A Barrier Certificate-Based Simplex Architecture with Application to Microgrids

Amol Damare, Shouvik Roy, Scott A. Smolka, and Scott D. Stoller(✉)

Stony Brook University, Stony Brook, NY, USA
{adamare,shroy,sas,stoller}@cs.stonybrook.edu

Abstract. We present *Barrier-based Simplex* (Bb-Simplex), a new, provably correct design for runtime assurance of continuous dynamical systems. Bb-Simplex is centered around the Simplex Control Architecture, which consists of a high-performance *advanced controller* which is not guaranteed to maintain safety of the plant, a verified-safe *baseline controller*, and a *decision module* that switches control of the plant between the two controllers to ensure safety without sacrificing performance. In Bb-Simplex, *Barrier certificates* are used to prove that the baseline controller ensures safety. Furthermore, Bb-Simplex features a new automated method for deriving, from the barrier certificate, the conditions for switching between the controllers. Our method is based on the Taylor expansion of the barrier certificate and yields computationally inexpensive switching conditions.

We consider a significant application of Bb-Simplex to a microgrid featuring an advanced controller in the form of a neural network trained using reinforcement learning. The microgrid is modeled in RTDS, an industry-standard high-fidelity, real-time power systems simulator. Our results demonstrate that Bb-Simplex can automatically derive switching conditions for complex systems, the switching conditions are not overly conservative, and Bb-Simplex ensures safety even in the presence of adversarial attacks on the neural controller.

1 Introduction

Barrier certificates (BaCs) [26,27] are a powerful method for verifying the safety of continuous dynamical systems without explicitly computing the set of reachable states. A BaC is a function of the state satisfying a set of inequalities on the value of the function and value of its time derivative along the dynamic flows of the system. Intuitively, the zero-level-set of a BaC forms a "barrier" between the reachable states and unsafe states. Existence of a BaC assures that starting from a state where the BaC is positive, safety is forever maintained [6,26,27]. Moreover, there are automated methods to synthesize BaCs, e.g., [13,31,34,38].

Proving safety of plants with complex controllers is difficult with any formal verification technique, including barrier certificates. As we now show, however, BaCs can play a crucial role in applying the well-established Simplex Control Architecture [29,30] to provide provably correct runtime safety assurance for systems with complex controllers.

T. Dang and V. Stolz (Eds.): RV 2022, LNCS 13498, pp. 105–123, 2022.
https://doi.org/10.1007/978-3-031-17196-3_6

We present *Barrier-based Simplex* (Bb-Simplex), a new, provably correct design for runtime assurance of continuous dynamical systems. Bb-Simplex is centered around the Simplex Control Architecture, which consists of a high-performance *advanced controller* (AC) that is not guaranteed to maintain safety of the plant, a verified-safe *baseline controller* (BC), and a *decision module* that switches control of the plant between the two controllers to ensure safety without sacrificing performance. In Bb-Simplex, *Barrier certificates* are used to prove that the baseline controller ensures safety. Furthermore, Bb-Simplex features a new scalable (relative to existing methods that require reachability analysis, e.g., [4,5,11]) and automated method for deriving, from the BaC, the conditions for switching between the controllers. Our method is based on the Taylor expansion of the BaC and yields computationally inexpensive switching conditions.

We consider a significant application of Bb-Simplex, namely *microgrid control*. A *microgrid* is an integrated energy system comprising distributed energy resources and multiple energy loads operating as a single controllable entity in parallel to, or islanded from, the existing power grid [33]. The microgrid we consider features an advanced controller (for voltage control) in the form of a neural network trained using reinforcement learning. For this purpose, we use Bb-Simplex in conjunction with the *Neural Simplex Architecture* (NSA) [24], where the AC is an AI-based *neural controller* (NC). NSA also includes an *adaptation module* (AM) for online retraining of the NC while the BC is in control.

The microgrid we consider is modeled in RTDS, an industry-standard high-fidelity, real-time power systems simulator. Our results demonstrate that Bb-Simplex can automatically derive switching conditions for complex systems, the switching conditions are not overly conservative, and Bb-Simplex ensures safety even in the presence of adversarial attacks on the neural controller. Please refer to [9] for a more in-depth exploration of our methodology and experiments.

Architectural Overview of Bb-Simplex. Fig. 1 shows the overall architecture of the combined Barrier-based Neural Simplex Architecture. The green part of the figure depicts our design methodology; the blue part illustrates NSA. Given the BC, the required safety properties, and a dynamic model of the plant, our methodology generates a BaC and then derives the switching condition from it. The reinforcement learning module learns a high-performance NC based on the performance objectives encoded in the reward function.

The structure of the rest of the paper is the following. Section 2 provides background material on barrier certificates. Section 3 features our new approach for deriving switching conditions from barrier certificates. Section 4 introduces our Microgrid case study and the associated controllers used for microgrid control. Section 5 presents the results of our microgrid case study. Section 6 discusses related work. Section 7 offers our concluding remarks.

2 Preliminaries

We use Barrier Certificates (BaCs) to prove that the BC ensures safety. We implemented two automated methods for BaC synthesis from the literature.

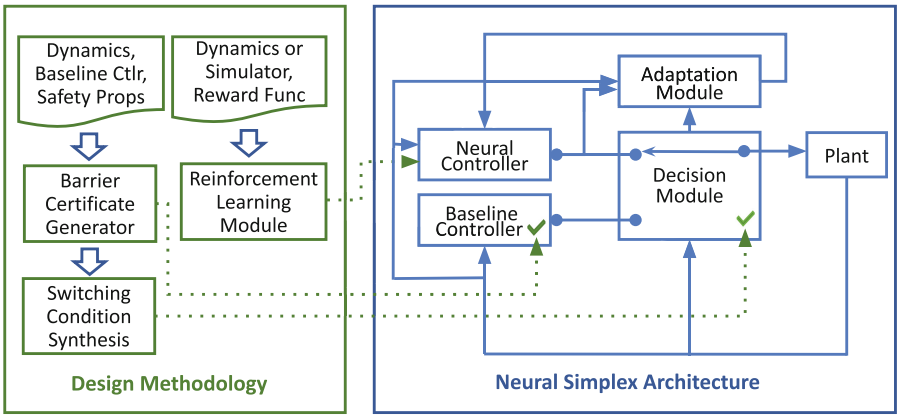

Fig. 1. Overview of the barrier certificate-based neural simplex architecture (Color figure online)

As discussed next, one of the methods is based on sum-of-squares optimization (SOS) and the other uses deep learning. Our design methodology for computing switching conditions (see Sect. 3) requires a BaC, but is independent of how the BaC is obtained.

