Computing Safe Control Inputs using Discrete-Time Matrix Control Barrier Functions via Convex Optimization

James Usevitch, Juan Augusto Paredes Salazar, and Ankit Goel

Abstract—**Control barrier functions (CBFs) have seen widespread success in providing forward invariance and safety guarantees for dynamical control systems. The majority of formulations consider continuous-time formulations, but recent focus has shifted to discrete-time dynamics. A crucial limitation of discrete-time formulations is that CBFs that are nonconcave in their argument require the solution of nonconvex optimization problems to compute safety-preserving control inputs, which inhibits real-time computation of control inputs guaranteeing forward invariance. This paper presents a novel method for computing safety-preserving control inputs for discrete-time systems with nonconvex safety sets, utilizing convex optimization and the recently developed class of matrix control barrier function techniques. The efficacy of our methods is demonstrated through numerical simulations on a bicopter system.**

I. INTRODUCTION

Guaranteeing safety and collision avoidance for dynamical systems is a fundamental problem in control theory and robotics. Control barrier functions (CBFs) have become a widely applied technique to obtain safety guarantees for systems with applications in aerospace [1], [2], robotics [3], [4], self-driving cars [5], [6], and numerous other domains [7]. The majority of prior work on CBFs has considered continuous-time systems, but a growing body of work has analyzed CBFs from a discrete-time perspective [8]–[12].

One key challenge of discrete-time CBFs is that the optimization formulation for computing a safety-preserving control input is generally nonconvex. Unlike continuous-time CBFs, the safety constraint is, in general, non-affine in the control input; therefore, the convexity of the optimization problem heavily depends on the form of the function defining the safe set. Many common collision avoidance scenarios require the use of nonconvex safe set functions, which necessitate the use of nonconvex optimization solvers that have longer runtimes and give few, if any, guarantees on finding a globally optimal solution.

James Usevitch is with the Department of Electrical and Computer Engineering, Brigham Young University, Provo, UT 84602. `james_usevitch@byu.edu`

Juan Augusto Paredes Salazar and Ankit Goel are with the Department of Mechanical Engineering, University of Maryland, Baltimore County,1000 Hilltop Circle, Baltimore, MD 21250. `japarede, ankgoel@umbc.edu`
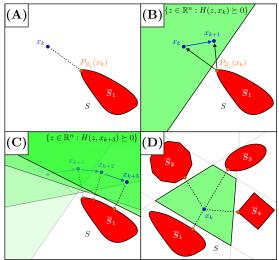
Fig. 1: Illustration of our proposed method to maintain convexity of computing safe control inputs for discrete-time systems. **(A):** The current state is projected onto the unsafe set. **(B):** A halfspace is formed which excludes the obstacle, and a control input keeping the next state in the halfspace is computed. **(C):** The process is repeated iteratively. **(D):** The proposed method can be applied to scenarios where the unsafe set is a subset of a union of convex regions.

Prior work has dealt with this challenge in several ways. In [8], the authors considered linear systems and convex quadratic discrete-time CBFs to maintain convexity of the optimization program computing safe control inputs. The authors of [9], [10], [13] assume that the composition of the CBF with the dynamics is concave in the control input. This results in a convex optimization problem; however, the assumption does not hold for many common collision avoidance scenarios involving convex obstacles. In [12], the notion of partially affine CBFs was introduced, which guarantees convexity of the optimization problem under a specific form of partitioned control-affine discrete-time dynamics. Despite the progress made by these prior works, the challenge of computing safe control inputs for discrete-time systems using solely convex optimization remains an open problem.

In addition, prior work has only considered scalar-valued control barrier functions. Recently, the notion of continuous-time matrix control barrier functions (MCBFs) was introduced [14], [15]. MCBFs, of which scalar-valued CBFs are a special case, allow for more

general classes of constraints over the cone of positive semidefinite matrices using the Loewner order. To our knowledge, the notion of MCBFs has not been extended to discrete-time dynamics.

This paper introduces the concept of exponential discrete-time matrix control barrier functions and presents a novel method for computing safe control inputs using solely convex optimization tools. Our proposed method operates by iteratively constructing convex subsets of the safe set obtained through a projection-based method. The method only requires solving two sequential convex optimization problems, and our experimental results suggest that the method is considerably faster and more reliable than nonconvex optimization approaches. The main contributions of this work thus are

- A novel definition of exponential discrete-time matrix control barrier functions,
- A novel method to compute safe control inputs for discrete-time systems with nonconvex safe sets using solely convex optimization. This includes the new notions of *safe subset functions* and *Subset-based Discrete-Time Exponential Matrix Control Barrier Functions* (SDTE-MCBFs).
- A novel extension of indefinite matrix safe sets to discrete-time matrix control barrier functions

We note that the work [11] considers a method of projecting onto convex obstacles, similar to the one presented in this paper. However, this prior work only considers the special case of scalar-valued safe set functions, does not consider the zeroing CBF property of asymptotic convergence to the safe set, and does not consider disjunctive boolean compositions of multiple CBFs.

The outline of this paper is as follows: notation and the problem formulation are given in Section II. Our main results are given in Section III. Numerical simulations demonstrating the efficacy of our proposed method are given in IV. A brief conclusion is given in Section V.

II. NOTATION AND PROBLEM FORMULATION

The closure of a set S is denoted $cl(S)$, the convex hull is denoted $co(S)$, and the power set is denoted 2^S. The set of symmetric real matrices of size $p \times p$ is denoted $\mathbb{S}^p$. The cone of positive semidefinite symmetric $p \times p$ matrices is denoted $\mathbb{S}^p_+$, and the set of positive definite $p \times p$ symmetric matrices is denoted $\mathbb{S}^p_{++}$. Given matrices $A, B \in \mathbb{R}^{n \times n}$, the notation $A \succeq 0$ and $B \succ 0$ indicate that A is positive semidefinite and B is positive definite as per the Loewner order.

The following theorem will be used to bound the eigenvalues of the matrices in $\mathbb{S}^p$.

Theorem 1 (Weyl's Inequality [16, Thm 4.3.1]). *Let $A, B \in \mathbb{S}^p$. Let the respective eigenvalues of A, B and $A+B$ be $\{\lambda_i(A)\}_{i=1}^p$, $\{\lambda_i(B)\}_{i=1}^p$, and $\{\lambda_i(A+B)\}_{i=1}^p$. Let the eigenvalues be ordered such that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_p$. Then the following holds:*

$$\lambda_{i-j+1}(A) + \lambda_j(B) \leq \lambda_i(A + B). \tag{1}$$

The following will be required for our analysis of convex sets.

Definition 1 (Projection Mapping). Given a closed, convex set $C \subset \mathbb{R}^n$, the projection mapping $P_C : \mathbb{R}^n \to \mathbb{R}^n$ is defined as $P_C(x) = \arg\min_{z \in C} \|z - x\|$.

Definition 2 (Normal Cone [17]). The normal cone mapping of a convex set $C \subset \mathbb{R}^n$, denoted $N_C : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is defined as $N_C(x) \triangleq \{v \in \mathbb{R}^n : v^\mathsf{T}(z - x) \leq 0 \text{ for all } z \in C\}$.

Theorem 2 ([17, Prop. 6.17]). *Given a convex set $C \subset \mathbb{R}^n$, the normal cone mapping N_C and projection mapping P_C are related as follows, where I indicates the identity mapping:*

$$N_C = P_C^{-1} - I, \qquad P_C = (I + N_C)^{-1}. \tag{2}$$

A. Background on Discrete-Time Safety Filters

Consider a discrete-time dynamical system with the control-affine form

$$x_{k+1} = f(x_k) + g(x_k)u_k. \tag{3}$$

Here, $x_k \in \mathbb{R}^n$ denotes the state at step k, $u \in \mathbb{R}^m$ denotes the control input at step k, and the functions $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ are locally Lipschitz in their arguments. Without loss of generality, we assume an initial step value of $k = 0$.

The state space $\mathbb{R}^n$ is divided into a safe set $S_h \subset \mathbb{R}^n$ and an unsafe set $\overline{S}_h = \mathbb{R}^n \backslash S_h$. The set S_h is defined as the superlevel sets of a function $h : \mathbb{R}^n \to \mathbb{R}$ as follows:

$$S_h = \{x \in \mathbb{R}^n : h(x) \geq 0\}, \tag{4a}$$
$$\text{int}(S_h) = \{x \in \mathbb{R}^n : h(x) > 0\}, \tag{4b}$$
$$\partial S_h = \{x \in \mathbb{R}^n : h(x) = 0\}. \tag{4c}$$

The objective is to render the set S_h forward invariant with respect to the dynamics (3), which is defined as follows:

Definition 3. A set $\mathcal{C} \subset \mathbb{R}^n$ is forward invariant with respect to the dynamics (3) if $x_k \in \mathcal{C}$ for all $k \geq 0$. The set $\mathcal{C}$ is called *safe* if it is both forward invariant and $\mathcal{C} \subseteq S_h$.

Let $\mathcal{D}_h \subseteq \mathbb{R}^n$ be a domain such that $S_h \subset \text{int}(\mathcal{D}_h)$. It has been shown in prior literature that satisfaction of

the following condition for all $x_k \in \mathcal{D}_h$ is sufficient to ensure forward invariance of the set S_h [8]:

$$\sup_u [h(x_{k+1}(x_k, u)) - h(x_k)] \geq -\gamma h(x_k), \qquad (5)$$

$$\iff \sup_u [h(f(x_k) + g(x_k)u)] \geq h(x_k) - \gamma h(x),$$

where $0 < \gamma \leq 1$.

Definition 4. A function $h : \mathbb{R}^n \to \mathbb{R}$ is called a discrete-time exponential control barrier function for the system (3) if the condition (5) holds for all $x \in \mathcal{D}_h$.

We define the set

$$K_h(x_k) =$$
$$\{u \in \mathbb{R}^m : h(f(x_k) + g(x_k)u) \geq h(x_k) - \alpha(h(x))\}. \qquad (6)$$

Theorem 3 ([8])**.** *Suppose that* (5) *holds for all* $x_k \in \mathcal{D}_h$ *and* $h(x_0) \geq 0$. *Then any control law* $u_k(\cdot)$ *such that* $u_k(x_k) \in K_h(x_k)$ *renders the set* S_h *forward invariant.*

In words, K_h is the set of control inputs that render the set S_h forward invariant. A control input $u_k \in K_h(x_k)$ is called *safe*. It is also straightforward to show that control inputs K_h also render the set S exponentially stable in $\mathcal{D}_h$.

Suppose a nominal feedback control law $u_{\text{nom}}(\cdot)$ is given for system (3) above. Suppose that several discrete-time CBFs $\{h_i(x_k)\}_{i=1,\dots,M}$ are given. If the condition (5) holds for all h_i, a safe control input $u^*(x_k) \in K_h(x_k)$ that minimally modifies u_{nom} in the sense of the Euclidean norm can be computed via the following (possibly nonconvex) optimization problem:

$$u^*(x_k) = \arg\min_{u \in \mathbb{R}^m} \|u - u_{\text{nom}}(x_k)\|_2 \qquad (7a)$$

$$\text{s.t. } h_i(f(x_k) + g(x_k)u) \geq h_i(x_k) - \gamma_i h_i(x_k) \quad (7b)$$

$$\forall i \in \{1, \dots, M\}. \qquad (7c)$$

Several limitations exist to this notion of discrete-time control barrier function and safety filtering as presented above. First, observe that the constraints in (7b) are convex if and only if each $h_i(\cdot)$ is concave in its argument.[1] If any $h_i(\cdot)$ is nonconcave, then the entire optimization problem (7) is nonconvex. This can inhibit real-time computation of the control input $u^*(x_k)$ on control systems.

In addition, the formulation (7) is unable to represent more complex semidefinite matrix constraints, including constraints on matrix eigenvalues and safe sets in the form of spectrahedra. The recent work [14] introduces the notion of *matrix control barrier functions* to address

---

[1]Recall that the composition of a function concave (convex) in its argument $h(\cdot)$ with an affine mapping $u \mapsto f(x_k) + g(x_k)u$ is likewise concave (convex) [18, Sec. 3.2.2].

this gap. More specifically, given a matrix valued function $H : \mathbb{R}^n \to \mathbb{S}^p$, they consider safe sets of the form

$$S = \{x \in \mathbb{R}^n : H(x) \succeq 0\}. \qquad (8)$$

However, [14] considers only continuous-time dynamical systems. Conditions to render sets of the form S forward invariant for discrete-time systems have not been considered in the literature.

This paper presents novel methods to address both of these gaps. Our methods are applicable to matrix-valued safe sets of the form (8), of which the scalar-valued safe sets of the form (4) are a special case. The problem addressed by this paper is as follows:

Problem II.1. Given the discrete-time dynamics (3), derive a method to compute control inputs u_k rendering sets of the form in (8) forward invariant using solely convex optimization methods.

III. MAIN RESULTS

A. Discrete-Time Exponential Matrix Control Barrier Functions

We consider safe sets S of the form (8). The following result presents a discrete-time condition under which the set S is rendered forward invariant.

