,\hat{x},\hat{u})\left\{ f(x,w) - \frac{\partial\pi}{\partial\hat{x}}\hat{f}(\hat{x},\hat{u}) \right\}\Big|_{x=\nu(e,\hat{x},\hat{u})}, && \text{(20)} \\
g_e(e,\hat{x},\hat{u},w) :=& \left\{ \frac{\partial\phi}{\partial x}(x-\pi(\hat{x},\hat{u})) \right. && \text{(21)} \\
& \left. + \phi(x,\hat{x},\hat{u}) \right\} g(x,w)\Big|_{x=\nu(e,\hat{x},\hat{u})},
\end{aligned}$$

and h_e can be computed but is not written explicitly since it multiplies $\dot{\hat{u}}$, which is zero within sampling periods.

Note that the planner input is applied in a zero order hold fashion within each sampling period as described in (3). As the tracking error dynamics (19) have a term containing $\dot{\hat{u}}$ (unlike (5)), these dynamics change discontinuously at each sampling instant τ_k. Therefore, we break up the error analysis into two parts. In Section 3.2, we bound the error *within* a single sampling period $[\tau_{k-1},\tau_k)$. In Section 3.3, we bound the error jump *across* sampling periods from τ_k^- to τ_k^+ induced by the input jump $\Delta\hat{u}_k$. This two-part approach can be visualized in Fig. 2.

3.2. Analysis within sampling periods

Since the signal $\hat{u}$ is piece-wise constant, within a single sampling period we have

$$\dot{\hat{u}}(t) = 0, \ \forall t \in [\tau_{k-1},\tau_k). \tag{22}$$

Therefore, the error dynamics (19) during the time interval $[\tau_{k-1}, \tau_k)$ are:

$$\dot{e} = f_e(e, \hat{x}, \hat{u}, w) + g_e(e, \hat{x}, \hat{u}, w)u. \tag{23}$$

We want to enforce the boundedness of the error state during $[0, T_s)$ by introducing a tracking controller

$$u(t) = \kappa(t, e(t), \hat{x}(t), \hat{u}(t)), \tag{24}$$

which is now defined by a *time-varying*, error-state feedback control law $\kappa : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \to \mathbb{R}^{n_u}$. We use a time-varying controller κ and a time-varying storage function V in this section for additional flexibility. In particular, we may find a storage functions whose level sets shrink as time increases. The main idea is that if the level set at the end of the sampling period is sufficiently smaller than the level set at the beginning of the sampling period, then this size difference can accommodate the error jump across sampling periods as in Fig. 2.

Below we provide conditions on κ and V for bounding the error in a level set of V within each sampling period, where now each level set of V is a *funnel* in (e, t) space.

Proposition 1. *Given the error dynamics (23) with mappings $f_e : \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$, $g_e : \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_x}$, and $\gamma \in \mathbb{R}$, $T_s > 0$, $\hat{\mathcal{X}} \subseteq \mathbb{R}^{\hat{n}_x}$, $\hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{n}_u}$, $\mathcal{W} \subseteq \mathbb{R}^{n_w}$, suppose there exists a C^1 function $V : \mathbb{R} \times \mathbb{R}^{n_x} \to \mathbb{R}$, and $\kappa : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \to \mathbb{R}^{n_u}$, such that*

$$\begin{aligned} &\frac{\partial V(t,e)}{\partial e} \cdot (f_e(e, \hat{x}, \hat{u}, w) + g_e(e, \hat{x}, \hat{u}, w) \cdot \kappa(t, e, \hat{x}, \hat{u})) \\ &\quad + \frac{\partial V(t,e)}{\partial t} < 0, \ \forall t, e, \hat{x}, \hat{u}, w, \ \text{s.t. } t \in [0, T_s), \\ &\quad V(t,e) = \gamma, \ \hat{x} \in \hat{\mathcal{X}}, \ \hat{u} \in \hat{\mathcal{U}}, \ w \in \mathcal{W}. \end{aligned} \tag{25}$$

Define the funnel $\Omega(V, t, \gamma) := \{e \in \mathbb{R}^{n_x} : V(t,e) \le \gamma\}$. If $e(0) \in \Omega(V, 0, \gamma)$, then $e(t) \in \Omega(V, t, \gamma)$ for all $t \in [0, T_s)$.

Proof. The proof is a simple modification of the proof of Theorem 1, where now $\dot{V}(t, e(t)) := \frac{\partial V(t,e)}{\partial e}\dot{e}(t) + \frac{\partial V(t,e)}{\partial t}$ contains the additional $\partial V/\partial t$ term from (25). □

Remark 3. Although Proposition 1 is stated for the first sampling period $[0, T_s)$, it can be used for any other sampling period $[\tau_k, \tau_{k+1})$ with $\tau_k = k \cdot T_s$. Let $e(\tau_k) \in \Omega(V, 0, \gamma)$. Then we have $e(\tau_k + t) \in \Omega(V, t, \gamma)$, for all $t \in [0, T_s)$, under the control signal $u(\tau_k + t) = \kappa(t, e(\tau_k + t), \hat{x}(\tau_k + t), \hat{u}(\tau_k + t))$.

