Compatibility checking of multiple control barrier functions for input constrained systems

Xiao Tan, and Dimos V. Dimarogonas

***Abstract*— State and input constraints are ubiquitous in control system design. One recently developed tool to deal with these constraints is control barrier functions (CBF) which transform state constraints into conditions in the input space. CBF-based controller design thus incorporates both the CBF conditions and input constraints in a quadratic program. However, the CBF-based controller is well-defined only if the CBF conditions are compatible. In the case of perturbed systems, robust compatibility is of relevance. In this work, we propose an algorithmic solution to verify or falsify the (robust) compatibility of given CBFs *a priori*. Leveraging the Lipschitz properties of the CBF conditions, a grid sampling and refinement method with theoretical analysis and guarantees is proposed.**

I. Introduction

Control design for dynamical systems with input and state constraints is an omnipresent problem in engineering and has been extensively investigated over the last few decades. Among all the investigated control methods, such as model predictive control (MPC) [1], reference governor [2], barrier Lyapunov functions (BLF) [3], and prescribed performance control (PPC) [4], the so-called control barrier functions (CBF) has revived recently [5], [6] and gained increasing popularity in the control and robotic community. The latter three methods relate a system state and possibly a time state with a real number. Unlike its counterparts as in BLF and PPC, CBF is negative if the system state is in the unsafe/undesired regions. By enforcing an inequality constraint on the system input, which is referred to as the CBF condition, the system trajectory is guaranteed to stay within a given safe region for all time. A CBF-based controller is thus formed as a quadratic program (QP) [6] with the CBF conditions and actuator limits as the constraints while trying to minimize its differences to a pre-designed task-satisfying controller. A moderate magnitude control signal is obtained when the system state is close to the boundary of the safety set, making it applicable even in the presence of noise. Compared to MPC schemes, the CBF formulation only requires to solve a small size quadratic program online and thus is more suitable for embedded systems thanks to today's increasing computational power.

Another nice property of the CBF formulation is its modular design feature. In the case that multiple state constraints are present for the system, i.e., the safe region is the intersection of all these state constraints, we can add multiple CBF conditions to the QP formulation each of which corresponds to one state constraint. Under the core assumption that the CBF-induced QP is always feasible, or equivalently, the CBF conditions are compatible, the satisfaction of all the state constraints is guaranteed for all time.

However, the compatibility of multiple CBFs is in general difficult to check. The problem is even more challenging if input limits are also present. In [7], sufficient and necessary conditions on CBF compatibility are discussed for SISO systems without input constraints. In the recent work [8], the authors propose a mixed-initiative control formulation that satisfies the input bound explicitly and enforces CBF conditions only in a neighborhood of the safety boundary. Under an assumption that the neighborhood around the boundary of safety region specified by each CBF should not overlap, the mixed-initiative formulation is applicable in the presence of multiple CBFs. In [9], a navigation problem is considered and a CBF-based QP for the image of obstacles in the "ball world" is employed. The QP feasibility is guaranteed thanks to the special structure of the problem, yet input constraints are missing.

More relevant to our work is the sum-of-square (SoS) approach in [10] for verifying worst-case and stochastic system safety by constructing a barrier certificate. A recent work [11] extends the SoS technique to verify if a candidate function is indeed a CBF by checking whether a polynomial control input exists and satisfies the CBF condition in the safe region. In [12], a SoS-based compatibility verification scheme for multiple CBFs is proposed. However, these results are only applicable to polynomial control systems. Moreover, the feasibility result is also subject to the order of the polynomials, and a failure to find such polynomial inputs provides no falsification guarantee.

In this paper, we consider the compatibility checking problem when multiple control barrier functions are present for input constrained systems. We aim to give a verification or falsification on the compatibility of multiple CBFs prior to their online implementation. In that respect, a grid sampling and refinement method is proposed leveraging the Lipschitz properties of the CBF conditions. We show that 1) the proposed algorithm will output the exact compatibility result if it terminates, 2) the algorithm is guaranteed to terminate in finite steps if the multiple CBFs are robustly compatible, and 3) the upper bound of the robustness level can be obtained if a lower bound of the lattice size is incorporated.

This work was supported in part by Swedish Research Council (VR), in part by Swedish Foundation for Strategic Research (SSF) COIN Project, in part by ERC CoG LEAFHOUND, in part by EU CANOPIES Project, and in part by Knut and Alice Wallenberg Foundation (KAW). The authors are with the School of EECS, Royal Institute of Technology (KTH), 100 44 Stockholm, Sweden (Email: xiaotan, dimos@kth.se).

II. PRELIMINARIES

Notation: The operator $\nabla : C^1(\mathbb{R}^n) \to \mathbb{R}^n$ is defined as the gradient $\frac{\partial}{\partial x}$ of a scalar-valued differentiable function with respect to $\boldsymbol{x}$. The Lie derivatives of a function $h(\boldsymbol{x})$ for the system $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x}) + \mathfrak{g}(\boldsymbol{x})\boldsymbol{u}$ are denoted by $L_{\mathfrak{f}}h = \nabla h^\top \mathfrak{f}(\boldsymbol{x}) \in \mathbb{R}$ and $L_{\mathfrak{g}}h = \nabla h^\top \mathfrak{g}(\boldsymbol{x}) \in \mathbb{R}^{1\times m}$, respectively. The interior and boundary of a set $\mathcal{A}$ are denoted $\text{Int}(\mathcal{A})$ and $\partial\mathcal{A}$, respectively. A continuous function $\alpha : [0, a) \to [0, \infty)$ for $a \in \mathbb{R}_{>0}$ is a *class $\mathcal{K}$ function* if it is strictly increasing and $\alpha(0) = 0$ [13]. A continuous function $\alpha : (-b, a) \to (-\infty, \infty)$ for $a, b \in \mathbb{R}_{>0}$ is an *extended class $\mathcal{K}$ function* if it is strictly increasing and $\alpha(0) = 0$. Vector inequalities are to be interpreted element-wise. $\mathbf{0}, \mathbf{1}$ refer to vectors of proper dimensions with all entries to be 0 or 1, respectively.

Consider the nonlinear control affine system

$$\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x}) + \mathfrak{g}(\boldsymbol{x})\boldsymbol{u}, \tag{1}$$

where the state $\boldsymbol{x} \in \mathbb{R}^n$, and the control input $\boldsymbol{u} \in \mathbb{U} \subset \mathbb{R}^m$. Assume that the vector fields $\mathfrak{f}(\boldsymbol{x})$ and $\mathfrak{g}(\boldsymbol{x})$ are locally Lipschitz functions in $\boldsymbol{x}$. A set $\mathcal{A} \subset \mathbb{R}^n$ is called *forward invariant*, if for any initial condition $\boldsymbol{x}_0 \in \mathcal{A}$, the system solution $\boldsymbol{x}(t, \boldsymbol{x}_0) \in \mathcal{A}$ for all t in the maximal time interval of existence.