Lemma III.1. Consider the dynamics (3) and the set S in (8). Let $\mathcal{D} \supset S$ be an open superset of S. The set S is forward invariant if there exists a constant $0 < \gamma \leq 1$ such that the following condition holds for all $x \in \mathcal{D}$:

$$H(x_{k+1}) - H(x_k) \succeq -\gamma H(x_k). \qquad (9)$$

Proof. We proceed by induction. By the definition of forward invariance, assume that $x_0 \in S$. This implies that $H(x_0) \succeq 0$. Observe that (9) can be rewritten as

$$H(x_{k+1}) \succeq (1 - \gamma)H(x_k). \qquad (10)$$

Since $0 < \gamma \leq 1$, the matrix $(1 - \gamma)H(x_0)$ is positive semidefinite. Equation (10) implies that $H(x_1)$ is therefore positive semidefinite. This can be seen by noting from (10) that $H(x_1) - (1 - \gamma)H(x_0) \succeq 0 \implies y^\intercal(H(x_1) - (1 - \gamma)H(x_0))y \geq 0$ for all $y \in \mathbb{R}^p$, implying that $y^\intercal H(x_1)y \geq y^T(1 - \gamma)H(x_0)y \geq 0$ since $(1 - \gamma)H(x_0)$ is positive semidefinite. It follows that $H(x_1) \succeq 0$. Now assume that $H(x_k)$ is positive semidefinite for any $k \geq 0$. This implies that $(1 - \gamma)H(x_k)$ is also positive semidefinite. Similar arguments can be used to conclude that $H(x_{k+1}) \succeq 0$, which concludes the proof. $\square$

Definition 5. A function $H : \mathbb{R}^n \to \mathbb{S}^p$ is called an discrete-time exponential matrix control barrier function

(DTE-MCBF) for the system (3) if there exists a constant $0 < \gamma \leq 1$ and a domain $\mathcal{D} \supset S$ such that for all $x \in \mathcal{D}$ there exists a $u \in \mathbb{R}^m$ satisfying

$$H(f(x) + g(x)u) \succeq (1 - \gamma)H(x) \qquad (11)$$

We define the set of safety-preserving control inputs $K_H : \mathbb{R}^n \to 2^{\mathbb{R}^m}$ as

$$K_H(x) \triangleq \{u \in \mathbb{R}^m : H(f(x) + g(x)u) \succeq (1-\gamma)H(x)\}. \qquad (12)$$

Let the eigenvalues of $H(x_k)$ be ordered as $\lambda_1(x_k) \leq \lambda_2(x_k) \leq \cdots \leq \lambda_p(x_k)$. The following theorem demonstrates that DTE-MCBFs are zeroing CBFs in the sense that all negative eigenvalues converge exponentially to zero under under any control law $u(x_k) \in K_H(x_k)$.

Theorem 4. *Let H be a DTE-MCBF for the system (3) with constant $0 < \gamma \leq 1$. For brevity, we denote $H_k \triangleq H(x_k)$. Let the eigenvalues of H_k be ordered as $\lambda_1(H_k) \leq \lambda_2(H_k) \leq \cdots \leq \lambda_p(H_k)$. Suppose that $u_k \in K_H(x_k)$ for all $k \geq 0$. Then the following holds:*

$$\lambda_1(H_k) \geq (1 - \gamma)^k \lambda_1(H_0). \qquad (13)$$

Proof. Define $\Delta_\gamma H_k \triangleq H_{k+1} - (1 - \gamma)H_k = H(f(x_k) + g(x_k)u_k) - (1 - \gamma)H(x_k)$. By Definition 5, the control input $u_k \in K_H(x_k) \ \forall k \geq 0$ implies that $\Delta_\gamma H \succeq 0$.

Observe that $H_{k+1} = (1-\gamma)H_k + \Delta_\gamma H_k$. By Weyl's Inequality (Theorem 1), it follows that

$$\lambda_1((1-\gamma)H_k) + \lambda_1(\Delta_\gamma H_k) \leq \lambda_1\left((1-\gamma)H_k + \Delta_\gamma H_k\right),$$
$$= \lambda_1(H_{k+1}). \qquad (14)$$

However, note that $\lambda_1(\Delta_\gamma H_k) \geq 0$ since $\Delta_\gamma H \succeq 0$ by the definition of K_H in (12). This implies that $\lambda_1((1-\gamma)H_k) \leq \lambda_1((1-\gamma)H_k) + \lambda_1(\Delta_\gamma H_k)$. It follows from (14) that

$$\lambda_1(H_{k+1}) \geq \lambda_1((1-\gamma)H_k) \qquad (15)$$
$$= (1-\gamma)\lambda_1(H_k) \ \forall k \geq 0. \qquad (16)$$

The result follows by repeated application of (16):

$$\lambda_1(H_k) \geq (1-\gamma)\lambda_1(H_{k-1}) \geq \cdots \geq (1-\gamma)^k \lambda_1(H_0).$$
$\square$

In particular, Theorem 4 implies that if $x_0 \notin S$, the state x_k will converge exponentially to the safe set S under a control law $u(x_k) \in K_H(x_k)$.

If H is a DTE-MCBF and a nominal feedback control law $u_{\text{nom}}(x_k)$ is given, a safe control input $u^*(x_k) \in K_H(x_k)$ can be computed via the following optimization problem:

$$u^*(x) = \underset{u \in \mathbb{R}^m}{\arg\min} \|u - u_{\text{nom}}(x)\|_2 \qquad (17a)$$
$$\text{s.t.} \quad H(f(x) + g(x)u) \succeq (1 - \gamma)H(x) \qquad (17b)$$

However, this optimization problem may not be convex in general due to the constraint (17b). In the next section we address this issue.

B. Computing Safe Inputs Using Convex Projection

We begin by considering the following two classes of safe set functions H. The definition of matrix concavity is given in the Appendix (Section VI).

- Safe sets that are matrix concave in their argument
- Safe sets that are not matrix concave in their argument.

For safe sets that are matrix concave in their argument, observe that the constraint (17b) is matrix convex in the optimization variable u. This follows because the composition of a convex function with an affine function is also convex. The optimization problem (17) is therefore convex and can be solved with standard conic optimization solvers.

We turn our attention to the more difficult case of safe sets H that are *not* matrix concave in their arguments. Recall that H defines the safe set S via its superlevel sets. The following assumption is made on the unsafe set $\overline{S}$:

Assumption 1. Given a safe set S defined by the sublevel sets of H, the closure of the unsafe set $\text{cl}(\overline{S}) = \text{cl}(\mathbb{R}^n \backslash S)$ is a subset of a finite number of closed, convex sets. More precisely, $\text{cl}(\overline{S}) \subset \bigcup_{i=1}^{Q} \overline{S}_i$. In addition, for each $\overline{S}_i$ there exists a matrix convex function $\overline{H}_i : \mathbb{R}^n \to \mathbb{S}$ such that $\overline{S}_i = \{x \in \mathbb{R}^n : \overline{H}_i \preceq 0\}$.

In the trivial case any nonconvex unsafe set $\overline{S}$ is a subset of its convex hull; i.e., $\text{cl}(\overline{S}) \subset \text{co}(\overline{S})$. As this can be overly conservative, however, Assumption 1 also includes cases where the unsafe set can be decomposed into a finite number of closed, convex sets.

Since the set S is nonconvex and therefore the function H is not matrix concave, the optimization constraint (17b) is nonconvex. However, we can instead iteratively consider a *convex subset* of S based on the current state x_k. Our proposed method is as follows: At each timestep k such that $x_k \notin \overline{S}_i$, observe that the sets $\{x_k\}$ and $\overline{S}_i$ are both nonempty, disjoint, closed convex sets, with the set $\{x_k\}$ being trivially compact. By the separating hyperplane theorem [18, Sec. 2.5.1], there exists a hyperplane defined by $a_i : \mathbb{R}^n \to \mathbb{R}^n, b_i : \mathbb{R}^n \to \mathbb{R}$ such that $(a_i(x_k))^\mathsf{T} x_k - b_i(x_k) > 0$ and $(a_i(x_k))^\mathsf{T} z < b_i(x_k)$ for all $z \in \overline{S}_i$.

The form of a_i, b_i is determined by the vector $x_k - P_{\overline{S}_i} x_k$, where $P_{\overline{S}_i} x_k$ is the projection of x_k onto the set $\overline{S}_i$. This projected point can be computed via the following convex optimization problem:

$$P_{\overline{S}_i}(x_k) = \underset{z \in \mathbb{R}^n}{\arg\min} \quad \|z - x_k\| \qquad (18a)$$
$$\text{s.t.} \quad \overline{H}_i(z) \preceq 0. \qquad (18b)$$

Once this projected point has been computed, the supporting hyperplane parameters $a_i(x_k), b_i(x_k)$ can be explicitly defined as

$$a_i(x_k) \triangleq \frac{x_k - P_{\overline{S}_i}(x_k)}{\left\| x_k - P_{\overline{S}_i}(x_k) \right\|_2}, \tag{19a}$$

$$b_i^\epsilon(x_k) \triangleq \epsilon + \left(\frac{x_k - P_{\overline{S}_i}(x_k)}{\left\| x_k - P_{\overline{S}_i}(x_k) \right\|_2} \right)^{\mathsf{T}} P_{\overline{S}_i}(x_k), \tag{19b}$$

for some $\epsilon > 0$.

Lemma III.2. Let a_i, b_i^ϵ be defined as in (19) for some $\epsilon > 0$. Suppose that $x_k \notin \overline{S}_i$. Then the following holds:

$$a_i(x_k)^{\mathsf{T}} z < b_i(x_k) \ \forall z \in \overline{S}_i. \tag{20}$$

Proof. We first demonstrate that the vector $x_k - P_{\overline{S}_i}(x_k)$ is in the normal cone $N_{\overline{S}_i}(P_{\overline{S}_i}(x_k))$. Using Theorem 2, observe that

$$x_k - P_{\overline{S}_i}(x_k) \in P_{\overline{S}_i}^{-1}\left(P_{\overline{S}_i}(x_k)\right) - I(P_{\overline{S}_i}(x_k)) \tag{21}$$

$$= N_{\overline{S}_i}\left(P_{\overline{S}_i}(x_k)\right) \tag{22}$$

Since $\left\| x_k - P_{\overline{S}_i}(x_k) \right\|_2 > 0$ it follows that $\frac{x_k - P_{\overline{S}_i}(x_k)}{\left\| x_k - P_{\overline{S}_i}(x_k) \right\|_2} \in N_{\overline{S}_i}(P_{\overline{S}_i}(x_k))$ Next, by Definition 2 it holds that $\left(\frac{x_k - P_{\overline{S}_i}(x_k)}{\left\| x_k - P_{\overline{S}_i}(x_k) \right\|_2} \right)^{\mathsf{T}} \left(z - P_{\overline{S}_i}(x_k) \right) \leq 0 < \epsilon \ \forall z \in \overline{S}_i$. Rearranging yields

$$\left(\frac{x_k - P_{\overline{S}_i}(x_k)}{\left\| x_k - P_{\overline{S}_i}(x_k) \right\|_2} \right)^{\mathsf{T}} z <$$
$$\epsilon - \left(\frac{x_k - P_{\overline{S}_i}(x_k)}{\left\| x_k - P_{\overline{S}_i}(x_k) \right\|_2} \right)^{\mathsf{T}} P_{\overline{S}_i}(x_k) \ \forall z \in \overline{S}_i$$

which by (19) is equivalent to $a_i(x_k)^{\mathsf{T}} z < b_i^\epsilon(x_k)$ for all $z \in \overline{S}_i$. □

Lemma III.2 implies that $\overline{S}_i$ is a subset of the interior of the halfspace defined by $\{y \in \mathbb{R}^n : a_i(x_k)^{\mathsf{T}} y < b_i^\epsilon(x_k)\}$, $y \in \mathbb{R}^m$ for all values of $\epsilon > 0$. On the other hand, by definition observe that the condition $a_i(x_k)^{\mathsf{T}} z - b_i^\epsilon \geq 0$ is equivalent to

$$\left(\frac{x - P_{\overline{S}_i}(x_k)}{\left\| x - P_{\overline{S}_i}(x_k) \right\|} \right)^{\mathsf{T}} \left(z - P_{\overline{S}_i}(x_k) \right) \geq \epsilon. \tag{23}$$

When $x_k \notin \overline{S}_i$, the parameter ϵ ensures that the equations (19) are well-defined by controlling the distance between the hyperplane $a_i(x_k)^{\mathsf{T}} y = b_i^\epsilon(x_k)$ and the boundary of the set $\overline{S}_i$. A positive ϵ adds a buffer between the boundary and the hyperplane; thus, the parameter ϵ is called the *buffer distance*.

Now consider the following revised safe set functions:

$$h_i^\epsilon(z, x_k) \triangleq a_i(x_k)^{\mathsf{T}} z - b_i^\epsilon(x_k), \ \forall i = 1, \ldots, Q, \tag{24}$$

$$H^\epsilon(z, x_k) \triangleq \begin{bmatrix} h_1^\epsilon(z, x_k) & & 0 \\ & \ddots & \\ 0 & & h_Q^\epsilon(z, x_k) \end{bmatrix} \tag{25}$$

Each h_i^ϵ is linear in z, which implies H^ϵ is therefore matrix concave in z. In particular, the constraint $H^\epsilon(z, x_k) \succeq 0$ enforces the inequality $\min_i \left(\{h_i^\epsilon(z, x_k)\}_{i=1}^Q \right) \geq 0$.

Lemma III.3. Let $x_k \notin \overline{S}_i$ for all $i = 1, \ldots, Q$. Then the set $\{z \in \mathbb{R}^n : H^\epsilon(z, x_k) \succeq 0\}$ is disjoint from $\overline{S}_i$ for all $i = 1, \ldots, Q$.

Proof. The result follows from Lemma III.2 and equations (24)-(25). □

By Lemma III.3, the set $\{z \in \mathbb{R}^n : H^\epsilon(z, x_k) \succeq 0\}$ is a convex subset of the safe set S. However, observe that each h_i^ϵ and H^ϵ is a two-argument function where the second argument determines the subset of the safe set being considered and the first argument is the point whose safety is being determined. This two-argument form motivates a new class of safe set functions and control barrier functions which are outlined in the next two definitions.

