Model Predictive Control with Reach-avoid Analysis

Dejin Ren[1,2] , **Wanli Lu**[3] , **Jidong Lv** [3] , **Lijun Zhang**[1,2] and **Bai Xue**[1,2*]
[1]State Key Lab. of Computer Science, Institute of Software, CAS, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]National Engineering Research Center of Rail Transportation Operation and Control System, Beijing Jiaotong University, Beijing, China
{rendj, zhanglj, xuebai}@ios.ac.cn, {luwanli, jdlv}@bjtu.edu.cn

Abstract

In this paper we investigate the optimal controller synthesis problem, so that the system under the controller can reach a specified target set while satisfying given constraints. Existing model predictive control (MPC) methods learn from a set of discrete states visited by previous (sub-)optimized trajectories and thus result in computationally expensive mixed-integer nonlinear optimization. In this paper a novel MPC method is proposed based on reach-avoid analysis to solve the controller synthesis problem iteratively. The reach-avoid analysis is concerned with computing a reach-avoid set which is a set of initial states such that the system can reach the target set successfully. It not only provides terminal constraints, which ensure feasibility of MPC, but also expands discrete states in existing methods into a continuous set (i.e., reach-avoid sets) and thus leads to nonlinear optimization which is more computationally tractable online due to the absence of integer variables. Finally, we evaluate the proposed method and make comparisons with state-of-the-art ones based on several examples.

1 Introduction

Control synthesis is a fundamental problem which automatically constructs a control strategy that induces a system to exhibit a desired behavior. Due to the ability of handling control and state constraints and yielding high performance control systems [Camacho and Alba, 2013], control design methods based on MPC have gained great popularity and found wide acceptance in industrial applications, ranging from autonomous driving [Verschueren *et al.*, 2014; Kabzan *et al.*, 2019] to large scale interconnected power systems [Ernst *et al.*, 2008; Mohamed *et al.*, 2011].

In MPC design methods one issue is to guarantee feasibility of the successive optimization problems [Scokaert and Rawlings, 1999]. Because MPC is 'greedy' in its nature, i.e., it only searches for the optimal strategy within a finite horizon, an MPC controller may steer the state to a region starting from where the violation of hard state constraints cannot be avoided. Although this feasibility issue could be solved by using a sufficiently long prediction horizon, we may not be able to afford the computation overhead due to the limited computational resources. Consequently, several solutions towards the feasibility issue are proposed [Zheng and Morari, 1995; Zeng *et al.*, 2021; Ma *et al.*, 2021]. Besides the feasibility issue of satisfying all hard state constraints [Zheng and Morari, 1995; Zeng *et al.*, 2021; Ma *et al.*, 2021], a system is also desired to achieve certain performance objective in an optimal sense. In existing literature, stability performance of approaching an equilibrium within the MPC framework is extensively studied. One common solution of achieving stability is adding Lyapunov functions as terminal costs and/or corresponding invariant sets as terminal sets [Michalska and Mayne, 1993; Limon *et al.*, 2005; Limón *et al.*, 2006; Mhaskar *et al.*, 2006; de la Peña and Christofides, 2008; Wu *et al.*, 2019; Grandia *et al.*, 2020]. This has motivated significant research work on applications of this control design to nonlinear processes [Yao and Shekhar, 2021]. However, in many real applications stability performance is demanding. For example, a spacecraft rendezvous may require the chaser vehicle to be at a certain position relative to the target, moving towards it with a certain velocity. All of these quantities are specified with some tolerance, forming a target region in the state space. When the chaser enters that region a physical connection would be made and the maneuver is complete. However, since the target velocity is non-zero, the region is not invariant and stability cannot be achieved. Regarding this practical issue, the formulation in this paper replaces the notion of stability with reachability: given a target set, the system will achieve the reachability objective of reaching the target set in finite time successfully. To the best of our knowledge, the learning model predictive control (LMPC) proposed in [Rosolia and Borrelli, 2017], which utilizes the historical data to improve suboptimal controllers iteratively, is the only method within the MPC framework which can solve this problem. However, it leads to mixed-integer nonlinear programming problems which are fundamentally challenging to solve.

In this work, we consider control tasks where the goal is to steer the system from a starting configuration to a target set in finite time, while satisfying state and input constraints. The control task is formalized as a reach-avoid optimization

*Corresponding author

problem. For solving the optimization problem, we propose a novel learning based MPC algorithm which fuses iterative learning control (ILC) [Arimoto *et al.*, 1984] and reach-avoid analysis in [Zhao *et al.*, 2022]. The proposed algorithm utilizes MPC to iteratively improve upon a suboptimal controller. Based on a suboptimal controller obtained in the previous iteration, the reach-avoid analysis is to compute a set of initial states such that the closed-loop system can reach the target set without a violation of the state and input constraints. In our algorithm, the reach-avoid analysis not only provides terminal constraints which ensure feasibility of the MPC, but also expands the viability space of the system and thus facilitates exploring better trajectories with lower costs. Finally, several examples demonstrate the benefits of our algorithm over state-of-the-art ones.

The closest work to the present one is [Rosolia and Borrelli, 2017]. Due to the use of a set of discrete states visited by previous (sub-)optimized trajectories, computationally expensive mixed-integer nonlinear programming problems have to be addressed online in the proposed LMPC, limiting its application in practice. Our method is inspired by [Rosolia and Borrelli, 2017]. However, it incorporates the recently developed reach-avoid analysis technique and expands discrete states into a continuous (reach-avoid) set, which not only facilitates exploring better trajectories but also leads to a novel MPC formulation which involves solving more computationally tractable nonlinear programming problems online. Besides, this work is also close to the ones on model-based safe reinforcement learning, such as [Thomas *et al.*, 2021; Luo and Ma, 2021; Wen and Topcu, 2018]. However, these studies prioritize enforcing the safety objective rather than achieving a joint objective of safety and reachability.

This paper is structured as follows. In Section 2 the reach-avoid problem of interest is introduced. Then, we detail our algorithm for solving the reach-avoid problem in Section 3, and evaluate it on several examples in Section 4. Finally, we conclude this paper in Section 5.

2 Preliminaries

In this section we introduce the reach-avoid optimization problem of interest. Throughout this paper, $\mathbb{R}^n$ and $\mathbb{N}$ denote the set of n-dimensional real vectors and non-negative integers, respectively. $\mathbb{R}[\cdot]$ denotes the ring of polynomials in variables given by the argument. $\sum[x]$ represents the set of sum-of-squares polynomials over variable x, i.e., $\sum[x] = \{p \in \mathbb{R}[x] \mid p = \sum_{i=1}^{k} q_i^2, q_i \in \mathbb{R}[x], i = 1, 2, \ldots, k\}$.