BaC Synthesis Using SOS Optimization. This method first derives a Lyapunov function V for the system using the expanding interior-point algorithm in [3]. It then uses the SOS-based algorithm in [34] to obtain a BaC from V. Note that the largest super-level set of a Lyapunov function within a safety region is a BaC. The algorithm in [13,34] computes a larger BaC by starting with that sub-level set and then expanding it, by allowing it to take shapes other than that of a sub-level set of the Lyapunov function. This method involves a search of Lyapunov functions and BaCs of various degrees by choosing different candidate polynomials and parameters of the SOS problem. It is limited to systems with polynomial dynamics. In some cases, non-polynomial dynamics can be recast as polynomial using, e.g., the techniques in [3].

BaC Synthesis Using Deep Learning. We also implemented *SyntheBC* [39], which uses deep learning to synthesize a BaC. First, training samples obtained by sampling different areas of the state space are used to train a feedforward ReLU neural network with two hidden layers as a candidate BaC. Second, the validity of this candidate BaC must be verified. The NN's structure allows the problem of checking whether the NN satisfies the defining conditions of a BaC to be transformed into mixed-integer linear programming (MILP) and mixed-integer quadratically-constrained programming (MIQCP) problems, which we solve using the Gurobi optimizer. If the verification fails, the Gurobi optimizer provides counter-examples which can be used to guide retraining of the NN. In this way, the training and verification steps can be iterated as needed.

3 Deriving the Switching Condition

We employ our novel methodology to derive the switching logic from the BaC. The Decision Module (DM) implements this switching logic for both forward and reverse switching. When the forward-switching condition (FSC) is true, control is switched from the NC to the BC; likewise, when the reverse-switching condition (RSC) is true, control is switched from the BC to the NC. The success of our approach rests on solving the complex problems discussed in this section to derive an FSC. Consider a continuous dynamical system of the form:

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

where $x \in \mathbb{R}^k$ is the state of the plant at time t and $u \in \Omega$ is the control input provided to the plant at time t. The set of all valid control actions is denoted by Ω. The set of *unsafe states* is denoted by $\mathcal{U}$. Let $x_{lb}, x_{ub} \in \mathbb{R}^k$ be *operational bounds* on the ranges of state variables, reflecting physical limits and simple safety requirements.

The set $\mathcal{A}$ of *admissible states* is given by: $\mathcal{A} = \{x : x_{lb} \leq x \leq x_{ub}\}$. A state of the plant is *recoverable* if the BC can take over in that state and keep the plant invariably safe. For a given BC, we denote the *recoverable region* by $\mathcal{R}$. Note that $\mathcal{U}$ and $\mathcal{R}$ are disjoint. The safety of such a system can be verified using a BaC $h(x) : \mathbb{R}^k \rightarrow \mathbb{R}$ of the following form [13,26,27,34]:

$$\begin{aligned} h(x) &\geq 0, \quad \forall x \in \mathbb{R}^k \setminus \mathcal{U} \\ h(x) &< 0, \quad \forall x \in \mathcal{U} \\ (\nabla_x h)^T f(x, u) + \sigma(h(x)) &\geq 0, \quad \forall x \in \mathbb{R}^k \end{aligned} \tag{2}$$

where $\sigma(.)$ is an extended class-$\mathcal{K}$ function. The BaC is negative over the unsafe region and non-negative otherwise. $\nabla_x h$ is the gradient of h w.r.t x and the expression $(\nabla_x h)^T f(x, u)$ is the time derivative of h. The zero-super-level set of a BaC h is $\mathcal{Z}(h) = \{x : h(x) > 0\}$. In [34], the invariance of this set is used to show $\mathcal{Z}(h) \subseteq \mathcal{R}$.

Let η denote the control period a.k.a. time step. Let $\hat{h}(x, u, \delta)$ denote the n^{th}-degree Taylor approximation of BaC h's value after time δ, if control action u is taken in state x. The approximation is computed at the current time to predict h's value δ time units later and is given by:

$$\hat{h}(x, u, \delta) = h(x) + \sum_{i=1}^{n} \frac{h^i(x, u)}{i!} \delta^i \tag{3}$$

where $h^i(x, u)$ denotes the i^{th} time derivative of h evaluated in state x if control action u is taken. The control action is needed to calculate the time derivatives of h from the definition of h and Eq. 1 by applying the chain rule. Since we are usually interested in predicting the value one time step in the future, we use $\hat{h}(x, u)$ as shorthand for $\hat{h}(x, u, \eta)$. By Taylor's theorem with the Lagrange form

of the remainder, the remainder error of the approximation $\hat{h}(x,u)$ is:

$$\frac{h^{n+1}(x,u,\delta)}{(n+1)!}\eta^{n+1} \text{ for some } \delta \in (0,\eta) \tag{4}$$

An upper bound on the remainder error, if the state remains in the admissible region during the time interval, is:

$$\lambda(u) = \sup\left\{\frac{|h^{n+1}(x,u)|}{(n+1)!}\eta^{n+1} : x \in \mathcal{A}\right\} \tag{5}$$

The FSC is based on checking recoverability during the next time step. For this purpose, the set $\mathcal{A}$ of admissible states is shrunk by margins of μ_{dec} and μ_{inc}, a vector of upper bounds on the amount by which each state variable can decrease and increase, respectively, in one time step, maximized over all admissible states. Formally,

$$\begin{aligned}\mu_{\text{dec}}(u) &= |\min(0, \eta\dot{x}_{min}(u))| \\ \mu_{\text{inc}}(u) &= |\max(0, \eta\dot{x}_{max}(u))|\end{aligned} \tag{6}$$

where $\dot{x}_{min}$ and $\dot{x}_{max}$ are vectors of solutions to the optimization problems:

$$\begin{aligned}\dot{x}_i^{min}(u) &= \inf\{\dot{x}_i(x,u) : x \in \mathcal{A}\} \\ \dot{x}_i^{max}(u) &= \sup\{\dot{x}_i(x,u) : x \in \mathcal{A}\}\end{aligned} \tag{7}$$

The difficulty of finding these extremal values depends on the complexity of the functions $\dot{x}_i(x,u)$. For example, it is relatively easy if they are convex. In our case study of a realistic microgrid model, they are multivariate polynomials with degree 1, and hence convex. The set $\mathcal{A}_r$ of *restricted admissible states* is given by:

$$\mathcal{A}_r(u) = \{x : x_{lb} + \mu_{\text{dec}}(u) < x < x_{ub} - \mu_{\text{inc}}(u)\} \tag{8}$$

Let $Reach_{=\eta}(x,u)$ denote the set of states reachable from state x after exactly time η if control action u is taken in state x. Let $Reach_{\leq\eta}(x,u)$ denote the set of states reachable from x within time η if control action u is taken in state x.