3.3. Analysis across sampling periods

Next, we focus on the effect of the input jump $\Delta\hat{u}$ at each sampling instant τ_k as in (3b). From (17), $\Delta\hat{u}$ induces a jump on the error. Let τ_k^- and τ_k^+ denote sampling instant τ_k before and after the discrete jump, respectively, and for simplicity use the notation $e_k^+ := e(\tau_k^+)$. Then we have

$$\begin{aligned} e_k^+ &= \phi(x_k^+, \hat{x}_k^+, \hat{u}_k^+)(x_k^+ - \pi(\hat{x}_k^+, \hat{u}_k^+)) \\ &= \phi(x_k^-, \hat{x}_k^-, \hat{u}_k^- + \Delta\hat{u}_k^+)(x_k^- - \pi(\hat{x}_k^-, \hat{u}_k^- + \Delta\hat{u}_k^+)) \\ &= \phi(\nu(e_k^-, \hat{x}_k^-, \hat{u}_k^- + \Delta\hat{u}_k^+), \hat{x}_k^-, \hat{u}_k^- + \Delta\hat{u}_k^+) \\ &\qquad \cdot (\nu(e_k^-, \hat{x}_k^-, \hat{u}_k^- + \Delta\hat{u}_k^+) - \pi(\hat{x}_k^-, \hat{u}_k^- + \Delta\hat{u}_k^+)) \\ &=: h(e_k^-, \hat{x}_k^-, \hat{u}_k^-, \Delta\hat{u}_k^+). \end{aligned} \tag{26}$$

We refer to h as the *jump function*, as it reflects how the error may jump from the end of one sampling period to the beginning of the next, due to the jump in the input.

We introduce the additional condition below to characterize the error jump induced by the control jump $\Delta\hat{u}$ in terms of the funnel $\Omega(V, t, \gamma)$.

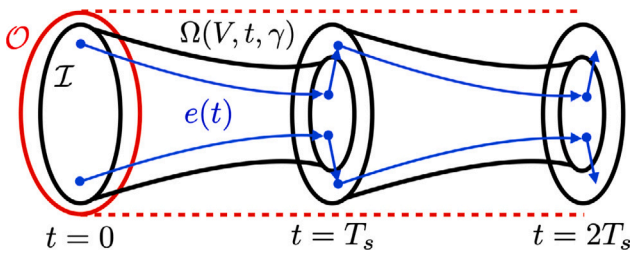


Fig. 2. Illustration of Theorem 2, with initial error set $\mathcal{I} = \Omega(V, 0, \gamma)$, funnels $\Omega(V, t, \gamma)$ over two sampling periods, bounded error jumps at sampling times, and TEB $\mathcal{O}$.

Proposition 2. *Given $\gamma \in \mathbb{R}$, $T_s \in \mathbb{R}$, $\hat{\mathcal{X}} \subseteq \mathbb{R}^{\hat{n}_x}$, $\hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{n}_u}$, $\Delta\hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{n}_u}$, $h : \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_x} \times \mathbb{R}^{\hat{n}_u} \times \mathbb{R}^{\hat{n}_u} \to \mathbb{R}^{n_x}$, if there exists a function $V : \mathbb{R} \times \mathbb{R}^{n_x} \to \mathbb{R}$ satisfying*

$$\begin{aligned} &V(0, h(e, \hat{x}, \hat{u}, \Delta\hat{u})) \le \gamma, \\ &\qquad \forall \hat{x} \in \hat{\mathcal{X}}, \hat{u} \in \hat{\mathcal{U}}, \Delta\hat{u} \in \Delta\hat{\mathcal{U}} \text{ and } \forall e \text{ s.t. } V(T_s, e) \le \gamma \end{aligned} \tag{27}$$

then for all $e_k^- \in \Omega(V, T_s, \gamma)$, $e_k^+ \in \Omega(V, 0, \gamma)$.

Proof. Suppose $e_k^- \in \Omega(V, T_s, \gamma)$, i.e., $V(T_s, e_k^-) \le \gamma$. By Eq. (26), $e_k^+ = h(e_k^-, \hat{x}_k^-, \hat{u}_k^-, \Delta\hat{u}_k^+)$. Thus by Eq. (27), $V(0, e_k^+) \le \gamma$, i.e., $e_k^+ \in \Omega(V, 0, \gamma)$. □

Remark 4. For the special case $\phi(x, \hat{x}, \hat{u}) = 1$ and $\pi(\hat{x}, \hat{u}) = \theta(\hat{x}) + Q\hat{u}$ for some $Q \in \mathbb{R}^{n_x \times \hat{n}_u}$, the $\hat{x}$ and $\hat{u}$ terms cancel, and so (27) simplifies to

$$V(0, e - Q\Delta\hat{u}) \le \gamma, \ \forall \Delta\hat{u} \in \Delta\hat{\mathcal{U}}, \ \forall e \text{ s.t. } V(T_s, e) \le \gamma.$$

3.4. Combining within- and across-sample analysis

We next combine the conditions for within- and across-sample error boundedness from Propositions 1 and 2, respectively, to obtain the main result on the boundedness of the error at all time, formulated below and illustrated in Fig. 2.

Theorem 2. *If there exist V and κ satisfying (25) and (27), define $\mathcal{O} \subset \mathbb{R}^{n_x}$ such that*

$$\cup_{t \in [0, T_s)} \Omega(V, t, \gamma) \subseteq \mathcal{O}.$$

Then for all $\hat{x}(t) \in \hat{\mathcal{X}}$, $\hat{u}(t) \in \hat{\mathcal{U}}$, $\Delta\hat{u}(t) \in \Delta\hat{\mathcal{U}}$, and $w(t) \in \mathcal{W}$, the error system (19) under control law $u(t) = \kappa(\tilde{t}, e(t), \hat{x}(t), \hat{u}(t))$ with $\tilde{t} = (t \bmod T_s) \in [0, T_s)$ satisfies:

$$e(0) \in \Omega(V, 0, \gamma) = \mathcal{I} \ \Rightarrow \ e(t) \in \mathcal{O}, \ \forall t \ge 0,$$

that is to say, $\mathcal{O}$ is a TEB from $\mathcal{I}$ achieved by the tracking control law κ.