2.1 Problem Statement

Consider a discrete-time dynamical system

$$x(t+1) = f(x(t), u(t)), t \in \mathbb{N}, \tag{1}$$

where $f(\cdot) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the system dynamics, $x(\cdot) : \mathbb{N} \to \mathbb{R}^n$ is the state and $u(\cdot) : \mathbb{N} \to \mathcal{U} \subseteq \mathbb{R}^m$ is the control.

Definition 1. *A control policy is a sequence of control inputs* $\pi = \{u(i)\}_{i\in\mathbb{N}}$*, where* $u(i) : \mathbb{N} \to \mathcal{U}$*. Furthermore, we define* Π *as the set of all control policies.*

Given a safe set $\mathcal{X} = \{x \in \mathbb{R}^n \mid w(x) \leq 0\}$ and a target set $\mathcal{T} = \{x \in \mathbb{R}^n \mid g(x) \leq 0\}$ with $\mathcal{T} \subseteq \mathcal{X}$, a control policy $\pi \in \Pi$ is safe with respect to a state x_0, if the control policy π will drive system (1) starting from $x(0) = x_0$ to reach the target set $\mathcal{T}$ in finite time while staying inside the safe set $\mathcal{X}$ before the first target hitting time. The associated cost of reaching the target set $\mathcal{T}$ safely is defined below.

Definition 2. *Given a state* $x(0) = x_0 \in \mathcal{X} \setminus \mathcal{T}$*, and a safe policy* $\pi = \{u(i)\}_{i\in\mathbb{N}} \in \Pi$*, the cost with respect to the state* x_0 *and the policy* π *is defined below:*

$$Q(x_0, \pi) = \sum_{k=0}^{L-1} h(x(k), u(k)), \tag{2}$$

where $x(k+1) = f(x(k), u(k))$*,* $L = \min\{i \in \mathbb{N} \mid x(i) \in \mathcal{T}\}$*, and* $h(\cdot,\cdot)$ *is continuous and satisfies*

$$h(x, u) \geq 0, \forall (x, u) \in \mathcal{X} \times \mathcal{U}. \tag{3}$$

In this paper, we would like to synthesize a safe control policy $\pi \in \Pi$ with respect to a specified initial state x_0 such that system (1) will enter the target set $\mathcal{T}$ with the minimal cost in finite time while staying inside the safe set $\mathcal{X}$ before the first target hitting time.

Problem 1. *We attempt to solve the following reach-avoid optimization problem:*

$$\begin{aligned} J(x_0) = &\min_{T, u(i), i=0,\ldots,T-1} \sum_{i=0}^{T-1} h(x(i), u(i)) \\ &s.t. \begin{cases} x(i+1) = f(x(i), u(i)), x(0) = x_0, \\ u(i) \in \mathcal{U}, x(i) \in \mathcal{X}, \\ x(T) \in \mathcal{T}, \\ i = 0, \ldots, T-1, \\ T \in \mathbb{N}, \end{cases} \end{aligned} \tag{4}$$

where T *is the first time of hitting the target set* $\mathcal{T}$*.*

Due to uncertainties on the target hitting time, it is challenging to solve optimization (4) directly. However, the computational complexity can be reduced when searching for a feasible sub-optimal solution. In this paper we will adopt a learning strategy to solving (4), which iteratively improves upon already known suboptimal policies as the LMPC algorithm in [Rosolia and Borrelli, 2017]. Consequently, an initial feasible policy is needed, as formulated in Assumption 1.

Assumption 1. *Assume an initial policy* $\pi^0 = \{u^0(i)\}_{i\in\mathbb{N}}$*, which can drive system* (1) *starting from* x_0 *to the target set* $\mathcal{T}$ *in finite time safely, is available. The corresponding trajectory of system* (1) *can be obtained and is denoted by* $\{x^0(i)\}_{0\leq i\leq L^0}$*, where,*

$$\begin{cases} x^0(i+1) = f(x^0(i), u^0(i)), i = 0, \ldots, L^0 - 1, \\ x^0(0) = x_0, x^0(L^0) \in \mathcal{T}. \end{cases}$$

Remark 1. *The availability of a feasible control policy is not restrictive in practice for a number of applications. For instance, with race cars one can always run a path at very low speed to obtain a control policy.*

2.2 Guidance-barrier Functions

In this subsection we introduce guidance-barrier functions. They not only provide terminal constraints in our MPC method escorting system (1) to the target set $\mathcal{T}$ safely, but also generate a set which curves out a viability space for system (1) to reach the target set $\mathcal{T}$ safely.

Definition 3. *Given the safe set $\mathcal{X}$, target set $\mathcal{T}$ and a factor $\lambda \in (1, \infty)$, a bounded function $v(x) : \mathcal{Y} \to \mathbb{R}$ is a guidance-barrier function of system* (1) *with the feedback controller $\hat{u}(\cdot) : \mathcal{X} \to \mathcal{U}$, if it satisfies the following constraints:*

$$\begin{cases} v(f(x, \hat{u}(x))) \geq \lambda v(x), \forall x \in \mathcal{X} \setminus \mathcal{T}, \\ v(x) \leq 0, \forall x \in \mathcal{Y} \setminus \mathcal{X}, \\ v(x) \leq M, \forall x \in \mathcal{T}, \\ v(x_0) > 0, \end{cases} \tag{5}$$

where $\mathcal{Y} = \{y \mid y = f(x, u), u \in \mathcal{U}, x \in \mathcal{X}\} \cup \mathcal{X}$, and M is a user-defined positive number.

When $f(x, \hat{u}(x))$ is polynomial over x, and $\mathcal{X}$ is semi-algebraic set, i.e., $f(x, \hat{u}(x)), w(x) \in \mathbb{R}[x]$, a set $\mathcal{Y}$ of the form $\{x \mid w_0(x) \leq 0\}$ with $w_0(x) \in \mathbb{R}[x]$ can be obtained using program (3) in [Zhao *et al.*, 2022].

Remark 2. *It is worth remarking here that if* (5) *holds, for system* (1) *with $\hat{u}(\cdot) : \mathcal{X} \to \mathcal{U}$, the induced trajectory starting from x_0 will hit the target set $\mathcal{T}$ within a bounded amount of time being less than or equal to $\log_\lambda \frac{M}{v(x_0)}$ (It can obtained according to $\lambda^T v(x_0) \leq v(x(T)) \leq M$, where T is the first hitting time of $\mathcal{T}$.).*

A reach-avoid set $\mathcal{R} = \{x \in \mathcal{X} \mid v(x) > 0\}$ can be computed via solving constraint (5), which is a set of states such that there exists a control policy $\pi \in \Pi$ driving system (1) to enter the target set $\mathcal{T}$ in finite time while staying inside the safe set $\mathcal{X}$ before the first target hitting time.