Consider the safety set $\mathcal{C}$ defined as an intersection of superlevel sets of continuously differentiable functions $h_i : \mathbb{R}^n \to \mathbb{R}, i \in \mathcal{I} = \{1, 2, ..., N\}$:

$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^n : h_i(\boldsymbol{x}) \geq 0, i \in \mathcal{I}\}. \tag{2}$$

Definition 1 (Compatible CBFs)**.** *The functions $h_i(\boldsymbol{x}), i \in \mathcal{I}$ are compatible control barrier functions (CBF) for* (1) *if there exists an open set $\mathcal{D} \supseteq \mathcal{C}$ and locally Lipschitz extended class $\mathcal{K}$ functions α_i such that, $\forall \boldsymbol{x} \in \mathcal{D}, \forall i \in \mathcal{I}$,*

$$\exists \boldsymbol{u} \in \mathbb{U},\ L_{\mathfrak{f}}h_i(\boldsymbol{x}) + L_{\mathfrak{g}}h_i(\boldsymbol{x})\boldsymbol{u} + \alpha_i(h_i(\boldsymbol{x})) \geq 0. \tag{3}$$

For given differentiable functions h_i and extended class $\mathcal{K}$ functions α_i, $i \in \mathcal{I}$, define

$$\mathcal{K}(\boldsymbol{x}) = \{\boldsymbol{u} \in \mathbb{U} : L_{\mathfrak{f}}h_i(\boldsymbol{x}) + L_{\mathfrak{g}}h_i(\boldsymbol{x})\boldsymbol{u} + \alpha_i(h_i(\boldsymbol{x})) \geq 0, \forall i \in \mathcal{I}\}. \tag{4}$$

Then $h_i(\boldsymbol{x}), i \in \mathcal{I}$, being compatible CBFs is equivalent to that $\mathcal{K}(\boldsymbol{x}) \neq \emptyset, \forall \boldsymbol{x} \in \mathcal{D}$.

Proposition 1. *If $h_i(\boldsymbol{x}), i \in \mathcal{I}$, are compatible CBFs, then any locally Lipschitz continuous feedback control law $\boldsymbol{u}(\boldsymbol{x}) \in \mathcal{K}(\boldsymbol{x})$ renders the safe set $\mathcal{C}$ forward invariant.*

Proof. This is evident from the Brezis version of Nagumo's Theorem at $\partial\mathcal{C}$. Please check [14, Theorem 4] for details. □

When the system is subject to disturbances/uncertainty, a relevant concept is that of *robustly compatible* CBFs.

Definition 2 (Robustly compatible CBFs)**.** *The functions $h_i(\boldsymbol{x}), i \in \mathcal{I}$ are robustly compatible control barrier functions with robustness level $\eta > 0$ for system* (1)*, if there exists an open set $\mathcal{D} \supseteq \mathcal{C}$ and locally Lipschitz extended class $\mathcal{K}$ functions α_i such that, $\forall \boldsymbol{x} \in \mathcal{D}, \forall i \in \mathcal{I}$,*

$$\exists \boldsymbol{u} \in \mathbb{U},\ L_{\mathfrak{f}}h_i(\boldsymbol{x}) + L_{\mathfrak{g}}h_i(\boldsymbol{x})\boldsymbol{u} + \alpha_i(h_i(\boldsymbol{x})) \geq \eta. \tag{5}$$

The condition in (5) is stricter compared to the condition in (3). If (5) holds, then the safety set $\mathcal{C}$ can be rendered forward invariant for the perturbed system $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x}) + \mathfrak{g}(\boldsymbol{x})\boldsymbol{u} + \mathfrak{p}(\boldsymbol{x})\boldsymbol{\omega}$ as long as $|L_{\mathfrak{p}\boldsymbol{\omega}}h_i(\boldsymbol{x})| \leq \eta$, $\forall i \in \mathcal{I}, \forall \boldsymbol{x} \in \mathcal{D}$. The analysis follows similarly as in [15] [16, Remark 3].

A CBF-based safety controller $\boldsymbol{u} : \mathcal{D} \to \mathbb{R}^m$ is given in the following form

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{x}) = \arg\min_{\boldsymbol{v}} &\|\boldsymbol{v} - \boldsymbol{u}_{nom}(\boldsymbol{x})\| \\ \text{s.t. } &\boldsymbol{v} \in \mathcal{K}(\boldsymbol{x}), \end{aligned} \tag{6}$$

where $\boldsymbol{u}_{nom}$ is a nominal controller focusing on task completion. For example, $\boldsymbol{u}_{nom}$ can be designed for state stabilization, reference tracking or can be given directly by a human user. One core problem for the CBF-based controller formulation in (6) is the compatibility, i.e., $\mathcal{K}(\boldsymbol{x}) \neq \emptyset, \forall \boldsymbol{x} \in \mathcal{D}$. When external disturbances are present, robustly compatibility is a desired property for practical implementation.

In this work, we propose an algorithmic solution to verify or falsify the hypothesis that $h_i(\boldsymbol{x}), i \in \mathcal{I}$ are (robustly) compatible. The compatibility verification algorithm only needs to be executed once and offline, before applying the CBF-based safety controller (6) online. For notational brevity, given $h_i(\boldsymbol{x}), \alpha_i(\cdot)$ and the control system in (1), we denote

$$A(\boldsymbol{x}) \triangleq \begin{pmatrix} L_{\mathfrak{g}}h_1(\boldsymbol{x}) \\ L_{\mathfrak{g}}h_2(\boldsymbol{x}) \\ \cdots \\ L_{\mathfrak{g}}h_N(\boldsymbol{x}) \end{pmatrix}, b(\boldsymbol{x}) \triangleq \begin{pmatrix} L_{\mathfrak{f}}h_1(\boldsymbol{x}) + \alpha_1(h_1(\boldsymbol{x})) \\ L_{\mathfrak{f}}h_2(\boldsymbol{x}) + \alpha_2(h_2(\boldsymbol{x})) \\ \cdots \\ L_{\mathfrak{f}}h_N(\boldsymbol{x}) + \alpha_N(h_N(\boldsymbol{x})) \end{pmatrix}.$$

The problem is thus to verify whether

$$\sup_{\boldsymbol{u}\in\mathbb{U}} A(\boldsymbol{x})\boldsymbol{u} + b(\boldsymbol{x}) \geq \mathbf{0}, \forall \boldsymbol{x} \in \mathcal{D}. \tag{7}$$

for compatibility, and whether

$$\sup_{\boldsymbol{u}\in\mathbb{U}} A(\boldsymbol{x})\boldsymbol{u} + b(\boldsymbol{x}) \geq \eta\mathbf{1}, \forall \boldsymbol{x} \in \mathcal{D}. \tag{8}$$

for robust compatibility with robustness level $\eta > 0$.