Proof. From Remark 3 and for all $\tau_k = k \cdot T_s$, we have if $e(\tau_k) \in \Omega(V, 0, \gamma)$, then $e(\tau_k + \tilde{t}) \in \Omega(V, \tilde{t}, \gamma)$ and $e(\tau_{k+1}^-) \in \Omega(V, T_s, \gamma)$. Then it follows from Proposition 2 that $e(\tau_{k+1}^+) \in \Omega(V, 0, \gamma)$. As a result, for all $e(0) \in \Omega(V, 0, \gamma)$, we have $e(k \cdot T_s + \tilde{t}) \in \Omega(V, \tilde{t}, \gamma) \subseteq \mathcal{O}$, for all $k \ge 0$, and $\tilde{t} \in [0, T_s)$. □

Example 2. We now revisit the planner/tracker dynamics from Example 1 and perform the analysis above to bound the error within and across sampling periods. Instead of using SOS to search for a storage function, we propose a fixed form of a controller, and we construct a storage function that satisfies the conditions of Theorem 2 above.

Within sampling periods, $\hat{u} = 0$, so the open loop error dynamics are

$$\dot{e} = \begin{bmatrix} \dot{s} - \dot{\hat{s}} \\ \dot{v} - \dot{\hat{v}} \end{bmatrix} = \begin{bmatrix} v - \hat{v} \\ u - 0 \end{bmatrix} = \begin{bmatrix} e_2 \\ u \end{bmatrix}. \tag{28}$$

Selecting a state-feedback controller $u(e) = -k_1 e_1 - k_2 e_2$, the closed loop error dynamics are

$$\dot{e} = \begin{bmatrix} 0 & 1 \\ -k_1 & -k_2 \end{bmatrix} e =: Ae. \tag{29}$$

Because this is a LTI system, constructing a Lyapunov function is straightforward. If there exists $P = P^\top > 0$ such that $PA + A^\top P < -\alpha P$ for some $\alpha > 0$, then it is simple to show that $V(t,e) = \exp(\alpha t) \cdot e^\top P e$ satisfies $\dot{V}(t,e) < 0$ for all $e \neq 0$. Hence, for any $\gamma > 0$, $\cup_{t\in[0,T_s)} \Omega(V,t,\gamma)$ forms a valid TEB from $\mathcal{I} = \Omega(V,0,\gamma)$ by Theorem 2. We can examine the form of the level sets to see how they shrink with time

$$\Omega(V,t,\gamma) = \left\{ e \in \mathbb{R}^{n_x} : e^\top P e \le \frac{\gamma}{\exp(\alpha t)} \right\}. \tag{30}$$

As t increases, e is forced to lie in smaller and smaller ellipsoids. Then the jump condition is

$$\left(e - \begin{bmatrix} 0 \\ \Delta\hat{u} \end{bmatrix}\right)^\top P \left(e - \begin{bmatrix} 0 \\ \Delta\hat{u} \end{bmatrix}\right) \le \gamma \tag{31}$$
$$\text{for all } \Delta\hat{u} \in \Delta\hat{\mathcal{U}} \text{ and } e \text{ s.t. } e^\top P e \le \frac{\gamma}{\exp(\alpha T_s)},$$

meaning that if the error lies in the smallest ellipsoid at the end of the sampling period, then for all values of $\Delta\hat{u}$ the perturbed error will lie in the largest ellipsoid at the start of the next sampling period, as illustrated in Fig. 2.

The variable e can be eliminated by maximizing the left hand side of (31) over e and observing that the optimizer e^* is aligned with $[0; \Delta\hat{u}]$. Then (31) can be simplified to

$$\Delta\hat{u}^\top P_{22} \Delta\hat{u} \le \gamma \left(1 - \exp(-\tfrac{1}{2}\alpha T_s)\right)^2 \;\; \forall\, \Delta\hat{u} \in \Delta\hat{\mathcal{U}} \tag{32}$$

where $P_{22} \in \mathbb{R}^{\hat{n}_u \times \hat{n}_u}$ is the lower right block of P. Furthermore, if $\Delta\hat{\mathcal{U}}$ is a polytope, condition (32) can simply be checked at the vertices of $\Delta\hat{\mathcal{U}}$ due to the convexity of the expression $\Delta\hat{u}^\top P_{22} \Delta\hat{u}$.

Remark 5. As we saw in Example 2, it is not always necessary to use a SOS tracking controller. SOS is a versatile option since it can handle general polynomial systems, but for a given system, a practitioner may wish to use a fixed controller or a fixed *form* of a controller with some parameters to be determined. This fixed or parameter-dependent κ can be plugged into the SOS optimization, rather than leaving κ as a totally free decision variable. Then the SOS optimization can search for V, γ, and any parameters in κ assuming the optimization remains convex in these parameters. Otherwise, an iterative search can be performed over the parameters of κ.