Theorem 1. *Given the safe set $\mathcal{X}$, target set $\mathcal{T}$ and a factor $\lambda \in (1, \infty)$, if $v(x) : \mathcal{Y} \to \mathbb{R}$ is a guidance-barrier function of system* (1) *with the feedback controller $\hat{u}(\cdot) : \mathcal{X} \to \mathcal{U}$, then $\mathcal{R} = \{x \in \mathcal{X} \mid v(x) > 0\}$ is a reach-avoid set.*

Theorem 1 can be assured by Corollary 1 in [Zhao *et al.*, 2022]. Due to space limitations, we omitted the proof herein.

Remark 3. *One of admissible control policies $\pi \in \Pi$ such that system* (1) *satisfies the reach-avoid specification can be constructed by the feedback controller $\hat{u}(\cdot) : \mathcal{X} \to \mathcal{U}$: when system* (1) *is in state $x(i) \in \mathcal{R}$, the corresponding control action is $u(i) = \hat{u}(x(i))$. We denote such a control policy by $\pi_{\hat{u}}$ in the rest of this paper.*

3 Reach-avoid Model Predictive Control

In this section we elucidate our learning-based algorithm for solving optimization (4) in Problem 1, which is built upon a so-called reach-avoid model predictive control (RAMPC). The proposed RAMPC is constructed based on a guidance-barrier function.

Our RAMPC algorithm is iterative and at each iteration it mainly consists of three steps. The first step is to synthesize a feedback controller by interpolating the suboptimal state-control pair obtained in the previous iteration. Then, a guidance-barrier function satisfying (5) with the synthesized feedback controller is computed. Finally, based on the computed guidance-barrier function a MPC controller, together with its resulting state trajectory, is generated online. The framework of the algorithm is summarized in Alg. 1.

For solving (5) in Alg. 1, when $f(x, \hat{u}(x))$ is polynomial over x, and $\mathcal{Y}$, $\mathcal{X}$ and $\mathcal{T}$ are semi-algebraic sets, i.e.,

Algorithm 1 The framework for solving optimization (4).

Require: system (1); initial state x_0; safe set $\mathcal{X}$; target set $\mathcal{T}$; control input set $\mathcal{U}$; feasible state-control trajectory $\{(x^0(i), u^0(i))\}_{i=0}^{L^0-1}$, of which the cost is $J^0(x_0) = \sum_{i=0}^{L^0-1} h(x^0(i), u^0(i))$; factor λ and bound M in (5); maximum iteration number K; prediction horizon N in RAMPC; termination threshold $\xi > 0$.
Ensure: Return $J^*(x_0)$.
for $j = 0 : K$ **do**
  apply interpolation techniques to compute a feedback controller $\hat{u}^j(\cdot) : \mathcal{X} \to \mathcal{U}$ based on the j-th state-control trajectory $\{(x^j(i), u^j(i))\}_{i=0}^{L^j-1}$;
  compute $v^j(x)$ via solving (5) with $\hat{u}(x) = \hat{u}^j(x)$;
  solve RAMPC optimization, which is constructed with $v^j(x)$, to obtain a state-control pair $\{(x^{j+1}(i), u^{j+1}(i))\}_{i=0}^{L^{j+1}-1}$ and the cost $J^{j+1}(x_0)$:
  if $J^{j+1}(x_0) - J^j(x_0) \geq -\xi$ **then**
    Return $J^*(x_0) = J^{j+1}(x_0)$;
  end if
end for

$f(x, \hat{u}(x)), w_0(x), w(x), g(x) \in \mathbb{R}[x]$, the problem of solving constraint (5) can be transformed into a semi-definite programming problem (6),

$$\begin{cases} v(f(x, \hat{u}(x))) - \lambda v(x) + s_1(x)w(x) - s_2(x)g(x) \in \sum[x] \\ -v(x) + s_3(x)w_0(x) - s_4(x)w(x) \in \sum[x] \\ M - v(x) + s_5(x)g(x) \in \sum[x] \\ v(x_0) > 0, \end{cases} \tag{6}$$

where $s_j(x) \in \sum[x], j = 1, \ldots, 5$, $v(x) \in \mathbb{R}[x]$.

Otherwise, sample-based approaches in the context of randomized algorithms [Tempo *et al.*, 2013] for robust convex optimization can be employed to solve constraint (5). The basis of these approaches is the Almost Lyapunov condition [Liu *et al.*, 2020], which allow the Lyapunov conditions to be violated in restricted subsets of the space while still ensuring stability properties. Although results from these approaches lack rigorous guarantees, nice empirical performances demonstrated the practicality of these approaches in existing literature (e.g., [Chang and Gao, 2021]). We will revisit it in our future work.

Remark 4. *In Alg. 1, the computations in the first two steps, i.e., synthesizing feedback controllers and solving* (5)*, can be carried out offline. Online computations occur only in the third step, which generate MPC controllers.*

In the following subsection we will introduce the RAMPC optimization in Alg. 1.

3.1 Reach-avoid Model Predictive Control

This subsection introduces the RAMPC optimization in Alg. 1, which involves solving the MPC optimization problem online :

$$J_{t \to t+N}^{\mathrm{RAMPC},j}(x_t^j) = \min_{u_{k|t}} \left[\sum_{k=0}^{N-1} h(x_{k|t}^j, u_{k|t}^j) + Q^j(x_{N|t}^j, \pi_{\hat{u}^{j-1}}) \right]$$

$$\text{s.t.} \begin{cases} x^j_{k+1|t} = f(x^j_{k|t}, u^j_{k|t}), \\ u^j_{k|t} \in \mathcal{U}, x^j_{k|t} \in \mathcal{X}, \\ k = 0, 1, \dots, N-1, \\ v^{j-1}(x^j_{N|t}) \geq \begin{cases} \lambda^N v^{j-1}(x_0), \text{if } t = 0, \\ \lambda v^{j-1}(x^j_{N|t-1}), \text{otherwise.} \end{cases} \\ x^j_{0|t} = x^j_t \end{cases} \tag{7}$$

where the superscript j is the iteration index, $x^j_{k|t}$ is the state predicted k time steps ahead, computed at time t, initialized at $x^j_{0|t} = x^j_t$ with $x^j_0 = x_0$, and similarly for $u^j_{k|t}$, and $Q^j(x^j_{N|t}) = \sum_{i=0}^{L-1} h(x^j_{i+N|t}, \hat{u}^{j-1}(x^j_{i+N|t}))$ is the cost with respect to the state $x^j_{N|t}$ and the control policy $\pi_{\hat{u}^{j-1}}$.

