A Distributed Simplex Architecture for Multi-Agent Systems

Usama Mehmood[1], Scott D. Stoller[1], Radu Grosu[2],
Shouvik Roy[1], Amol Damare[1], and Scott A. Smolka[1]

[1] Department of Computer Science, Stony Brook University, USA
[2] Department of Computer Engineering, Technische Universität Wien, Austria

Abstract. We present *Distributed Simplex Architecture* (DSA), a new runtime assurance technique that provides safety guarantees for multi-agent systems (MASs). DSA is inspired by the Simplex control architecture of Sha et al., but with some significant differences. The traditional Simplex approach is limited to single-agent systems or a MAS with a centralized control scheme. DSA addresses this limitation by extending the scope of Simplex to include MASs under distributed control. In DSA, each agent has a local instance of traditional Simplex such that the preservation of safety in the local instances implies safety for the entire MAS. We provide a proof of safety for DSA, and present experimental results for several case studies, including flocking with collision avoidance, safe navigation of ground rovers through way-points, and the safe operation of a microgrid.

Keywords: Runtime assurance · Simplex architecture · Control Barrier Functions · Distributed flocking · Reverse switching.

1 Introduction

A multi-agent system (MAS) is a group of autonomous, intelligent agents that work together to solve tasks and carry out missions. MAS applications include the design of power systems and smart-grids [1,2], autonomous control of robotic swarms for monitoring, disaster management, military battle systems, etc. [3], and sensor networks. Many MAS applications are safety-critical. It is therefore paramount that MAS control strategies ensure safety.

In this paper, we present the *Distributed Simplex Architecture* (DSA), a new runtime assurance technique that provides safety guarantees for MASs under distributed control. DSA is inspired by Sha et al.'s Simplex Architecture [4,5], but differs from it in significant aspects. The Simplex Architecture provides runtime assurance of safety by switching control from an unverified (hence potentially unsafe) *advanced controller* (AC) to a verified-safe *baseline controller* (BC), if the action produced by the AC could result in a safety violation in the near future. The switching logic is implemented in a verified *decision module* (DM). The applicability of the traditional Simplex Architecture is limited to systems with a centralized control architecture.

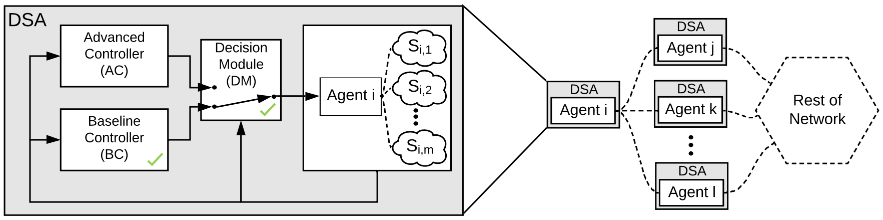


Fig. 1: The DSA for the MAS on the right. All agents in the MAS are homogeneous and operate under DSA, but the figure shows the DSA components for only agent i. The sensed state of agent i's j^{th} neighbor is denoted as $S_{i,j}$. The AC, BC, and DM take as input the state of the agent and its neighbors.

DSA, illustrated in Fig. 1, addresses this limitation by making necessary additions to the traditional Simplex to widen its scope to include MASs. Also, as in [6], it implements *reverse switching* by reverting control back to the AC when it is safe to do so.

In DSA, for each agent, there is a verified-safe BC and a certified switching logic such that if all the agents operate under DSA, then safety of the MAS is guaranteed. The BC and DM along with the AC are distributed and depend only on local information. DSA itself is *distributed* in the sense that it involves one local instance of traditional Simplex per agent such that the conjunction of their respective safety properties yields the desired safety property for the entire MAS. For example, consider our flocking case study, where we want to establish collision-freedom for the entire MAS. This can be accomplished in a distributed manner by showing that each local instance of Simplex, say for agent i, ensures collision-freedom for agent i and its neighboring agents.

DSA allows agents to switch their mode of operation independently. At any given time, some agents may be operating in AC mode while others are operating in BC mode. Our approach to the design of the BC and DM leverages *Control Barrier Functions* (CBFs), which have been used to synthesize safe controllers [7–9], and are closely related to Barrier Certificates used for safety verification of closed dynamical systems [10, 11]. A CBF is a mapping from the state space to a real number, with its zero level-set partitioning the state space into safe and unsafe regions. If certain inequalities on the Lie derivative of the CBF are satisfied, then the corresponding control actions are considered safe (admissible).

In DSA, the BC is designed as an optimal controller with the goal of increasing a utility function based on the Lie derivatives of the CBFs. As CBFs are a measure of the safety of a state, optimizing for control actions with a higher Lie derivative values gives a direct way to make the state safer. The safety of the BC is further guaranteed by constraining the control action to remain in a set of admissible actions that satisfy certain inequalities on the Lie derivatives of the CBFs. CBFs are also used in the design of the switching logic, as they provide an efficient method for checking whether an action could lead to a safety violation during the next time step.

We demonstrate the effectiveness of DSA on several example MASs, including a flock of robots moving coherently while avoiding inter-agent collisions, ground rovers safely navigating through a series of way-points, and safe operation of a microgrid.

2 Background

2.1 Simplex Architecture

The Simplex Control Architecture relies on a verified-safe baseline controller (BC) in conjunction with the verified switching logic of the Decision Module (DM) to guarantee the safety of the plant (Agent i in the Fig. 1) while permitting the use of an unverifiable, high-performance advanced controller (AC).

Let the *admissible* states be those which satisfy all safety constraints and operational limits. Other states are are called *inadmissible*. The goal of the Simplex Architecture is to ensure the system never enters an inadmissible state. The set $\mathcal{R}$ of *recoverable* states is a subset of the admissible states such that the BC, starting from any state in $\mathcal{R}$ guarantees that all future states are also in $\mathcal{R}$. The recoverable set takes into account the inertia of the physical system, giving the BC enough time to preserve safety.

The DM's *forward switching condition* (FSC) evaluates the control action proposed by the AC and decides whether to switch to the BC. A common technique to develop a FSC is to shrink the recoverable region by a margin based on the maximum time derivative of the state and the length of a timestep, and switch to BC if the current state lies outside this smaller set.

2.2 Control Barrier Functions

Control Barrier Functions (CBFs) [12, 13] are an extension of the Barrier Certificates used for safety verification of hybrid systems [10, 11]. CBFs are a class of Lyapunov-like functions used to guarantee safety for nonlinear control systems by assisting in the design of a class of safe controllers that establish the forward-invariance of safe sets [9, 14]. Our presentation of CBFs is based on [13].