3.5. SOS optimization

Again, to use SOS optimization to search for V and κ, we restrict them to polynomials: $V \in \mathbb{R}[(t,e)]$, and $\kappa \in \mathbb{R}[(t,e,\hat{x},\hat{u})]$. We further assume that f_e(20), g_e(21), and the jump function h(26) are polynomials. In addition to Assumption 1, we assume $\Delta\hat{\mathcal{U}} = \{\Delta\hat{u} \in \mathbb{R}^{\hat{n}_u} : p_\Delta(\Delta\hat{u}) \le 0\}$, where $p_\Delta \in \mathbb{R}[\Delta\hat{u}]$. Similarly to (11), we enforce tracking input constraints $u \in \mathcal{U}$ via the constraint

$$\{e \in \mathbb{R}^{n_x} : V(t,e) \le \gamma\} \subseteq \{e \in \mathbb{R}^{n_x} : \kappa(t,e,\hat{x},\hat{u}) \in \mathcal{U}\},$$
$$\forall (t,\hat{x},\hat{u}) \in [0,T_s) \times \hat{\mathcal{X}} \times \hat{\mathcal{U}}. \tag{33}$$

By choosing the integral of the volume of $\Omega(V,t,\gamma)$ over the time interval $[0,T_s]$ as the cost function, and applying the generalized S-procedure to (12), (25), (27), and (33), we obtain the following optimization problem:

$$\min_{V,\kappa,\gamma,s,l} \int_0^{T_s} \text{volume}(\Omega(V,t,\gamma))dt$$
$$\text{s.t. } s_{1\to 4} \in \Sigma[(t,e,\hat{x},\hat{u},w)], s_{5\to 6} \in \Sigma[(e,\Delta\hat{u})],$$
$$l \in \mathbb{R}[(t,e,\hat{x},\hat{u},w)], s_0 \in \Sigma[e],$$
$$s_{7\to 14,i} \in \Sigma[(t,e,\hat{x},\hat{u})], i \in \{1,\dots,n_u\}, \tag{34a}$$
$$\gamma - V(0,e) + s_0 \cdot p_0 \in \Sigma[e], \tag{34b}$$
$$-\left(\frac{\partial V}{\partial t} + \frac{\partial V}{\partial e} \cdot (f_e + g_e \kappa)\right) - \epsilon e^\top e + l \cdot (V - \gamma)$$
$$+ s_1 \cdot p_{\hat{x}} + s_2 \cdot p_{\hat{u}} + s_3 \cdot p_w - s_4 \cdot t(T_s - t)$$
$$\in \Sigma[(t,e,\hat{x},\hat{u},w)], \tag{34c}$$
$$-(V(0,h(e,\hat{x},\hat{u},\Delta\hat{u})) - \gamma) + s_5 \cdot (V(T_s,e) - \gamma)$$
$$+ s_6 \cdot p_\Delta \in \Sigma[(e,\Delta\hat{u})], \tag{34d}$$
$$\bar{u}_i - \kappa_i + s_{7,i} \cdot (V - \gamma) - s_{8,i} \cdot t(T_s - t) + s_{9,i} \cdot p_{\hat{x}}$$
$$+ s_{10,i} \cdot p_{\hat{u}} \in \Sigma[(t,e,\hat{x},\hat{u})], i \in \{1,\dots,n_u\}, \tag{34e}$$
$$\kappa_i - \underline{u}_i + s_{11,i} \cdot (V - \gamma) - s_{12,i} \cdot t(T_s - t) + s_{13,i} \cdot p_{\hat{x}}$$
$$+ s_{14,i} \cdot p_{\hat{u}} \in \Sigma[(t,e,\hat{x},\hat{u})], i \in \{1,\dots,n_u\}. \tag{34f}$$

Note that the condition $t \in [0,T_s]$ is reformulated in (34e)–(34f) via the inequality $-t(T_s - t) \le 0$. The optimization is bilinear in two groups of decision variables V and $(\kappa, l, s_5, s_{7,i}, s_{11,i})$, and can also be solved using alternating direction method similar to Algorithm 1 in Appendix.

After the funnel $\Omega(V,t,\gamma)$ is found, the next step is to compute a TEB $\mathcal{O}$ by solving a convex optimization:

$$\begin{aligned} &\min \;\; \text{volume}(\mathcal{O}) \\ &\text{s.t.} \;\; \Omega(V,t,\gamma) \subseteq \mathcal{O}, \;\; \forall t \in [0,T_s]. \end{aligned} \tag{35}$$

The set $\mathcal{O}$ is restricted to a semi-algebraic set in order to convert the set containment constraint into a SOS constraint. Depending on the parameterization of $\mathcal{O}$, different cost functions can be chosen. For example, if $\mathcal{O}$ is an ellipsoid, $\mathcal{O} = \{e \in \mathbb{R}^{n_x} : e^\top P_\mathcal{O} e \le 1\}$, where $P_\mathcal{O} \in \mathbb{S}^{n_x}_{++}$ is a decision variable, then $-\log\det(P_\mathcal{O})$ can be used as a cost function. If $\mathcal{O}$ is a polytope, $\mathcal{O} = \{e \in \mathbb{R}^{n_x} : A_\mathcal{O} e \le b_\mathcal{O}\}$, where $A_\mathcal{O} \in \mathbb{R}^{n_\mathcal{O} \times n_x}$ is fixed, and $b_\mathcal{O} \in \mathbb{R}^{n_\mathcal{O}}$ is a decision variable, then $\sum_{i=1}^{n_\mathcal{O}} b_{\mathcal{O},i}$ can be used as a cost function, where $b_{\mathcal{O},i}$ is the ith element of $b_\mathcal{O}$.

Once a TEB $\mathcal{O}$ is computed from the SOS optimization (34)–(35), we can check the following safety condition, which is a generalized version of (14):

$$\nu(\mathcal{O}, \hat{\mathcal{X}}, \hat{\mathcal{U}}) \subseteq \mathcal{X}. \tag{36}$$

If (36) is satisfied, then the tracker state x is guaranteed to satisfy state constraints $\mathcal{X}$ and the design is considered successful. If (14) is *not* satisfied, we shrink the planner sets $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$ and repeat the process.