Definition 6. Let $S \subset \mathbb{R}^n$. A function $H : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{S}^p$ is called a *weak safe subset function* for S if there exists a domain $\mathcal{D} \supset S$ such that the following holds for all $x \in \mathcal{D}$:

$$\{z \in \mathbb{R}^n : H(z, x) \succeq 0\} \subseteq S. \tag{26}$$

A function H is called a *safe subset function* if it is a weak safe subset function and $H(x, x) \succeq 0$ for all $x \in S$.

Definition 7. A safe subset function $H : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{S}^p$ is called a Subset-based Discrete-Time Exponential Matrix Control Barrier Function (SDTE-MCBF) for the system (3) if there exists a constant $0 < \gamma \leq 1$ and a domain $\mathcal{D} \supset S$ such that for all $x \in \mathcal{D}$ there exists a control input $u \in \mathbb{R}^m$ satisfying

$$H(f(x) + g(x)u, x) \succeq (1 - \gamma)H(x, x). \tag{27}$$

When the safe subset described by H is obtained through projection-based methods (as described previously in this section), we will refer to H as a *Projection-based Discrete-Time Exponential Matrix Control Barrier Function* (PDTE-MCBF). A PDTE-MCBF is a special case of SDTE-CBFs since it is possible that a safe subset could be obtained by means other than projections. However, we leave further investigation of this distinction for future work.

Given a safe subset function H, consider the following set of control inputs:

$$K_H(x_k) \triangleq \tag{28}$$
$$\{u \in \mathbb{R}^m : H(x_{k+1}, x_k) \succeq (1 - \gamma)H(x_k, x_k)\}$$
$$= \{u \in \mathbb{R}^m : H(f(x_k) + g(x_k)u, x_k) \succeq (1 - \gamma)H^\epsilon(x_k, x_k)\} \tag{29}$$

The following theorem demonstrates that $K_H(x_k)$ represents the set of control inputs u that render the set S forward invariant.

Theorem 5. *Let H be an SDTE-MCBF on a domain $\mathcal{D} \supset S$ for the system (3). Then any control law $u(x_k)$ such that $u(x_k) \in K_H(x_k)$ for all $k \geq 0$ renders the safe set S forward invariant.*

Proof. Suppose $x_0 \in S$. The fact that H is a SDTE-MCBF implies that it is a safe subset function by Definition 7, which implies $H(x_0, x_0) \succeq 0$ by Definition 6. The fact that $u(x_0) \in K_H(x_0)$ implies that $H(f(x_0) + g(x_0)u(x_0)) \succeq (1 - \gamma)H(x_0, x_0) \succeq 0$. It follows that $x_1 \in S$. Proceeding inductively, assume that $x_k \in S$. By similar arguments it follows that $x_{k+1} \in S$ and $H(f(x_k) + g(x_k)u(x_k), x_k) \succeq (1-\gamma)H(x_k, x_k) \succeq 0$, which concludes the proof. $\square$

Given these definitions, we now formally prove that the function H^ϵ can be classified as a safe subset function.

Lemma III.4. *Let S be nonempty, and suppose $\overline{S}$ satisfies Assumption 1. Let H^ϵ be defined as in (25). Choose $\mathcal{D} = S$ and define $S^\epsilon = \{z \in S : \mathrm{dist}(x, \overline{S}) \geq \epsilon\} \subset \mathcal{D}$, where the distance is defined in terms of the Euclidean norm. Then H^ϵ is a safe subset function for S^ϵ.*

Proof. By the properties of convex sets and Lemma III.2, $a_i(x)^\intercal z - b_i^\epsilon(x) \geq 0 \implies \left(\frac{x - P_{\overline{S}_i}(x)}{\|x - P_{\overline{S}_i}(x)\|}\right)^\intercal \left(z - P_{\overline{S}_i}(x)\right) \geq \epsilon$ implies that $\mathrm{dist}(z, \overline{S}_i) \geq \epsilon$ for all $i = 1, \ldots, Q$. By equation (25), it follows that all points in the set $\{z \in \mathbb{R}^n : H^\epsilon(z, x) \succeq 0\}$ satisfy $\mathrm{dist}(z, \overline{S}) \geq \epsilon$, and therefore $\{z \in \mathbb{R}^n : H^\epsilon(z, x) \succeq 0\} \subset S^\epsilon$. The function H^ϵ is therefore a weak safe set function.

Next, choose any $x \in S^\epsilon$. It holds that $\|x - P_{\overline{S}_i}(x)\|_2 \geq \epsilon$, which implies that $a_i(x)^\intercal x - b_i^\epsilon(x) \geq 0$ for all $i = 1, \ldots, Q$. It follows that $H^\epsilon(x, x) \succeq 0$ for all $x \in S^\epsilon$. $\square$

Since the definition of SDTE-MCBF relies upon the specific dynamics under consideration, proofs that H^ϵ form a SDTE-MCBF will require case-by-case analysis. Note that any H^ϵ as defined in (25) that qualifies as a SDTE-MCBF is also a PDTE-MCBF since the subset of the safe set is obtained via projections. Under the

assumption that H^ϵ is an SDTE-MCBF (or PDTE-MCBF) and given a nominal control input $u_{\mathrm{nom}}(x_k)$ and the projected points $\{P_{\overline{S}_i}(x_k)\}_{i=1}^Q$, it is possible to compute a safety-preserving control input $u^*(x_k) \in K_H^\epsilon(x_k)$ using the following convex QP:

$$u^*(x_k) = \arg\min_u \|u - u_{\mathrm{nom}}\| \tag{30a}$$
$$\text{s.t. } H^\epsilon(f(x_k) + g(x_k)u, x_k) \succeq (1 - \gamma)H^\epsilon(x_k, x_k) \tag{30b}$$

Algorithm 1: Safe Input Computation for PDTE-MCBF

---

1 **Inputs:** State x_k, nominal control input $u_{\mathrm{nom}}(x_k)$, parameter $\epsilon > 0$
2 **Output:** Safety-preserving control input $u^*(x_k)$
3 **for** $i = 1, \ldots, Q$ **do**
4   $\quad P_{\overline{S}_i} \leftarrow \arg\min_{z \in \mathbb{R}^n} \|z - x_k\|$ s.t. $\overline{H}_i(x) \preceq 0$
5   $\quad\quad$ (See Eqs. (18) or (31))
6 $u^*(x_k) \leftarrow$
  $\quad \arg\min_{u \in \mathbb{R}^m} \|u - u_{\mathrm{nom}}\|$ s.t. $H^\epsilon(f(x_k) + g(x_k)u, x_k) \succeq 0$
7   $\quad$ (See Eqs. (24), (25), (30))
8 **return** $u^*(x_k)$

Algorithm 1 summarizes the process of computing safe control inputs $u^*(x_k)$ for the form of PDTE-MCBFs defined in this paper. For simplicity, algorithm 1 shows each projection point $P_{\overline{S}_i}(x_k)$ being computed by a separate optimization problem instance. However, we note that all projection points $\vec{P} \triangleq \begin{bmatrix} P_{\overline{S}_1} & \cdots & P_{\overline{S}_Q} \end{bmatrix}^\intercal$ can be computed simultaneously in a single optimization instance as follows:

$$\vec{P} = \arg\min_{\vec{z} \triangleq [z_1, \cdots, z_Q]^\intercal} \sum_{i=1}^Q \|z_i - x_k\| \tag{31a}$$
$$\text{s.t. } \overline{H}_i(z_i) \preceq 0 \ \forall i = 1, \ldots, Q. \tag{31b}$$

We point out that our proposed method only requires solving two sequential convex optimization problems per time step k, the second of which is a QP. In special cases where $\overline{S}_i$ takes a specific form such as a circle or ellipse, it is possible to obtain the projection point directly in closed-form. Compared to prior nonconvex formulations for discrete-time CBFs, our convex optimization method has polynomial-time computational complexity and guarantees convergence to globally optimal solutions.

C. Zeroing Properties of the Projection Method

The original notion of discrete-time exponential control barrier functions renders safe sets exponentially stable in the state space [8]. In other words, if the

initial state x_0 is in the unsafe set $\overline{S}$, a safe control input derived from a discrete-time exponential CBF causes x_k to converge exponentially to the safe set S. The convex projection method under Assumption 1 in general operates only when the state x_k does not coincide with any of the projections $P_{\overline{S}_i}(x_k)$ onto the unsafe sets $\overline{S}_i$, $i = 1, \ldots, Q$. If $x_k \in \overline{S}_i$, then $P_{\overline{S}_i}(x_k) = x_k$ and the prior formulas for a_i, b_i result in ill-defined constraints. Deriving methods to ensure $x_k \in \overline{S}$ converges exponentially to S is more difficult in general for the projection method proposed previously; however this section presents methods to render S asymptotically stable when $x_k \in \overline{S}$.

If $x_k \in \partial \overline{S}_i$, we may use the normal cone $N_{\overline{S}_i}(x_k)$ to compute a safe control input. Choose any $x_N(x_k) \in N_{\overline{S}_i}(x_k)$ such that $x_N(x_k) \neq x_k$ and $x_N(x_k) - x_k \in N_{\overline{S}_i}(x_k)$. Define the functions $\widehat{a}_i^\partial : \mathbb{R}^n \to \mathbb{R}^n, \widehat{b}_i^{\partial,\epsilon} : \mathbb{R}^n \to \mathbb{R}$ as

$$\widehat{a}_i^\partial(x_k) = \frac{x_N(x_k) - x_k}{\|x_N(x_k) - x_k\|}, \tag{32a}$$

$$\widehat{b}_i^{\partial,\epsilon}(x_k) = \epsilon + \frac{(x_N(x_k) - x_k)}{\|x_N(x_k) - x_k\|}^\mathsf{T} x_k. \tag{32b}$$

We can then define the safe set function

$$\widehat{h}_i^{\partial,\epsilon}(z, x_k) \triangleq \left(\widehat{a}_i^\partial(x_k)\right)^\mathsf{T} z - \widehat{b}_i^{\partial,\epsilon}(x_k). \tag{33}$$

Lemma III.5. Let $x_k \in \partial \overline{S}_i$ for all $i = 1, \ldots, Q$. Let $\epsilon > 0$. Then the set $\{z \in \mathbb{R}^n : h_i^{\partial,\epsilon}(z, x_k) \geq 0\}$ is disjoint from $\overline{S}_i$ for all $i = 1, \ldots, Q$.

Proof. By definition, $x_N(x_k) - x_k \in N_{\overline{S}_i}(x_k)$. The result follows using similar arguments as Lemma III.2 and Lemma III.3 $\qquad\square$

The safe set function $\widehat{h}_i^{\partial,\epsilon}$ may be used in place of h_i^ϵ in Algorithm 1 when $x_k \in \partial \overline{S}_i$.

The more difficult case is when the state is in the interior of the unsafe set; i.e., $x_k \in \text{int}\left(\overline{S}_i\right)$. In this case the normal cone is trivially the zero vector: $N_{\overline{S}_i}(z) = \{0\} \ \forall z \in \overline{S}_i$. We therefore use the notion of *depth* of a convex set. The depth function depth $: \mathbb{R}^n \times 2^{\mathbb{R}^n} \to \mathbb{R}$ is defined as

$$\text{depth}(x, C) \triangleq \text{dist}(x, \mathbb{R}^n \backslash S), \tag{34}$$

where C is a subset of $\mathbb{R}^n$. For convex sets such as $\overline{S}_i$ under Assumption 1, the depth can be computed as [18, Sec. 8.5.1]

$$\text{depth}(x_k, \overline{S}_i) = \max_R \quad R \tag{35}$$
$$\text{s.t.} \quad \mathfrak{g}_j(x, R) \leq 0 \ j = 1, \ldots, J',$$

This optimization problem is convex, but the functions $\mathfrak{g}_j$ are difficult to evaluate in general. However, the optimization problem in (35) is tractable for specific classes of convex functions. When $\overline{S}_i$ is a polytope

in halfspace form $a_{i,j}'^\mathsf{T} x_k \leq b_{i,j}'$, depth$(x_k, \overline{S}_i)$ can be computed via the following linear program [18, Sec. 8.5.1]:

$$\text{depth}(x_k, \overline{S}_i) = \max_R \quad R \tag{36}$$
$$\text{s.t.} \quad a_{i,j}'^\mathsf{T} x_k + R\left\|a_{i,j}'\right\| \leq b_{i,j} \tag{37}$$

When $\overline{S}_i$ takes the form of an intersection of ellipsoids defined by quadratic inequalities $\overline{S}_i = \{x : x^\mathsf{T} M_{i,j} x + 2v_{i,j}^\mathsf{T} x + d_{i,j} \leq 0, \ j = 1, \ldots, J', \ M_{i,j} \in \mathbb{R}^{n\times n}, v_{i,j} \in \mathbb{R}^n, d_{i,j} \in \mathbb{R}$, the depth can be computed via the following convex SDP [18, Sec. 8.5.1]:

$$\text{depth}(x_k, \overline{S}_i) =$$
$$\max_{R \in \mathbb{R}, \ \lambda_{i,j} \in \mathbb{R}} R$$
$$\text{s.t.} \begin{bmatrix} -\lambda_{i,j} - d_{i,j} + v_{i,j}^\mathsf{T} M_{i,j}^{-1} v_{i,j} & 0 & (x + M_{i,j}^{-1} v_{i,j})^\mathsf{T} \\ 0 & \lambda_{i,j} I & RI \\ x_k + M_{i,j}^{-1} v_{i,j} & RI & M_i^{-1} \end{bmatrix} \succeq 0,$$
$$j = 1, \ldots, J'$$

Due to the nature of the objective function (35), the value of depth$(x_k, \overline{S}_i)$ is equal to the value of the optimal point $R^*(x_k)$. Using the implicit function theorem, it is possible to obtain a descent direction for the depth by computing $\frac{\partial}{\partial x}\text{depth}(x_k, \overline{S}_i) = \frac{\partial R^*(x_k)}{\partial x}$ using standard methods to backpropagate through convex optimization programs [19]. The following halfspaces can then be defined:

$$a_i^R(x_k) \triangleq \frac{-\partial R_i^*(x_k)/\partial x}{\|\partial R_i^*(x_k)/\partial x\|}, \tag{38a}$$

$$b_i^R(x_k) \triangleq \left(\frac{-\partial R_i^*(x_k)/\partial x}{\|\partial R_i^*(x_k)/\partial x\|}\right)^\mathsf{T} x_k - R_i^*(x_k) \tag{38b}$$

We can then define the function

$$h_i^R(z, x_k) = a_i^R(x_k)^\mathsf{T} z - b_i^R(x_k), \tag{39}$$

which may be used in place of h_i^ϵ in Algorithm 1 when $x_k \in \text{int}(\overline{S}_i)$.