Lemma 1. *For all $x \in \mathcal{A}_r(u)$ and all control actions u, $Reach_{\leq\eta}(x,u) \subseteq \mathcal{A}$.*

Proof. The derivative of x is bounded by $\dot{x}_{min}(u)$ and $\dot{x}_{max}(u)$ for all states in $\mathcal{A}$. This implies that μ_{dec} and μ_{inc} are the largest amounts by which the state x can decrease and increase, respectively, during time η, as long as x remains within $\mathcal{A}$ during the time step. Since $\mathcal{A}_r(u)$ is obtained by shrinking $\mathcal{A}$ by μ_{dec} and μ_{inc} (i.e., by moving the lower and upper bounds, respectively, of each variable inwards by those amounts), the state cannot move outside of $\mathcal{A}$ during time η.

3.1 Forward Switching Condition

To ensure safety, a forward-switching condition (FSC) should switch control from the NC to the BC if using the control action u proposed by NC causes any unsafe states to be reachable from the current state x during the next control period, or causes any unrecoverable states to be reachable at the end of the next control period. These two conditions are captured in the following definition:

Definition 1 (Forward Switching Condition). *A condition $FSC(x,u)$ is a* forward switching condition *if for every recoverable state x, every control action u, and control period η, $Reach_{\leq\eta}(x,u) \cap \mathcal{U} \neq \emptyset \vee Reach_{=\eta}(x,u) \not\subset \mathcal{R}$ implies $FSC(x,u)$ is true.*

Theorem 1. *A Simplex architecture whose forward switching condition satisfies Definition 1 keeps the system invariably safe provided the system starts in a recoverable state.*

Proof. Our definition of an FSC is based directly on the switching logic in Algorithm 1 of [36]. The proof of Theorem 1 in [36] shows that an FSC that is exactly the disjunction of the two conditions in our definition invariantly ensures system safety. It is easy to see that any weaker FSC also ensures safety. ■

We now propose a new and general procedure for constructing a switching condition from a BaC and prove its correctness.

Theorem 2. *Given a barrier certificate h, the following condition is a forward switching condition: $FSC(x,u) = \alpha \vee \beta$ where $\alpha \equiv \hat{h}(x,u) - \lambda(u) \leq 0$ and $\beta \equiv x \notin \mathcal{A}_r(u)$*

Proof. Intuitively, $\alpha \vee \beta$ is an FSC because (1) if condition α is false, then control action u does not lead to an unsafe or unrecoverable state during the next control period, provided the state remains admissible during that period; and (2) if condition β is false, then the state will remain admissible during that period. Thus, if α and β are both false, then nothing bad can happen during the control period, and there is no need to switch to the BC.

Formally, suppose x is a recoverable state, u is a control action, and $Reach_{\leq\eta}(x,u) \cap \mathcal{U} \neq \emptyset \ \vee \ Reach_{=\eta}(x,u) \not\subset \mathcal{R}$, i.e., there is an unsafe state in $Reach_{\leq\eta}(x,u)$ or an unrecoverable state in $Reach_{=\eta}(x,u)$. Let x' denote that unsafe or unrecoverable state. Recall that $\mathcal{Z}(h) \subseteq \mathcal{R}$, and $\mathcal{R} \cap \mathcal{U} = \emptyset$. Therefore, $h(x',u) \leq 0$. We need to show that $\alpha \vee \beta$ holds. We do a case analysis based on whether x is in $\mathcal{A}_r(u)$.

Case 1: $x \in \mathcal{A}_r(u)$. In this case, we use a lower bound on the value of the BaC h to show that states reachable in the next control period are safe and recoverable. Using Lemma 1, we have $Reach_{\leq\eta}(x,u) \subseteq \mathcal{A}$. This implies that $\lambda(u)$, whose definition maximizes over $x \in \mathcal{A}$, is an upper bound on the error in the Taylor approximation $\hat{h}(x,u,\delta)$ for $\delta \leq \eta$. This implies that $\hat{h}(x,u) - \lambda(u)$ is a lower bound on value of BaC for all states in $Reach_{\leq\eta}(x,u)$. As shown above,

there is a state x' in $Reach_{\leq\eta}(x,u)$ with $h(x',u) \leq 0$. $\hat{h}(x,u) - \lambda(u)$ is lower bound on $h(x',u)$ and hence must also be less than or equal to 0. Thus, α holds.

Case 2: $x \notin \mathcal{A}_r(u)$. In this case, β holds. Note that in this case, the truth value of α is not significant (and not relevant, since $FSC(x,u)$ holds regardless), because the state might not remain admissible during the next control period. Hence, the error bound obtained using Eq. 5 is not applicable. ■

3.2 Reverse Switching Condition

The RSC is designed with a heuristic approach, since it does not affect safety of the system. To prevent frequent switching between the NC and BC, we design the RSC to hold if the FSC is likely to remain false for at least m time steps, with $m > 1$. The RSC, like the FSC, is the disjunction of two conditions. The first condition is $h(x) \geq m\eta|\dot{h}(x)|$, since h is likely to remain non-negative for at least m time steps if its current value is at least that duration times its rate of change. The second condition ensures that the state will remain admissible for m time steps. In particular, we take:

$$RSC(x) = h(x) \geq m\eta|\dot{h}(x)| \wedge x \in \mathcal{A}_{r,m}, \tag{9}$$

where the m-times-restricted admissible region is:

$$\mathcal{A}_{r,m} = \{x : x_{lb} + m\mu_{\text{dec}} < x < x_{ub} - m\mu_{\text{inc}}\}, \tag{10}$$

where vectors μ_{dec} and μ_{inc} are defined in the same way as $\mu_{\text{dec}}(u)$ and $\mu_{\text{inc}}(u)$ in Eqs. 6 and 7 except with optimization over all control actions u.

3.3 Decision Logic

The DM's switching logic has three inputs: the current state x, the control action u currently proposed by the NC, and the name c of the controller currently in control (as a special case, we take $c = NC$ in the first time step). The switching logic is defined by cases as follows: $DM(x,u,c)$ returns BC if $c = NC \wedge FSC(x,u)$, returns NC if $c = BC \wedge RSC(x)$, and returns c otherwise.

4 Application to Microgrids

A *microgrid* (MG) is an integrated energy system comprising distributed energy resources (DERs) and multiple energy loads. DERs tend to be *renewable* energy resources and include solar panels, wind turbines, batteries, and emergency diesel generators. By satisfying energy needs from local renewable energy resources, MGs can reduce energy costs and improve energy supply reliability for energy consumers. Some of the major control requirements for an MG are power control, load sharing, and frequency and voltage regulation.