In (7), the terminal constraint

$$v^{j-1}(x^j_{N|t}) \geq \begin{cases} \lambda^N v^{j-1}(x_0), \text{if } t = 0, \\ \lambda v^{j-1}(x^j_{N|t-1}), \text{otherwise.} \end{cases} \tag{8}$$

guarantees that the terminal state $x^j_{N|t}$ lies in the reach-avoid set $\mathcal{R}^{j-1} = \{x \in \mathcal{X} \mid v^{j-1}(x) > 0\}$, which carves out a larger continuous viability space (visualized as the cyan region in Fig. 1) for system (1) to be capable of achieving the reach-avoid objective. This is different from the LMCP method in [Rosolia and Borrelli, 2017], which restricts terminal states within previously explored discrete states (visualized as pink points in Fig. 1) and thus leads to computationally demanding mixed-integer nonlinear optimization. As to the practical computations of the cost $Q^j(\cdot, \pi_{\hat{u}^{j-1}}) : \mathcal{R}^{j-1} \to \mathbb{R}$, we will give a detailed introduction in Subsection 3.2.

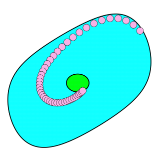

Figure 1: An illustration of continuous (reach-avoid) sets in our RAMPC method and discrete states in the LMPC method, the green region denotes a target set.

Let

$$\mathbf{u}^{*,j}_{0:N|t} = \{u^{*,j}_{i|t}\}_{i=0}^{N-1} \text{ and } \mathbf{x}^{*,j}_{0:N|t} = \{x^{*,j}_{i|t}\}_{i=0}^{N} \tag{9}$$

be the optimal solution to (7) at time t and $J^{\text{RAMPC},j}_{t\to t+N}(x^j_t)$ the corresponding optimal cost. Then, at time t of the iteration j, the first element $u^j(t) = u^{*,j}_{0|t}$ of $\mathbf{u}^{*,j}_{0:N|t}$ is applied to the system (1) and thus the state of system (1) turns into

$$x^j_{t+1} = x^{*,j}_{1|t} = f(x^j(t), u^j(t)), \tag{10}$$

where $x^j(t) = x^{*,j}_{0|t}$, which will be the used to update the initial state in (7) for subsequent computations. Once there exist $t \in \mathbb{N}$ and $l \leq N$ s.t. $x^j_{l|t} \in \mathcal{T}$, we terminate computations in this iteration: from the state $x^{*,j}_{0|t}$ on, we will apply the control actions $\{u^{*,j}_{i|t}\}_{i=0}^{l-1}$ successively to system (1).

Let $\pi^j = \{u^j(t)\}_{0\leq t\leq L^j-1}$ be the control policy applied to system (1) by solving optimization (7). The resulting state-control pair $\{x^j(i), u^j(i)\}_{0\leq i\leq L^j-1}$, where $x^j(L^j) \in \mathcal{T}$, is obtained. We can conclude that the performance of π^j is no worse than that of π^{j-1}, i.e., $J^{j-1}(x_0) = \sum_{i=0}^{L^{j-1}-1} h(x^{j-1}(i), u^{j-1}(i)) \geq \sum_{i=0}^{L^j-1} h(x^j(i), u^j(i)) = J^j(x_0)$. This conclusion can be justified by Theorem 3.

Under Assumption 2, we in Theorem 2 show that the RAMPC (7) is feasible and system (1) with controllers computed by solving (7) satisfies the reach-avoid property, and in Theorem 3 show that the j-th iteration cost $J^j(x_0)$ is non-increasing as j increases in RAMPC (7). Their proofs can be found in the appendix of the extended version [Ren *et al.*, 2023].

Assumption 2. *At each iteration $0 \leq j \leq K$ the computed reach-avoid set $\mathcal{R}^j = \{x \in \mathcal{X} \mid v^j(x) > 0\}$ via solving (5) is non-empty. In addition, we assume that it includes the state trajectory $\{x^j(i)\}_{i=0}^{L^j-1}$*[1].

Theorem 2. *For each iteration $j \leq K$, the RAMPC (7) is feasible for $1 \leq t \leq L^j - 1$; also, system (1) with controllers obtained by solving RAMPC (7) can reach the target set $\mathcal{T}$ within the time interval $[0, \log_\lambda \frac{M}{v_0}]$ while satisfying all of state and input constraints.*

Theorem 3. *Consider system (1) with controllers obtained by solving (7). Then, the iteration cost $J^j(x_0)$ is non-increasing with the iteration index j.*

3.2 Estimating Terminal Costs via Scenario Optimization

In practice, the exact terminal cost $Q^j(\cdot, \pi_{\hat{u}^{j-1}}) : \mathcal{R}^{j-1} \to \mathbb{R}$ is challenging, even impossible to obtain. In this subsection, we present a linear programming method based on the scenario optimization [Calafiore and Campi, 2006] to compute an approximation of the terminal cost $Q^j(\cdot, \pi_{\hat{u}^{j-1}}) : \mathcal{R}^{j-1} \to \mathbb{R}$ in the probably approximately correct (PAC) sense.

Let $\{x_i\}_{i=1}^{N'}$ be the independent samples extracted from $\mathcal{R}^{j-1} = \{x \in \mathcal{X} \mid v^{j-1}(x) > 0\}$ according to the uniform distribution $\mathbb{P}$, and $Q^j(x_i, \pi_{\hat{u}^{j-1}})$ be the corresponding cost of the roll-out from x_i with the controller $\pi_{\hat{u}^{j-1}}$. Such cost can be simply computed by summing up the cost along the closed-loop realized trajectory until the target set $\mathcal{T}$ is reached. Finally, using the set of sample states and correspding costs $\{(x_i, Q^j(x_i, \pi_{\hat{u}^{j-1}}))\}_{i=1}^{N'}$, we approximate the terminal cost $Q^j(\cdot, \pi_{\hat{u}^{j-1}}) : \mathcal{R}^{j-1} \to \mathbb{R}$ using the scenario optimization [Calafiore and Campi, 2006].

A linearly parameterized model template $Q^j_a(c_1, \dots, c_l, x)$, $k \geq 1$ is utilized, which is a linear function in unknown parameters $\boldsymbol{c} = (c_1, \dots, c_l)$ but can be nonlinear over x. Then we construct the following linear program over $\boldsymbol{c}$ based on the family of given datum

[1]This assumption can be realized by interpolating the feedback controller $\hat{u}^j(\cdot) : \mathcal{X} \to \mathcal{U}$ s.t. $\hat{u}^j(x^j(i)) = u^j(i), i = 0, \dots, L^j - 1$. This requirement mainly serves for our theoretical analysis since it can guarantee that our algorithm can iteratively improve the policy (Theorem 3). However, in practical this requirement is not indispensable regarding efficient computations.