Lemma III.6. Let $\overline{S}_i$ be defined as in Assumption 1. Let h_i^R be defined as in (39). Let $x_k \in \text{int}(\overline{S}_i)$ and let $z \in \mathbb{R}^n$ be any vector z satisfying $h_i^R(z, x_k) \geq (1 - \gamma)h_i^R(x_k, x_k)$ for some $0 < \gamma \leq 1$. Then depth$(z, \overline{S}_i) < $ depth$(x_k, \overline{S}_i)$.

Proof. By definition, $R_i^*(x_k) = \text{depth}(x_k, \overline{S}_i)$. The vector $\frac{-\partial R_i^*(x_k)/\partial x}{\|\partial R_i^*(x_k)/\partial x\|}$ can be considered an element of the normal cone to the convex set $C_i(x_k) = \{y \in \mathbb{R}^n : R_i^*(x_k) \leq R_i^*(y)\}$, where $x_k \in \partial C_i(x_k)$. Observe that $h_i^R(x_k, x_k) = R_i^*(x_k)$. The equation $h_i^R(z, x_k) \geq h_i^R(x_k, x_k)$ can therefore be rewritten as

$$\frac{-\partial R_i^*(x_k)/\partial x}{\|\partial R_i^*(x_k)/\partial x\|}^\mathsf{T} (z - x_k) \geq (1 - \gamma)R_i^*(x_k). \tag{40}$$

Since $R_i^*(x_k) > 0$ for $x_k \in \overline{S}_i$ and $0 < \gamma \leq 1$, we therefore have $R_i^*(z) < R_i^*(x_k)$, or equivalently $\text{depth}(z, \overline{S}_i) < \text{depth}(x_k, \overline{S}_i)$. $\quad\square$

Finally, observe that $h_i^R(f(x) + g(x)u, x_k)$ is affine and therefore concave in u, which empowers rapid computation of feasible control inputs u using convex optimization techniques.

D. Indefinite Matrix Safe Sets

The recent work [14] introduced the notion of *indefinite safe sets* defined as

$$S = \{x \in \mathbb{R}^n : H(x) \not\prec 0\}, \quad (41)$$

$$= \{x \in \mathbb{R}^n : \lambda_p(H(x)) \geq 0\}. \quad (42)$$

Here, recall that by convention the eigenvalues of H are ordered $\lambda_1(H(x)) \leq \lambda_2(H(x)) \leq \cdots \leq \lambda_p(H(x))$. Indefinite safe sets have application to enforcing disjunctive boolean constraints on multiple CBFs [14]. We generalize this notion of indefinite safe sets as follows:

$$S^{(j)} = \{x \in \mathbb{R}^n : \lambda_j(H(x)) \geq 0\}, \ 1 \leq j \leq p. \quad (43)$$

In words, $S^{(j)}$ represents the states where eigenvalues j through p are non-negative. If we let $r = p - j$, then this represents enforcing r out of p constraints to be active at any time. We next show that the set $S^{(j)}$ may be rendered forward invariant by enforcing the following constraint at every time step $k \geq 0$ for some $0 < \gamma \leq 1$:

$$H(x_{k+1}) - H(x_k) \succeq -\gamma \lambda_j(H(x_k)) I_{p \times p}. \quad (44)$$

Given the dynamics (3), define the set-valued mapping $K_H^{(j)} : \mathbb{R}^n \to 2^{\mathbb{R}^m}$ as

$$K_H^{(j)}(x_k) \triangleq \{ u \in \mathbb{R}^m : \quad (45)$$
$$H(f(x) + g(x)u) \succeq H(x_k) - \gamma \lambda_j(H(x_k)) I_{p \times p} \}.$$

Theorem 6. *Let $S^{(j)}$ be defined as in (43) and let $K_H^{(j)}$ be defined as in (45). Let $x_0 \in S^{(j)}$. Then any control input $u(x_k) \in K_H^{(j)}(x_k) \ \forall k \geq 0$ renders the set $S^{(j)}$ forward invariant. Furthermore, such a $u(x_k)$ renders the set $S^{(j)}$ exponentially stable in $\mathbb{R}^n$.*

Proof. For brevity denote $H_k = H(x_k)$. Define $\Delta^{(j)} H_k \triangleq H_{k+1} - (H_k - \gamma \lambda_j(H_k) I_{p \times p})$. Using Weyl's inequality (Theorem 1) and setting $i = j$, we have

$$\lambda_1 \left(\Delta^{(j)} H_k \right) + \lambda_j \left(H_k - \gamma \lambda_j(H_k) I_{p \times p} \right) \leq$$
$$\lambda_j \left(\Delta^{(j)} H_k + H_k - \gamma \lambda_j(H_k) I_{p \times p} \right),$$
$$= \lambda_j \left(H_{k+1} \right).$$

However, $u(x_k) \in K_H^{(j)}(x_k)$ implies that $\lambda_1(\Delta^{(j)} H_k) \geq 0$. It follows that

$$\lambda_j \left(H_k - \gamma \lambda_j(H_k) I_{p \times p} \right) \leq \lambda_j(H_{k+1}),$$
$$\implies (1 - \gamma) \lambda_j(H_k) \leq \lambda_j(H_{k+1}).$$

By induction, we have

$$\lambda_j(H_{k+1}) \geq (1 - \lambda)^k \lambda_j(H_0). \quad (46)$$

It follows that $S^{(j)}$ is both forward invariant and exponentially stable in $\mathbb{R}^n$. $\quad\square$

The set of control inputs within $K_H^{(j)}(x_k)$ ensure that the largest eigenvalues λ_j through λ_p remain positive, but the smallest eigenvalues λ_1 through λ_{j-1} are free to remain negative. This behavior relates to recent work on combinatorial "p-choose-r" CBFs [15], however our contributions here apply to discrete-time CBFs. Given a nominal control input $u_{\text{nom}}(x_k)$, control inputs within $K_H^{(j)}(x_k)$ may be computed using the following optimization formulation:

$$u^{(j)*}(x_k) = \min_{u \in \mathbb{R}^m} \quad \|u - u_{\text{nom}}(x_k)\| \quad (47)$$

$$\text{s.t. } H(f(x_k) + g(x_k)u) - H(x_k) + \gamma \lambda_j(H(x_k)) I_{p \times p} \succeq 0.$$

The results in preceding sections may be applied to ensure that this optimization formulation is convex.

IV. NUMERICAL SIMULATIONS

In this section, the PDTE-MCBF technique developed in the previous section and shown in Algorithm 1 is implemented in numerical examples. This section also provides a performance comparison between the proposed PDTE-MCBF technique and nonconvex CBF formulations. Note that in this work, the numerical examples are constrained to a plane in which only horizontal and vertical motion is permitted and thus only two-dimensional obstacles are considered. Consequently, the subindices h and v used in the variables to denote a connection to horizontal and vertical coordinates, respectively. For instance, p_{h} and p_{v} are used in these examples to denotes positions in the horizontal and vertical directions, respectively.

First, an overview of the obstacles considered in the numerical examples is provided, in which the unsafe sets, matrix convex functions, the projected point calculations, and the safe sets associated with each obstacle are discussed; in particular, analogous parameters to the previously defined buffer distance ϵ are defined for the safe sets of each obstacle. Then, two systems are considered: a system in a plane composed of two double integrators and the outer-loop controller of a bicopter lateral flight system. The nominal controllers in the examples are given by reference tracking LQR controllers. The details of this implementation are omitted for brevity and since the main focus of the paper is the performance of the CBF. The implementation of the PDTE-MCBF technique requires the solution of a constrained linear least-squares optimization problem, as shown in (30). In all examples, this problem is solved by using the lsqlin solver from Matlab with the active-set

algorithm. The solver for nonconvex CBF formulations is discussed in each numerical example.

Furthermore, in this work, the safe sets for obstacle avoidance depend only on the position states. Hence, there exists $C_{\mathrm{pos}} \in \mathbb{R}^{2 \times n}$, such that $C_{\mathrm{pos}} x \in \mathbb{R}^2$ yields the position state vector related to state x. Thus, (19) is rewritten as

$$a_i(x_k) \triangleq \frac{C_{\mathrm{pos}}^{\mathrm{T}} C_{\mathrm{pos}}(x_k - P_{\overline{S}_i}(x_k))}{\left\| C_{\mathrm{pos}}(x_k - P_{\overline{S}_i}(x_k)) \right\|_2}, \tag{48a}$$

$$b_i^\epsilon(x_k) \triangleq \epsilon + \left(\frac{C_{\mathrm{pos}}(x_k - P_{\overline{S}_i}(x_k))}{\left\| C_{\mathrm{pos}}(x_k - P_{\overline{S}_i}(x_k)) \right\|_2} \right)^{\mathsf{T}} (C_{\mathrm{pos}} P_{\overline{S}_i}(x_k)). \tag{48b}$$

The matrix convex functions presented in the obstacle subsection are also defined in terms of the C_{pos} operator.

All computational results in this paper were obtained using a laptop PC running Windows 10 Pro, version 22H2, OS build 19045.6332, with an Intel Core i7-10750H processor running at 2.60 GHz and a 32GB, 3200 MHz RAM, with MATLAB version R2024b Update 5.

A. Obstacles

The four obstacles considered in this work are a circle, an ellipse, a convex polytope, and a spectrahedron, which are shown in Figure 2. Their properties and the projected point calculation procedure and the CBF associated with each of these obstacles are presented next.

1) Circle: A circle $\mathfrak{C}$ is defined by a center position $p_{\mathfrak{C}} \in \mathbb{R}^2$ and a radius $r_{\mathfrak{C}}$. For a circle $\mathfrak{C}$ obstacle, the unsafe set is given by

$$\overline{S}_{\mathfrak{C}} = \{p \in \mathbb{R}^2 \colon \|p - p_{\mathfrak{C}}\|_2 \leq r_{\mathfrak{C}}\}, \tag{49}$$

and thus, the corresponding matrix convex function describing the unsafe set is given by

$$\overline{H}_{\mathfrak{C}}(x) = \|C_{\mathrm{pos}} x - p_{\mathfrak{C}}\|_2 - r_{\mathfrak{C}}. \tag{50}$$

Note that the projection of a state vector x_k onto the set $\overline{S}_{\mathfrak{C}}$, given by (18) can be expressed in closed-form as

$$P_{\overline{S}_{\mathfrak{C}}}(x_k) = \frac{\min\{r_{\mathfrak{C}}, \|C_{\mathrm{pos}} x_k - p_{\mathfrak{C}}\|_2\}}{\|C_{\mathrm{pos}} x_k - p_{\mathfrak{C}}\|_2}(C_{\mathrm{pos}} x_k - p_{\mathfrak{C}}) + p_{\mathfrak{C}}, \tag{51}$$

which can be used to calculate the projected point without solving an optimization problem. Finally, it follows from the unsafe set (49) that the CBF associated with the circle obstacle safe set $S_{\mathfrak{C}}$ with buffer distance $\epsilon_{\mathfrak{C}}$ is given by

$$h_{\mathfrak{C}}(x_k) = (r_{\mathfrak{C}} + \epsilon_{\mathfrak{C}})^2 - (C_{\mathrm{pos}} x_k - p_{\mathfrak{C}})^{\mathrm{T}}(C_{\mathrm{pos}} x_k - p_{\mathfrak{C}}), \tag{52}$$

which can be used to implement the safe control input constraints shown in (7b). Note that $\epsilon_{\mathfrak{C}}$ increases the radius of the unsafe set considered by the CBF.