4. Vehicle obstacle avoidance example

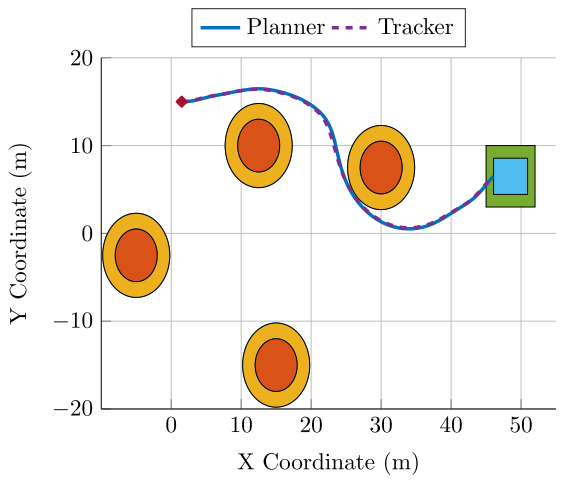
We now apply the planner–tracker control scheme to a vehicle obstacle avoidance example. For the high-fidelity tracking model, we use the dynamic bicycle model from Kong, Pfeiffer, Schildbach, and Borrelli (2015):

$$\begin{aligned} \dot{x}_1(t) &= x_5(t)\cos(x_3(t)) - x_6(t)\sin(x_3(t)), \\ \dot{x}_2(t) &= x_5(t)\sin(x_3(t)) + x_6(t)\cos(x_3(t)), \\ \dot{x}_3(t) &= x_4(t), \\ \dot{x}_4(t) &= \frac{2}{I_z}(l_f F_{c,f}(t) - l_r F_{c,r}(t)), \\ \dot{x}_5(t) &= x_4(t)x_6(t) + u_2(t), \\ \dot{x}_6(t) &= -x_4(t)x_5(t) + \frac{2}{m}(F_{c,f}(t) + F_{c,r}(t)) \end{aligned} \tag{37}$$

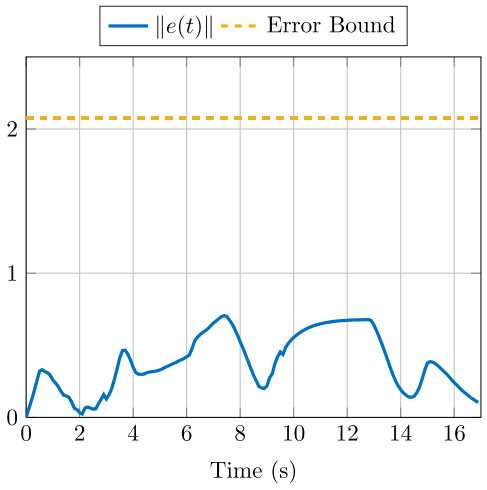
with

$$F_{c,f} = -C_{\alpha,f}\alpha_f, \;\; F_{c,r} = -C_{\alpha,r}\alpha_r \tag{38}$$
$$\alpha_f = \frac{x_6 + l_f x_4}{x_5} - u_1, \;\; \alpha_r = \frac{x_6 - l_r x_4}{x_5} \tag{39}$$

where x_1 to x_6 represent x, y positions in an inertial frame, inertial heading, yaw rate, and longitudinal and lateral speeds in the body frame. Variables u_1, u_2 represent front wheel steering angle and longitudinal acceleration, m and I_z denote the vehicle's mass and yaw inertia,



(a) Planner and tracker system trajectories.



(b) Error trajectory.

Fig. 3. Simulation results for the vehicle obstacle avoidance example. In Fig. 3(a) we plot the trajectories of the planner and tracker systems through the environment, and in Fig. 3(b) we plot $\|e(t)\|$ and its guaranteed upper bound. In Fig. 3(a), the initial position of the vehicle is marked with a red diamond, the goal set is represented with a green box, and the shrunken goal set is represented with a blue box. The four orange circles are the obstacles the vehicle must avoid. For each obstacle, the expanded unsafe region is shown in yellow. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and l_f and l_r represent the distance from the center of mass of the vehicle to the front and rear axles. $C_{\alpha,i}$ is the tire cornering stiffness, where $i \in \{f, r\}$. We use the parameter values $m = 1.67 \times 10^3$ kg, $I_z = 2.1 \times 10^3$ kg·m^2, $l_f = 0.99$ m, $l_r = 1.7$ m, $C_{\alpha,f} = 6.1595 \times 10^4$ N/rad, and $C_{\alpha,r} = 5.2095 \times 10^4$ N/rad.

The planning model is a Dubin's vehicle model:

$$\begin{aligned}\dot{\hat{x}}_1(t) &= \hat{u}_2(t)\cos(\hat{x}_3(t)),\\ \dot{\hat{x}}_2(t) &= \hat{u}_2(t)\sin(\hat{x}_3(t)),\\ \dot{\hat{x}}_3(t) &= \hat{u}_1(t),\end{aligned}$$

where $\hat{x}_1$ to $\hat{x}_3$ represent x, y positions and heading angle, and $\hat{u}_1$ and $\hat{u}_2$ represent angular velocity and velocity. If we use the map $\pi(\hat{x}) = [\hat{x}; 0_{3\times 1}]$, where $\hat{x} = [\hat{x}_1; \hat{x}_2; \hat{x}_3]$, then x_4 and x_5 will become part of the resulting error state. As a result, the magnitude of the absolute state x_4 and x_5 will be minimized in optimization (13), which is practically undesirable. To eliminate this issue, we use a map $\pi(\hat{x}, \hat{u}) = [\hat{x}; \hat{u}; 0]$, where $\hat{u} = [\hat{u}_1; \hat{u}_2]$, which also provides reference signals for x_4 and x_5.