To simplify our analysis though without jeopardizing the generality, we assume the following:

Assumption 1. *The safe set $\mathcal{C}$ is compact.*

Assumption 2. *The input set $\mathbb{U}$ is convex.*

Under Assumption 2 and in view of (4), we know that $\mathcal{K}(\boldsymbol{x})$ is either empty or convex, for any $\boldsymbol{x} \in \mathcal{D}$.

III. PROPOSED SOLUTIONS

A. Grid sampling algorithm using n-cubes

We recall some basic notions for approximating a compact set in $\mathbb{R}^n$ using n-cubes. Let $\mathcal{S} \subset \mathbb{R}^n$ be a compact set, and $\{\boldsymbol{e}_i, i = 1, 2, ..., n\}$ the canonical basis of $\mathbb{R}^n$. For $\boldsymbol{x} \in \mathbb{R}^n, r > 0$, define

$$P_{\text{lattice}}(\boldsymbol{x}, r) = \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{y} = \boldsymbol{x} + \sum_{i\in\{1,2,...,n\}} a_i r \boldsymbol{e}_i, \forall a_i \in \mathbb{N}, i = \{1, 2, ..., n\}\}, \tag{9}$$

$$B(\boldsymbol{x}, r) = \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{y} = \boldsymbol{x} + \sum_{i \in \{1,2,...,n\}} k_i r \boldsymbol{e}_i, \\ \forall k_i \in [-1/2, 1/2], i \in \{1, 2, ..., n\}\}. \quad (10)$$

Here P_{lattice} denotes a set of points that forms a regular lattice with size r in $\mathbb{R}^n$ and $\boldsymbol{x} \in P_{\text{lattice}}$; $B(\boldsymbol{x}, r)$ denotes a n-cube in $\mathbb{R}^n$ centered at $\boldsymbol{x}$ with size r.

Now we propose the following grid sampling algorithm. First we calculate the range limit $\rho^{\min}_{\boldsymbol{e}_i}$ and $\rho^{\max}_{\boldsymbol{e}_i}, i = 1, 2, ..., n$ of the set $\mathcal{S}$ (Line 1 of Algorithm 1). Since $\mathcal{S}$ is compact, $\mathcal{S}$ is a subset of the hyperrectangle $[\rho^{\min}_{\boldsymbol{e}_1}, \rho^{\max}_{\boldsymbol{e}_1}] \times [\rho^{\min}_{\boldsymbol{e}_2}, \rho^{\max}_{\boldsymbol{e}_2}] \times ... \times [\rho^{\min}_{\boldsymbol{e}_n}, \rho^{\max}_{\boldsymbol{e}_n}]$ (Line 2). Then we construct a regular lattice P_{lattice} around the center point of the hyperrectangle with size r. In Line 3, we obtain a set P_{cand} by intersecting P_{lattice} with the inflated hyperrectangle $[\rho^{\min}_{\boldsymbol{e}_1} - r/2, \rho^{\max}_{\boldsymbol{e}_1} + r/2] \times [\rho^{\min}_{\boldsymbol{e}_2} - r/2, \rho^{\max}_{\boldsymbol{e}_2} + r/2] \times ... \times [\rho^{\min}_{\boldsymbol{e}_n} - r/2, \rho^{\max}_{\boldsymbol{e}_n} + r/2]$. We then collect all the points $\boldsymbol{p}$ in P_{cand} around which the n-cube with size r intersects with the set $\mathcal{S}$ (Line 4). The algorithm returns G as a Cartesian product of P and the singleton $\{r\}$.

Algorithm 1 `GridSampling`

Require: Compact set $\mathcal{S} \subset \mathbb{R}^n$, lattice size r

1: Calculate $\rho^{\min}_{\boldsymbol{e}_i} = \min_{\boldsymbol{x} \in \mathcal{S}} \boldsymbol{e}_i^\top \boldsymbol{x}, \rho^{\max}_{\boldsymbol{e}_i} = \max_{\boldsymbol{x} \in \mathcal{S}} \boldsymbol{e}_i^\top \boldsymbol{x}$ for $i \in \{1, 2, ..., n\}$.

2: Construct a regular lattice P_{lattice} around $(\frac{\rho^{\min}_{\boldsymbol{e}_1} + \rho^{\max}_{\boldsymbol{e}_1}}{2}, \frac{\rho^{\min}_{\boldsymbol{e}_2} + \rho^{\max}_{\boldsymbol{e}_2}}{2}, ..., \frac{\rho^{\min}_{\boldsymbol{e}_n} + \rho^{\max}_{\boldsymbol{e}_n}}{2})$ with size r.

3: Construct $P_{\text{cand}} = P_{\text{lattice}} \cap [\rho^{\min}_{\boldsymbol{e}_1} - r/2, \rho^{\max}_{\boldsymbol{e}_1} + r/2] \times [\rho^{\min}_{\boldsymbol{e}_2} - r/2, \rho^{\max}_{\boldsymbol{e}_2} + r/2] \times ... \times [\rho^{\min}_{\boldsymbol{e}_n} - r/2, \rho^{\max}_{\boldsymbol{e}_n} + r/2]$.

4: $P = \{\boldsymbol{p} \in P_{\text{cand}} : B(\boldsymbol{p}, r) \cap \mathcal{S} \neq \emptyset\}, G = P \times \{r\}$.

5: **return** G.