2) Ellipse: An ellipse $\mathfrak{El}$ is defined by a center position $p_{\mathfrak{El}} \in \mathbb{R}^2$, the semi-major axis unit vector $v_{\mathfrak{El}} \in \mathbb{R}^2$, such that $\|v_{\mathfrak{El}}\|_2 = 1$, and the semi-major and semi-minor axes lengths $\ell_{\mathfrak{El},1} \geq \ell_{\mathfrak{El},2} > 0$, respectively. Let $Q_{\mathfrak{El}} \triangleq \begin{bmatrix} v_{\mathfrak{El}} & v_{\mathfrak{El},\perp} \end{bmatrix} \in \mathbb{R}^{2 \times 2}$, where $v_{\mathfrak{El},\perp} \triangleq \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} v_{\mathfrak{El}}$, and let

$$M_{\mathfrak{El}} \triangleq Q_{\mathfrak{El}} \begin{bmatrix} \ell_{\mathfrak{El},1}^{-2} & 0 \\ 0 & \ell_{\mathfrak{El},2}^{-2} \end{bmatrix} Q_{\mathfrak{El}}^{\mathrm{T}}.$$

Then, for an ellipse $\mathfrak{El}$ obstacle, the unsafe set is given by

$$\overline{S}_{\mathfrak{El}} = \{p \in \mathbb{R}^2 \colon (p - p_{\mathfrak{El}})^{\mathrm{T}} M_{\mathfrak{El}}(p - p_{\mathfrak{El}}) \leq 1\}, \tag{53}$$

and thus, the corresponding matrix convex function is given by

$$\overline{H}_{\mathfrak{El}}(x) = (C_{\mathrm{pos}} x - p_{\mathfrak{El}})^{\mathrm{T}} M_{\mathfrak{El}}(C_{\mathrm{pos}} x - p_{\mathfrak{El}}) - 1. \tag{54}$$

Note that the projection of a state vector x_k onto the set $\overline{S}_{\mathfrak{El}}$, given by (18) can be formulated as a second-order cone optimization problem; in this work, this optimization problem is solved by using the `coneprog` solver from Matlab. Finally, it follows from the unsafe set (53) that the CBF associated with the ellipse obstacle safe set $S_{\mathfrak{El}}$ with buffer distance $\epsilon_{\mathfrak{El}}$ is given by

$$h_{\mathfrak{El}}(x_k) = 1 - (C_{\mathrm{pos}} x_k - p_{\mathfrak{El}})^{\mathrm{T}} M_{\mathfrak{El},\epsilon_{\mathfrak{El}}}(C_{\mathrm{pos}} x_k - p_{\mathfrak{El}}), \tag{55}$$

where

$$M_{\mathfrak{El},\epsilon_{\mathfrak{El}}} \triangleq Q_{\mathfrak{El}} \begin{bmatrix} (\ell_{\mathfrak{El},1} + \epsilon_{\mathfrak{El}})^{-2} & 0 \\ 0 & (\ell_{\mathfrak{El},2} + \epsilon_{\mathfrak{El}})^{-2} \end{bmatrix} Q_{\mathfrak{El}}^{\mathrm{T}},$$

and which can be used to implement safe control input constraints shown in (7b). Note that $\epsilon_{\mathfrak{El}}$ increases the lengths of the semi-major and semi-minor aces of the ellipse corresponding to the unsafe set considered by the CBF.

3) Convex Polytope: A convex polytope $\mathfrak{P}$ is defined by a matrix composed by $n_{\mathfrak{P}}$ vertices, given by $v_{\mathfrak{P}} \in \mathbb{R}^{2 \times n_{\mathfrak{P}}}$. Then, for a polytope $\mathfrak{P}$ obstacle, the unsafe set is given by

$$\overline{S}_{\mathfrak{P}} = \left\{ \bigcap_{i=1}^{n_{\mathfrak{P}}} \{p \in \mathbb{R}^2 \colon -A_{\mathfrak{P},i} p - b_{\mathfrak{P},i} \leq 0\} \right\}, \tag{56}$$

where $-A_{\mathfrak{P},i} \in \mathbb{R}^{1 \times 2}, -b_{\mathfrak{P},i} \in \mathbb{R}$ are associated with the halfspace that is determined by the i-th edge of $\mathfrak{P}$ and that does not contain the mean of all $n_{\mathfrak{P}}$ vertices. Hence, the corresponding matrix convex function is given by

$$\overline{H}_{\mathfrak{P}}(x) = \mathrm{diag}(-A_{\mathfrak{P},1} C_{\mathrm{pos}} x - b_{\mathfrak{P},1}, \dots, \\ -A_{\mathfrak{P},n_{\mathfrak{P}}} C_{\mathrm{pos}} x - b_{\mathfrak{P},n_{\mathfrak{P}}}). \tag{57}$$

Note that the projection of a state vector x_k onto the set $\overline{S}_{\mathfrak{P}}$, given by (18) can be written as

$$P_{\overline{S}_{\mathfrak{P}}}(x_k) = \arg\min_{z \in \mathbb{R}^n} \|z - C_{\text{pos}}x_k\| \tag{58a}$$

$$\text{s.t.} \quad \begin{array}{c} v_{\mathfrak{P}}\lambda_{\mathfrak{P}} = z, \\ \lambda_{\mathfrak{P}} \in \mathbb{R}^{n_{\mathfrak{P}}}, \lambda_{\mathfrak{P}} \geq 0, \mathbb{1}_{1 \times n_{\mathfrak{P}}}\lambda_{\mathfrak{P}} = 1, \end{array} \tag{58b}$$

and thus can be formulated as a constrained linear least-squares problem; in this work, this optimization problem is solved by using the `lsqlin` solver from Matlab with the `active-set` algorithm option. This problem is warm-started by providing the closest vertex to $C_{\text{pos}}x_k$ as an initial solution. Next, it follows from the unsafe set (56) that the safe set corresponding to the polytope obstacle is given by

$$S_{\mathfrak{P}} = \left\{\bigcup_{i=1}^{n_{\mathfrak{P}}}\{p \in \mathbb{R}^2 \colon A_{\mathfrak{P},i}p + b_{\mathfrak{P},i} \geq 0\}\right\}. \tag{59}$$

In order to derive a CBF for this union of sets, a discrete-time version of the CBF constraint proposed in (15) from [15] is formulated. Using a combinatorial CBF was required for the implementation of the CBF associated with the polytope, and we will explore this connection in more detail in future work. Hence, the safe input constraints associated with the polytope safe set $S_{\mathfrak{P}}$ with buffer distance $\epsilon_{\mathfrak{P}}$ that replace (7b) are given by

$$\begin{aligned} h_{\mathfrak{P},i}(f(x_k) + g(x_k)u) \\ \geq h_{\mathfrak{P},i}(x_k) - \gamma\left(h_{\mathfrak{P}}(x_k) + |h_{\mathfrak{P},i}(x_k) - h_{\mathfrak{P}}(x_k)|\right) \\ \forall i \in \{1, \ldots, n_{\mathfrak{P}}\}, \end{aligned} \tag{60}$$

where $\gamma \in [0, 1]$,

$$h_{\mathfrak{P},i}(x_k) \triangleq A_{\mathfrak{P},i}C_{\text{pos}}x_k + b_{\mathfrak{P},i} + \eta_i\epsilon_{\mathfrak{P}} \tag{61}$$
$$\forall i \in \{1, \ldots, n_{\mathfrak{P}}\},$$

$$h_{\mathfrak{P},\max}(x_k) \triangleq \max_{i \in \{1, \ldots, n_{\mathfrak{P}}\}} h_{\mathfrak{P},i}(x_k), \tag{62}$$

where, for all $i \in \{1, \ldots, n_{\mathfrak{P}}\}$, $\eta_i \in \{-1, 1\}$ is chosen so that the distance between the hyperplane given by $h_{\mathfrak{P},i}$ and the center of $\mathfrak{P}$ is increased by $\epsilon_{\mathfrak{P}}$ relative to the case where $\eta_i = 0$.

4) Spectrahedron: A spectrahedron $\mathfrak{Sp}$ in $\mathbb{R}^2$ is defined by a center position $p_{\mathfrak{Sp}} \in \mathbb{R}^2$, matrix size $n_{\mathfrak{Sp}}$, symmetric matrices $A_{0,\mathfrak{Sp}}, A_{1,\mathfrak{Sp}}, A_{2,\mathfrak{Sp}} \in \mathbb{R}^{n_{\mathfrak{Sp}} \times n_{\mathfrak{Sp}}}$, and rotation angle $\theta_{\mathfrak{Sp}}$ in radians. Then, for a polyhedron $\mathfrak{Sp}$ obstacle, the unsafe set is given by

$$\overline{S}_{\mathfrak{Sp}} = \{p \in \mathbb{R}^2 \colon A_{0,\mathfrak{Sp}} + z_1 A_{1,\mathfrak{Sp}} + z_2 A_{2,\mathfrak{Sp}} \succeq 0,$$
$$\text{where } \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = R_{\theta_{\mathfrak{Sp}}}(p - p_{\mathfrak{Sp}})\}, \tag{63}$$

where $R_{\theta_{\mathfrak{Sp}}} \in \mathbb{R}^{2 \times 2}$ is the rotation matrix associated with $\theta_{\mathfrak{Sp}}$. Hence, the corresponding matrix convex function is given by

$$\overline{H}_{\mathfrak{Sp}}(x) = -A_{\mathfrak{Sp}}\left(\begin{bmatrix} 1 \\ R_{\theta_{\mathfrak{Sp}}}(C_{\text{pos}}x - p_{\mathfrak{Sp}}) \end{bmatrix} \otimes I_{n_{\mathfrak{Sp}}}\right), \tag{64}$$

$$A_{\mathfrak{Sp}} \triangleq \begin{bmatrix} A_{0,\mathfrak{Sp}} & A_{1,\mathfrak{Sp}} & A_{2,\mathfrak{Sp}} \end{bmatrix}.$$

Note that the projection of a state vector x_k onto the set $\overline{S}_{\mathfrak{Sp}}$ requires the solution of the convex, semidefinite optimization problem given by (18); in this work, this optimization problem is solved using the `MOSEK` solver interfaced with the `YALMIP` toolbox. Next, it follows from the unsafe set (63) that the safe set corresponding to the spectrahedron obstacle is given by

$$S_{\mathfrak{Sp}} = \{p \in \mathbb{R}^2 \colon -(A_{0,\mathfrak{Sp}} + z_1 A_{1,\mathfrak{Sp}} + z_2 A_{2,\mathfrak{Sp}}) \nsucceq 0,$$
$$\text{where } \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = R_{\theta_{\mathfrak{Sp}}}(p - p_{\mathfrak{Sp}})\}, \tag{65}$$

since positive semidefiniteness does not hold if at least one of the eigenvalues of the associated real symmetric matrix is negative. In order to derive a CBF for this indefinite inequality, a discrete-time version of the CBF constraint proposed in (19) from [14] is formulated. Using an indefinite matrix CBF was required for the implementation of the CBF associated with the spectrahedron, and we will explore this connection in more detail in future work. Hence, the safe input constraint associated with the spectrahedron safe set $S_{\mathfrak{Sp}}$ with buffer ratio $\epsilon_{\mathfrak{Sp}} \in [0, 1)$ that replaces (7b) is given by

$$\begin{aligned} H_{\mathfrak{Sp}}(f(x_k) + g(x_k)u) \\ \geq (1 + c_\perp)H_{\mathfrak{Sp}}(x_k) - (\gamma + c_\perp)\,\lambda_{\mathfrak{Sp},\max}\,I_{n_{\mathfrak{Sp}}}, \end{aligned} \tag{66}$$

where $\gamma \in [0, 1]$, $c_\perp \in [0, 1]$, and

$$H_{\mathfrak{Sp}} \triangleq -A_{\mathfrak{Sp}}\left(\begin{bmatrix} 1 \\ (1 - \epsilon_{\mathfrak{Sp}})R_{\theta_{\mathfrak{Sp}}}(C_{\text{pos}}x - p_{\mathfrak{Sp}}) \end{bmatrix} \otimes I_{n_{\mathfrak{Sp}}}\right),$$
$$\lambda_{\mathfrak{Sp},\max} \triangleq \max(\text{eig}(H_{\mathfrak{Sp}}(x_k))).$$

Note that, unlike the other obstacles, $\epsilon_{\mathfrak{Sp}}$ represents a ratio which increases the size of the unsafe set considered by the CBF the closer its value is to 1, since distance cannot be intuitively defined relative to any of the spectrahedron parameters.

B. Discretized double integrators

Robotic agents are frequently represented using double-integrator dynamics, which capture the relationship between acceleration, velocity, and position. In this example, we consider the design of a safety-critical controller for an agent whose dynamics are governed by a double-integrator model. In particular, consider two discretized double integrators

$$\begin{bmatrix} p_{\text{h},k+1} \\ v_{\text{h},k+1} \end{bmatrix} = A_{\text{d}}\begin{bmatrix} p_{\text{h},k} \\ v_{\text{h},k} \end{bmatrix} + B_{\text{d}}u_{\text{h},k}, \tag{67}$$

$$\begin{bmatrix} p_{\text{v},k+1} \\ v_{\text{v},k+1} \end{bmatrix} = A_{\text{d}}\begin{bmatrix} p_{\text{v},k} \\ v_{\text{v},k} \end{bmatrix} + B_{\text{d}}u_{\text{v},k},, \tag{68}$$
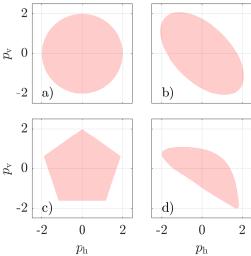
Fig. 2: Typical obstacle geometries. The figure shows a) circle (IV-A.1), b) ellipse (IV-A.2), c) convex polytope (IV-A.3), and d) spectrahedron (IV-A.4) shaped obstacles.

where $p_{\mathrm{h},k}, p_{\mathrm{v},k} \in \mathbb{R}$ denote the translations in the horizontal and vertical directions, respectively, $v_{\mathrm{h},k}, v_{\mathrm{v},k} \in \mathbb{R}$ denote the velocities in the horizontal and vertical directions, respectively, $u_{\mathrm{h},l}, u_{\mathrm{v},k} \in \mathbb{R}$ are horizontal and vertical acceleration commands, respectively, and the discretized state transition matrices are

$$A_{\mathrm{d}} \triangleq \begin{bmatrix} 1 & T_{\mathrm{s}} \\ 0 & 1 \end{bmatrix}, \quad B_{\mathrm{d}} \triangleq \begin{bmatrix} T_{\mathrm{s}}^2/2 \\ T_{\mathrm{s}} \end{bmatrix}. \qquad (69)$$

Note that (67), (68) can be written as

$$x_{k+1} = Ax_k + Bu_k, \qquad (70)$$

where $x_k \triangleq \begin{bmatrix} p_{\mathrm{h},k} & v_{\mathrm{h},k} & p_{\mathrm{v},k} & v_{\mathrm{v},k} \end{bmatrix}^{\mathrm{T}}$, $u_k \triangleq \begin{bmatrix} u_{\mathrm{h},k} & u_{\mathrm{v},k} \end{bmatrix}^{\mathrm{T}}$, $A \triangleq \mathrm{diag}(A_{\mathrm{d}}, A_{\mathrm{d}})$, and $B \triangleq \mathrm{diag}(B_{\mathrm{d}}, B_{\mathrm{d}})$. In this example, $C_{\mathrm{pos}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$, and we set $T_{\mathrm{s}} = 0.01$ s. Let $r_k \in \mathbb{R}^4$ be a reference state for x_k. Hence, the objective of the nominal controller is to minimize $\sum_{k=0}^{\infty} \|r_k - x_k\|$.