The error is defined as in (17), with $\pi(\hat{x}, \hat{u}) = [\hat{x}; \hat{u}; 0]$ and $\phi(\hat{x}) = \text{diag}(R^{-1}(\hat{x}_3), I_4)$, where $R(\psi) = \begin{bmatrix}\cos(\psi) & -\sin(\psi)\\ \sin(\psi) & \cos(\psi)\end{bmatrix}$. In this example, ϕ allows us to replace the trigonometric functions in $\hat{x}_3$ in the error dynamics by trigonometric functions in $e_3 = (x_3 - \hat{x}_3)$, which can easily be approximated by polynomials in a certain range of e_3. The sampling time used in this example is $T_s = 0.1$ s. The input and input jump spaces for the planning model are $\hat{\mathcal{U}} = [-\pi/8, \pi/8] \times [2, 4]$, and $\Delta\hat{\mathcal{U}} = [-\pi/50, \pi/50] \times [-0.075, 0.075]$. The tracking control is unconstrained, i.e., $\mathcal{U} = \mathbb{R}^3$.

In this example, the SOS optimizations are formulated using SOSOPT (Seiler, 2013), and solved by MOSEK. To compute the tracking controller, we parameterize the storage function V, control law κ, and multipliers s, l as degree-2 polynomials. We solve optimization (34) with these decision variables and then solve optimization (35) with the error bound $\mathcal{O}$ parameterized as a hypercube. The resulting error bound on (e_1, e_2, e_3) is $[-1.07, 1.07] \times [-1.44, 1.44] \times [-1.05, 1.05]$.

The resulting tracking controller is then tested in simulation with a corresponding planner; see Fig. 3. The objective for the planner system is to generate a pathway through the environment that avoids all obstacles and eventually reaches a goal set, which is accomplished using a standard model-predictive controller (using the solver Ipopt (Wächter & Biegler, 2006)):

$$\min_{\hat{u}(\cdot)} \quad J = \ell_f(\hat{x}(t + N_p + 1)) + \sum_{k=t}^{t+N_p} \ell(\hat{x}(k), \hat{u}(k)) \tag{40a}$$

$$\text{s.t.} \quad \hat{x}(k+1) = \hat{x}(k) + T_s \cdot \hat{f}_{\text{approx}}(\hat{x}(k), \hat{u}(k)), \tag{40b}$$

$$\hat{x}(k) \in \hat{\mathcal{X}}, \tag{40c}$$

$$\hat{u}(k) \in \hat{\mathcal{U}}, \tag{40d}$$

$$\hat{x}(t) = \hat{x}_0, \tag{40e}$$

$$\forall k = t, \dots, t + N_p,$$

$$\hat{u}(k) - \hat{u}(k-1) \in \Delta\hat{\mathcal{U}}, \tag{40f}$$

$$\hat{u}(t) - \hat{u}_0 \in \Delta\hat{\mathcal{U}}, \tag{40g}$$

$$\forall k = t+1, \dots, t + N_p,$$

where $\ell(\cdot,\cdot)$ in (40a) is the state/input cost at each time step, $\ell_f(\cdot)$ is the final state cost, (40b) is the polynomial approximation of the discretized Dubin's vehicle dynamics, (40c) and (40d) ensure state and input constraints are obeyed, and (40e) is the initial state constraint. Furthermore, $\hat{u}_0$ is the input that was applied at the previous time step, and therefore (40f) and (40g) ensure the input jump constraints are respected. The objective to reach the goal set is encoded using the functions ℓ and ℓ_f. The initial state of the vehicle is $\hat{x}_0 = [0;\ 15;\ 0]$ and the goal set is a square region centered at $(48.5, 6.5)$ with a height and width of 7 m. This goal set is shrunk by the error bound to ensure that if the planner state reaches the shrunk goal set, the tracker state will reach the true goal set. There are four circular obstacles centered at $(-5, -2.5)$, $(12.5, 10)$, $(30, 7.5)$, and $(15, -15)$, each with a radius of 3 m. Since the maximum position tracking error the vehicle will experience is $\sqrt{1.07^2 + 1.44^2} = 1.79$m, for each obstacle, we constrain the vehicle to avoid a circular region centered at the obstacle coordinates with an expanded radius of 4.79 m. This ensures the vehicle will not collide with any of the obstacles. Indeed, in simulation the vehicle successfully navigates past each obstacle and eventually reaches the goal set, as shown in Fig. 3.

5. Conclusion

In this tutorial, we address robust trajectory planning and control design for nonlinear systems. A hierarchical trajectory planning and control framework is proposed, where a low-fidelity model is used to plan trajectories satisfying planning constraints, and a high-fidelity model is used for synthesizing tracking controllers guaranteeing the boundedness of the error state between the low- and high-fidelity models. We consider error states that are functions of both planner

states and inputs, which offers more freedom in the choice of the low-fidelity model. SOS optimizations are formulated for computing the tracking controllers and their associated tracking error bound simultaneously. Finally, we demonstrate the planner–tracker control scheme on a vehicle obstacle avoidance example.

When implementing the planner–tracker framework in real-time, there are still challenges for providing a full guarantee of safety. Two sources of error in the planner dynamics are present in the example above: (1) the discretization error from the forward Euler discretization, and (2) the polynomial approximation error from the trigonometric terms. If a bound on these errors were known, it would be possible to incorporate them into the design process, ensuring instead that the planner constraints, when augmented with the discretization error, polynomial approximation error, *and* the tracking error still satisfy the tracker constraints. We do not perform such an analysis in this tutorial. One could also avoid discretization error by using a MPC solver that is designed to perform numerical integration for *continuous-time* dynamical systems, such as in the software package `acados` (Verschueren etal, 2021).