An MG can operate in two modes: grid-connected and islanded. When operated in grid-connected mode, DERs act as constant source of power which can be injected into the network on demand. In contrast, in islanded or autonomous

mode, the DERs form a grid of their own, meaning not only do they supply power to the local loads, but they also maintain the MG's voltage and frequency within the specified limits [25]. For our case study, we focus on voltage regulation in both grid-connected and islanded modes. Specifically, we apply Bb-Simplex to the controller for the inverter for a Photovoltaic (PV) DER.

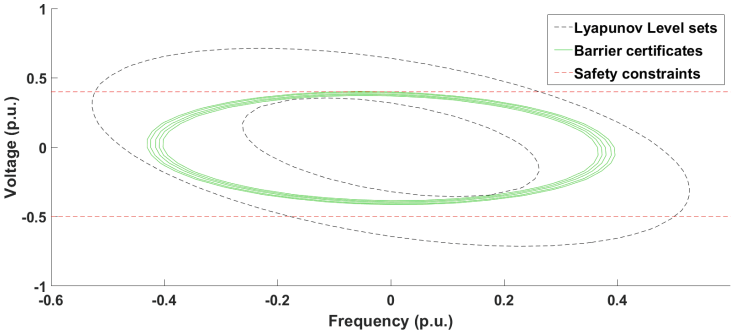

Fig. 2. Lyapunov-function level sets (black-dotted ellipses). Innermost ellipse also indicates initial BaC, which is optimized iteratively (green ellipses). Red lines are voltage safety limits. (Color figure online)

4.1 Baseline Controller

For our experiments, we used the SOS-based methodology described in Sect. 2 to derive a Barrier Certificate (as a proof of safety) for the baseline controller. We use a droop controller as the BC. A droop controller is a type of proportional controller, traditionally used in power systems for control objectives such as voltage regulation, power regulation, and current sharing [10,14,40]. The droop controller tries to balance the electrical power with voltage and frequency. Variations in the active and reactive powers result in frequency and voltage magnitude deviations, respectively [20]. The dynamic model for a voltage droop controller for an inverter has the form $\dot{v} = v^* - v + \lambda_q(Q^* - Q)$, where v^*, v, Q^*, Q are voltage reference, voltage, reactive power reference and reactive power of inverter, respectively, and λ_q is the controller's droop coefficient. Detailed dynamic models for an MG with multiple inverters connected by transmission lines and with droop controllers for frequency and voltage are given in [3,13].

Figure 2 shows this process of incrementally expanding the Lyapunov function to obtain the BaC. SOS-based algorithms apply only to polynomial dynamics so we first recast our droop controller dynamics to be polynomial using a DQ0 transformation [22] to AC waveforms. This transformation is exact; i.e., it does not introduce any approximation error. In our experimental evaluation (Sect. 5), we obtain the BaCs for BCs in the form of droop controllers for voltage regulation, in the context of MGs containing up to three DERs of different types. Note that battery DERs operate in two distinct modes, charging and discharging, resulting in a hybrid system model with different dynamics in different modes. For now, we consider only runs in which the battery remains in the same mode for the duration of the run. Extending our framework to hybrid systems is future work.

4.2 Neural Controller

To help address the control challenges related to microgrids, the application of *neural networks for microgrid control* is on the rise [16]. Increasingly, Reinforcement learning (RL) is being used to train powerful Deep Neural Networks (DNNs) to produce high-performance MG controllers.

We present our approach for learning neural controllers (NCs) in the form of DNNs representing deterministic control policies. Such a DNN maps system states (or raw sensor readings) to control inputs. We use RL in form of Deep Deterministic Policy Gradient (DDPG) algorithm, with the safe learning strategy of penalizing unrecoverable actions [24]. DDPG was chosen because it works with deterministic policies and is compatible with continuous action spaces.

We consider a standard RL setup consisting of an agent interacting with an environment in discrete time. At each time step t, the agent receives a (microgrid) state x_t as input, takes an action a_t, and receives a scalar reward r_t. The DDPG algorithm employs an *actor-critic framework*. The actor generates a control action and the critic evaluates its quality. In order to learn from prior knowledge, DDPG uses a replay buffer to store training samples of the form (x_t, a_t, r_t, x_{t+1}). At every training iteration, a set of samples is randomly chosen from the replay buffer. For further details regarding the implementation of the DDPG algorithm, please refer to Algorithm 1 in [15].

To learn an NC for DER voltage control, we designed the following reward function, which guides the actor network to learn the desired control objective.

$$r(x_t, a_t) = \begin{cases} -1000 & \text{if FSC}(x_t, a_t) \\ 100 & \text{if } v_{od} \in [v_{ref} - \epsilon, v_{ref} + \epsilon] \\ -w \cdot (v_{od} - v_{ref})^2 & \text{otherwise} \end{cases} \tag{11}$$

where w is a weight ($w = 100$ in our experiments), v_{od} is the d-component of the output voltage of the DER whose controller is being learned, v_{ref} is the reference or nominal voltage, and ϵ is the tolerance threshold. We assign a high negative reward for triggering the FSC, and a high positive reward for reaching the tolerance region, i.e., $v_{ref} \pm \epsilon$. The third clause rewards actions that lead to a state in which the DER voltage is close to its reference value.

Adversarial Inputs. Controllers obtained via deep RL algorithms are vulnerable to *adversarial inputs* (AIs): those that lead to a state in which the NC produces an unrecoverable action, even though the NC behaves safely on very similar inputs. NSA provides a defense against these kinds of attacks. If the NC proposes a potentially unsafe action, the BC takes over in a timely manner, thereby guaranteeing the safety of the system. To demonstrate NSA's resilience to AIs, we use a gradient-based attack (Algorithm 4) [23] to construct such inputs, and show that the DM switches control to the BC in time to ensure safety.

The gradient-based algorithm takes as input the critic network, actor network, adversarial attack constant c, parameters a, b of beta distribution $\beta(a, b)$, and the number of times n noise is sampled. For a given (microgrid) state x,

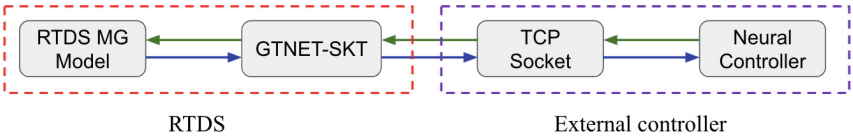

Fig. 3. Integration of External NC with RTDS

the critic network is used to ascertain its $\mathcal{Q}$-value and the actor network determines its optimal action. Once the gradient of the critic network's loss function is computed using the $\mathcal{Q}$-value and the action, the l_2-constrained norm of the gradient ($grad\_dir$) is obtained. An initial (microgrid) state x_0, to be provided as input to the actor network, is then perturbed to obtain a potential adversarial state x_{adv}, determined by the sampled noise in the direction of the gradient: $x_{adv} = x_0 - c \cdot \beta(a, b) \cdot grad\_dir$.