Consider a reference tracking nominal controller

$$u_{\mathrm{nom},k} = f_{\mathrm{lqr}}(x_k, r_k),$$

where $u_{\mathrm{nom},k} \in \mathbb{R}$ is the requested control input and f_{lqr} encodes an LQR controller. As mentioned at the beginning of this section, the details of the LQR controller implementation are omitted for brevity and since the main focus of the paper is the performance of the CBF.

The obstacles considered in this example are the following:

- A circle obstacle $\mathfrak{C}$ with $p_{\mathfrak{C}} = \begin{bmatrix} 18 & 16 \end{bmatrix}^{\mathrm{T}}$ m, and $r_{\mathfrak{C}} = 1.5$ m.

- An ellipse obstacle $\mathfrak{El}$ with $p_{\mathfrak{El}} = \begin{bmatrix} 2.5 & 5 \end{bmatrix}^{\mathrm{T}}$ m, $v_{\mathfrak{El}} = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}^{\mathrm{T}}$, and $\ell_{\mathfrak{El},1} = \sqrt{10}$ m, $\ell_{\mathfrak{El},2} = \sqrt{2}$,
- A convex polytope obstacle $\mathfrak{P}$ given by a regular pentagon ($n_{\mathfrak{P}} = 5$) with center at $\begin{bmatrix} 14 & 11 \end{bmatrix}^{\mathrm{T}}$ and a distance of 2 m from the center to each of its vertices.
- A spectrahedron obstacle $\mathfrak{Sp}$ with $p_{\mathfrak{Sp}} = \begin{bmatrix} 7.5 & 7.5 \end{bmatrix}^{\mathrm{T}}$ m, $n_{\mathfrak{Sp}} = 3$, $\theta_{\mathfrak{Sp}} = 0$, and

$$A_{0,\mathfrak{Sp}} = 2I_3, \quad A_{1,\mathfrak{Sp}} = \begin{bmatrix} 0 & 0.8 & 0 \\ 0.8 & 0 & 0.8 \\ 0 & 0.8 & 0 \end{bmatrix},$$

$$A_{2,\mathfrak{Sp}} = \begin{bmatrix} 0 & 0 & 1.6 \\ 0 & 0 & 2.4 \\ 1.6 & 2.4 & 0 \end{bmatrix}.$$

Two different CBF formulations are compared in the following example, the PDTE-MCBF formulation and a nonconvex CBF formulation:

- For the **PDTE-MCBF** formulation, the control input u_k is obtained by solving the constrained linear least-squares optimization problem

$$u_k = \operatorname*{argmin}_{\nu \in \mathbb{R}} \|\nu - u_{\mathrm{nom},k}\|_2, \qquad (71)$$

$$\text{s.t. } H^\epsilon(Ax_k + B\nu, x_k) \succeq (1-\gamma)H^\epsilon(x_k, x_k),$$

where H^ϵ is given by (24), (25) with $Q = 4$, each of the indices $i \in \{1,2,3,4\}$ corresponding to each of the obstacles, a_i, b_i^ϵ given by (48), and the projections $P_{\overline{S}_i}$ are calulated using (18) with the corresponding $\overline{H}_i$ for each obstacle given by (50), (54), (57), (64). Algorithm 1 is used to formulate and solve (71). This constrained linear least-squares optimization problem is solved at each iteration using the `lsqlin` Matlab solver with the `active-set` algorithm option. This problem is warm-started by providing the nominal input $u_{\mathrm{nom},k}$ as an initial solution. Furthermore, the optimization problem formulations to calculate the projections corresponding to each obstacle are discussed in Subsection IV-A. For this example, $\gamma = 0.2$, and $\varepsilon = 0.4$.

- For the **nonconvex CBF** formulation, the control input u_k is obtained by solving the nonconvex, semidefinite optimization problem

$$u_k = \operatorname*{argmin}_{\nu \in \mathbb{R}} \|\nu - u_{\mathrm{nom},k}\|_2, \qquad (72)$$

$$\text{s.t. } \begin{matrix} h_{\mathfrak{C}}(Ax_k + B\nu) \geq (1-\gamma)h_{\mathfrak{C}}(x_k) \\ h_{\mathfrak{El}}(Ax_k + B\nu) \geq (1-\gamma)h_{\mathfrak{El}}(x_k) \\ (60) \\ (66) \end{matrix}, \qquad (73)$$

with $h_{\mathfrak{C}}, h_{\mathfrak{El}}$ given by (52), (55), respectively, and $f(x_k) = Ax_k$, $g(x_k) = B$ in (60), (66). This

nonconvex, semidefinite optimization problem is solved at each iteration using the `bmibnb` solver from the `YALMIP` toolbox, interfaced with `MOSEK`. This problem is warm-started by providing the nominal input $u_{\mathrm{nom},k}$ as an initial solution. For this example, $\gamma = 0.2$, $\epsilon_{\mathfrak{C}} = \epsilon_{\mathfrak{El}} = \epsilon_{\mathfrak{P}} = 0.4$, $\epsilon_{\mathfrak{Sp}} = 0.25$, and $c_\perp = 0.05$.

For all simulations, we set $x_0 = 0$, and $r_k \equiv \begin{bmatrix} 18 & 0 & 20 & 0 \end{bmatrix}^{\mathrm{T}}$. The simulations are run for all $k \in \{0, k_{\mathrm{end}}\}$, with $k_{\mathrm{end}} = 1200$. The results of the simulations with the PDTE-MCBF and nonconvex CBF formulations are shown in Figures 3 and 4, respectively. The optimization problems run at each iteration returned feasible solutions in both simulations. Figures 3 and 4 show the trajectory of the double integrators, the unsafe sets given by the obstacles, and the buffer sets, which correspond to the unsafe sets considered by the corresponding CBF formulations and facilitated by the buffer distances $\epsilon, \epsilon_{\mathfrak{C}}, \epsilon_{\mathfrak{El}}, \epsilon_{\mathfrak{P}}$ and the buffer ratio $\epsilon_{\mathfrak{Sp}}$. The following can be concluded from both figures:

- The boundaries of the buffer sets resulting from the PDTE-MCBF formulation are more equidistant from the obstacle boundaries than the boundaries of the buffer sets resulting from the nonconvex CBF formulation; this is better observed when comparing the buffer sets corresponding to the pentagon and spectahedron obstacles in both figures.
- The buffer sets resulting from the PDTE-MCBF formulation are similar to the Minkowski sum of the unsafe sets and a circle of radius ϵ m.
- The tracking and obstacle avoidance performances of both CBF formulations are very similar.

Finally, Table I shows the average runtime per iteration for all CBF formulations for each of the simulations. In the case of PDTE-MCBF, the runtime without calculation of projected points refers to the runtime for solving (71) without calculating the projections $P_{\overline{S}_i}$, and the runtime with calculation of projected points includes the runtime for solving (71) considering the entirety of Algorithm 1. The results in Table I show that the total runtime of the PDTE-MCBF formulation is more than an order of magnitude faster than the nonconvex CBF formulation.

C. Bicopter lateral flight

Consider the bicopter in the vertical plane shown in Figure 5, which consists of a rigid frame with two rotors that generate thrust along their respective axes. The bicopter has mass m, center of mass c, moment of inertia J about c, and the distance between the rotors is ℓ_{mc}. Let T_1, T_2 denote the thrusts produced by the left and right rotors, respectively, as shown in Figure 5. Define the total thrust $T \overset{\triangle}{=} T_1 + T_2$ and the total moment $\tau \overset{\triangle}{=} (T_1 - T_2)/\ell_{\mathrm{mc}}$. Then, the dynamics of the bicopter
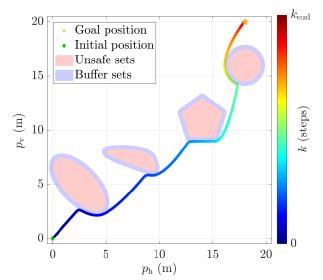

Fig. 3: **Discretized double integrators with PDTE-MCBF formulation.** The figure shows the trajectory of an agent modeled by double integrators for all $k \in [0, k_{\mathrm{end}}]$ with $k_{\mathrm{end}} = 1200$, the unsafe sets given by the obstacles, and the buffer sets, which are obtained by setting the buffer distance $\epsilon = 0.4$.
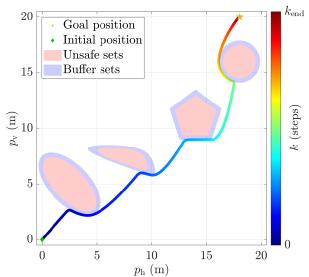

Fig. 4: **Discretized double integrators with nonconvex CBF formulation.** This figure shows the trajectory of an agent modeled by double integrators for all $k \in [0, k_{\mathrm{end}}]$ with $k_{\mathrm{end}} = 1200$, the unsafe sets given by the obstacles, and the buffer sets, which are obtained by setting the buffer distances $\epsilon_{\mathfrak{C}} = \epsilon_{\mathfrak{El}} = \epsilon_{\mathfrak{P}} = 0.4$ and the buffer ratio $\epsilon_{\mathfrak{Sp}} = 0.25$.

in the vertical plane are then given by

$$\dot{p}_{\mathrm{h}} = v_{\mathrm{h}}, \qquad \dot{v}_{\mathrm{h}} = \frac{T}{m} \sin\theta, \qquad (74)$$

$$\dot{p}_{\mathrm{v}} = v_{\mathrm{v}}, \qquad \dot{v}_{\mathrm{v}} = -\frac{T}{m} \cos\theta + g, \qquad (75)$$

$$\dot{\theta} = \omega, \qquad \dot{\omega} = \frac{\tau}{J}, \qquad (76)$$

where $p_{\mathrm{h}}, p_{\mathrm{v}} \in \mathbb{R}$ are the horizontal and vertical positions of c, respectively, $v_{\mathrm{h}}, v_{\mathrm{v}} \in \mathbb{R}$ are the horizontal

CBF formulation	Average runtime per iteration (s)
PDTE-MCBF (without calculation of projected points)	1.036×10^{-3}
PDTE-MCBF (with calculation of projected points)	6.002×10^{-2}
Nonconvex CBF	2.543

and vertical velocities of c, respectively, $\theta \in \mathbb{R}$ is the bicopter tilt, $\omega \in \mathbb{R}$ is the bicopter angular velocity, and g is the acceleration due to gravity.
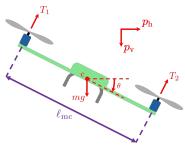

Fig. 5: A bicopter system whose motion is constrained in a vertical plane.

A discrete-time controller is implemented to control the continuous-time dynamics shown in (74)-(76). Hence, the states $p_{\mathrm{h}}, v_{\mathrm{h}}, p_{\mathrm{h}}, v_{\mathrm{h}}, \theta, \omega$ are sampled to produce the sampled states

$$p_{\mathrm{h},k} \triangleq p_{\mathrm{h}}(kT_{\mathrm{s}}), \qquad v_{\mathrm{h},k} \triangleq v_{\mathrm{h}}(kT_{\mathrm{s}}),$$
$$p_{\mathrm{v},k} \triangleq p_{\mathrm{v}}(kT_{\mathrm{s}}), \qquad v_{\mathrm{v},k} \triangleq v_{\mathrm{v}}(kT_{\mathrm{s}}),$$
$$\theta_k \triangleq \theta(kT_{\mathrm{s}}), \qquad \omega_k \triangleq \omega(kT_{\mathrm{s}}),$$

where $k \geq 0$ is the discrete-time step, and $T_{\mathrm{s}} > 0$ is the sampling time. The controller generates the total thrust $T_k \geq 0$ and the total torque $\tau_k \in \mathbb{R}$. The continuous-time signals T and τ applied to the bicopter are generated by applying a zero-order hold operation to T_k and τ_k, that is, for all $k \geq 0$,

$$T(t) = T_k, \ \tau(t) = \tau_k, \ \text{for all } t \in [kT_{\mathrm{s}}, (k+1)T_{\mathrm{s}}). \tag{77}$$