Proposition 2. *Given a compact set $\mathcal{S} \subset \mathbb{R}^n$and a lattice size $r > 0$, then the following hold:*

1) *G, from Algorithm 1, is of finite cardinality, and*
2) *$\mathcal{S} \subseteq \cup_{\boldsymbol{p} \in P} B(\boldsymbol{p}, r)$, where P is given in Algorithm 1, Line 4.*

Proof. Since the set $\mathcal{S}$ is compact, the lower and upper range limit $\rho^{\min}_{\boldsymbol{e}_i}$ and $\rho^{\max}_{\boldsymbol{e}_i}$, $i = 1, 2, .., n$, given in Line 1 of Algorithm 1, are finite for every dimension. This leads to the fact that the hyperrectangle $[\rho^{\min}_{\boldsymbol{e}_1} - r/2, \rho^{\max}_{\boldsymbol{e}_1} + r/2] \times [\rho^{\min}_{\boldsymbol{e}_2} - r/2, \rho^{\max}_{\boldsymbol{e}_2} + r/2] \times ... \times [\rho^{\min}_{\boldsymbol{e}_n} - r/2, \rho^{\max}_{\boldsymbol{e}_n} + r/2]$ is bounded. Recall that by definition (9), P_{lattice} denotes a regular lattice in $\mathbb{R}^n$, and we thus know that P_{cand} has a finite cardinality, which implies that G also has a finite cardinality. Now we show Property 2) by contradiction. Assume that there exists $\boldsymbol{x} \in \mathcal{S}$ and $\boldsymbol{x} \notin \cup_{\boldsymbol{p} \in P} B(\boldsymbol{p}, r)$. In view of the definition of P, this implies that $\boldsymbol{x} \notin \cup_{\boldsymbol{p} \in P_{\text{cand}}} B(\boldsymbol{p}, r)$. This yields a contradiction since $\boldsymbol{x} \in \mathcal{S} \subseteq [\rho^{\min}_{\boldsymbol{e}_1}, \rho^{\max}_{\boldsymbol{e}_1}] \times [\rho^{\min}_{\boldsymbol{e}_2}, \rho^{\max}_{\boldsymbol{e}_2}] \times ... \times [\rho^{\min}_{\boldsymbol{e}_n}, \rho^{\max}_{\boldsymbol{e}_n}] \subseteq \cup_{\boldsymbol{p} \in P_{\text{cand}}} B(\boldsymbol{p}, r)$. The former set inclusion is trivial in view of the definition of $\rho^{\min}_{\boldsymbol{e}_i}, \rho^{\max}_{\boldsymbol{e}_i}$. The latter set inclusions can be straightforwardly checked by discussing all possible relations of the points in P_{cand} and the hyberrectangle. □

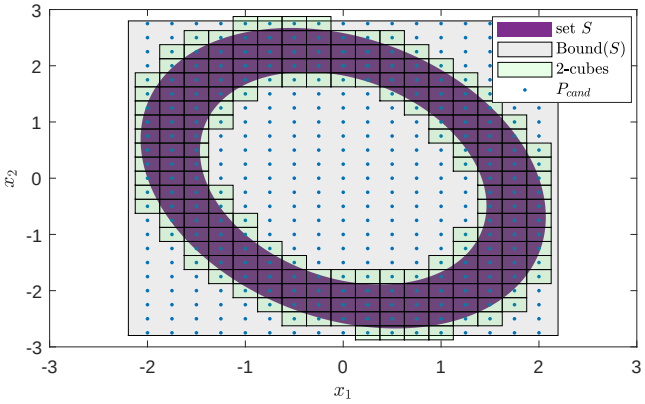


Fig. 1: Grid sampling of a set $\mathcal{S}$ in 2-D. Here the set $\mathcal{S}$ is shown in violet, Bound($\mathcal{S}$) is shown in gray, the 2-cubes generated from Algorithm 1 are in light green, and all the points in blue form P_{cand}. We observe that $\mathcal{S} \subset$ Bound($\mathcal{S}$) and $\mathcal{S}$ is over-approximated by the union of the 2-cubes.

From now on, we denote Bound($\mathcal{S}$) the bounding box $[\rho^{\min}_{\boldsymbol{e}_1} - r/2, \rho^{\max}_{\boldsymbol{e}_1} + r/2] \times [\rho^{\min}_{\boldsymbol{e}_2} - r/2, \rho^{\max}_{\boldsymbol{e}_2} + r/2] \times ... \times [\rho^{\min}_{\boldsymbol{e}_n} - r/2, \rho^{\max}_{\boldsymbol{e}_n} + r/2]$ of a compact set $\mathcal{S}$.

Example 1. Here we show an example of Algorithm 1 with the set $\mathcal{S} = \{\boldsymbol{x} \in \mathbb{R}^2 : 1 \leq \boldsymbol{x}^\top Q \boldsymbol{x} \leq 2$, where $Q = \left(\begin{smallmatrix} 0.5 & 0.1 \\ 0.1 & 0.3 \end{smallmatrix}\right)\}$ and $r = 0.25$. From Fig. 1, we observe that G has a finite cardinality and $\mathcal{S} \subseteq \cup_{\boldsymbol{p} \in P} B(\boldsymbol{p}, r)$. It is worth noting that $\cup_{\boldsymbol{p} \in P} B(\boldsymbol{p}, r) \nsubseteq$ Bound($\mathcal{S}$), where P is given in Algorithm 1 Line 4, as shown in Fig. 1.

B. Proposed verification algorithm

Now consider the compatibility verification problem in (7). For any $\boldsymbol{x} \in$ Bound($\mathcal{C}$), define

$$\begin{aligned} c(\boldsymbol{x}) = \max_{\boldsymbol{u}, t} \; & t \\ \text{s.t. } & A(\boldsymbol{x})\boldsymbol{u} + b(\boldsymbol{x}) \geq t \mathbf{1}_N, \\ & \boldsymbol{u} \in \mathbb{U}. \end{aligned} \quad (11)$$

In the case that $\mathbb{U}$ is a polytopic set, $c(\boldsymbol{x})$ is obtained by solving a linear program. In the general case where $\mathbb{U}$ is convex, $c(\boldsymbol{x})$ is obtained from a convex optimization. One interpretation is that $c(\boldsymbol{x})$ indicates the largest robustness level at $\boldsymbol{x}$ up to which the CBF conditions or the input constraints are to be breached.

Recall that the candidate CBFs $h_i(\boldsymbol{x}), i = 1, 2, ..., N$ are continuously differentiable, the vector fields $\mathbf{f}(\boldsymbol{x})$ and $\mathbf{g}(\boldsymbol{x})$ are locally Lipschitz, and thus $A(\boldsymbol{x}), b(\boldsymbol{x})$ in (7) are locally Lipschitz. Specifically, denote the respective Lipschitz constants in the bounding box Bound($\mathcal{C}$) with respect to the l_∞ norm as $L_{A,\infty}, L_{b,\infty}$, i.e.,

$$\begin{aligned} \|A(\boldsymbol{x}) - A(\boldsymbol{x}')\|_\infty &\leq L_{A,\infty} \|\boldsymbol{x} - \boldsymbol{x}'\|_\infty, \\ \|b(\boldsymbol{x}) - b(\boldsymbol{x}')\|_\infty &\leq L_{b,\infty} \|\boldsymbol{x} - \boldsymbol{x}'\|_\infty, \end{aligned} \quad (12)$$

for all $\boldsymbol{x}, \boldsymbol{x}' \in$ Bound($\mathcal{C}$)[1].