In this tutorial, we also do not address the question of MPC terminal sets and costs for stability and persistent feasibility guarantees for the MPC problem. These sets/costs can be computed in simple cases but may increase the computational burden, both offline and online. Real-time reliability of solvers for MPC, especially for nonlinear models, should also be considered in practical applications. Finally, defining the error variable can require clever selection of the function ϕ to make terms in the dynamics cancel, which is not always intuitive. The shortcomings mentioned above also provide directions for further research.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by ONR, United States of America Grant N00014-18-1-2209, National Science Foundation, United States of America Grant ECCS-1906164, and AFOSR, United States of America Grant FA9550-21-1-0288. The authors would like to thank Monimoy Bujarbaruah for pointing to Tube MPC references.

Appendix

The algorithm to solve the bilinear optimization (13) is summarized below, the (κ,γ)-step of which treats γ as a decision variable. By minimizing γ, the volume of $\Omega(V^{j-1},\gamma)$ can be shrunk. In the V-step, (41) enforces $\Omega(V^j,\gamma^j)\subseteq\Omega(V^{j-1},\gamma^j)$.

The input to Algorithm 1 is a feasible initial guess V^0. One candidate might be a quadratic Lyapunov function $\bar{V}$ obtained by solving Lyapunov equations using the linearized error dynamics with LQR controllers. However, $\bar{V}$ might be too coarse to be feasible for the constraints (13). Here, we introduced a slack variable $\lambda>0$ to the constraint (13c) to relax the constraint, and quantify how far $\bar{V}$ is away from a feasible candidate:

$$-\frac{\partial V}{\partial e}\cdot(f_e+g_e\cdot\kappa)+\lambda-\epsilon e^\top e+l\cdot(V-\gamma)+s_1\cdot p_{\hat{x}} + s_2\cdot p_{\hat{u}}+s_3\cdot p_w\in\Sigma[(e,\hat{x},\hat{u},w)]. \tag{42}$$

By iteratively searching over two bilinear groups of decision variables, we minimize λ until $\lambda\le 0$. Based on this idea, an algorithm to compute V^0 from $\bar{V}$ is proposed as Algorithm 2.

Algorithm 1 Alternating direction method

Input: function V^0 such that constraints (13) are feasible by proper choice of s,l,κ,γ.
Output: κ,γ,V.
1: **for** $j=1:N_{\text{iter}}$ **do**
2: **(κ,γ)-step**: decision variables (s,l,κ,γ). Minimize γ subject to (13) using $V=V^{j-1}$. This yields $(l^j,s_{4,i}^j,s_{7,i}^j,\kappa^j)$ and the cost γ^j.
3: **V-step**: decision variables $(s_{1\to3},s_{5\to6,i},s_{8\to9,i},V)$; Maximize the feasibility subject to (13) as well as $s_{10}-\epsilon\in\Sigma[e]$, and

$$-s_{10}\cdot(V^{j-1}-\gamma^j)+(V-\gamma^j)\in\Sigma[e], \tag{41}$$

using $(\gamma=\gamma^j,s_{4,i}=s_{4,i}^j,s_{7,i}=s_{7,i}^j,\kappa=\kappa^j,l=l^j)$. This yields V^j.
4: **end for**

Algorithm 2 Computation of V^0

Input: function $\bar{V}$, and $\bar{\gamma}>0$.
Output: V^0.
1: $V^{\text{pre}}\leftarrow\bar{V}$
2: **while** $\lambda>0$ **do**
3: **κ-step**: decision variables (s,l,κ). Minimize λ subject to (13a)–(13b), (42), (13d)–(13e), using $V=V^{\text{pre}}$, $\gamma=\bar{\gamma}$. $(l^{\text{pre}},s_{4,i}^{\text{pre}},s_{7,i}^{\text{pre}},\kappa^{\text{pre}})\leftarrow(l,s_{4,i},s_{7,i},\kappa)$
4: **V-step**: decision variables $(s_{1\to3},s_{5\to6,i},s_{8\to9,i},V)$; Minimize λ subject to (13a)–(13b), (42), (13d)–(13e) using $(\gamma=\bar{\gamma},s_{4,i}=s_{4,i}^{\text{pre}},s_{7,i}=s_{7,i}^{\text{pre}},\kappa=\kappa^{\text{pre}},l=l^{\text{pre}})$. $V^{\text{pre}}\leftarrow V$
5: **end while**
6: $V^0\leftarrow V^{\text{pre}}$

References

Allgöwer, F., & Zheng, A. (2012). *Nonlinear model predictive control. Vol. 26*. Birkhäuser.
Angeli, D. (2002). A Lyapunov approach to incremental stability properties. *IEEE Transactions on Automatic Control*, *47*(3), 410–421.
Bujarbaruah, M., Rosolia, U., Stürz, Y. R., Zhang, X., & Borrelli, F. (2020). Robust MPC for linear systems with parametric and additive uncertainty: A novel constraint tightening approach. arXiv preprint arXiv:2007.00930.
Cannon, M., Buerger, J., Kouvaritakis, B., & Rakovic, S. (2011). Robust tubes in nonlinear model predictive control. *IEEE Transactions on Automatic Control*, *56*(8), 1942–1947.
Chisci, L., Rossiter, J. A., & Zappa, G. (2001). Systems with persistent disturbances: predictive control with restricted constraints. *Automatica*, *37*(7), 1019–1028.
Fleming, J., Kouvaritakis, B., & Cannon, M. (2014). Robust tube MPC for linear systems with multiplicative uncertainty. *IEEE Transactions on Automatic Control*, *60*(4), 1087–1092.
Girard, A., & Pappas, G. J. (2009). Hierarchical control system design using approximate simulation. *Automatica*, *45*(2), 566–571.
Goulart, P. J., Kerrigan, E. C., & Maciejowski, J. M. (2006). Optimization over state feedback policies for robust control with constraints. *Automatica*, *42*(4), 523–533.
Herbert, S. L., Chen, M., Han, S., Bansal, S., Fisac, J. F., & Tomlin, C. J. (2017). FaSTrack: A modular framework for fast and guaranteed safe motion planning. In *2017 IEEE conference on decision and control* (pp. 1517–1522).
Köhler, J., Müller, M. A., & Allgöwer, F. (2018). A novel constraint tightening approach for nonlinear robust model predictive control. In *2018 Annual American control conference* (pp. 728–734). IEEE.
Köhler, J., Soloperto, R., Müller, M. A., & Allgöwer, F. (2021). A computationally efficient robust model predictive control framework for uncertain nonlinear systems. *IEEE Transactions on Automatic Control*, *66*(2), 794–801.
Koller, T., Berkenkamp, F., Turchetta, M., & Krause, A. (2018). Learning-based model predictive control for safe exploration. In *2018 IEEE conference on decision and control* (pp. 6059–6066).
Kong, J., Pfeiffer, M., Schildbach, G., & Borrelli, F. (2015). Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE intelligent vehicles symposium* (pp. 1094–1099).