We can now compute the $\mathcal{Q}$-value of x_{adv} and its (potentially adversarial) action a_{adv}. If this value is less tha $\mathcal{Q}(x_0, a_0)$, then x_{adv} leads to a sub-optimal action. The gradient-based attack algorithm does not guarantee the successful generation of AIs every time it is executed. The success rate is inversely related to the quality of the training of the NC. In our experiments (see Sect. 5.3), the highest success rate for AI generation that we observed is 0.008%.

4.3 Adaptation Module

The Adaptation Module (AM) retrains the NC in an online manner when the NC produces an unrecoverable action that causes the DM to failover to the BC. With retraining, the NC is less likely to repeat the same or similar mistakes in the future, allowing it to remain in control of the system more often, thereby improving performance. We use Reinforcement Learning with the reward function defined in Eq. 11 for online retraining.

As in initial training, we use the DDPG algorithm (with the same settings) for online retraining. When the NC outputs an unrecoverable action, the DM switches control to the BC, and the AM computes the (negative) reward for this action and adds it to a pool of training samples. As in [24], we found that reusing the pool of training samples (DDPG's experience replay buffer) from initial training of the NC evolves the policy in a more stable fashion, as retraining samples gradually replace initial training samples in the pool. Another benefit of reusing the initial training pool is that retraining of the NC can start almost immediately, without having to wait for enough samples to be collected online.

We use off-policy retraining i.e., at every time step while the BC is active, the BC's action is used in the training sample. The reward for the BC's action is based on the observed next state of the system.

5 Experimental Evaluation

We apply our Bb-Simplex methodology to a model of a microgrid [21] with three DERs: a battery, photovoltaic (PV, a.k.a. solar panels), and diesel generator. The three DERs are connected to the main grid via bus lines. We are primarily interested in PV control, since we apply Bb-Simplex to PV voltage regulation. The PV control includes multiple components, such as "three-phase to DQ0 voltage and current" transformer, average voltage and current control, power and voltage measurements, inner-loop dq current control, and outer-loop Maximum Power Point Tracking (MPPT) control. Our experimental evaluation of Bb-Simplex was carried out on RTDS, a high-fidelity power systems simulator.

We ran experiments on a configuration where the PV is in islanded mode, and the diesel generator and battery (in discharging mode) DERs are connected within the MG. The state of the MG plant is given by $[i_d\ i_q\ i_{od}\ i_{oq}\ v_{od}\ v_{oq}\ i_{ld}\ i_{lq}\ m_d\ m_q]$, where i_d and i_q are the d- and q-components of the dq current measured at the local load of the inverter, i_{od} and i_{oq} are the d- and q-components of the output current of the inverter measured at point of coupling to the main grid, v_{od} and v_{oq} are the d- and q-components of the output voltage of the inverter measured at point of coupling to the main grid, i_{ld} and i_{lq} are the d- and q-components of the input current to the current controller, m_d and m_q are the d- and q-components of the output voltage from the current controller used to generate the next state.

We use Bb-Simplex to ensure the safety property that the d-component of the output voltage (v_{od}) of the inverter for the PV DER is within $\pm 3\%$ of the reference voltage $v_{ref} = 0.48$ kV. We adopted a 3% tolerance based on the discussion in [21]. Bb-Simplex could similarly be used to ensure additional desired safety properties. All experiments use runs of length 10 s, with the control period, RTDS time step, and simulation time step in MATLAB all equal to 3.2 milliseconds (msec), the largest time step allowed by RTDS.

5.1 Integration of Bb-Simplex in RTDS

The BC is the original droop controller described in [21], implemented in RTDS using components in the RTDS standard libraries. The DM is implemented as an RTDS *custom component* written in C. For an MG configuration, expressions for the BaC, λ and μ (see Sect. 3) are derived in MATLAB, converted to C data structures, and then included in a header file of the custom component.

The BaCs are polynomials comprising 92 monomials for our configuration.

The NC is trained and implemented using Keras [8], a high-level neural network API written in Python, running on top of TensorFlow [1]. For training, we customized an existing skeleton implementation of DDPG in Keras, which we then used with the Adam optimizer [12].

RTDS imposes limitations on custom components that make it difficult to implement complex NNs within RTDS. Existing NN libraries for RTDS, such as [17,18], severely limit the NN's size and the types of activation functions.

Therefore, we implemented the NC external to RTDS, following the *software-defined microgrid control* approach in [35]. Figure 3 shows our setup. We used RTDS's GTNET-SKT communication protocol to establish a TCP connection between the NC running on a PC and an "NC-to-DM" relay component in the RTDS MG model. This relay component repeatedly sends the plant state to the NC, which computes its control action and sends it to the relay component, which in turn sends it to the DM.

5.2 Evaluation of Forward Switching Condition

We derive a BaC using the SOS-based methodology presented in Sect. 2, and then derive a switching condition from the BaC, as described in Sect. 3.1. To find values of λ and μ, we use MATLAB's fmincon function to solve the constrained optimization problems given in Eqs. 6 and 7.

An ideal FSC triggers a switch to BC only if an unrecoverable state is reachable in one time step. For systems with complex dynamics, switching conditions derived in practice are conservative, i.e., may switch sooner. To show that our FSC is not overly conservative, we performed experiments using an AC that continuously increases the voltage and hence soon violates safety. The PV voltage controller has two outputs, m_d and m_q, for the d and q components of the voltage, respectively. The dummy AC simply uses constant values for its outputs, with $m_d = 0.5$ and $m_q = 1e - 6$.

These experiments were performed with PV DER in grid connected mode, with reference voltage and voltage safety threshold of 0.48 kV and 0.4944 kV, respectively, and a FSC derived using a 4^{th}-order Taylor approximation of the BaC. We averaged over 100 runs from initial states with initial voltage selected uniformly at random from the range 0.48 kV $\pm$ 1%. The mean voltage at switching is 0.4921 kV (with standard deviation 0.0002314 kV), which is only 0.46% below the safety threshold. The mean numbers of time steps before switching, and before a safety violation if Bb-Simplex is not used, are 127.4 and 130.2, respectively. Thus, our FSC triggered a switch about three time steps, on average, before a safety violation would have occurred.