[1]Here $\|A\|_\infty$, where A is a matrix, refers to the induced matrix norm and can be calculated as the maximum absolute row sum of A.

If $c(\boldsymbol{x}) > 0$ for some $\boldsymbol{x}$, based on the Lipschitz continuity of $A(\boldsymbol{x})$ and $b(\boldsymbol{x})$, there must exist a neighborhood around $\boldsymbol{x}$ where the CBFs $h_i(\boldsymbol{x})$ are compatible. This is formally shown below.

Proposition 3. *For any $\boldsymbol{x} \in$ Bound($\mathcal{C}$), if $c(\boldsymbol{x}) > 0$, then $\sup_{\boldsymbol{v}\in\mathbb{U}} A(\boldsymbol{x}')\boldsymbol{v} + b(\boldsymbol{x}') \geq \mathbf{0}$ for all $\boldsymbol{x}' \in B(\boldsymbol{x}, \rho(\boldsymbol{x})) \cap$ Bound($\mathcal{C}$) with*

$$\rho(\boldsymbol{x}) = \frac{2c(\boldsymbol{x})}{L_{A,\infty}\|\boldsymbol{u}^\star(\boldsymbol{x})\|_\infty + L_{b,\infty}}, \tag{13}$$

where $L_{A,\infty}, L_{b,\infty}$ are the Lipschitz constants of $A(\boldsymbol{x}), b(\boldsymbol{x})$ with respect to the l_∞ norm as per (12)*, respectively, and $\boldsymbol{u}^\star(\boldsymbol{x})$ is the optimal solution to* (11) *at $\boldsymbol{x}$.*

Proof. For any $\boldsymbol{x}' \in B(\boldsymbol{x}, \rho) \cap$ Bound($\mathcal{C}$), we have

$$\begin{aligned} A(\boldsymbol{x}')\boldsymbol{u}^\star(\boldsymbol{x}) + b(\boldsymbol{x}') &= (A(\boldsymbol{x}') - A(\boldsymbol{x}))\boldsymbol{u}^\star(\boldsymbol{x}) \\ &\quad + (b(\boldsymbol{x}') - b(\boldsymbol{x})) + A(\boldsymbol{x})\boldsymbol{u}^\star(\boldsymbol{x}) + b(\boldsymbol{x}) \end{aligned} \tag{14}$$

In view of (12), we have

$$\begin{aligned} &\|(A(\boldsymbol{x}') - A(\boldsymbol{x}))\boldsymbol{u}^\star(\boldsymbol{x}) + b(\boldsymbol{x}') - b(\boldsymbol{x})\|_\infty \\ &\leq \|A(\boldsymbol{x}') - A(\boldsymbol{x})\|_\infty \|\boldsymbol{u}^\star(\boldsymbol{x})\|_\infty + \|b(\boldsymbol{x}') - b(\boldsymbol{x})\|_\infty \\ &\leq L_{A,\infty}\|\boldsymbol{x} - \boldsymbol{x}'\|_\infty \|\boldsymbol{u}^\star(\boldsymbol{x})\|_\infty + L_{b,\infty}\|\boldsymbol{x} - \boldsymbol{x}'\|_\infty \end{aligned} \tag{15}$$

In view of $\boldsymbol{x}' \in B(\boldsymbol{x}, \rho)$, and ρ in (13), we obtain $\|\boldsymbol{x} - \boldsymbol{x}'\|_\infty \leq \rho/2 = \frac{c(\boldsymbol{x})}{L_{A,\infty}\|\boldsymbol{u}^\star(\boldsymbol{x})\|_\infty + L_{b,\infty}}$. Thus, $\|(A(\boldsymbol{x}') - A(\boldsymbol{x}))\boldsymbol{u}^\star(\boldsymbol{x}) + b(\boldsymbol{x}') - b(\boldsymbol{x})\|_\infty \leq c(\boldsymbol{x})$. From (14) and $A(\boldsymbol{x})\boldsymbol{u}^\star(\boldsymbol{x}) + b(\boldsymbol{x}) \geq c(\boldsymbol{x})\mathbf{1}$, we further obtain $A(\boldsymbol{x}')\boldsymbol{u}^\star + b(\boldsymbol{x}') \geq \mathbf{0}$, which completes the proof. □

Algorithm 2 `CompatibilityChecking`

Require: $h_i(\boldsymbol{x}), \alpha_i(\cdot)$, initial size r_0, decaying factor λ

1: **Initialization:**
2: $k = 0$, obtain $\mathcal{C}$ from (2), $G_0 \leftarrow$ `GS`$(\mathcal{C}, r_0)$, $G_1 = \emptyset$.
3: **while** $G_k \neq \emptyset$ **do**
4: **for** each $(\boldsymbol{x}, r) \in G_k$ **do**
5: $c \leftarrow c(\boldsymbol{x})$ from (11), $\rho \leftarrow \rho(\boldsymbol{x})$ from (13).
6: **if** $c < 0$ **then** ▷ Found an incompatible state;
7: **return False**.
8: **else if** $\rho \geq r$ **then** ▷ Compatibility checked;
9: remove $(\boldsymbol{x}, r)$ from G_k.
10: **else** ▷ Compatibility partially checked;
11: remove $(\boldsymbol{x}, r)$ from G_k, $r' \leftarrow \lambda r$.
12: $G_{k+1} \leftarrow G_{k+1} \cup$ `GS`$(B(\boldsymbol{x}, r) \setminus B(\boldsymbol{x}, \rho), r')$.
13: **end if**
14: **end for**
15: $k = k + 1, G_{k+2} = \emptyset$.
16: **end while**
17: **return True**.

*`GS` stands for `GridSampling` given in Algorithm 1.

Built on above analysis, we design a compatibility checking algorithm using grid sampling and refinement. As given in Algorithm 2, the safety set $\mathcal{C}$ is firstly over-approximated using `GridSampling` Algorithm with an initial lattice size r_0. This will yield a finite set G_0 of n-cubes that is to be checked later. Recall that in Problem formulation (7) and (8), we need to check the compatibility over a set $\mathcal{D} \supseteq \mathcal{C}$. Here we take $\mathcal{D} = \cup_{(\boldsymbol{x}_i, r_0)\in G_0} B(\boldsymbol{x}_i, r_0)$, which is a super set of $\mathcal{C}$ from Proposition 2, item 2). Choosing r_0 is important and depends on how large buffering zone one allows outside the safety set. For each n-cube $B(\boldsymbol{x}, r)$ in G_k, represented as a $(\boldsymbol{x}, r)$ pair in Line 4, we calculate the robustness level c and the size ρ of a guaranteed compatible n-cube centered at $\boldsymbol{x}$ from (11) and (13), respectively. If $c < 0$, then an incompatible state is found and the algorithm terminates and returns `False`. If $\rho \geq r$, then we know that the CBFs are compatible for all the states within the n-cube $B(\boldsymbol{x}, r)$ and we remove $(\boldsymbol{x}, r)$ from G_k; otherwise, we refine the remaining unchecked region $B(\boldsymbol{x}, r) \setminus B(\boldsymbol{x}, \rho)$ with a discounted lattice size $r' = \lambda r$ and include the new n-cubes in G_{k+1}. After checking all the n-cubes in G_k, we iterate the process again for G_{k+1}. Once $G_{k+1} = \emptyset$, the algorithm terminates and returns `True`.