Kothare, M. V., Balakrishnan, V., & Morari, M. (1996). Robust constrained model predictive control using linear matrix inequalities. *Automatica*, *32*(10), 1361–1379.

Kousik, S., Vaskov, S., Bu, F., Johnson-Roberson, M., & Vasudevan, R. (2020). Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots. *International Journal of Robotics Research*, *39*(12), 1419–1469.

Langson, W., Chryssochoos, I., Rakovic, S. V., & Mayne, D. Q. (2004). Robust model predictive control using tubes. *Automatica*, *40*, 125–133.

Löhning, M., Reble, M., Hasenauer, J., Yu, S., & Allgöwer, F. (2014). Model predictive control using reduced order models: Guaranteed stability for constrained linear systems. *Journal of Process Control*, *24*(11), 1647–1659.

Majumdar, A., & Tedrake, R. (2017). Funnel libraries for real-time robust feedback motion planning. *International Journal of Robotics Research*, *36*(8), 947–982.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., & Scokaert, P. O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, *36*(6), 789–814.

Meyer, P.-J., Yin, H., Brodtkorb, A. H., Arcak, M., & Sørensen, A. J. (2020). Continuous and discrete abstractions for planning, applied to ship docking. In *21st IFAC world congress*.

Muñoz-Carpintero, D., Cannon, M., & Kouvaritakis, B. (2013). Recursively feasible robust MPC for linear systems with additive and multiplicative uncertainty using optimized polytopic dynamics. In *2013 IEEE conference on decision and control* (pp. 1101–1106).

Pant, Y., Yin, H., Arcak, M., & Seshia, S. (2021). Co-design of control and planning for multi-rotor UAVs with signal temporal logic specifications. In *2021 Annual American control conference* (pp. 4199–4206). IEEE.

Parrilo, P. (2000). *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. (Ph.D. thesis), California Institute of Technology.

Raković, S. V., & Cheng, Q. (2013). Homothetic tube MPC for constrained linear difference inclusions. In *Chinese control and decision conference* (pp. 754–761). IEEE.

Raković, S. V., Kouvaritakis, B., Cannon, M., Panos, C., & Findeisen, R. (2012). Parameterized tube model predictive control. *IEEE Transactions on Automatic Control*, *57*(11), 2746–2761.

Raković, S. V., Levine, W. S., & Açikmese, B. (2016). Elastic tube model predictive control. In *2016 Annual American control conference* (pp. 3594–3599). IEEE.

Rosolia, U., & Ames, A. D. (2021). Multi-rate control design leveraging control barrier functions and model predictive control policies. *IEEE Control Systems Letters*, *5*(3), 1007–1012.

Seiler, P. (2013). SOSOPT: A toolbox for polynomial optimization. arXiv preprint arXiv:1308.1889.

Singh, S., Chen, M., Herbert, S. L., Tomlin, C. J., & Pavone, M. (2020). Robust tracking with model mismatch for fast and safe planning: An SOS optimization approach. In M. Morales, L. Tapia, G. Sánchez-Ante, & S. Hutchinson (Eds.), *Algorithmic foundations of robotics XIII* (pp. 545–564). Cham: Springer International Publishing.

Singh, S., Majumdar, A., Slotine, J., & Pavone, M. (2017). Robust online motion planning via contraction theory and convex optimization. In *2017 IEEE international conference on robotics and automation* (pp. 5883–5890).

Smith, S. W., Yin, H., & Arcak, M. (2019). Continuous abstraction of nonlinear systems using sum-of-squares programming. In *2019 IEEE conference on decision and control* (pp. 8093–8098).

Tedrake, R., Manchester, I. R., Tobenkin, M., & Roberts, J. W. (2010). LQR-trees: Feedback motion planning via sums-of-squares verification. *International Journal of Robotics Research*, *29*(8), 1038–1052.

Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., Duijkeren, N. v., Zanelli, A., et al. (2021). Acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 1–37.

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*(1), 25–57.

Yin, H., Arcak, M., Packard, A. K., & Seiler, P. (2021). Backward reachability for polynomial systems on a finite horizon. *IEEE Transactions on Automatic Control*, *66*(12), 6025–6032.

Yin, H., Bujarbaruah, M., Arcak, M., & Packard, A. (2020). Optimization based planner–tracker design for safety guarantees. In *2020 Annual American control conference* (pp. 5194–5200). IEEE.

Yu, S., Maier, C., Chen, H., & Allgöwer, F. (2013). Tube MPC scheme based on robust control invariant set with application to Lipschitz nonlinear systems. *Systems & Control Letters*, *62*(2), 194–200.