We also derived a neural network-based BaC using deep learning and verified it using the Gurobi optimizer as discussed in Sect. 2. We then derived the switching conditions from the verified neural BaC, again using a 4^{th}-order Taylor approximation. We performed the same experiments as above to determine the conservativeness of this FSC. The mean voltage at switching is 0.4923 kV (with standard deviation 0.0002132 kV). The mean numbers of time steps before switching, and before a safety violation if Bb-Simplex is not used, are 128.1 and 130.2, respectively. Thus, our neural FSC triggered a switch about two time steps, on average, before a safety violation would have occurred.

5.3 Evaluation of Neural Controller

The NC for a microgrid configuration is a DNN with four fully-connected hidden layers of 128 neurons each and one output layer. The hidden layers and output

Table 1. Performance evaluation of NC

	CT	$\sigma(CT)$	δ	$\sigma(\delta)$
NC	84.3	7.6	1.7e−4	1.4e−5
BC	115.7	9.8	5.8e−4	3.8e−5

(a) Performance comparison of NC and BC

	CT	$\sigma(CT)$	δ	$\sigma(\delta)$
Gen 1	112.5	11.1	2.5e−4	1.9e−5
Gen 2	89.1	8.7	1.8e−4	1.3e−5

(b) Generalization performance of NC

layer use the ReLU and tanh activation function, respectively. The input state to the NC (DNN) is the same as the inputs to the BC (droop controller) i.e., $[i_{ld}\ i_{lq}]$, where i_{ld} and i_{lq} are the d- and q-components of the input current to the droop controller. Thus the NC has same inputs and outputs as the BC. The NC is trained on 1 million samples (one-step transitions) from MATLAB simulations, processed in batches of 200. Transitions start from random states, with initial values uniformly sampled from $[0.646, 0.714]$ for i_{ld} and $[-0.001, 0.001]$ for i_{lq} [21]. Training takes approximately 2 h.

Performance. We evaluate a controller's performance based on three metrics: convergence rate (CR), the percentage of trajectories in which the DER voltage converges to the tolerance region $v_{ref} \pm \epsilon$; average convergence time (CT), the average time required for convergence of the DER voltage to the tolerance region; and mean deviation (δ), the average deviation of the DER voltage from v_{ref} after the voltage enters the tolerance region. We always report CR as a percentage, CT in milliseconds, and δ in kV.

We show that the NC outperforms the BC. For this experiment, we used RTDS to run the BC and NC starting from the same 100 initial states. The CR is 100% for the NC and BC. Table 1a compares their performance, averaged over 100 runs, with $\epsilon = 0.001$. We observe that the NC outperforms the BC both in terms of average convergence time and mean deviation. We also report the standard deviations (σ) for these metrics and note that they are small compared to the average values. The FSC was not triggered even once during these runs, showing that the NC is well-trained.

Generalization. Generalization refers to the NC's ability to perform well in contexts beyond the ones in which it was trained. First, we consider two kinds of generalization with respect to the microgrid state:

- Gen 1: the initial states of the DERs are randomly chosen from a range outside of the range used during training.
- Gen 2: the power set-point $P^\star$ is randomly chosen from the range $[0.2, 1]$, whereas all training was done with $P^\star = 1$.

Table 1b presents the NC's performance in these two cases, based on 100 runs for each case. We see that the NC performs well in both cases.

Second, we consider generalization with respect to the microgrid configuration. Here we evaluate how the NC handles dynamic changes to the microgrid configuration during runtime. For the first experiment, we start with all the 3

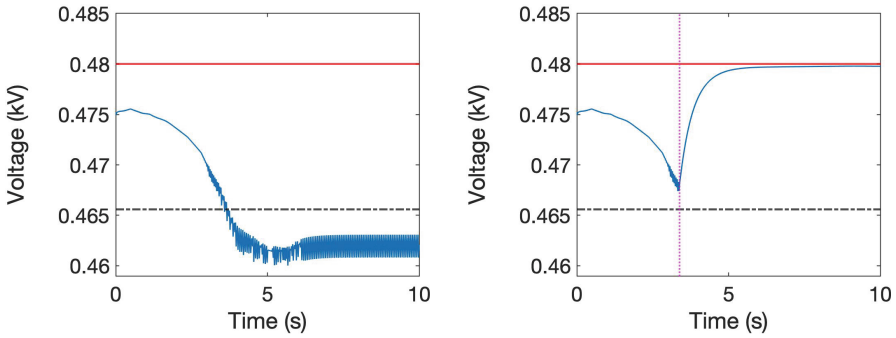

Fig. 4. NC with adversarial inputs (left: without NSA, right: with NSA)

DERs connected, but the diesel generator DER is disconnected after the voltage has converged. For the second experiment, we again start with all the 3 DERs connected, but both the diesel generator and battery DER are disconnected after the voltage has converged. For both instances, the NC succeeded in continuously keeping the voltage in the tolerance region ($v_{ref} \pm \epsilon$) after the disconnection. The disconnection caused a slight drop in the subsequent steady-state voltage, a drop of 0.114% and 0.132%, averaged over 100 runs for each case.

Adversarial Input Attacks. We demonstrate that RL-based neural controllers are vulnerable to adversarial input attacks. We use the gradient-based attack algorithm described in Sect. 4.2 to generate adversarial inputs for our NCs. We use an adversarial attack constant $c = 0.05$ and the parameters for the beta distributions are $a = 2$ and $b = 4$. From 100,000 unique initial states, we obtain 5 adversarial states for our MG configuration. In these experiments, we perturb all state variables simultaneously.

We confirmed with simulations that all generated adversarial states lead to safety violations when the NC alone is used, and that safety is maintained when Bb-Simplex is used. Figure 4 (left) shows one such case, where the NC commits a voltage safety violation. The red horizontal line shows the reference voltage $v_{ref} = 0.48$ kV. The black dashed horizontal line shows the lower boundary of the safety region, 3% below v_{ref}. Figure 4 (right) shows how Bb-Simplex prevents the safety violation. The pink vertical line marks the switch from NC to BC.

We also confirmed that for all generated adversarial states, the forward switch is followed by a reverse switch. The time between forward switch and reverse switch depends on the choice of m (see Sect. 3.2). In the run shown in Fig. 4 (right), they are 5 time steps (0.016 sec) apart; the time of the reverse switch is not depicted explicitly, because the line for it would mostly overlap the line marking the forward switch. For $m = 2, 3, 4$, the average number of time steps between them are 8 (0.0256 s), 13 (0.0416 s), and 18 (0.0576 s).