Consider a nonlinear affine control system

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

with state $x \in D \subset \mathbb{R}^n$, control input $u \in U$, and functions f and g that are locally Lipschitz. The set $\mathcal{R}$ of recoverable states is defined as the super-level set of a continuously differentiable function $h : D \subset \mathbb{R}^n \to \mathbb{R}$. The recoverable set $\mathcal{R}$ and its boundary $\delta\mathcal{R}$ are given by:

$$\mathcal{R} = \{x \in D \subset \mathbb{R}^n | h(x) \geq 0\} \tag{2}$$

$$\delta\mathcal{R} = \{x \in D \subset \mathbb{R}^n | h(x) = 0\} \tag{3}$$

For all $x \in \mathcal{D}$, if there exists an extended class $\mathcal{K}$ function $\alpha : \mathbb{R} \rightarrow \mathbb{R}$ (strictly increasing and $\alpha(0) = 0$) such that the following condition on the Lie-derivative of h is satisfied:

$$\underset{u \in U}{sup}[L_f h(x) + L_g h(x)u + \alpha(h(x) \geq 0 \tag{4}$$

then the function $h(x)$ is a valid CBF. Condition (4) implies the existence of a control action for all $x \in D$, such that the Lie-derivative of h is bounded from below by $-\alpha(h(x))$. Furthermore, for $x \in \delta\mathcal{R}$, condition (4) reduces to a result for set invariance known as Nagumo's theorem [15,16]. Condition (4) is used to define the set $K(x)$ of control actions that establish the forward invariance of set $\mathcal{R}$; i.e., starting from $x \in \mathcal{R}$, the state will always remain inside $\mathcal{R}$:

$$K(x) = \{u \in U : L_f h(x) + L_g h(x)u + \alpha(h(x)] \geq 0\} \tag{5}$$

Theorem 1. *[13] For the control system given in Eq. (1) and recoverable set $\mathcal{R}$ defined in (2) as the super-level set of some continuously differentiable function $h : \mathbb{R}^n \rightarrow \mathbb{R}$, if h is a control barrier function for all $x \in D$ and $\frac{\delta h}{\delta x} \neq 0$ for all $x \in \delta\mathcal{R}$, then any controller u such that $\forall x \in D : u(x) \in K(x)$ ensures forward-invariance of $\mathcal{R}$.*

Proof. Condition (4) on the Lie-derivative of h reduces, on the boundary of $\mathcal{R}$, to the set invariance condition of Nagumo's theorem: for $x \in \delta\mathcal{R}$, $\dot{h} \geq -\alpha(h(x)) = 0$. Hence, according to Nagumo's theorem [15,16] the set $\mathcal{R}$ is forward-invariant.

3 Distributed Simplex Architecture

This section describes the Distributed Simplex Architecture (DSA). We formally introduce the MAS safety problem and then discuss the main components of DSA, namely, the distributed baseline controller (BC) and the distributed decision module (DM).

We say that an instance of DSA is *symmetric* if every agent uses the same switching condition and baseline controller. Moreover, DSA, or more precisely the MAS it is controlling, is *homogeneous* if every constituent agent is an instance of the same plant model.

Consider a MAS consisting of k homogeneous agents, denoted as $\mathcal{M} = \{1, ..., k\}$, where the nonlinear control affine dynamics for the i^{th} agent are:

$$\dot{x}_i = f(x_i) + g(x)u_i, \tag{6}$$

Here, $x_i \in D \in \mathbb{R}^n$ is the state of agent i and $u_i \in U \subset \mathbb{R}^m$ is its control input. For an agent i, we define the set of its *neighbors* $\mathcal{N}_i \subseteq \mathcal{M}$ as the agents whose state is accessible to the agent i either through sensing or communication. Depending on the application, the set of neighbors could be fixed or vary dynamically. For example, in our flocking case study (Section 4), agent i's neighbors (in a given state) are the agents within a fixed distance r of agent i; we assume agent

i can accurately sense the positions and velocities those agents. We denote the combined state of all the agents in the MAS as the vector $\mathbf{x} = \{x_1^T, x_2^T, ...x_k^T\}^T$ and denote the state of the neighbors of agent i (including agent i) as $x_{\mathcal{N}_i}$. DSA uses discrete-time control: the DM and controllers are evaluated every η seconds. We assume that all agents evaluate DM and controllers simultaneously; this assumption simplifies the analysis but can be relaxed.

Admissible States Set of admissible states $\mathcal{A} \subset \mathbb{R}^{kn}$ consists of all states that satisfy the safety constraints. A constraint C is a function from k-agent MAS states to the reals; $C : D^k \rightarrow \mathbb{R}$. In this paper, we are primarily concerned with binary constraints (between neighboring agents) $C_{ij} : D \times D \rightarrow \mathbb{R}$, and *unary constraints* $C_i : D \rightarrow \mathbb{R}$. Hence, set of admissible states, $\mathcal{A} \subset \mathbb{R}^{kn}$ are the states of MAS $\mathbf{x} \in \mathbb{R}^{kn}$, such that all the unary and binary constraints are satisfied.

Formally, a symmetric instance of DSA aims to solve the following problem. Given a MAS defined as in Eq. (1) and $\mathbf{x}(0) \in \mathcal{A}$, design a BC and DM to be used by all agents such that the MAS remains safe; i.e. $\mathbf{x}(t) \in \mathcal{A}$, $\forall\ t > 0$.

Recoverable States For each agent i, the local admissible set $\mathcal{A}_i \subset \mathbb{R}^n$ is the set of states $x_i \in \mathbb{R}^n$ which satisfy all the unary constraints. The set $\mathcal{S}_i \subset \mathcal{A}_i$ is defined as the super-level set of the CBF $h_i : \mathbb{R}^n \rightarrow \mathbb{R}$, which is designed to ensure forward-invariance of $\mathcal{A}_i$. Similarly, for a pair of neighboring agents i, j where $i \in \mathcal{M}, j \in \mathcal{N}_i$, the pairwise admissible set $\mathcal{A}_{ij} \subset \mathbb{R}^{2n}$ is the set of pairs of states which satisfy all the binary constraints. The set $\mathcal{S}_{ij} \subset \mathcal{A}_{ij}$ is defined as the super-level set of the CBF $h_{ij} : \mathbb{R}^{2n} \rightarrow \mathbb{R}$ designed to ensure forward-invariance of $\mathcal{A}_{ij}$. The recoverable set $\mathcal{R}_{ij} \subset \mathbb{R}^{2n}$, for a pair of neighboring agents i, j where $i \in \mathcal{M}, j \in \mathcal{N}_i$, is defined in terms of $\mathcal{S}_i$, $\mathcal{S}_j$ and $\mathcal{S}_{ij}$.

$$\mathcal{S}_i = \{x_i \in \mathbb{R}^n | h_i(x_i) \geq 0\} \tag{7}$$

$$\mathcal{S}_{ij} = \{(x_i, x_j) \in \mathbb{R}^{2n} | h_{ij}(x_i, x_j) \geq 0\} \tag{8}$$

$$\mathcal{R}_{ij} = (\mathcal{S}_i \times \mathcal{S}_j) \cap \mathcal{S}_{ij} \tag{9}$$