$\{(x_i, Q^j(x_i, \pi_{\hat{u}^{j-1}}))\}_{i=1}^{N'}$:

$$\min_{\boldsymbol{c},\delta} \delta$$
$$\text{s.t.} \begin{cases} Q_a^j(\boldsymbol{c}, x_i) - Q^j(x_i, \pi_{\hat{u}^{j-1}}) \leq \delta, \\ Q^j(x_i, \pi_{\hat{u}^{j-1}}) - Q_a^j(\boldsymbol{c}, x_i) \leq \delta, \\ -U_c \leq c_l \leq U_c, \\ i = 1, \ldots, N', \\ 0 \leq \delta, \end{cases} \tag{11}$$

where $U_c \geq 0$ is a pre-specified upper bound for c_i, $i = 1, \ldots, l$.

Denote the optimal solution to (11) by $(\boldsymbol{c}^*, \delta^*)$. The discrepancy between $Q_a^j(\boldsymbol{c}^*, x)$ and $Q^j(x, \pi_{\hat{u}^{j-1}})$ is formally characterized by two parameters: the error probability $\epsilon \in (0, 1)$ and confidence level $\beta \in (0, 1)$.

Theorem 4. *Let* $(\boldsymbol{c}^*, \delta^*)$ *be an optimal solution to* (11), $\epsilon \in (0, 1)$, $\beta \in (0, 1)$ *and*

$$\epsilon \geq \frac{2}{N'}(\ln \frac{1}{\beta} + l + 1).$$

Then we have that with at least $1 - \beta$ *confidence,*

$$\mathbb{P}(\{x \in \mathcal{R} \mid |Q_a^j(\boldsymbol{c}^*, x) - Q^j(x, \pi_{\hat{u}^{j-1}})| \leq \delta^*\}) \geq 1 - \epsilon.$$

Thus, we relax optimization (4) into the following form:

$$J_{a,t\to t+N}^{\text{RAMPC},j}(x_t^j) = \min_{u_{k|t}} \left[\sum_{k=0}^{N-1} h(x_{k|t}, u_{k|t}) + Q_a^{j-1}(\boldsymbol{c}^*, x_{N|t}) \right]$$
$$\text{s.t.} \begin{cases} x_{k+1|t} = f(x_{k|t}, u_{k|t}), \\ u_{k|t} \in \mathcal{U}, x_{k|t} \in \mathcal{X}, \\ k = 0, 1, \ldots, N-1, \\ v(x_{N|t}) \geq \begin{cases} \lambda^N v(x_0), \text{if } t = 0, \\ \lambda v(x_{N|t-1}), \text{otherwise.} \end{cases} \\ x_{0|t} = x_t^j \end{cases} \tag{12}$$

where $Q_a^{j-1}(\boldsymbol{c}^*, x_{N|t})$ is the approximate terminal cost at the state $x_{N|t}$, which is obtained via solving (11).

Proposition 1. *With at least* $1 - \beta$ *confidence,*

$$|J_{a,t\to t+N}^{\text{RAMPC},j}(x_t^j) - J_{t\to t+N}^{\text{RAMPC},j}(x_t^j)| \leq \delta^*$$

holds with the probability larger than or equal to $1 - \epsilon$*, i.e.,*

$$\mathbb{P}(\{x_t^j \in \mathcal{R}^{j-1} \mid |J_{a,t\to t+N}^{\text{RAMPC},j}(x_t^j) - J_{t\to t+N}^{\text{RAMPC},j}(x_t^j)| \leq \delta^*\}) \geq 1 - \epsilon.$$

Relying on (12), we summarize our RAMPC algorithm for solving optimization (4) in Algorithm 2.

Algorithm 2 The RAMPC algorithm for solving (4).

Require: system (1) with an initial state x_0, a safe set $\mathcal{X}$, a target set $\mathcal{T}$ a control input set $\mathcal{U}$ and a feasible state-control trajectory $\{(x^0(i), u^0(i))\}_{i=0}^{L^0-1}$, of which the corresponding cost is $J^0(x_0) = \sum_{i=0}^{L^0-1} h(x^0(i), u^0(i))$; factor λ and bound M in (5); iteration number K, prediction horizon N and termination threshold $\xi > 0$; probability error ϵ and confidence level δ in PAC approximations.
Ensure: Return $J^*(x_0)$.
for all $j = 0 : K$ **do**
1 apply interpolation techniques to compute $\hat{u}^j(\cdot) : \mathcal{X} \to \mathcal{U}$ for the j-th state-control trajectory $\{(x_i^j, u_i^j)\}_{i=0}^{L^1-1}$;
2 obtain $\mathcal{R}^j = \{x \in \mathcal{X} \mid v^j(x) > 0\}$ via solving (5) with $\hat{u}(x) = \hat{u}^j(x)$;
3 compute the PAC terminal cost $Q_a^j(\cdot) : \mathcal{R}^j \to \mathbb{R}$ via solving optimization (11);
4 Solving MPC optimization (12) with v^j to obtain a state-control pair $\{(x^{j+1}(i), u^{j+1}(i))\}_{i=0}^{L^{j+1}-1}$ and the cost $J^{j+1}(x_0)$:
  if $|J^{j+1}(x_0) - J^j(x_0)| \leq \xi$ **then**
    Return $J^*(x_0) = J^{j+1}(x_0)$;
  end if
end for

4 Examples

In this section we evaluate our RAMPC algorithm, i.e., Alg. 2, and make comparisons with the LMPC algorithm in [Rosolia and Borrelli, 2017] on several examples. All the experiments were run on MATLAB 2022b with CPU 12th Gen Intel(R) Core(TM) i9-12900K and RAM 64 GB. Constraint (5) is solved by encoding it into sum-of-squares constraints which is treated by the semi-definite programming solver MOSEK; the nonlinear programming (7) and the mixed-integer nonlinear programming in the LMPC algorithm in [Rosolia and Borrelli, 2017] are solved using YALMIP [Lofberg, 2004]. In addition, we take $\hat{u}^j(\cdot)$ as a linear function to interpolate the j-th state-control trajectory $\{(x_i^j, u_i^j)\}_{i=0}^{L^j-1}$ at each iteration $j \geq 1$. The configuration parameters in Alg. 2 for all examples are shown in Table 1.

Example	λ	M	N	K	ξ	δ	ϵ
Ex. 1	1.001	1	4	8	0.1	0.1	0.1
Ex. 2	1.001	1	3	8	0.1	0.05	0.05
Ex. 3	1.001	1	4	6	0.002	0.1	0.1

Table 1: Configuration parameters in Alg. 2 for examples.

Example 1. *Consider the drone system from [Rosolia and Borrelli, 2017],*

$$x_{t+1} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_t$$

where $x_t = (p_t, v_t)^\top$, p_t *and* v_t *are respectively the position and velocity of the drone at time* t, u_t *is the control input and* dt *is the control interval which is equal to* 0.1.