Table 2. Performance comparison of original NC and NC retrained by AM

NC	CR	CT	$\sigma(CT)$	δ	$\sigma(\delta)$
Retrained	100	70.2	5.7	1.4e−4	1.3e−5
Original	100	81.1	7.7	1.5e−4	1.3e−5

5.4 Evaluation of Adaptation Module

To measure the benefits of online retraining, we used the adversarial inputs described above to trigger switches to BC. We used the switching conditions derived using the SOS-based methodology. We ran the original NC from the first adversarial input state, performed online retraining while the BC is in control, and repeated this procedure for the remaining adversarial states except starting with the updated NC from the previous step. As such, the retraining is cumulative. We performed this entire procedure separately for different RSCs corresponding to different values of m. After the cumulative retraining, we ran the retrained controller from all of the adversarial states, to check whether the retrained NC was still vulnerable (i.e., whether those states caused violations).

The BC was in control for a total of 40, 70, and 95 time steps for $m = 2, 3, 4$, respectively. For $m = 2$, the retrained controllers were still vulnerable to some adversarial states. For $m = 3, 4$, the retrained controllers were not vulnerable to any of the adversarial states, and voltage always converged to the tolerance region. Table 2 demonstrates performance comparison of the original and retrained NCs, averaged over 100 runs starting from random (non-adversarial) states which shows a slight improvement in the performance of the retrained NC (13.4% for CT and 6% for δ). Thus, retraining improves both safety and performance.

6 Related Work

The use of BaCs in the Simplex architecture originated in [36]. There are, however, significant differences between their method for obtaining the switching condition and ours. Their switching logic involves computing, at each decision period, the set of states reachable from the current state within one control period, and then checking whether that set of states is a subset of the zero-level set of the BaC. Our approach avoids the need for reachability calculations by using a Taylor approximation of the BaC, and bounds on the BaC's derivatives, to bound the possible values of the BaC during the next control period and thereby determine recoverability of states reachable during that time. Our approach is computationally much cheaper: a reachability computation is expensive compared to evaluating a polynomial. Their framework can handle hybrid systems. Extending our method to hybrid systems is a direction for future work.

Mehmood et al. [19] propose a distributed Simplex architecture with BCs synthesized using control barrier functions (CBFs) and with switching conditions

derived from the CBFs, which are BaCs satisfying additional constraints. A derivation of switching conditions based on Taylor approximation of CBFs is briefly described but does not consider the remainder error, admissible states, or restricted admissible states, and does not include a proof of correctness (which requires an analysis of the remainder error).

Kundu et al. [13] and Wang et al. [34] use BaCs for safety of microgrids, and Prajna et al. [28] propose an approach for stochastic safety verification of continuous and hybrid systems using BaCs. These approaches are based on the use of verified-safe controllers; they do not allow the use of unverified high-performance controllers, do not consider switching conditions, etc.

The application of neural networks for microgrid control is gaining in popularity [16]. Amoateng et al. [2] use adaptive neural networks and cooperative control theory to develop microgrid controllers for inverter-based DERs. Using Lyapunov analysis, they prove that their error-function values and weight-estimation errors are uniformly ultimately bounded. Tan et al. [32] use Recurrent Probabilistic Wavelet Fuzzy Neural Networks (RPWFNNs) for microgrid control, since they work well under uncertainty and generalize well. We used more traditional DNNs, since they are already high performing, and our focus is on safety assurance. Our Bb-Simplex framework, however, allows any kind of neural network to be used as the AC and can provide the safety guarantees lacking in their work. Unlike our approach, none of these works provide safety guarantees.

7 Conclusion

We have presented Bb-Simplex, a new, provably correct design for runtime assurance of continuous dynamical systems. Bb-Simplex features a new scalable automated method for deriving, from the barrier certificate, computationally inexpensive conditions for switching between advanced and baseline controllers.

We combined Bb-Simplex with the Neural Simplex Architecture and applied the combined framework to micgrogrid control. We conducted an extensive experimental evaluation of the framework on a realistic model of a microgrid with multiple types of energy sources. The experiments demonstrate that the framework can be used to develop high-performance, generalizable neural controllers (NCs) while assuring specified safety properties, even in the presence of adversarial input attacks on the NC. Our experiments also demonstrate that the derived forward switching conditions are not too conservative, i.e., they switch control from the NC to the BC only a short time before a safety violation becomes unavoidable, and that online retraining of the NC is effective in preventing subsequent safety violations by the NC.

As future work, we plan to extend our framework to systems with noise or other sources of uncertainty in the dynamics. We also plan to eliminate the need for manually developed analytical dynamic models by learning neural ODEs [7,41] that capture unknown parts of the dynamics, and deriving BaCs and switching conditions from the resulting dynamics. We also intend to apply our approach to networked microgrids [37].

Acknowledgement. We thank the anonymous reviewers for their valuable feedback. This work was supported in part by NSF grants ITE-2134840, ITE-2040599, CCF-1954837, CCF-1918225, and CPS-1446832.

References

1. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). https://www.tensorflow.org/
2. Amoateng, D.O., Al Hosani, M., Elmoursi, M.S., Turitsyn, K., Kirtley, J.L.: Adaptive voltage and frequency control of islanded multi-microgrids. IEEE Trans. Power Syst. **33**(4), 4454–4465 (2018)
3. Anghel, M., Milano, F., Papachristodoulou, A.: Algorithmic construction of Lyapunov functions for power system stability analysis. IEEE Trans. Circuits Syst. I Regul. Pap. **60**(9), 2533–2546 (2013)
4. Bak, S., Greer, A., Mitra, S.: Hybrid cyberphysical system verification with Simplex using discrete abstractions. In: 16th IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 143–152 (2010)
5. Bak, S., Manamcheri, K., Mitra, S., Caccamo, M.: Sandboxing controllers for cyberphysical systems. In: Proceedings of the IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS 2011), pp. 3–12 April 2011
6. Borrmann, U., Wang, L., Ames, A.D., Egerstedt, M.: Control barrier certificates for safe swarm behavior. In: Egerstedt, M., Wardi, Y. (eds.) Analysis and Design of Hybrid Systems. IFAC-PapersOnLine, vol. 48, pp. 68–73. Elsevier (2015)
7. Chen, T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. In: Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS 2018), pp. 6572–6583 (2018)
8. Chollet, F., et al.: Keras (2015). https://github.com/keras-team/keras.git
9. Damare, A., Roy, S., Smolka, S.A., Stoller, S.D.: A barrier certificate-based Simplex architecture with application to microgrids (2022). https://doi.org/10.48550/ARXIV.2202.09710
10. Guerrero, J.M., Vasquez, J.C., Matas, J., de Vicuna, L.G., Castilla, M.: Hierarchical control of droop-controlled AC and DC microgrids - a general approach toward standardization. IEEE Trans. Industr. Electron. **58**(1), 158–172 (2011)
11. Johnson, T.T., Bak, S., Caccamo, M., Sha, L.: Real-time reachability for verified Simplex design. ACM Trans. Embedded Comput. Syst. **15**(2), 26:1–26:27 (2016)
12. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: 3rd International Conference on Learning Representations, pp. 1–15 (2015)
13. Kundu, S., Geng, S., Nandanoori, S.P., Hiskens, I.A., Kalsi, K.: Distributed barrier certificates for safe operation of inverter-based microgrids. In: 2019 American Control Conference, pp. 1042–1047 (2019)
14. Lasseter, R., Paigi, P.: Microgrid: a conceptual solution. In: 2004 IEEE 35th Annual Power Electronics Specialists Conference (IEEE Cat. No. 04CH37551), vol. 6, pp. 4285–4290 (2004)
15. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. In: 4th International Conference on Learning Representations, pp. 1–14 (2016)
16. Lopez-Garcia, T.B., Coronado-Mendoza, A., Domínguez-Navarro, J.A.: Artificial neural networks in microgrids: a review. Eng. Appl. Artif. Intell. **95**(103894), 1–14 (2020)