The recoverable set $\mathcal{R} \subset \mathcal{A}$ for the entire MAS is defined as the set of system states in which $(x_i, x_j) \in \mathcal{R}_{ij}$ for every pair of neighboring agents i, j. The CBFs can be computed using sum-of-squares programming [17] or the technique in [18]. An application of these techniques for the synthesis of CBFs for several systems can be found in [13]. Note that if agent i and j's controllers satisfy the following constraints based on the Lie derivatives of h_i, h_j and h_{ij}, similar to the constraints in (5), the pairwise state of agents i and j will remain in $\mathcal{R}_{ij}$ according to Theorem 1.

$$L_f h_i(x_i) + L_g h_i(x_i) u_i + \alpha(h_i(x_i)) \geq 0 \tag{10a}$$

$$L_f h_j(x_j) + L_g h_j(x_j) u_j + \alpha(h_j(x_j)) \geq 0 \tag{10b}$$

$$L_f h_{ij}(x_i, x_j) + L_g h_{ij}(x_i, x_j) \begin{bmatrix} u_i \\ u_j \end{bmatrix} + \alpha(h_{ij}(x_i, x_j)) \geq 0 \tag{10c}$$

Constraint Partitioning Note that the constraints in (10) are linear in the control variable. For ease of notation we write the unary constraints as $A_i u_i \leq b_i$ and the binary constraints as $\left[\begin{smallmatrix} P_{ij} & Q_{ij} \end{smallmatrix}\right]\left[\begin{smallmatrix} u_i \\ u_j \end{smallmatrix}\right] \leq b_{ij}$.

The binary constraint in (10c) is a condition on the control action of a pair of agents. For a centralized MAS, the global controller can pick coordinated actions for agents i and j to ensure the binary constraint (10c) is satisfied. However, for a decentralized MAS, the distributed control of the two agents cannot independently satisfy the binary constraint without running an agreement protocol.

As DSA is a distributed control framework, we solve the problem of the satisfaction of the binary constraint by partitioning the binary constraint into two unary constraints such that the satisfaction of the unary constraints implies the satisfaction of the binary constraint (but not vice versa) [9].

$$\begin{bmatrix} P_{ij} \; Q_{ij} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \end{bmatrix} \leq b_{ij} \;\rightarrow \begin{cases} P_{ij} u_i \leq b_{ij}/2 \\ Q_{ij} u_j \leq b_{ij}/2 \end{cases} \tag{11}$$

The satisfaction of the two unary constraints in (11) by the respective controllers of agents i and j guarantees safety because the binary constraint still holds. Moreover, the equal partitioning ensures that the agents share an equal responsibility to keep the pairwise state safe. The admissible control space for an agent i, denoted by $\mathcal{L}_i$, is an intersection of half-spaces of the hyper-planes defined by the linear constraints.

$$\mathcal{L}_i = \{u_i \in U \mid \forall j \in \mathcal{N}_i : A_i u_i \leq b_i \wedge P_{ij} u_i \leq b_{ij}\} \tag{12}$$

Theorem 2. *Given a MAS indexed by $\mathcal{M}$ and with dynamics in (6), if the controller for each agent $i \in \mathcal{M}$ chooses an action $u_i \in \mathcal{L}_i$, thereby satisfying the Lie-derivative constraints on the respective CBFs, and $\mathbf{x}(0) \in \mathcal{R}$, then the MAS is guaranteed to be safe.*

Proof. If all the agents choose an action from their respective admissible control spaces $\mathcal{L}_i$, then the forward invariance of the set $\mathcal{S}_i$ for all $i \in \mathcal{M}$ and $\mathcal{S}_{ij}$ for all $i \in \mathcal{M}$, $j \in \mathcal{N}_i$ is established by Theorem 1. Therefore, $\mathcal{R}_{ij}$ is forward invariant for all $i \in \mathcal{M}, j \in \mathcal{N}_i$ and consequently $\mathcal{R}$ is forward invariant.

3.1 Baseline Controller

The BC is a distributed controller with the task to keep the state of the agent in the safe region. For an agent i, the control law of the BC depends on its state x_i and the states of its . In our design, the BC considers only the safety-critical aspects, leaving the mission-critical objectives to the AC. Specifically, the BC is designed to move the system toward safer states as quickly as possible. This reduces the width of the necessary "safety margin" between unsafe and recoverable states, allowing a looser FSC (i.e., allowing the AC to stay in control more often).

We design the BC as the solution to the following constrained multi-objective optimization (MOO) problem where the utility function is a weighted sum of objective functions based on the Lie derivatives of the CBFs h_i and h_{ij} introduced above:

$$\begin{aligned} u_i^* = \underset{u_i}{argmax} \quad & \frac{1}{h_i}(L_f h_i + L_g h_i u_i) + \sum_{j \in \mathcal{N}_i} \frac{1}{h_{ij}}(L_f h_{ij} + L_g h_{ij} \begin{bmatrix} u_i \\ 0 \end{bmatrix}) \\ \text{s.t.} \quad & u_i \in \mathcal{L}_i \end{aligned} \tag{13}$$

The bottom component of the column vector in the last term is agent i's prediction for agent j's next control action u_j. Since we consider MASs in which agents are unable to communicate planned control actions, agent i simply predicts that $u_j = 0$. This approach has been shown to work well in prior work on distributed model-predictive control for flocking [19].

Recall that, by definition, the CBFs quantify the degree of safety of the state with respect to the given safety constraints, with larger (positive) values indicating safer states. For a given state, the Lie derivative of a CBF is a linear function in the control action. A positive value of the Lie derivative indicates that the proposed action will lead to a state which has a higher CBF value and therefore is safer.

The solution to the optimization problem in (13) is a control action that maximizes the weighted sum of the Lie derivatives of the CBFs. We note that in a weighted-sum formulation of a MOO problem, it is possible that some objective functions are negative in the optimal solution. We ensure the selected action u_i is safe by constraining u_i to be in the admissible control space $\mathcal{L}_i$, defined in (12).

The weights in the utility function in (13) prioritize certain safety constraints over others. We use state-dependent weights which are the inverses of the CBF functions, thereby giving more weight to maximizing the Lie derivatives of CBFs corresponding to safety constraints that are closer to being violated.