We assume $\mathcal{X} = \{(p, v)^\top \mid \frac{p^2}{8^2} + \frac{v^2}{8^2} - 1 \leq 0\}$*, initial state* $x_0 = (4, -6)^\top$*, target set* $\mathcal{T} = \{(p, v)^\top \mid p^2 + v^2 - 0.5^2 \leq 0\}$ *and control input set* $\mathcal{U} = \{u \mid -0.5 \leq u \leq 0.5\}$*. The set* $\mathcal{Y} = \{(p, v)^\top \mid \frac{p^2}{8^2} + \frac{v^2}{8^2} - 2 \leq 0\}$ *in constraint* (5) *is obtained by solving a semi-define program as in [Zhao* et al.*, 2022]. The cost in optimization* (4) *is* $h(x, u) = \|x\|_2^2 + \|u\|_2^2$*. For simplicity of presentation herein, we do not present the initial state-control trajectory which is a long sequence. The initial state-control trajectory is induced by the feedback controller* $\hat{u}^0(p, v) = -0.04p - 0.1v$*. For scenario optimization* (11),

Iteration	Iteration Cost		Time Cost(seconds)	
	RAMPC	LMPC	RAMPC	LMPC
0	369.8267	369.8267		
1	215.1007	222.2482	3.3204	3.1741
2	215.1002	217.3604	3.0100	1.5634
3	-	215.3008	-	1.1980
4	-	215.1003	-	1.1023
5	-	215.1044	-	1.1881
total time			**6.3304**	8.2259

Table 2: Iteration cost and computation time for Example 1.

we use a template of the polynomial form $Q_a^j(\boldsymbol{c}, p, v) = c_1 + c_2 p + c_3 v + c_4 pv + c_5 p^2 + c_6 v^2$ *and* $N' = 207$ *sampled points which can be computed via Theorem 4.*

We compare our RAMPC algorithm with the LMPC one in [Rosolia and Borrelli, 2017] with the same prediction horizon and termination conditions (i.e., N *and* ξ*). Since this system is linear, instead of using a set of explored discrete states, the terminal constraint in the LMPC algorithm can be constructed by its convex hull, thus resulting in non-linear programs instead of mixed-integer nonlinear programs as mentioned in [Rosolia and Borrelli, 2017]. The performances of trajectories (shown in Figure 2) generated by both are compared. The iteration costs and computation times at each iteration of RAMPC and LMPC are presented in Table 2. It is observed that the iteration costs in our algorithm drop more quickly than those in the LMPC one. Also, our RAMPC algorithm is more efficient than the LMPC one. RAMPC terminates after* 2 *iterations with the computation time of* $6.3304s$*, but LMPC converges more slowly and has to take* 5 *iterations with the computation time of* $8.2259s$*.*

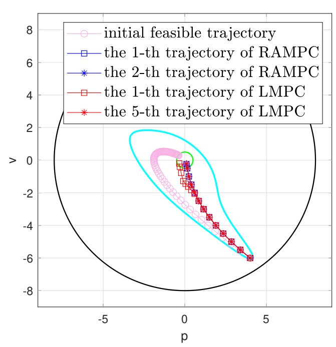

Figure 2: Trajectories in Example 1 (Green, black and cyan curve-$\partial\mathcal{T}$, $\partial\mathcal{X}$ and $\partial\mathcal{R}^0$).

Example 2. *Consider the Euler version with the time step* $\Delta t = 0.05$ *of the controlled reversed-time Van der Pol oscillator in [Drazin and Drazin, 1992]:*

$$\begin{cases} x_1(t+1) = x_1(t) - \Delta t x_2(t) \\ x_2(t+1) = x_2(t) - \Delta t((1 - x_1^2(t))x_2(t) - x_1(t)) + u(t), \end{cases}$$

with the safe set $\mathcal{X} = \{(x_1, x_2)^\top \mid \left(\frac{x_1}{2}\right)^2 + \left(\frac{x_2}{2}\right)^2 - 1 \le 0\}$*, initial state* $x_0 = (1.2, 1)^\top$*, target set* $\mathcal{T} = \{(x_1, x_2)^\top \mid x_1^2 + x_2^2 - 0.2^2 \le 0\}$ *and input set* $\mathcal{U} = \{u \mid -0.5 \le u \le 0.5\}$*.*

In (5), the set $\mathcal{Y} = \{(x_1, x_2)^\top \mid \left(\frac{x_1}{2}\right)^2 + \left(\frac{x_2}{2}\right)^2 - 2 \le 0\}$ *is computed by solving a semi-define program in [Zhao* et al.*, 2022]. In addition, the cost* $h(x, u) = \|x\|_2^2 + \|u\|_2^2$*, where* $x = (x_1, x_2)^\top$*, is adopted. The initial trajectory is induced by the controller* $\hat{u}^0(\boldsymbol{x}) \equiv 0$*. For scenario optimization (11), we use a template of the polynomial form* $Q_a^j(\boldsymbol{c}, x) = c_1 + c_2 x_1 + c_3 x_2 + c_4 x_1 x_2 + c_5 x_1^2 + c_6 x_2^2$ *and* $N' = 428$ *samples which can be computed via Theorem 4. LMPC uses the same prediction horizon and termination conditions (i.e.,* N *and* ξ*).*

Some trajectories generated by our RAMPC algorithm and the LMPC algorithm are visualized in Figure 3. Our RAMPC algorithm terminates after the third iteration, but the LMPC one does not terminate after eight iterations. Figure 3 shows that the initial trajectory and the trajectory generated by the first iteration in the LMPC algorithm are too close to be indistinguishable within the first few time steps, which on the other hand further reflects that the controller generated in the first iteration of the LMPC algorithm may not improve the performance induced by the initial control policy. Besides, it is observed that the eighth trajectory from the LMPC algorithm looks not stable and has strong oscillations, which are not expected in practice. In contrast, the trajectory generated by our algorithm is smoother.

Table 3 summarizes the iteration costs and the computation times at each iteration of our RAMPC algorithm and the LMPC one. The iteration cost induced by the initial trajectory is 64.3087*, after three iterations our RAMPC algorithm reduces the cost to* 29.2824 *with the computation time of* $31.2841s$*. In contrast, the LMPC algorithm needs more than 8 iterations with the computation time of almost more than* 3 *hours to achieve the same cost. This striking contrast definitely demonstrates that our RAMPC algorithm can solve optimization* (4) *more efficiently for some cases.*

Iteration	Iteration Cost		Time Cost(seconds)	
	RAMPC	LMPC	RAMPC	LMPC
0	64.3087	64.3087		
1	30.9693	40.0714	10.6583	45.0
2	29.2919	39.3270	10.2248	59.4
3	29.2824	38.5239	10.4010	558.5
4	-	37.6402	-	1298
5	-	36.7088	-	1653.3
6	-	35.7561	-	2033.6
7	-	35.2567	-	2115.2
8	-	34.9022	-	1706.1
total time			**31.2841**	9469.1

Table 3: Iteration cost and computation time for Example 2.