The following properties provide a guarantee on the finite-step termination of the algorithm and the compatibility property certified from its termination.

Theorem 1. *Given control barrier functions $h_i(\boldsymbol{x})$, extended class $\mathcal{K}$ functions $\alpha_i(\cdot)$ with $i \in \mathcal{I}$, an initial lattice size $r_0 > 0$ and a decaying factor $0 < \lambda < 1$, we have:*

1) *If Algorithm 2 terminates, it gives verification or falsification on the CBF compatibility as per Def. 1;*
2) *if $\mathbb{U}$ is bounded, and the CBFs $h_i(\boldsymbol{x})$ are robustly compatible with robustness level $\eta > 0$ in* Bound($\mathcal{C}$)*, then Algorithm* (2) *terminates in finite steps.*
3) *If $\mathbb{U}$ is bounded, and a lower bound of the lattice size $\underline{r}$ is incorporated, i.e., Algorithm 2 terminates if $r \leq \underline{r}$ in Line 4, then Algorithm 2 terminates in finite steps and gives one of the following three results:*
   i. *$h_i(\boldsymbol{x}), i \in \mathcal{I}$ are compatible;*
   ii. *$h_i(\boldsymbol{x}), i \in \mathcal{I}$ are incompatible;*
   iii. *$h_i(\boldsymbol{x}), i \in \mathcal{I}$ are not robustly compatible with robust level greater than*

$$\eta' = \lambda^{-1}\underline{r}(\max_{\boldsymbol{u}\in\mathbb{U}} L_{A,\infty}\|\boldsymbol{u}\|_\infty + L_{b,\infty})/2. \tag{16}$$

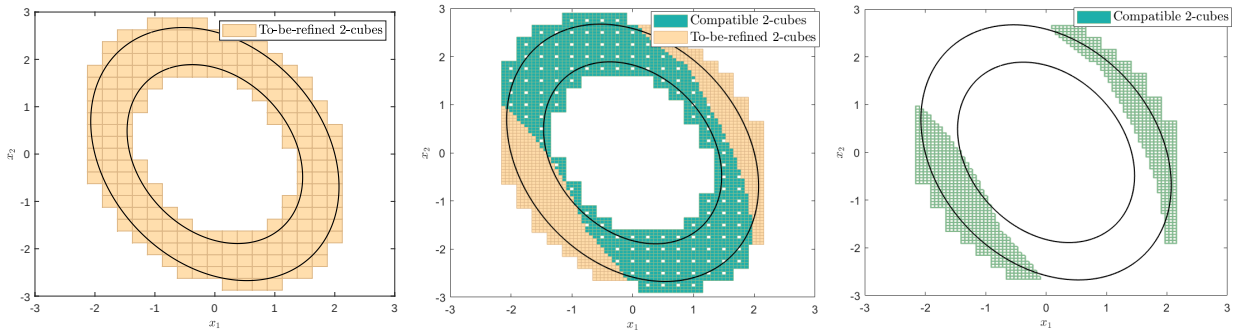
Proof to this theorem is omitted here due to page limits. Interested readers are referred to [17] for details as well as a thorough discussion on the proposed algorithm.

IV. Case Studies

In this section we show more details on the algorithm implementation, especially the Lipschitz constant calculation, and demonstrate the efficacy of our proposed verification algorithm in several different scenarios. All the simulations are done using Matlab Parallel Computing Toolbox on an Intel i7-8650U CPU laptop.

Example 2. Consider a $2-D$ system with state variable $\boldsymbol{x} = (x_1, x_2)$, input variable $\boldsymbol{u} = (u_1, u_2)$, dynamics

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \underbrace{\begin{pmatrix} x_1 + x_2 \\ -x_1^2/2 \end{pmatrix}}_{\mathrm{f}(\boldsymbol{x})} + \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{\mathrm{g}(\boldsymbol{x})} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \tag{17}$$



(a) First iteration, $r = 0.25$. (b) Second iteration, $r = 0.0625$. (c) Third iteration, $r = 0.0156$.

Fig. 2: Execution process of `CompatibilityChecking` in Example 2. The safety region is between the two ellipsoids. Compatible 2-cubes: 2-cubes within which the CBFs are verified to be compatible (in green); to-be-refined 2-cubes: 2-cubes that need further refinement (in yellow). (a) First iteration with the lattice size $r = 0.25$. All of the total 200 2-cubes are to-be-refined 2-cubes. (b) Second iteration with the lattice size $r = 0.0625$. The refined 2-cubes are checked and 3204 out of 4869 are verified to be compatible 2-cubes, and 1665 2-cubes are refined again. (c) Third iteration with the lattice size $r = 0.0156$. All the 2-cubes are compatible. Algorithm 2 thus gives verification on the compatibility of the CBFs.

and the input constraint set $\mathbb{U} = \{(u_1, u_2) : |u_1| \leq 3, |u_2| \leq 3\}$. The two CBF candidates are $h_1(\boldsymbol{x}) = \boldsymbol{x}^\top Q \boldsymbol{x} - 1, h_2(\boldsymbol{x}) = 2 - \boldsymbol{x}^\top Q \boldsymbol{x}$, where $Q = \left(\begin{smallmatrix} 0.5 & 0.1 \\ 0.1 & 0.3 \end{smallmatrix}\right)$. The safety set is $\mathcal{C} = \{\boldsymbol{x} : h_i(\boldsymbol{x}) \geq 0, i = 1, 2\}$. The corresponding extended class $\mathcal{K}$ functions are chosen as $\alpha_1(v) = v, \alpha_2(v) = v, v \in \mathbb{R}$. The CBF conditions are then given by

$$A(\boldsymbol{x}) := \begin{pmatrix} \nabla^\top h_1(\boldsymbol{x}) \\ \nabla^\top h_2(\boldsymbol{x}) \end{pmatrix}, b(\boldsymbol{x}) := \begin{pmatrix} \nabla^\top h_1(\boldsymbol{x}) \mathfrak{f}(\boldsymbol{x}) + h_1(\boldsymbol{x}) \\ \nabla^\top h_2(\boldsymbol{x}) \mathfrak{f}(\boldsymbol{x}) + h_2(\boldsymbol{x}) \end{pmatrix}.$$

where $\nabla h_1(\boldsymbol{x}) = (Q + Q^\top)\boldsymbol{x} = 2Q\boldsymbol{x}, \nabla h_2(\boldsymbol{x}) = -2Q\boldsymbol{x}$. Choose the initial lattice size $r_0 = 0.25$. By applying `GridSampling`(C, r_0), we obtain, as shown in Fig. 1, Bound$(\mathcal{C}) = [-2.2, 2.2] \times [-2.8, 2.8]$.