3.2 Decision Module

Each agent's DM implements the switching logic for both forward switching and reverse switching. Control is switched from the AC to the BC if the *forward switching condition* (FSC) is true. Similarly, control is reverted back to the AC (from the BC) if the *reverse switching condition* (RSC) is true. For an agent i, the state of the DM is denoted as $DM_i \in \{AC, BC\}$, with $DM_i = AC$ ($DM_i = BC$) indicating that the advanced (baseline) controller is in control. DSA starts with all agents in the AC mode; i.e., $DM_i(t) = AC$ for all $t \leq 0$ and $i \in \mathcal{M}$; this is justified by the assumption that $\mathbf{x}(0) \in \mathcal{R}$. For $t > 0$, the DM state is given by:

$$DM_i(t) = \begin{cases} AC & \text{if } DM_i(t-1) = BC \text{ and } RSC(x_{\mathcal{N}_i}) \\ BC & \text{if } DM_i(t-1) = AC \text{ and } FSC(x_{\mathcal{N}_i}) \\ DM_i(t-1) & \text{otherwise} \end{cases} \tag{14}$$

where $x_{\mathcal{N}_i}$ is the vector containing the states of agent i and its neighbors.

We derive the switching conditions from the CBFs as follows. To ensure safety, the FSC must be true in a state $x_{\mathcal{N}_i}(t)$ if an unrecoverable state is reachable from $x_{\mathcal{N}_i}(t)$ in one time step η. For a CBF function, in a given state, we define a *worst-case* action to be an action that minimizes the Lie derivative of the CBF. The check for one-step reachability of an unrecoverable state is based on the minimum value of the Lie derivative of the CBFs, which corresponds to the worst-case actions by the agents. Hence, for each CBF h, we define a minimum threshold value $\lambda_h(x_{\mathcal{N}_i})$ equal to the magnitude of the minimum of the Lie derivative of the CBF times η, and we switch to BC if, in the current state, the value of any CBF h is less than $\lambda_h(x_{\mathcal{N}_i})$. This results in a FSC of the following form:

$$FSC(x_{\mathcal{N}_i}) = (h_i < \lambda_{h_i}(x_{\mathcal{N}_i})) \vee (\exists j \in \mathcal{N}_i \mid h_{ij} < \lambda_{h_{ij}}(x_{\mathcal{N}_i})) \tag{15}$$

Thus, the one-step reachability check shrinks the size of the recoverable set by an amount equal to the maximum change that can occur from the current state in one control period, and the switch occurs if the current state is outside this smaller set.

We derive the RSC using a similar approach, except based on an m-time-step reachability check with $m > 1$, in order to prevent frequent switching between AC and BC. The RSC holds if, in the current state, the value of each CBF h is greater than the threshold $m\lambda_h(x_{\mathcal{N}_i})$.

$$RSC(x_{\mathcal{N}_i}) = (h_i > m\lambda_{h_i}(x_{\mathcal{N}_i})) \wedge (\forall j \in \mathcal{N}_i \mid h_{ij} > m\lambda_{h_{ij}}(x_{\mathcal{N}_i})) \tag{16}$$

This definition of the RSC ensures that, when control is switched to AC, the state is safe and the FSC will not hold for at least m time steps.

3.3 Safety Theorem

Our main result is the following safety theorem for DSA.

Theorem 3. *Given an MAS indexed by $\mathcal{M}$ with dynamics in (6), if each agent operates under DSA with the BC as defined in (13), the DM as defined in (14), and $\mathbf{x}(0) \in \mathcal{R}$, then the MAS will remain safe.*

Proof. The proof proceeds by considering both DM states for an arbitrary agent i and establishing that its next state is safe. First, consider an agent i at time t with $DM_i(t) = AC$. As the FSC is false, the one-step reachability check in the FSC ensures that the CBFs for unary and binary safety constraints are strictly positive at the next state $x_i(t+\eta)$, i.e. $h_i(x_i(t+\eta)) > 0$ and $\forall j \in \mathcal{N}_i : h_{ij}(x_i(t+\eta), x_j(t+\eta)) > 0$, hence the next state is recoverable. Next, consider an agent i at time t with $DM_i(t) = BC$. We divide the neighbors of i into two sets based on their DM states: the sets of neighbors in AC mode and BC mode are denoted as $\mathcal{N}_i^{AC}$ and $\mathcal{N}_i^{BC}$, respectively. The agents in BC mode choose their control actions from their corresponding admissible control spaces as defined in Eq. 12. Hence, according to *Theorem* 2, these agents will satisfy unary safety constraints and pairwise safety constraints among themselves. As for the neighbors in AC mode, due to the one-step reachability check in their

FSC, in the state $x_i(t+\eta)$, the pairwise CBFs satisfy $h_{ij}(x_i(t+\eta), x_j(t+\eta)) \geq 0$ for all $j \in \mathcal{N}_i^{AC}$. Hence, $x_i(t+\eta)$ is recoverable for $DM_i(t) = BC$. We have proven that, for any agent i and time step t, if $x_i(t)$ is recoverable, then $x_i(t+\eta)$ is recoverable. By assumption, $\mathbf{x}(0) \in \mathcal{R}$. Therefore, by induction, $\mathbf{x}(t) \in \mathcal{R}$ for $t > 0$.

4 Flocking Case Study

We evaluate DSA on the distributed flocking problem with the goal of preventing inter-agent collisions. Consider an MAS consisting of n robotic agents, indexed by $\mathcal{M} = \{1, \ldots, n\}$ with double integrator dynamics:

$$\begin{bmatrix} \dot{p}_i \\ \dot{v}_i \end{bmatrix} = \begin{bmatrix} 0 & I_{2\times 2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p_i \\ v_i \end{bmatrix} + \begin{bmatrix} 0 \\ I_{2\times 2} \end{bmatrix} a_i \tag{17}$$

where p_i, v_i, $a_i \in \mathbb{R}^2$ are the position, velocity and acceleration of agent $i \in \mathcal{M}$, respectively. The magnitudes of velocities and accelerations are bounded by $\bar{v}$ and $\bar{a}$, respectively. Acceleration a_i is the control input for agent i. As DSA is a discrete-time protocol, the state of the DM and the a_i's are updated every η seconds. The *state* of an agent i is denoted by the vector $s_i = [p_i^T v_i^T]^T$. The *state* of the entire flock at time t, is denoted by the vector $\mathbf{s}(t) = [\mathbf{p}(t)^T\ \mathbf{v}(t)^T]^T \in \mathbb{R}^{4n}$, where $\mathbf{p}(t) = [p_1^T(t) \cdots p_n^T(t)]^T$ and $\mathbf{v}(t) = [v_1^T(t) \cdots v_n^T(t)]^T$ are the vectors respectively denoting the positions and velocities of the flock at time t.

We assume that an agent can accurately sense the positions and velocities of nearby agents within a fixed distance r. The set of the *spatial neighbors* of agent i is defined as $\mathcal{N}_i(\mathbf{p}) = \{j \in \mathcal{M} \mid j \neq i \land \|p_i - p_j\| < r\}$, where $\|\cdot\|$ denotes the Euclidean norm. For ease of notation, we sometimes use $\mathbf{s}$ and $\mathbf{s}_i$ to refer to the state variables $\mathbf{s}(t)$ and $\mathbf{s}_i(t)$, respectively, without the time index.