Example 3. *Consider the Euler version with the time step* $\Delta t = 0.1$ *of the controlled 3D Van der Pol oscillator from*

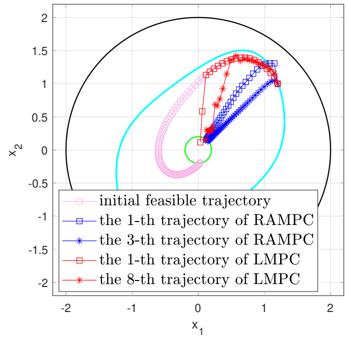

Figure 3: Trajectories in Example 2 (Green, black and cyan curve-$\partial\mathcal{T}$, $\partial\mathcal{X}$ and $\partial\mathcal{R}^0$).

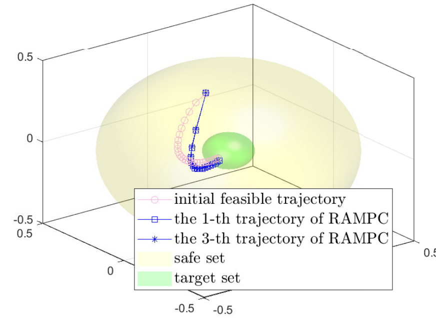

Figure 4: Trajectories in Example 3

[Korda et al.*, 2014]:*

$$\begin{cases} x_1(t+1) = & x_1(t) + \Delta t(-2x_2(t)) \\ x_2(t+1) = & x_2(t) + \Delta t(0.8x_1(t) - 2.1x_2(t) \\ & +x_3(t) + 10x_1^2(t)x_2(t)) \\ x_3(t+1) = & x_3(t) + \Delta t(-x_3(t) + x_3^3(t)) + u(t) \end{cases}$$

with the safe set $\mathcal{X} = \{(x_1, x_2, x_3)^\top \mid x_1^2 + x_2^2 + x_3^2 - 0.5^2 \leq 0\}$*, initial state* $x_0 = (0.2, 0.4, 0.1)^\top$*, target set* $\mathcal{T} = \{(x_1, x_2, x_3)^\top \mid x_1^2 + x_2^2 + x_3^2 - 0.1^2 \leq 0\}$ *and input set* $\mathcal{U} = \{u \mid -2 \leq u \leq 2\}$*.*

The set $\mathcal{Y} = \{(x_1, x_2, x_3)^\top \mid x_1^2 + x_2^2 + x_3^2 - 0.5 \leq 0\}$ *is utilized in* (5) *by solving a semi-define program, as described in [Zhao* et al.*, 2022]. We use the cost function* $h(x, u) = \|x\|_2^2 + \|u\|_2^2$ *in this example, where* $x = (x_1, x_2, x_3)^\top$*. The initial trajectory is generated by the controller* $\hat{u}^0(\boldsymbol{x}) \equiv 0$*. The template of the polynomial form* $Q_a^j(\boldsymbol{c}, x) = c_1 + c_2x_1 + c_3x_2 + c_4x_3 + c_5x_1x_2 + c_6x_1x_3 + c_7x_2x_3 + c_8x_1^2 + c_9x_2^2 + c_{10}x_3^2$ *and sample number* $N' = 267$ *computed through Theorem 4 are adopted for scenario optimization* (11)*. The prediction horizon* N *and termination condition* ξ *are same for LMPC.*

In this example, LMPC is unable to achieve the desired task of reducing the iteration cost and generating a suitable controller. In contrast, our RAMPC algorithm demonstrates good performance and can generate an effective controller. Table 4 summarizes the iteration costs and computation times at each iteration of our RAMPC algorithm. The initial trajectory and ones generated in the first and last iterations of our RAMPC algorithm are visualized in Figure 4.

Iteration	**Iteration Cost**	**Time Cost**(seconds)
0	1.3489	
1	0.8344	7.8186
2	0.8295	7.8022
3	0.8291	7.7416
total time		23.3623

Table 4: Iteration cost and computation time of RAMPC algorithm for Example 3.

Discussions on configuration parameters. Here we give a brief discussion on the configuration parameters: $\lambda, M, K, \xi, N, \delta, \epsilon$ based on the experiments (some are presented in the appendix of the extended version [Ren *et al.*, 2023]). The less conservative the computed reach-avoid set in each iteration is, the smaller the iteration cost is. Therefore, λ is recommended to be as close to one as possible, but it cannot be one. For details, please refer to [Zhao *et al.*, 2022]; the upper bound M can be any positive number since if $v(x)$ satisfies constraint (5) with $M = M'$, then $\frac{v(x)}{M'}$ also satisfies constraint (5) with $M = 1$; K and ξ, especially ξ, affect the number of iterations and thus the quality of controllers. Generally, a larger K and/or smaller ξ will improve the control performance, but the computational burden increases; the approximation error for terminal costs is formally characterized in the PAC sense using ϵ and δ. These two parameters and the size of parameters $\boldsymbol{c}$ determine the minimal number of samples for approximating terminal costs according to Theorem 4. Smaller ϵ and δ will lead to more accurate approximations of terminal costs in the PAC sense. However, regarding the overfitting issue and the computational burden of solving (11), too small ϵ and δ are not desired in practice; the prediction horizon N controls how far into the future the MPC predicts the system response. Like other MPC schemes, a longer horizon generally increases the control performance, but requires an increasingly powerful computing platform (due to solving (7) online), excluding certain control applications. In practice, an appropriate choice strongly depends on computational resources and real-time requirements. In addition, it is worth remarking here that the costs obtained by RAMPC are not sensitive to N for some cases. For details, please refer to experimental results in the appendix of the extended version [Ren *et al.*, 2023].

5 Conclusion

In this paper we proposed a novel RAMPC algorithm for solving a reach-avoid optimization problem. The RAMPC algorithm is built upon MPC and reach-avoid analysis. Rather than solving computationally intractable mixed-integer nonlinear optimization online, it addresses computationally more tractable nonlinear optimization. Moreover, due to the incorporation of reach-avoid analysis, which expands the viability space, the RAMPC algorithm can provide better controllers with a smaller number of iterations. Several numerical examples demonstrated the advantages of our RAMPC algorithm.

Acknowledgments

This research was supported by the NSFC under grant No. 61836005, CAS Project for Young Scientists in Basic Research under grant No. YSBR-040, NSFC under grant No. 52272329, and CAS Pioneer Hundred Talents Program.