Now we calculate the Lipschitz constants $L_{A,\infty}, L_{b,\infty}$ in Bound$(\mathcal{C})$. We note that the Lipschitz constants can be obtained by considering each CBF individually, taking advantage of the fact that each row of $A(\boldsymbol{x})$ and $b(\boldsymbol{x})$ corresponds to one CBF. By definition, $L_{A,\infty}$ needs to satisfy

$$\|A(\boldsymbol{x}) - A(\boldsymbol{x}')\|_\infty = \left\| \begin{pmatrix} 2(\boldsymbol{x} - \boldsymbol{x}')^\top Q \\ -2(\boldsymbol{x} - \boldsymbol{x}')^\top Q \end{pmatrix} \right\|_\infty \leq L_{A,\infty} \|\boldsymbol{x} - \boldsymbol{x}'\|_\infty \tag{18}$$

for any $\boldsymbol{x}, \boldsymbol{x}'$ in Bound$(\mathcal{C})$. Let $\boldsymbol{a}_i(\boldsymbol{x})$ be the ith row of $A(\boldsymbol{x})$. Condition (18) is equivalent to

$$\begin{aligned} \|\boldsymbol{a}_i(\boldsymbol{x}) - \boldsymbol{a}_i(\boldsymbol{x}')\|_1 &= \|2Q(\boldsymbol{x} - \boldsymbol{x}')\|_1 \\ &\leq L_{A,\infty} \|\boldsymbol{x} - \boldsymbol{x}'\|_\infty, \forall i \in \{1, 2\}. \end{aligned} \tag{19}$$

This is implied by the condition $\max_{\boldsymbol{x}, \boldsymbol{x}'} \frac{\|2Q(\boldsymbol{x} - \boldsymbol{x}')\|_1}{\|\boldsymbol{x} - \boldsymbol{x}'\|_\infty} \leq L_{A,\infty}$. Using the inequality $\|\boldsymbol{v}\|_\infty \leq \|\boldsymbol{v}\|_1 \leq n\|\boldsymbol{v}\|_\infty, \forall \boldsymbol{v} \in \mathbb{R}^n$, we have

$$\begin{aligned} &\max_{\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{x} \neq \boldsymbol{x}'} \frac{\|2Q(\boldsymbol{x} - \boldsymbol{x}')\|_1}{\|\boldsymbol{x} - \boldsymbol{x}'\|_\infty} \\ &\leq 2 \max_{\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{x} \neq \boldsymbol{x}'} \frac{\|2Q(\boldsymbol{x} - \boldsymbol{x}')\|_1}{\|\boldsymbol{x} - \boldsymbol{x}'\|_1} = 4\|Q\|_1 \end{aligned} \tag{20}$$

The equality holds due to the definition of induced matrix norm. This reveals that $L_{A,\infty} = 4\|Q\|_1 = 2.4$ satisfies (18).[2]

Similarly, $L_{b,\infty}$ needs to satisfy

$$\|b(\boldsymbol{x}) - b(\boldsymbol{x}')\|_\infty \leq L_{b,\infty} \|\boldsymbol{x} - \boldsymbol{x}'\|_\infty \tag{21}$$

for any $\boldsymbol{x}, \boldsymbol{x}'$ in Bound$(\mathcal{C})$. Let $b_i(\boldsymbol{x})$ be the ith row of $b(\boldsymbol{x})$. Thus, (21) is equivalent to $|b_i(\boldsymbol{x}) - b_i(\boldsymbol{x}')| \leq L_{b,\infty} \|\boldsymbol{x} - \boldsymbol{x}'\|_\infty, \forall i = \{1, 2\}$, for any $\boldsymbol{x}, \boldsymbol{x}'$ in Bound$(\mathcal{C})$. Recall that

$$b_1(\boldsymbol{x}) = 2\boldsymbol{x}^\top Q \mathfrak{f}(\boldsymbol{x}) + \boldsymbol{x}^\top Q \boldsymbol{x} - 1 \tag{22}$$

$$b_2(\boldsymbol{x}) = 2\boldsymbol{x}^\top Q \mathfrak{f}(\boldsymbol{x}) - \boldsymbol{x}^\top Q \boldsymbol{x} + 2 \tag{23}$$

We have $\nabla b_1(\boldsymbol{x}) = 2Q\mathfrak{f}(\boldsymbol{x}) + 2\frac{\partial \mathfrak{f}}{\partial \boldsymbol{x}}(\boldsymbol{x})Q\boldsymbol{x} + 2Q\boldsymbol{x}, \nabla b_2(\boldsymbol{x}) = 2Q\mathfrak{f}(\boldsymbol{x}) + 2\frac{\partial \mathfrak{f}}{\partial \boldsymbol{x}}(\boldsymbol{x})Q\boldsymbol{x} - 2Q\boldsymbol{x}$, where $\frac{\partial \mathfrak{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}) = \left(\begin{smallmatrix} 1 & 1 \\ -x_1 & 0 \end{smallmatrix}\right)$. Following Mean Value Theorem, we know

$$|b_i(\boldsymbol{x}) - b_i(\boldsymbol{x}')| \leq \max_{\boldsymbol{v} \in \text{Bound}(\mathcal{C})} \|\nabla b_i(\boldsymbol{v})\|_\infty \|\boldsymbol{x} - \boldsymbol{x}'\|_\infty,$$

$\forall \boldsymbol{x}, \boldsymbol{x}' \in$ Bound$(\mathcal{C}), \forall i \in \{1, 2\}$. Thus we choose

$$L_{b,\infty} \geq \max_{i=1,2} \max_{\boldsymbol{v} \in \text{Bound}(\mathcal{C})} (\|\nabla b_i(\boldsymbol{v})\|_\infty).$$

This leads to solve two quadratic programs and we obtain $L_{b,\infty} = 13$.