The MAS is characterized by a set of operational constraints which include physical limits and safety properties. States that satisfy the operational constraints are called *admissible*, and are denoted by the set $\mathcal{A} \in \mathbb{R}^{4n}$. The desired safety property is that no pair of agents is in a "state of collision". A pair of agents is considered to be in a *state of collision* if the Euclidean distance between them is less than a threshold distance $d_{min} \in \mathbb{R}^+$, resulting in a binary safety constraint of the form: $\|p_i - p_j\| - d_{min} \geq 0\ \forall\ i \in \mathcal{M}, j \in \mathcal{N}_i$. Similarly, a state $\mathbf{s}$ is *recoverable* if all pairs of agents can brake (de-accelerate) relative to each other without colliding. Otherwise, the state $\mathbf{s}$ is considered *unrecoverable*.

4.1 Synthesis of Control Barrier Function

Let $\mathcal{R}_{ij} \subset \mathbb{R}^8$ be the set of recoverable states for a pair of agents $i, j \in \mathcal{M}$. The flock-wide set of recoverable states, denoted by $\mathcal{R} \subset \mathbb{R}^{4n}$, is defined in terms of $\mathcal{R}_{ij}$. As in [14], the set $\mathcal{R}_{ij}$ is defined as the super-level set of a pairwise CBF $h_{ij} : \mathbb{R}^8 \to \mathbb{R}$: $\mathcal{R}_{ij} = \{s_i, s_j \mid h_{ij}(s_i, s_j) \geq 0\}$. The flock-wide set of recoverable states $\mathcal{R} \subset \mathcal{A}$ is defined as the set of system states in which $(s_i, s_j) \in \mathcal{R}_{ij}$ for every pair of neighboring agents i, j.

In accordance with [14], the function $h_{ij}(s_i, s_j)$ is based on a safety constraint over a pair of agents $i, j \in \mathcal{M}$. The safety constraint ensures that for any pair of agents, the maximum braking force can always keep the agents at a distance greater than d_{min} from each other. As introduced earlier, d_{min} is the threshold distance that defines a collision. Considering that the tangential component of the relative velocity, denoted by Δv, causes a collision, the constraint regulates Δv by application of maximum acceleration to reduce Δv to zero. Hence, the safety constraint can be represented as the following condition on the inter-agent distance $\|\Delta \mathbf{p}_{ij}\| = \|p_i - p_j\|$, the braking distance $(\Delta v)^2/4\bar{a}$, and the safety threshold distance d_{min}:

$$\|\Delta \mathbf{p}_{ij}\| - \frac{(\Delta v)^2}{4\bar{a}} \geq d_{min} \tag{18}$$

$$h_{ij}(s_i, s_j) = \sqrt{4\bar{a}(\|\Delta \mathbf{p}_{ij}\| - d_{min})} - \Delta v \geq 0 \tag{19}$$

The braking distance is the distance covered while the relative speed reduces from Δv to zero under a deceleration of $2\bar{a}$. The constraint in Eq. (18) is rearranged to get the CBF h_{ij} given in Eq. (19).

Combining (19) and (10c), we arrive at the linear constraint on the accelerations for agents i and j, which constrains the Lie derivative of the CBF in (19) to be greater than $-\alpha(h_{ij})$. We set $\alpha(h_{ij}) = \gamma h_{ij}^3$, as in [14], resulting in the following constraint on the accelerations of agents i, j:

$$\begin{aligned} \frac{\Delta \mathbf{p}_{ij}^T(\Delta \mathbf{a}_{ij})}{\|\Delta \mathbf{p}_{ij}\|} - \frac{(\Delta \mathbf{v}_{ij}^T \Delta \mathbf{p}_{ij})^2}{\|\Delta \mathbf{p}_{ij}\|^3} + \frac{\|\Delta \mathbf{v}_{ij}\|^2}{\|\Delta \mathbf{p}_{ij}\|} \\ + \frac{2\bar{a}\Delta \mathbf{v}_{ij}^T \Delta \mathbf{p}_{ij}}{\|\Delta \mathbf{p}_{ij}\| \sqrt{4\bar{a}(\|\Delta \mathbf{p}_{ij}\| - d_{min})}} \geq -\gamma h_{ij}^3 \end{aligned} \tag{20}$$

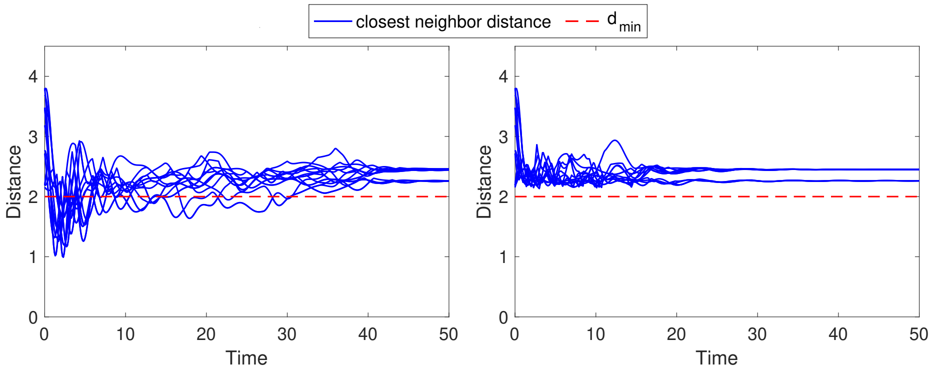
where the left-hand side is the Lie derivative of the CBF h_{ij} and $\Delta \mathbf{p}_{ij} = p_i - p_j$, $\Delta \mathbf{v}_{ij} = v_i - v_j$, and $\Delta \mathbf{a}_{ij} = a_i - a_j$ are the vectors representing the relative position, the relative velocity, and the relative acceleration of agents i and j, respectively. We further note that the binary constraint (20) can be represented as $\left[\begin{smallmatrix} P_{ij} & Q_{ij} \end{smallmatrix}\right]\left[\begin{smallmatrix} a_i \\ a_j \end{smallmatrix}\right] \leq b_{ij}$, and hence it can be split into two unary constraints ($P_{ij}u_i \leq b_{ij}/2$ and $Q_{ij}u_j \leq b_{ij}/2$), following the convention in (11). The set of safe accelerations for an agent $i \in \mathcal{A}$, denoted by $\mathcal{K}_i(\mathbf{s}_i) \subset \mathbb{R}^2$, is defined as the intersection of the half-planes defined by the Lie-derivative-based constraints, where each neighboring agent contributes a single constraint:

$$\mathcal{K}_i(\mathbf{s}_i) = \left\{a_i \in \mathbb{R}^2 \mid P_{ij}u_i \leq b_{ij}/2,\ \forall j \in \mathcal{N}_i\right\} \tag{21}$$

With the CBF for collision-free flocking defined in (19) and the admissible control space defined in (21), the BC, RSC, and FSC follow from (13), (15), and (16), respectively.