17. Luitel, B., Venayagamoorthy, G.K.: Neural networks in RSCAD for intelligent real-time power system applications. In: 2013 IEEE Power Energy Society General Meeting, pp. 1–5 (2013)
18. Luitel, B., Venayagamoorthy, G.K., Oliveira, G.: Developing neural networks library in RSCAD for real-time power system simulation. In: 2013 IEEE Computational Intelligence Applications in Smart Grid (CIASG 2013), pp. 130–137 (2013)
19. Mehmood, U., Stoller, S.D., Grosu, R., Roy, S., Damare, A., Smolka, S.A.: A distributed Simplex architecture for multi-agent systems. In: Proceedings of the Symposium on Dependable Software Engineering: Theories, Tools and Applications (SETTA 2021). Lecture Notes in Computer Science, vol. 13071, pp. 239–257. Springer (2021). https://doi.org/10.1007/978-3-030-91265-9_13
20. Mehrizi-Sani, A.: Distributed control techniques in microgrids. In: Mahmoud, M.S. (ed.) Microgrid: Advanced Control Methods and Renewable Energy System Integration, pp. 43–62. Butterworth-Heinemann (2017)
21. Nzimako, O., Rajapakse, A.: Real time simulation of a microgrid with multiple distributed energy resources. In: International Conference on Cogeneration, Small Power Plants and District Energy (ICUE 2016), pp. 1–6 (2016)
22. O'Rourke, C.J., Qasim, M.M., Overlin, M.R., Kirtley, J.L.: A geometric interpretation of reference frames and transformations: dq0, Clarke, and Park. IEEE Trans. Energy Convers. **34**(4), 2070–2083 (2019)
23. Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., Chowdhary, G.: Robust deep reinforcement learning with adversarial attacks. https://doi.org/10.48550/ARXIV.1712.03632
24. Phan, D., Grosu, R., Jansen, N., Paoletti, N., Smolka, S.A., Stoller, S.D.: Neural Simplex architecture. In: NASA Formal Methods Symposium, pp. 97–114. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-55754-6_6
25. Pogaku, N., Prodanovic, M., Green, T.C.: Modeling, analysis and testing of autonomous operation of an inverter-based microgrid. IEEE Trans. Power Electron. **22**(2), 613–625 (2007)
26. Prajna, S.: Barrier certificates for nonlinear model validation. Automatica **42**(1), 117–126 (2006)
27. Prajna, S., Jadbabaie, A.: Safety verification of hybrid systems using barrier certificates. In: Alur, R., Pappas, G.J. (eds.) HSCC 2004. LNCS, vol. 2993, pp. 477–492. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24743-2_32
28. Prajna, S., Jadbabaie, A., Pappas, G.J.: A framework for worst-case and stochastic safety verification using barrier certificates. IEEE Trans. Autom. Control **52**(8), 1415–1428 (2007)
29. Seto, D., Krogh, B., Sha, L., Chutinan, A.: The Simplex architecture for safe online control system upgrades. In: Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No. 98CH36207), vol. 6, pp. 3504–3508 (1998)
30. Sha, L.: Using simplicity to control complexity. IEEE Softw. **18**(4), 20–28 (2001)
31. Sha, M., et al.: Synthesizing barrier certificates of neural network controlled continuous systems via approximations. In: 2021 58th ACM/IEEE Design Automation Conference, pp. 631–636 (2021)
32. Tan, K.H., Lin, F.J., Shih, C.M., Kuo, C.N.: Intelligent control of microgrid with virtual inertia using recurrent probabilistic wavelet fuzzy neural network. IEEE Trans. Power Electron. **35**(7), 7451–7464 (2020)
33. Ton, D.T., Smith, M.A.: The U.S. department of energy's microgrid initiative. Electr. J. **25**(8), 84–94 (2012)

34. Wang, L., Han, D., Egerstedt, M.: Permissive barrier certificates for safe stabilization using sum-of-squares. In: 2018 Annual American Control Conference, pp. 585–590 (2018)
35. Wang, L., Qin, Y., Tang, Z., Zhang, P.: Software-defined microgrid control: the genesis of decoupled cyber-physical microgrids. IEEE Open Access J. Power Energy **7**, 173–182 (2020)
36. Yang, J., Islam, M.A., Murthy, A., Smolka, S.A., Stoller, S.D.: A simplex architecture for hybrid systems using barrier certificates. In: Tonetta, S., Schoitsch, E., Bitsch, F. (eds.) SAFECOMP 2017. LNCS, vol. 10488, pp. 117–131. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66266-4_8
37. Zhang, P.: Networked Microgrids. Cambridge University Press (2021)
38. Zhao, H., Zeng, X., Chen, T., Liu, Z.: Synthesizing barrier certificates using neural networks. In: Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control (HSCC 2020), pp. 1–11. Association for Computing Machinery (2020)
39. Zhao, Q., et al: Synthesizing ReLU neural networks with two hidden layers as barrier certificates for hybrid systems. In: Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control (HSCC 2021), pp. 1–11. Association for Computing Machinery (2021)
40. Zhou, Y., Ngai-Man Ho, C.: A review on microgrid architectures and control methods. In: 2016 IEEE 8th International Power Electronics and Motion Control Conference (IPEMC-ECCE Asia 2016), pp. 3149–3156 (2016)
41. Zhou, Y., Zhang, P.: Neuro-reachability of networked microgrids. IEEE Trans. Power Syst. **37**(1), 142–152 (2021)