References

[Arimoto *et al.*, 1984] Suguru Arimoto, Sadao Kawamura, and Fumio Miyazaki. Bettering operation of robots by learning. *Journal of Robotic systems*, 1(2):123–140, 1984.

[Calafiore and Campi, 2006] Giuseppe Carlo Calafiore and Marco C Campi. The scenario approach to robust control design. *IEEE Transactions on Automatic Control*, 51(5):742–753, 2006.

[Camacho and Alba, 2013] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

[Chang and Gao, 2021] Ya-Chien Chang and Sicun Gao. Stabilizing neural control using self-learned almost lyapunov critics. In *Proceedings of the 2021 IEEE International Conference on Robotics and Automation*, pages 1803–1809. IEEE, 2021.

[de la Peña and Christofides, 2008] David Muñoz de la Peña and Panagiotis D Christofides. Lyapunov-based model predictive control of nonlinear systems subject to data losses. *IEEE Transactions on Automatic Control*, 53(9):2076–2089, 2008.

[Drazin and Drazin, 1992] Philip G Drazin and Philip Drazin Drazin. *Nonlinear systems*. Number 10. Cambridge University Press, 1992.

[Ernst *et al.*, 2008] Damien Ernst, Mevludin Glavic, Florin Capitanescu, and Louis Wehenkel. Reinforcement learning versus model predictive control: a comparison on a power system problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):517–529, 2008.

[Grandia *et al.*, 2020] Ruben Grandia, Andrew J Taylor, Andrew Singletary, Marco Hutter, and Aaron D Ames. Nonlinear model predictive control of robotic systems with control lyapunov functions. *arXiv preprint arXiv:2006.01229*, 2020.

[Kabzan *et al.*, 2019] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.

[Korda *et al.*, 2014] Milan Korda, Didier Henrion, and Colin N Jones. Controller design and region of attraction estimation for nonlinear dynamical systems. *IFAC Proceedings Volumes*, 47(3):2310–2316, 2014.

[Limon *et al.*, 2005] Daniel Limon, Teodoro Alamo, and Eduardo F Camacho. Enlarging the domain of attraction of mpc controllers. *Automatica*, 41(4):629–635, 2005.

[Limón *et al.*, 2006] Daniel Limón, Teodoro Alamo, Francisco Salas, and Eduardo F Camacho. On the stability of constrained mpc without terminal constraint. *IEEE Transactions on Automatic Control*, 51(5):832–836, 2006.

[Liu *et al.*, 2020] Shenyu Liu, Daniel Liberzon, and Vadim Zharnitsky. Almost lyapunov functions for nonlinear systems. *Automatica*, 113:108758, 2020.

[Lofberg, 2004] Johan Lofberg. Yalmip: A toolbox for modeling and optimization in matlab. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*, pages 284–289. IEEE, 2004.

[Luo and Ma, 2021] Yuping Luo and Tengyu Ma. Learning barrier certificates: Towards safe reinforcement learning with zero training-time violations. *Advances in Neural Information Processing Systems*, 34:25621–25632, 2021.

[Ma *et al.*, 2021] Haitong Ma, Xiangteng Zhang, Shengbo Eben Li, Ziyu Lin, Yao Lyu, and Sifa Zheng. Feasibility enhancement of constrained receding horizon control using generalized control barrier function. In *Proceedings of the 4th IEEE International Conference on Industrial Cyber-Physical Systems*, pages 551–557. IEEE, 2021.

[Mhaskar *et al.*, 2006] Prashant Mhaskar, Nael H El-Farra, and Panagiotis D Christofides. Stabilization of nonlinear systems with state and control constraints using lyapunov-based predictive control. *Systems & Control Letters*, 55(8):650–659, 2006.

[Michalska and Mayne, 1993] Hannah Michalska and David Q Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 38(11):1623–1633, 1993.

[Mohamed *et al.*, 2011] TH Mohamed, H Bevrani, AA Hassan, and T Hiyama. Decentralized model predictive based load frequency control in an interconnected power system. *Energy Convers. Manag.*, 52(2):1208–1214, 2011.

[Ren *et al.*, 2023] Dejin Ren, Wanli Lu, Jidong Lv, Lijun Zhang, and Bai Xue. Model predictive control with reach-avoid analysis. *arXiv preprint arXiv:2305.08712*, 2023.

[Rosolia and Borrelli, 2017] Ugo Rosolia and Francesco Borrelli. Learning model predictive control for iterative tasks. a data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7):1883–1896, 2017.

[Scokaert and Rawlings, 1999] Pierre OM Scokaert and James B Rawlings. Feasibility issues in linear model predictive control. *AIChE Journal*, 45(8):1649–1659, 1999.

[Tempo *et al.*, 2013] Roberto Tempo, Giuseppe Calafiore, and Fabrizio Dabbene. *Randomized algorithms for analysis and control of uncertain systems: with applications*. Springer, 2013.

[Thomas *et al.*, 2021] Garrett Thomas, Yuping Luo, and Tengyu Ma. Safe reinforcement learning by imagining the near future. *Advances in Neural Information Processing Systems*, 34:13859–13869, 2021.

[Verschueren *et al.*, 2014] Robin Verschueren, Stijn De Bruyne, Mario Zanon, Janick V Frasch, and Moritz

Diehl. Towards time-optimal race car driving using nonlinear mpc in real-time. In *Proceedings of the 53rd IEEE Conference on Decision and Control*, pages 2505–2510. IEEE, 2014.

[Wen and Topcu, 2018] Min Wen and Ufuk Topcu. Constrained cross-entropy method for safe reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[Wu *et al.*, 2019] Zhe Wu, Fahad Albalawi, Zhihao Zhang, Junfeng Zhang, Helen Durand, and Panagiotis D Christofides. Control lyapunov-barrier function-based model predictive control of nonlinear systems. *Automatica*, 109:108508, 2019.

[Yao and Shekhar, 2021] Ye Yao and Divyanshu Kumar Shekhar. State of the art review on model predictive control (mpc) in heating ventilation and air-conditioning (hvac) field. *Build. Environ.*, 200:107952, 2021.

[Zeng *et al.*, 2021] Jun Zeng, Bike Zhang, and Koushil Sreenath. Safety-critical model predictive control with discrete-time control barrier function. In *Proceedings of the 2021 American Control Conference*, pages 3882–3889. IEEE, 2021.

[Zhao *et al.*, 2022] Changyuan Zhao, Shuyuan Zhang, Lei Wang, and Bai Xue. Inner-approximating robust reach-avoid sets for discrete-time polynomial dynamical systems. *IEEE Transactions on Automatic Control*, 2022.

[Zheng and Morari, 1995] Alex Zheng and Manfred Morari. Stability of model predictive control with mixed constraints. *IEEE Transactions on Automatic Control*, 40(10):1818–1823, 1995.