4.2 Advanced Controller

We use the Reynolds flocking model [20] as the AC. In the Reynolds model, the acceleration a_i for each agent is a weighted sum of three acceleration terms,



(a) Reynolds Model (b) Reynolds Model with DSA

Fig. 2: Results for a flock of size 15, with and without DSA.

based on simple rules of interaction with the neighboring agents: *separation* (move away from your close neighbors), *cohesion* (move towards the centroid of your neighbors), and *alignment* (match your velocity with the average velocity of your neighbors). The acceleration for agent i is $a_i = w_s a_i^s + w_c a_i^c + w_{al} a_i^{al}$, where $w_s, w_c, w_{al} \in \mathbb{R}^+$ are the scalar weights and $a_i^s, a_i^c, a_i^{al} \in \mathbb{R}^2$ are the acceleration terms corresponding to separation, cohesion, and alignment, respectively. We note that the Reynolds model does not guarantee collision avoidance. Nevertheless, when the flock stabilizes, the average distance to the closest neighbors are determined by the choice of the weights of the interaction terms.

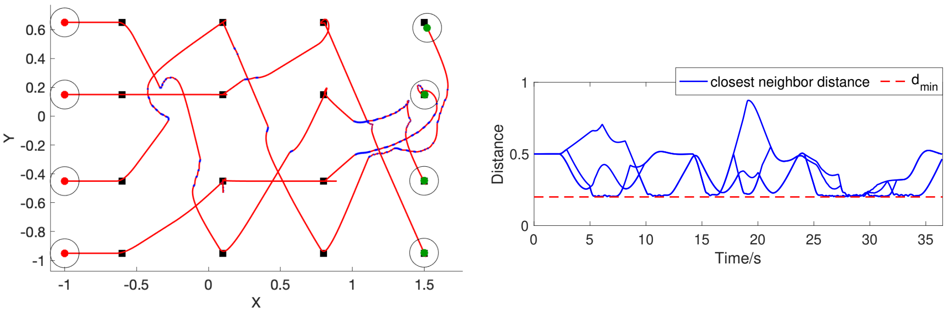
4.3 Experimental Results

The number of agents in the MAS is $n = 15$. The other parameters used in the experiments are $r = 4$, $\bar{a} = 5$, $\bar{v} = 2.5$, $d_{min} = 2$, and $\eta = 0.1$s. The length of the simulations is 50 seconds. The initial positions and the initial velocities are uniformly sampled from $[-10, 10]^2$ and $[-1, 1]^2$, respectively, and we ensure that the initial state is recoverable. The weights of the Reynolds' model terms are picked experimentally to ensure that no pair of agents are in a state of collision in the steady state. They are set to $w_s = 3$, $w_c = 1.5$, and $w_{al} = 0.5$.

We performed two simulations, starting from the same initial configuration. In the first simulation, Reynolds model is used to control all agents for the duration of the simulation. In the second simulation, Reynolds model is wrapped with a verified safe BC and DM designed using DSA. Videos of both simulations are available online.[3]

To recall, the safety property is that all pairs of agents maintain a distance greater than d_{min} from each other. Fig. 2 plots, for the duration of the simulations, the distance between the agents and their closest neighbors, i.e., $\min_{j \in \mathcal{N}_i} \|p_i - p_j\|$. As evident from Fig. 2(a), Reynolds model results in multiple safety violations before the flock stabilizes at around 40 seconds. In contrast, as shown

[3] `https://streamable.com/zn2bl5`
`https://streamable.com/hetraw`



(a) The trajectories of the agents passing through the way-points. Red/blue segments indicate AC/BC modes.

(b) Distance to the closest neighbor for all agents.

Fig. 3: Experimental results for the way-point control study.

in Fig. 2(b), DSA preserves safety, maintaining a separation greater than d_{min} between all agents. DSA has an additional benefit of stabilizing the flock much earlier (around 18 seconds). We further note that the average time the agents spent in BC mode is only 2.47 percent of the total duration of the simulation, indicating that DSA is largely non-invasive.

5 Way-point Control Case Study

This section describes the problem setup and experimental results for the way-point (WP) control case study. For this study, the model of the agents is the same as the one used for the flocking case study, given in Eq. (17). The experimental setup is shown in Fig. 3, where the agents initially positioned on the left-hand side are to sequentially navigate through a series of WPs while maintaining a safe distance from each other. The WPs are represented by the black squares.

The CBF, BC and DM are same as those defined for the flocking problem; see Section 4. The AC is a rule-based controller where each agent accelerates towards its next WP (ignoring the other agents) until the final WP is reached. Agents are assigned one WP from each column such that they are on a collision course if they follow the AC's commands.

5.1 Experimental Results

The number of agents used in the experiment is four as is the number of WPs an agent is required to visit. Initially, the agents are at rest with their positions represented by the red dots in Fig. 3(a). The final configuration is shown in green. The duration of the simulation is 37 seconds. The other parameters used in the experiments are $r = 1.0$, $\bar{a} = 0.8$, $\bar{v} = 0.2$, $d_{min} = 0.2$, and $\eta = 0.05$s. The trajectories of the agents are given in Fig. 3(a), where the segments in blue indicate when the BC is in control. Fig. 3(b) plots the smallest inter-agent distances, indicating that the agents maintain a safe distance from one another. A video of the simulation is available online.[4]

6 Microgrid Case Study

With an increasing prevalence of distributed energy resources (DERs) such as wind and solar power, electrification using microgrids (MGs) has witnessed unprecedented growth. Unlike traditional power systems, MG DERs do not have rotating components such as turbines. The lack of rotating components can lead to low inertia, making MGs susceptible to oscillations resulting from transient disturbances [21]. Ensuring the safe operation of an MG is thus a challenging problem. In this case study, we demonstrate the effectiveness of DSA in maintaining MG voltage levels within safe limits.

The MG we consider is a network of n droop-controlled inverters, indexed by $\mathcal{M} = \{1, \ldots, n\}$. The dynamics of each inverter is modeled as [21–24]:

$$\dot{\theta}_i = \omega_i \tag{22a}$$

$$\dot{\omega}_i = \omega_i^0 - \omega_i + \lambda_i^p (P_i^{set} - P_i) \tag{22b}$$

$$\dot{v}_i = v_i^0 - v_i + \lambda_i^q (Q_i^{set} - Q_i) \tag{22c}$$

where θ_i, ω_i, and v_i are respectively the phase angle, frequency, and voltage of inverter i, $i \in \mathcal{M}$. The state vector for the MG is denoted by $\mathbf{s} = [\theta^T \ \omega^T \ v^T]^T \in \mathbb{R}^{3n}$, where θ, ω, and v are respectively vectors representing the voltage phase angle, frequency, and voltage at each node of the MG. A pair of inverters are considered neighbors if they are connected by a transmission line. Also, λ_i^p and λ_i^q are droop coefficients of “active power vs frequency” and “reactive power vs voltage” droop controllers, respectively. Finally, ω_i^0 and v_i^0 are the nominal frequency and voltage values.