Now we have all the necessary elements to execute `CompatibilityChecking`. Choose $\lambda = 0.25$. The Algorithm terminates after 3 iterations and verifies the compatibility of the two CBFs. The execution process takes 169s and is shown in Fig. 2.

Example 3. Now we consider the same scenario as in Example 2 but with a more stringent input set $\mathbb{U} = \{(u_1, u_2) : |u_1| \leq 2, |u_2| \leq 2\}$. This time, `CompatibilityChecking` gives a falsification on the compatibility. It finds an incompatible state $\boldsymbol{x}_{in} = (-1.5, -1.25)$, at which point

[2]Here $\|Q\|_1$, where Q is a matrix, refers to the induced matrix norm and can be calculated as the maximum absolute column sum of Q.

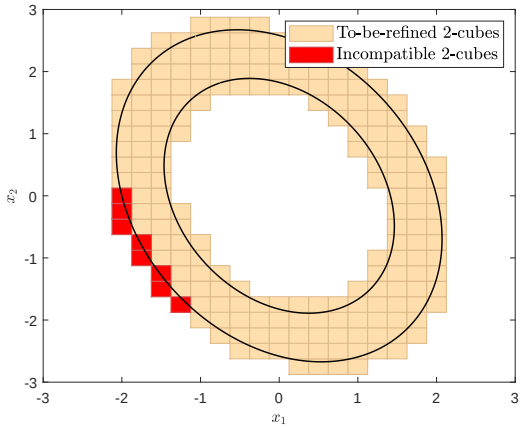


Fig. 3: Execution process of `CompatibilityChecking` in Example 3. The safety region is between the two ellipsoids. To-be-refined 2-cubes: 2-cubes that need further refinement (in yellow); incompatible 2-cubes: 2-cubes whose center point is an incompatible state (in red). At the first iteration with the lattice size $r = 0.25$, Algorithm 2 finds incompatible 2-cubes and terminates, falsifying the CBF compatibility.

$h_1(\boldsymbol{x}_{in}) = 0.96, h_2(\boldsymbol{x}_{in}) = 0.03, c(\boldsymbol{x}_{in}) = -0.36$. The execution process takes 23s and is shown in Fig. 3.

Example 4. In this example, a lower bound $\underline{r}$ of the size of the 2-cubes is incorporated in Algorithm 2 in a fashion described in Property 3) of Theorem 1. The input set is again $\mathbb{U} = \{(u_1, u_2) : |u_1| \leq 3, |u_2| \leq 3\}$. In Example 2, we have already showed that the multiple CBFs are compatible. Now we would like to obtain an upper bound of the robustness level the multiple CBFs can attain. From Fig. 2, if we set $\underline{r} = 0.016$, then `CompatibilityChecking` takes 73s and terminates after 2 iterations due to the lattice size decreases as the algorithm iterates and is smaller than $\underline{r}$ after the second iteration. Thus the termination belongs to **Case c** in the proof of Theorem 1. Based on (16), we thus know the multiple CBFs are at most robustly compatible with a robustness level $\eta = 0.6464$. This is validated by, for example, considering that $c(-1.5, -1.25) = 0.5$.

V. Conclusions

In this work, we propose a grid sampling and refinement verification scheme for the compatibility checking of multiple control barrier functions for input constrained control systems. We provide both implementation details and theoretical guarantees for the verification problem. In particular, we show that if the algorithm terminates, it will give an exact answer to the compatibility problem. If the multiple CBFs are robustly compatible, then the algorithm is guaranteed to terminate in finite steps. If we also incorporate a lower bound on the size of the n-cubes, then we can also obtain the largest robustness level the multiple CBFs can possibly attain. We demonstrate the efficacy of the proposed algorithm in several scenarios that corroborates our theoretical results.

This work focuses on the compatibility checking of multiple CBFs. Future directions include how to determine the extended class $\mathcal{K}$ functions that mitigate the possible incompatibility and/or increase the robustness level, and how to incorporate the compatibility as a constraint with the online QP to ensure recursive feasibility.

References

[1] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[2] A. Bemporad, "Reference governor for constrained nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 3, pp. 415–419, 1998.

[3] K. P. Tee, S. S. Ge, and E. H. Tay, "Barrier lyapunov functions for the control of output-constrained nonlinear systems," *Automatica*, vol. 45, no. 4, pp. 918–927, 2009.

[4] C. P. Bechlioulis and G. A. Rovithakis, "Robust adaptive control of feedback linearizable MIMO nonlinear systems with prescribed performance," *IEEE Transactions on Automatic Control*, vol. 53, no. 9, pp. 2090–2099, 2008.

[5] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 462–467, 2007.

[6] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," in *Proc. IFAC Conf. Anal. Design Hybrid Syst.*, vol. 48, 2015, pp. 54–61.

[7] X. Xu, "Constrained control of input–output linearizable systems using control sharing barrier functions," *Automatica*, vol. 87, pp. 195–201, 2018.

[8] W. S. Cortez, X. Tan, and D. V. Dimarogonas, "A robust, multiple control barrier function framework for input constrained systems," *IEEE Control Systems Letters*, vol. 6, pp. 1742–1747, 2021.

[9] G. Notomista and M. Saveriano, "Safety of dynamical systems with multiple non-convex unsafe sets using control barrier functions," *IEEE Control Systems Letters*, vol. 6, pp. 1136–1141, 2021.

[10] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, 2007.

[11] A. Clark, "Verification and synthesis of control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 6105–6112.

[12] I. Axton, G. Masoumeh, G. S. Ricardo, and E. D. Warren, "On the feasibility and continuity of feedback controllers defined by multiple control barrier functions," in *2022 American Control Conference (ACC)*. IEEE, 2022.

[13] H. K. Khalil, *Nonlinear Systems*. Upper Saddle River, N.J. : Prentice Hall, c2002., 2002.

[14] R. Redheffer, "The theorems of Bony and Brezis on flow-invariant sets," *The American Mathematical Monthly*, vol. 79, no. 7, pp. 740–747, 1972.

[15] M. Jankovic, "Robust control barrier functions for constrained stabilization of nonlinear systems," *Automatica*, vol. 96, pp. 359–367, 2018.

[16] X. Tan, W. S. Cortez, and D. V. Dimarogonas, "High-order barrier functions: Robustness, safety, and performance-critical control," *IEEE Transactions on Automatic Control*, vol. 67, no. 6, pp. 3021–3028, 2021.

[17] X. Tan and D. V. Dimarogonas, "Compatibility checking of multiple control barrier functions for input constrained systems," *arXiv preprint arXiv:2209.02284*, 2022.