The nominal controller is designed so that $p_{\mathrm{h},k}$ and $p_{\mathrm{v},k}$ follow reference signals $r_{\mathrm{h},k}$ and $r_{\mathrm{v},k}$, respectively, such that the objective of the nominal controller is to minimize $\sum_{k=0}^{\infty} \left\| \begin{bmatrix} r_{\mathrm{h},k} & r_{\mathrm{v},k} \end{bmatrix}^{\mathrm{T}} - \begin{bmatrix} p_{\mathrm{h},k} & p_{\mathrm{v},k} \end{bmatrix}^{\mathrm{T}} \right\|$. For this purpose, the inner-loop, outer-loop control architecture shown in Figure 6 is adopted. An advantage of this architecture is that it allows the dynamics shown in (74)-(76) to be decoupled into linear systems, such that the

resulting decoupled, discretized dynamics are given by

$$x_{\mathrm{h},k+1} = A_{\mathrm{pos}}x_{\mathrm{h},k} + B_{\mathrm{pos}}u_{\mathrm{h},k}, \tag{78}$$
$$x_{\mathrm{v},k+1} = A_{\mathrm{pos}}x_{\mathrm{v},k} + B_{\mathrm{pos}}u_{\mathrm{v},k}, \tag{79}$$
$$x_{\mathrm{att},k+1} = A_{\mathrm{att}}x_{\mathrm{att},k} + B_{\mathrm{att}}\tau_k, \tag{80}$$

where $x_{\mathrm{h},k} \triangleq \begin{bmatrix} p_{\mathrm{h},k} & v_{\mathrm{h},k} \end{bmatrix}^{\mathrm{T}}$, $x_{\mathrm{v},k} \triangleq \begin{bmatrix} p_{\mathrm{v},k} & v_{\mathrm{v},k} \end{bmatrix}^{\mathrm{T}}$, $x_{\mathrm{att},k} \triangleq \begin{bmatrix} \theta_k & \omega_k \end{bmatrix}^{\mathrm{T}}$, $u_{\mathrm{h},k}, u_{\mathrm{v},k} \in \mathbb{R}$ are horizontal and vertical acceleration commands, respectively, and

$$A_{\mathrm{pos}} \triangleq A_{\mathrm{att}} \triangleq \begin{bmatrix} 1 & T_{\mathrm{s}} \\ 0 & 1 \end{bmatrix},$$
$$B_{\mathrm{pos}} \triangleq \begin{bmatrix} T_{\mathrm{s}}^2/(2m) \\ T_{\mathrm{s}}/m \end{bmatrix}, \quad B_{\mathrm{att}} \triangleq \begin{bmatrix} T_{\mathrm{s}}^2/(2J) \\ T_{\mathrm{s}}/J \end{bmatrix}.$$

The dynamics shown in (78)–(80) only hold near hover conditions ($\theta \approx 0$ deg) and are used to design the outer-loop and inner-loop controllers

Let the outer-loop controller $G_{\mathrm{c,ol}}$ be given by two LQR controllers for the horizontal and vertical states separately, such that

$$u_{\mathrm{nom,h},k} = f_{\mathrm{lqr,i,h}}\left(\begin{bmatrix} r_{\mathrm{h},k} - p_{\mathrm{h},k} \\ -v_{\mathrm{h},k} \end{bmatrix} \right), \tag{81}$$
$$u_{\mathrm{nom,v},k} = f_{\mathrm{lqr,i,v}}\left(\begin{bmatrix} r_{\mathrm{v},k} - p_{\mathrm{v},k} \\ -v_{\mathrm{v},k} \end{bmatrix} \right), \tag{82}$$

where $f_{\mathrm{lqr,i,h}}, f_{\mathrm{lqr,i,v}}$ implement LQR controllers with integrator states and different sets of gains; Algorithm 1 in [20] shows an implementation the LQR controllers mentioned above. As mentioned at the beginning of this section, more details of the LQR controllers are omitted for brevity and since the main focus of the paper is the performance of the CBF.

Next, in this example, the function f_{cbf} is a safety filter that implements either the PDTE-MCBF formulation or the nonconvex CBF formulation. The inputs of f_{cbf} are $u_{\mathrm{nom},k} \triangleq \begin{bmatrix} u_{\mathrm{nom,h},k} & u_{\mathrm{nom,v},k} \end{bmatrix}^{\mathrm{T}}$ and $x_k \triangleq \begin{bmatrix} x_{\mathrm{h},k}^{\mathrm{T}} & x_{\mathrm{v},k}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, and the output of f_{cbf} is $u_k \triangleq \begin{bmatrix} u_{\mathrm{h},k} & u_{\mathrm{v},k} \end{bmatrix}^{\mathrm{T}}$. In this example, $C_{\mathrm{pos}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$. The implementation details for both CBF formulations are given later.

The outer and inner loops are linked by a nonlinear mapping function f_{map} that can be used to obtain the thrust T_k and a reference tilt value $\theta_{\mathrm{r},k}$ from u_k, such that

$$\begin{bmatrix} T_k \\ \theta_{\mathrm{r},k} \end{bmatrix} = f_{\mathrm{map}}(u_k) = \begin{bmatrix} \sqrt{u_{\mathrm{h},k}^2 + (mg - u_{\mathrm{v},k})^2} \\ \mathrm{atan2}(u_{\mathrm{h},k}, \ mg - u_{\mathrm{v},k}) \end{bmatrix}. \tag{83}$$

Then, the inner-loop controller $G_{\mathrm{c,il}}$ is given by a LQR controller for the attitude states, such that

$$\tau_k = f_{\mathrm{lqr,i,att}}\left(\begin{bmatrix} \theta_{\mathrm{r},k} - \theta_k \\ -\omega_k \end{bmatrix} \right), \tag{84}$$

where $f_{\mathrm{lqr,i,att}}$ implements the LQR controller whose implementation is shown in Algorithm 1 in [20]. As mentioned at the beginning of this section, more details of the LQR controllers are omitted for brevity and since the main focus of the paper is the performance of the CBF.
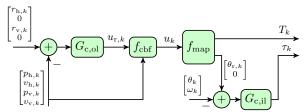

Fig. 6: Inner-loop, outer-loop control architecture used for bicopter tracking control.

Next, safe set details are discussed. In this example, the only obstacle considered is a circle obstacle with $p_{\mathfrak{C}} = \begin{bmatrix} 2 & -0.75 \end{bmatrix}^{\mathrm{T}}$ m, and $r_{\mathfrak{C}} = 1$ m, which entails constraints on the position states, addressed by the previously introduced CBF formulations. Aside from these constraints, velocity constraints and reference tilt constraints are considered to increase the likelihood that the safe inputs obtained by the CBFs do not cause a large deviation from hover conditions, which are assumed in the design of both the outer-loop and the inner-loop controllers. Hence, for all steps $k \geq 0$, the following velocity and reference tilt constraints are considered

$$v_{\mathrm{h},k} \in [-v_{\mathrm{h,max}}, v_{\mathrm{h,max}}] \text{ m/s}, \tag{85}$$

$$v_{\mathrm{v},k} \in [-v_{\mathrm{v,max}}, v_{\mathrm{v,max}}] \text{ m/s}, \tag{86}$$

$$\theta_{\mathrm{r},k} \in [-\theta_{\mathrm{r,max}}, \theta_{\mathrm{r,max}}] \text{ deg}, \tag{87}$$

where $v_{\mathrm{h,max}}, v_{\mathrm{v,max}}, \theta_{\mathrm{r,max}} > 0$ are the maximum absolute values of the horizontal and vertical velocities, and the reference tilt, respectively. The CBF associated with the constraints (85), (86) is given by

$$h_v(x_k) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} x_k + \begin{bmatrix} v_{\mathrm{h,max}} \\ v_{\mathrm{h,max}} \\ v_{\mathrm{v,max}} \\ v_{\mathrm{v,max}} \end{bmatrix}, \tag{88}$$

and thus the safe input constraints associated with the velocity states are given by

$$h_v(Ax_k + Bu_k) \geq (1 - \gamma)h_v(x_k), \tag{89}$$

where $\gamma \in [0, 1]$, $A \triangleq \mathrm{diag}(A_{\mathrm{pos}}, A_{\mathrm{pos}})$, and $B \triangleq \mathrm{diag}(B_{\mathrm{pos}}, B_{\mathrm{pos}})$, which results in an affine, convex constraint. Next, it follows from (83) that safe input constraints associated with the reference tilt constraint (87) is given by

$$\begin{bmatrix} 1 & -\mathrm{t}_{\theta_{\mathrm{r,max}}} \\ -1 & -\mathrm{t}_{\theta_{\mathrm{r,max}}} \end{bmatrix} u_k \geq -mg \ \mathrm{t}_{\theta_{\mathrm{r,max}}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \tag{90}$$

where $\mathrm{t}_{\theta_{\mathrm{r,max}}} \triangleq \tan\left(\frac{\pi}{180}\theta_{\mathrm{r,max}}\right)$.

Two different CBF formulations are compared in the following example, the PDTE-MCBF formulation and a nonconvex CBF formulation:

- For the **PDTE-MCBF** formulation, the control input u_k is obtained by solving the constrained linear least-squares optimization problem

$$u_k = \underset{\nu \in \mathbb{R}}{\mathrm{argmin}} \|\nu - u_{\mathrm{nom},k}\|_2, \tag{91}$$

$$\text{s.t.} \quad \begin{array}{c} H^\epsilon(Ax_k + B\nu, x_k) \succeq (1 - \gamma)H^\epsilon(x_k, x_k) \\ (89) \\ (90) \end{array},$$

where H^ϵ is given by (24), (25) with $Q = 1$, and index $i = 1$ corresponding to the circle obstacle, a_i, b_i^ϵ given by (48), and the projection $P_{\overline{S}_i}$ are calulated using (18) with the corresponding $\overline{H}_i$ for the circle obstacle given by (50); in this case, the projection has a closed-loop solution given by (51). Algorithm 1 is used to formulate and solve (91). This constrained linear least-squares optimization problem is solved at each iteration using the `lsqlin` Matlab solver with the `active-set` algorithm option. This problem is warm-started by providing the nominal input $u_{\mathrm{nom},k}$ as an initial solution. For this example, $\gamma = 0.1$, and $\varepsilon = 0.2$.

- For the **nonconvex CBF** formulation, the control input u_k is obtained by solving the nonconvex, semidefinite optimization problem

$$u_k = \underset{\nu \in \mathbb{R}}{\mathrm{argmin}} \|\nu - u_{\mathrm{nom},k}\|_2, \tag{92}$$

$$\text{s.t.} \quad \begin{array}{c} h_{\mathfrak{C}}(Ax_k + B\nu) \geq (1 - \gamma)h_{\mathfrak{C}}(x_k) \\ (89) \\ (90) \end{array},$$

with $h_{\mathfrak{C}}$ given by (52). This nonconvex, nonlinear optimization problem is solved at each iteration using two different solvers in separate simulations. The first considered solver is the `fmincon` Matlab solver with the `sqp` algorithm option and the maximum number of iterations set to 10^5. The second considered solver is the `bmibnb` solver from the YALMIP toolbox, interfaced with MOSEK. In both instances, this problem is warm-started by providing the nominal input $u_{\mathrm{nom},k}$ as an initial solution. For this example, $\gamma = 0.1$, and $\epsilon_{\mathfrak{C}} = 0.2$.

Remark 1. The `lsqlin` and `fmincon` solvers can converge to unfeasible results that do not meet the optimization constraints and output these values. In contrast, the YALMIP-based solver does not output a numerical value if it determines that the problem is unfeasible. In this case, the YALMIP-based solver is used a second

time to solve (92) with slack variables. If the `YALMIP`-based determines that the problem with slack variables is also unfeasible, the nominal control input is given as the output of the nonconvex, nonlinear optimization problem.

For all simulations, $x(t) = 0, T(t) = mg, \tau(t) = 0$ for all $t \leq 0$, $u_{h,k} = u_{v,k} = \theta_{r,k} = 0$ for all $k \leq 0$, $r_{h,k} \equiv 18$ m, $r_{v,k} \equiv 20$ m, and $T_s = 0.005$ s. The simulations are run for all $t \in [0, t_{end}]$ s, with $t_{end} = 10$ s. The Simulink[2] environment is used for numerical simulation with the `ode45` solver to solve the bicopter continuous-time dynamics. The discrete-time dynamics corresponding to the controller and either of the CBF formulations are evaluated every T_s seconds.

The results of the simulations with the PDTE-MCBF and nonconvex CBF formulations are shown in Figure 7. Furthermore, the feasibility of the solutions that each solver converged to at each iteration step for the PDTE-MCBF and nonconvex CBF formulations is shown in Figure 8; for the `YALMIP`-based solver, the problem solution is considered unfeasible if the first run without slack variables is considered unfeasible, as discussed in Remark 1. Finally, Table II shows the average runtime per iteration for all CBF formulations for each of the solvers used in the simulations; for the `YALMIP`-based solver, the runtime per iteration accounts for the runtime of the optimization problem with slack variables in the iterations where the first run without slack variables is considered unfeasible, as discussed in Remark 1.