P_i and Q_i are the active and reactive powers injected by inverter i into the system:

$$\begin{aligned} P_i &= v_i \sum_{k \in \mathcal{N}_i} v_k (G_{i,k} \cos\theta_{i,k} + B_{i,k} \sin\theta_{i,k}) \\ Q_i &= v_i \sum_{k \in \mathcal{N}_i} v_k (G_{i,k} \sin\theta_{i,k} - B_{i,k} \cos\theta_{i,k}) \end{aligned} \tag{23}$$

where $\theta_{i,k} = \theta_i - \theta_k$, and $\mathcal{N}_i \subseteq \mathcal{M}$ is the set of neighbors. $G_{i,k}, B_{i,k}$ are respectively conductance and susceptance values of the transmission line connecting inverters i and k.

P_i^{set} and Q_i^{set} are the active power and reactive power setpoints. The inverters have the ability to change their respective power setpoints according to the MG's operating conditions. This is modeled as:

$$P_i^{set} = P_i^0 + u_i^p, \ Q_i^{set} = Q_i^0 + u_i^q \tag{24}$$

where P_i^0 and Q_i^0 are the setpoints for the nominal operating condition, and u_i^p and u_i^q are control inputs.

[4] https://streamable.com/e9rnqd

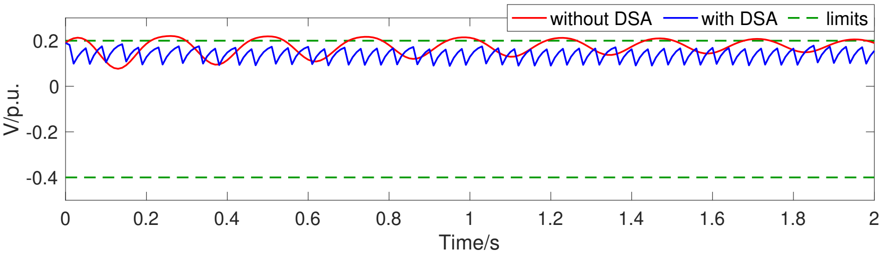


Fig. 4: Voltage graph at node 4 of the MG network.

6.1 Synthesis of Control Barrier Function

The safety property for the MG network is a set of unary constraints restricting the voltages at each node to remain within safe limits. The recoverable set $\mathcal{R}_i \subset \mathbb{R}^3$ for inverter i is defined as the super-level set of a CBF $h_i : \mathbb{R}^3 \to \mathbb{R}$. We follow the SOS-optimization technique given in [24] to synthesize the CBF.

Since the power flow equations (22) are nonlinear, we apply a third-order Taylor series expansion to approximate the dynamics in polynomial form. We then follow the three-step process given in [24] to obtain the CBF for each MG node. We then calculate the admissible control space according to (12), and the BC, FSC, and RSC follow from (13), (15), and (16), respectively.

6.2 Advanced Controller

The AC sets the active/reactive power setpoints to their nominal values. Thus, the AC does not limit voltage and frequency magnitudes but is only concerned with stabilizing frequency and voltage magnitudes to their nominal values.

6.3 Experimental Results

We consider a 6-bus MG [24]. Disconnecting the MG from the main utility, we replace bus 0 with a droop-controlled inverter (Eq. (22)), with inverters also placed on buses 1, 4 and 5. Bus 0 is the reference bus for the phase angle. Nominal values of voltage and frequency, as well as the active/reactive power set-points, were obtained by solving the steady-state power-flow equations given in Eq. (23); these were then used to shift the equilibrium point to the origin. Droop coefficients λ_i^p and λ_i^q were set to 2.43 rad/s/p.u. and 0.20 p.u./p.u., and τ_i was set to 0.5 s. Loads are modeled as constant power loads, and a Kron-reduced network [25] with only the inverter nodes was used for analysis. The unsafe set is defined in terms of the shifted (around the 0 p.u.) nodal voltage magnitudes as follows: $v_i < -0.4\,\text{p.u.} \quad \text{or} \quad v_i > 0.2\,\text{p.u.}$

The duration of the simulation is two seconds. Our results show that with DSA, the voltages at all the nodes remain within the safe limits throughout the simulation. Fig. 4 gives the voltage plot at node 4. When the MG is operating under the control of DSA, forward and reverse switching cause sudden changes in the voltage. As the voltage approaches the upper limit, a switch from AC to BC occurs. Subsequently, the BC reduces the voltage inducing a reverse switch. The voltage profile at the other nodes is similar.

7 Related Work

The original Simplex architecture [4,26] was developed for a systems comprising a single controller and a single (non-distributed) plant. With DSA, we extend the scope of Simplex to MASs under distributed control. RTA [27,28] is a runtime assurance technique that can be applied to component-based systems. In this case, however, each RTA wrapper (i.e., each Simplex-like instance) independently ensures a local safety property of a component. For example, in [27], RTA instances for an inner-loop controller and a guidance system are uncoordinated and operate independently. In contrast, in DSA, each agent takes the states of neighboring agents into account when making control decisions, in order to ensure that pairwise safety constraints are satisfied.

A runtime verification framework for dynamically adaptive multi-agent systems (DAMS-RV) is proposed in [29]. DAMS-RV is activated every time the system adapts to a change in the system itself or its environment. It takes a feedback loop- and model-based approach to verifying dynamic agent collaboration. However this method relies on a *monitoring* phase to observe and identify changes that occur in agent collaboration so that verification can be carried out on the system operating in new contexts. DSA does not require such intermediary supervision. In [30], a dynamic policy model that can be used to express constraints on agent behavior is presented. These constraints limit agent autonomy to lie within well-defined boundaries. Constraint specifications are kept simple by allowing the policy designer to decompose a specification into components and define the overall policy as a composition of these small units. In contrast, DSA uses CBFs to compute the requisite safety regions.

CBF-based methodologies [9,13,14,31] have been widely used for MAS runtime safety assurance. In [9,14], a formal framework for collision avoidance in multi-robot systems is presented. CBFs are used to design a wrapper around an AC that guarantees forward invariance of a safe set. The wrapper solves an optimization problem involving the Lie derivative of the CBF to compute minimal changes to the advanced controller's output needed to ensure safety.

8 Conclusion

We have presented Distributed Simplex Architecture, a runtime assurance technique for the safety of multi-agent systems. DSA is distributed in the sense that it involves one local instance of traditional Simplex per agent such that the conjunction of their respective safety properties yields the desired safety property for the entire MAS. Moreover, an agent's switching logic depends only on its own state and that of neighboring agents. We demonstrated the effectiveness of DSA by successfully applying it to flocking, way-point visiting, and microgrid control. As future work, we plan to apply DSA to non-homogenous MASs and implement it on a physical platform.

References

1. M. Nasir, Z. Jin, H. A. Khan, N. A. Zaffar, J. C. Vasquez, and J. M. Guerrero, “A decentralized control architecture applied to DC nanogrid clusters for rural electrification in developing regions,” *IEEE Transactions on Power Electronics*, vol. 34, no. 2, pp. 1773–1785, 2019.
2. Z. Boussaada, O. Curea, H. Camblong, N. Bellaaj Mrabet, and A. Hacala, “Multi-agent systems for the dependability and safety of microgrids,” *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 2016.
3. A. Tahir, J. Böling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila, “Swarms of unmanned aerial vehicles — a survey,” *Journal of Industrial Information Integration*, vol. 16, p. 100106, 2019.
4. D. Seto and L. Sha, “A case study on analytical analysis of the inverted pendulum real-time control system,” Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-99-TR-023, 1999. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13511
5. L. Sha, “Using simplicity to control complexity,” *IEEE Software*, vol. 18, no. 4, pp. 20–28, 2001.
6. D. Phan, R. Grosu, N. Jansen, N. Paoletti, S. A. Smolka, and S. D. Stoller, “Neural simplex architecture,” in *NASA Formal Methods Symposium (NFM 2020)*, 2020.
7. T. Gurriet, A. Singletary, J. Reher, L. Ciarletta, E. Feron, and A. Ames, “Towards a framework for realizable safety critical control through active set invariance,” in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, 2018, pp. 98–106.
8. M. Egerstedt, J. N. Pauli, G. Notomista, and S. Hutchinson, “Robot ecology: Constraint-based control design for long duration autonomy,” *Annual Reviews in Control*, vol. 46, pp. 1 – 7, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S136757881830141X
9. L. Wang, A. D. Ames, and M. Egerstedt, “Safety barrier certificates for heterogeneous multi-robot systems,” in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 5213–5218.
10. S. Prajna and A. Jadbabaie, “Safety verification of hybrid systems using barrier certificates,” in *Hybrid Systems: Computation and Control, 7th International Workshop, HSCC 2004, Philadelphia, PA, USA, March 25-27, 2004, Proceedings*, ser. Lecture Notes in Computer Science, R. Alur and G. J. Pappas, Eds., vol. 2993. Springer, 2004, pp. 477–492.
11. S. Prajna, “Barrier certificates for nonlinear model validation,” *Autom.*, vol. 42, no. 1, pp. 117–126, 2006.
12. P. Wieland and F. Allgöwer, “Constructive safety using control barrier functions,” *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 462 – 467, 2007, 7th IFAC Symposium on Nonlinear Control Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016355690
13. A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, “Control barrier functions: Theory and applications,” in *18th European Control Conference, ECC 2019, Naples, Italy, June 25-28, 2019*. IEEE, 2019, pp. 3420–3431.
14. U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, “Control barrier certificates for safe swarm behavior,” in *ADHS*, ser. IFAC-PapersOnLine, M. Egerstedt and Y. Wardi, Eds., vol. 48, no. 27. Elsevier, 2015, pp. 68–73.

15. F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*, 1st ed. Birkhäuser Basel, 2007.
16. F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747 – 1767, 1999.
17. L. Wang, D. Han, and M. Egerstedt, "Permissive barrier certificates for safe stabilization using sum-of-squares," in *2018 Annual American Control Conference, ACC 2018, Milwaukee, WI, USA, June 27-29, 2018*. IEEE, 2018, pp. 585–590.
18. E. Squires, P. Pierpaoli, and M. Egerstedt, "Constructive barrier certificates with applications to fixed-wing aircraft collision avoidance," in *IEEE Conference on Control Technology and Applications, CCTA 2018, Copenhagen, Denmark, August 21-24, 2018*. IEEE, 2018, pp. 1656–1661.
19. U. Mehmood, N. Paoletti, D. Phan, R. Grosu, S. Lin, S. D. Stoller, A. Tiwari, J. Yang, and S. A. Smolka, "Declarative vs rule-based control for flocking dynamics," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018.
20. C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 25–34, Aug. 1987. [Online]. Available: http://doi.acm.org/10.1145/37402.37406
21. N. Pogaku, M. Prodanovic, and T. C. Green, "Modeling, analysis and testing of autonomous operation of an inverter-based microgrid," *IEEE Transactions on Power Electronics*, vol. 22, no. 2, pp. 613–625, 2007.
22. J. Schiffer, R. Ortega, A. Astolfi, J. Raisch, and T. Sezi, "Conditions for stability of droop-controlled inverter-based microgrids," *Automatica*, vol. 50, no. 10, pp. 2457 – 2469, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109814003100
23. E. A. A. Coelho, P. C. Cortizo, and P. F. D. Garcia, "Small-signal stability for parallel-connected inverters in stand-alone ac supply systems," *IEEE Transactions on Industry Applications*, vol. 38, no. 2, pp. 533–542, 2002.
24. S. Kundu, S. Geng, S. P. Nandanoori, I. A. Hiskens, and K. Kalsi, "Distributed barrier certificates for safe operation of inverter-based microgrids," in *2019 American Control Conference (ACC)*, 2019, pp. 1042–1047.
25. P. Kundur, N. Balu, and M. Lauby, *Power System Stability and Control*, ser. EPRI power system engineering series. McGraw-Hill Education, 1994. [Online]. Available: https://books.google.com.pk/books?id=2cbvyf8Ly4AC
26. D. Seto, B. Krogh, L. Sha, and A. Chutinan, "The simplex architecture for safe online control system upgrades," in *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*, vol. 6, 1998, pp. 3504–3508.
27. M. Aiello, J. Berryman, J. Grohs, and J. Schierman, *Run-Time Assurance for Advanced Flight-Critical Control Systems*, 2010.
28. J. Schierman, D. Ward, B. Dutoi, A. Aiello, J. Berryman, M. DeVore, W. Storm, and J. Wadley, *Run-Time Verification and Validation for Safety-Critical Flight Control Systems*, 2012.
29. Y. J. Lim, G. Hong, D. Shin, E. Jee, and D.-H. Bae, "A runtime verification framework for dynamically adaptive multi-agent systems," in *2016 International Conference on Big Data and Smart Computing (BigComp)*, 2016, pp. 509–512.
30. H. Alotaibi and H. Zedan, "Runtime verification of safety properties in multi-agents systems," in *2010 10th International Conference on Intelligent Systems Design and Applications*, 2010, pp. 356–362.
31. A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.