In all cases, the optimization problem becomes unfeasible as the bicopter enters the buffer set. The safety filter is unable to prevent the bicopter from entering the buffer set since it only modifies the outer-loop controller outputs, which results in the inner-loop controller and the tilt angle dynamics becoming unmodeled input dynamics, which, in turn, can lead to safety violations, as discussed in [21]. However, while the nonconvex CBF formulation with the `fmincon` solver is unable to exit the buffer set and enters the unsafe set, the PDTE-MCBF formulation and the nonconvex CBF formulation with the `YALMIP`-based solver are able to recover from entering the buffer set and go back to the safe set before approaching the goal. Furthermore, as shown in Table II, the total runtime of the PDTE-MCBF formulation is faster than the nonconvex CBF formulation with the `fmincon` solver and is more than an order of magnitude faster than the nonconvex CBF formulation with the `YALMIP`-based solver. A video illustrating these simulation results in the accompanying animation at https://youtu.be/eYDTqa__lE4.
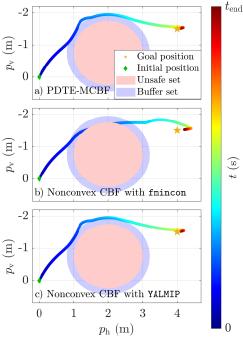

Fig. 7: **PDTE-MCBF and nonconvex CBF closed-loop performance comparison in bicopter lateral flight system.** This figure shows the trajectory of the bicopters for all $t \in [0, t_{end}]$ s, with $t_{end} = 10$ s, the unsafe set given by the circle, and the buffer sets, which correspond to the unsafe sets considered by the CBF formulations and facilitated by the buffer distances $\epsilon = \epsilon_{\mathfrak{C}} = 0.2$. The plots show the results for a) the PDTE-MCBF formulation solved with `lsqlin`, b) the nonconvex CBF formulation solved with `fmincon`, and c) the nonconvex CBF formulation solved with the `YALMIP`-based solver.
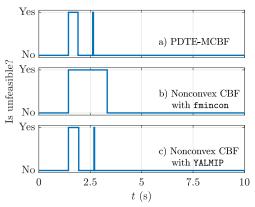

Fig. 8: **PDTE-MCBF and nonconvex CBF closed-loop unfeasibility comparison in bicopter lateral flight system.** The iterations for which the implemented solvers converged to unfeasible solutions is shown in these plots. The plots show the results for a) the PDTE-MCBF formulation solved with `lsqlin`, b) the nonconvex CBF formulation solved with `fmincon`, and c) the nonconvex CBF formulation solved with the `YALMIP`-based solver. For the `YALMIP`-based solver, the problem solution is considered unfeasible if the first run without slack variables is considered unfeasible, as discussed in Remark 1.

---

[2]Special instructions for interfacing `YALMIP`-based solvers with Simulink are given in https://yalmip.github.io/example/simulink/

TABLE II: Average runtime per iteration for all CBF formulations with their respective solvers.

CBF formulation	Average runtime per iteration (s)
PDTE-MCBF with `lsqlin`	5.251×10^{-6}
Nonconvex CBF with `fmincon`	3.113×10^{-5}
Nonconvex CBF with `YALMIP`	9.414×10^{-1}

V. CONCLUSION

This paper introduced the notion of exponential discrete-time matrix control barrier functions, presented a novel method to compute safe control inputs using solely convex optimization called *Projection-based Discrete-Time Exponential Matrix Control Barrier Function* (PDTE-MCBF), and demonstrated methods to handle disjunctive boolean constraints using discrete-time matrix control barrier functions. Numerical simulations showed the following advantages of the proposed method relative to nonconvex optimization methods:

- The tracking and obstacle avoidance performances of both the PDTE-MCBF and the nonconvex CBF formulations are very similar.
- The boundaries of the buffer sets resulting from the PDTE-MCBF formulation are more equidistant from the obstacle boundaries than The boundaries of the buffer sets resulting from the nonconvex CBF formulation.
- The total runtime of the PDTE-MCBF formulation is more than an order of magnitude faster than the nonconvex CBF formulation.

Future work will investigate extensions of our results to multi-agent systems and systems with noise and uncertainty.

VI. APPENDIX: MATRIX CONVEXITY

The following is an abbreviated overview of matrix convexity as defined in [18, Sec. 3.6.2]. More precisely, this definition of matrix convexity is simply $\mathbb{S}^p$-convexity for the proper cone $\mathbb{S}^p$.

Let $H : \mathbb{R}^n \to \mathbb{S}^p$ be a function with output in the space of symmetric real-valued $p \times p$ matrices.

Definition 8 (Matrix Convexity)**.** A mapping $H : \mathbb{R}^n \to \mathbb{S}^p$ is matrix convex if for all $x, y \in \mathbb{R}^n$ and for all $\theta \in [0, 1]$, the following holds:

$$H(\theta x + (1 - \theta)y) \preceq \theta H(x) + (1 - \theta)H(y). \quad (93)$$

Equivalently, H is matrix convex if and only if $x \mapsto y^\mathsf{T} H(x)y$ is convex in x for all $y \in \mathbb{R}^p$.

Definition 9 (Matrix Concavity)**.** A mapping $H : \mathbb{R}^n \to \mathbb{S}^p$ is matrix concave if for all $x, y \in \mathbb{R}^n$ and for all $\theta \in [0, 1]$, the following holds:

$$H(\theta x + (1 - \theta)y) \succeq \theta H(x) + (1 - \theta)H(y). \quad (94)$$

Equivalently, H is matrix concave if and only if $x \mapsto y^\mathsf{T} H(x)y$ is concave in x for all $y \in \mathbb{R}^p$.

Lemma VI.1. Let $A \in \mathbb{S}_+^p$. Then for all $a, b \in \mathbb{R}$,

$$a \leq b \iff aA \preceq bA. \quad (95)$$

Proof. Let $U\Lambda U^T$ be the eigendecomposition of A. We have

$$(b - a)A = U((b - a)\Lambda)U^T. \quad (96)$$

Since $A \in \mathbb{S}_+^p$, we have $\lambda_i(A) \geq 0$. In addition, $b - a \geq 0$ by assumption. It follows that $(b - a)A \succeq 0$, which holds if and only if $aA \preceq bA$. $\square$

Lemma VI.2. Let $\{f_i : \mathbb{R}^n \to \mathbb{R}\}_{i=1}^q$ be convex functions, let $\{A_i \in \mathbb{S}_+^p\}_{i=1}^q$ be positive semidefinite matrices, and let $B \in \mathbb{S}^p$ be a symmetric matrix. Then the following function is matrix convex:

$$H(x) \triangleq B + \sum_{i=1}^q A_i f_i(x). \quad (97)$$

Proof. Let $\theta \in [0, 1]$ and $x, y \in \mathbb{R}^n$. By Lemma VI.1 and the convexity of each f_i it follows for all $i = 1, \ldots, q$ that

$$A_i f_i(\theta x + (1 - \theta)y) \preceq A_i (\theta f_i(x) + (1 - \theta)f_i(y)), \quad (98)$$

$$= \theta A_i f_i(x) + (1 - \theta)A_i f_i(y). \quad (99)$$

Rearranging yields

$$\theta A_i f_i(x) + (1 - \theta)A_i f_i(y) - A_i f_i(\theta x + (1 - \theta)y) \succeq 0. \quad (100)$$

Since the sum of PSD matrices is also PSD, we have

$$\sum_{i=1}^q \theta A_i f_i(x) + (1 - \theta)A_i f_i(y) - A_i f_i(\theta x + (1 - \theta)y) \succeq 0. \quad (101)$$

Adding zero in the form $0 = B - B$ yields

$$\left(\sum_{i=1}^q \theta A_i f_i(x) + (1 - \theta)A_i f_i(y) + B\right)$$
$$- \left(\sum_{i=1}^q A_i f_i(\theta x + (1 - \theta)y) + B\right) \succeq 0.$$

Rearranging yields the desired inequality proving convexity:

$$\sum_{i=1}^q A_i f_i(\theta x + (1 - \theta)y) + B \preceq$$
$$\sum_{i=1}^q \theta A_i f_i(x) + (1 - \theta)A_i f_i(y) + B$$

$\square$

Lemma VI.3. The mapping $H : \mathbb{R}^n \to \mathbb{S}^p$ is matrix convex if H is in the form

$$H(x) = B + \sum_{j=1}^{J} A_j f_j(x) \qquad (102)$$

for some symmetric $B \in \mathbb{S}^p$ and $f_j : \mathbb{R}^n \to \mathbb{R}$, and at least one of the following conditions holds for each $j \in 1, \ldots, J$:

1) f_j is affine in x and $A_j \in \mathbb{S}^p$,
2) f_j is convex in x and $A_j \succeq 0$,
3) f_j is concave in x and $A_j \preceq 0$.

Proof. As per Definition 8, we proceed by proving that $x \mapsto y^\mathsf{T} H(x) y$ is convex for all $y \in \mathbb{R}^p$. Consider the function

$$x \mapsto y^\mathsf{T} \left(B + \sum_{j=1}^{J} A_j f_j(x) \right) y, \qquad (103)$$

$$\mapsto y^\mathsf{T} B y + \sum_{j=1}^{J} \left(y^\mathsf{T} A_j y \right) f_j(x) \qquad (104)$$

Case 1): If f_j is linear in x, then $y^\mathsf{T} B y + \sum_{j=1}^{J} \left(y^\mathsf{T} A_j y \right) f_j(x)$ is an affine function of x which is both convex and concave.

Case 2): Observe that $y^\mathsf{T} A_j y \geq 0$ since $A_j \succeq 0$. This implies that $y^\mathsf{T} B y + \sum_{j=1}^{J} \left(y^\mathsf{T} A_j y \right) f_j(x)$ is the sum of convex functions each multiplied by a nonnegative scalar, added to a constant value. This results in a convex function.

Case 3): Observe that $y^\mathsf{T} A_j y \leq 0$ since $A_j \preceq 0$. This implies that $y^\mathsf{T} B y + \sum_{j=1}^{J} \left(y^\mathsf{T} A_j y \right) f_j(x)$ is the sum of concave functions each multiplied by a nonpositive scalar, added to a constant value. This results in a convex function. $\square$

Lemma VI.4. The mapping $H : \mathbb{R}^n \to \mathbb{S}^p$ is matrix concave if H is in the form

$$H(x) = B + \sum_{j=1}^{J} A_j f_j(x) \qquad (105)$$

for some symmetric $B \in \mathbb{S}^p$ and at least one of the following conditions holds for each $j \in 1, \ldots, J$:

- f_j is affine in x and $A_j \in \mathbb{S}^p$,
- f_j is concave in x and $A_j \succeq 0$,
- f_j is convex in x and $A_j \preceq 0$.

Proof. Follows from similar arguments as Lemma VI.3, using Definition 9. $\square$

REFERENCES

[1] J. Breeden and D. Panagou, "Autonomous spacecraft attitude reorientation using robust sampled-data control barrier functions," *Journal of Guidance, Control, and Dynamics*, vol. 46, no. 10, pp. 1874–1891, 2023.

[2] T. G. Molnar, S. K. Kannan, J. Cunningham, K. Dunlap, K. L. Hobbs, and A. D. Ames, "Collision avoidance and geofencing for fixed-wing aircraft with control barrier functions," *IEEE Transactions on Control Systems Technology*, 2025.

[3] F. Ferraguti, C. T. Landi, A. Singletary, H.-C. Lin, A. Ames, C. Secchi, and M. Bonfe, "Safety and efficiency in robotics: The control barrier functions approach," *IEEE Robotics & Automation Magazine*, vol. 29, no. 3, pp. 139–151, 2022.

[4] W. S. Cortez, D. Oetomo, C. Manzie, and P. Choong, "Control barrier functions for mechanical systems: Theory and application to robotic grasping," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 530–545, 2019.

[5] Y. Chen, H. Peng, and J. Grizzle, "Obstacle avoidance for low-speed autonomous vehicles with barrier function," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 194–206, 2017.

[6] A. Alan, A. J. Taylor, C. R. He, A. D. Ames, and G. Orosz, "Control barrier functions and input-to-state safety with application to automated vehicles," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 6, pp. 2744–2759, 2023.

[7] K. Garg, J. Usevitch, J. Breeden, M. Black, D. Agrawal, H. Parwana, and D. Panagou, "Advances in the theory of control barrier functions: Addressing practical challenges in safe control synthesis for autonomous and robotic systems," *Annual Reviews in Control*, vol. 57, p. 100 945, 2024.

[8] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation.," in *Robotics: Science and Systems*, Cambridge, MA, USA, vol. 13, 2017, pp. 1–10.

[9] R. Cosner, P. Culbertson, A. Taylor, and A. Ames, "Robust Safety under Stochastic Uncertainty with Discrete-Time Control Barrier Functions," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, 2023.

[10] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *2021 American Control Conference (ACC)*, IEEE, 2021, pp. 3882–3889.

[11] S. Liu, J. Zeng, K. Sreenath, and C. A. Belta, "Iterative convex optimization for model predictive control with discrete-time high-order control barrier functions," in *2023 American Control Conference (ACC)*, IEEE, 2023, pp. 3368–3375.

[12] M. Khajenejad, M. Cavorsi, R. Niu, Q. Shen, and S. Z. Yong, "Tractable compositions of discrete-time control barrier functions with application to driving safety control," in *2021 European Control Conference (ECC)*, IEEE, 2021, pp. 1303–1309.

[13] M. Ahmadi, A. Singletary, J. W. Burdick, and A. D. Ames, "Safe policy synthesis in multi-agent pomdps via discrete-time barrier functions," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, IEEE, 2019, pp. 4797–4803.

[14] P. Ong, Y. Xu, R. M. Bena, F. Jabbari, and A. D. Ames, "Matrix control barrier functions," *arXiv preprint arXiv:2508.11795*, 2025.

[15] P. Ong, H. Lee, T. G. Molnar, D. Panagou, and A. D. Ames, "Combinatorial control barrier functions: Nested boolean and p-choose-r compositions of safety constraints," *arXiv preprint arXiv:2509.10716*, 2025.

[16] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[17] R. T. Rockafellar and R. J. Wets, *Variational analysis*. Springer, 1998.

[18] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[19] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," *Advances in neural information processing systems*, vol. 32, 2019.

[20] J. A. Paredes Salazar, J. Usevitch, and A. Goel, "Predictive control barrier functions for discrete-time linear systems with unmodeled delays," *arXiv preprint arXiv:2510.01059*, 2025.

[21] P. Seiler, M. Jankovic, and E. Hellstrom, "Control barrier functions with unmodeled input dynamics using integral quadratic constraints," *IEEE Contr. Syst. Lett.*, vol. 6, pp. 1664–1669, 2021.