A Set-Theoretic Approach to Safe Reinforcement Learning in Power Systems

Daniel Tabas [1] **Baosen Zhang** [1]

Abstract

Reducing the carbon footprint of the energy sector will be a vital part of the fight against climate change, and doing so will require the widespread adoption of renewable energy resources. Optimally integrating a large number of these resources requires new control techniques that can both compensate for the variability of renewables and satisfy hard engineering constraints. Reinforcement learning (RL) is a promising approach to data-driven control, but it is difficult to verify that the policies derived from data will be safe. In this paper, we combine RL with set-theoretic control to propose a computationally efficient approach to safe RL. We demonstrate the method on a simplified power system model and compare it with other RL techniques.

1. Introduction

Distributed energy resources (DERs) such as wind, solar, and energy storage systems are emerging as a crucial means to achieve greenhouse gas emissions reductions (Steinberg et al., 2017). However, integrating DERs into existing power infrastructure while maintaining reliability proves to be challenging (Kroposki et al., 2017). For example, as DERs come to dominate the energy supply, the uncertainty in wind and solar generation will narrow the transient stability margins of electric generators (Kundur et al., 1994). As a result, the electric grid will be more prone to failures under large disturbances such as line outages.

The flexibility and speed of electronically-controlled DERs can be leveraged for control algorithms that achieve higher efficiency in maintaining system stability. However, when DERs are integrated at large scale, it is challenging to find or implement the optimal strategy. Further, power systems are a quintessential example of safety-critical infrastructure, in which the violation of operational constraints can lead to large blackouts with high economic and human cost. Safe reinforcement learning (safe RL) offers the possibility to design efficient controllers for safety-critical applications without the need for detailed power system models (Chen et al., 2021; Glavic, 2019; Cao et al., 2020; Cui & Zhang, 2021). Safe RL can take several perspectives (García & Fernández, 2015), but in this paper, we are concerned with the ability to guarantee satisfaction of hard state and action constraints during both exploration and execution of a policy. Since guaranteeing constraint satisfaction is impossible without some degree of knowledge about the system (Wachi & Sui, 2020), we assume that the system model is at least approximately known. Even with this model, the task at hand may be too complex for traditional model-based control.

Contributions. We propose a safe and computationally efficient reinforcement learning paradigm using results from *set-theoretic control*. Our method exploits the geometry of polytopic *robust controlled-invariant sets* (RCIs), obtained either from models (Blanchini & Miani, 2015) or directly from data (Chen et al., 2019). We design a policy neural network with an output layer that uses iterative linear projections to guarantee satisfaction of hard state constraints during exploration and execution. Our method applies to safety-critical control tasks which require computationally efficient decision-making on short timescales. We apply the technique to a transient stability problem in a small power system model.

Approaches to safe RL that use weaker notions of safety include restricting actions to those with Lyapunov stability guarantees (Perkins & Barto, 2002; Berkenkamp et al., 2017; Cui & Zhang, 2021) or robustness guarantees (Kretchmar et al., 2001; Donti et al., 2021). However, stability does not always translate to constraint satisfaction. Other approaches use optimization-based "safety filters" to project an action recommended by a policy network into a set of safe control actions (Cheng et al., 2019; Wabersich & Zeilinger, 2021). However, it may not be possible to solve optimization problems in real time. The work in (Zheng et al., 2020) takes a geometric approach in order to guarantee safe exploration without solving an optimization problem, but the resulting "one-step" safety guarantee can still lead a system into states from which no action is safe. Our work extends this one-step safety guarantee to an infinite-horizon guarantee for a

[1]Department of Electrical and Computer Engineering, University of Washington, Seattle, WA, United States. Correspondence to: Daniel Tabas <dtabas@uw.edu>, Baosen Zhang <zhangbao@uw.edu>.

Tackling Climate Change with Machine Learning workshop at ICML 2021.

class of systems with bounded uncertainty.

2. Problem Model

Consider the discrete-time, discounted, infinite-horizon, constrained robust optimal control problem given by

$$\min_{x_{(\cdot)},u_{(\cdot)}} \mathbb{E}_{d_{(\cdot)}} \sum_{t=1}^{\infty} \gamma^t J(x_t,u_t) \tag{1a}$$

$$\text{subject to, } \forall\, t\text{: } x_{t+1} = f(x_t,u_t,d_t) \tag{1b}$$

$$x_{t+1} \in \mathbf{X}\ \forall\, d_t \in \mathbf{D} \tag{1c}$$

$$u_t \in \mathbf{U} \tag{1d}$$

where $x_t \in \mathbb{R}^n$, $u_t \in \mathbb{R}^m$, and $d_t \in \mathbb{R}^d$ are the system state, control input, and exogenous disturbance, respectively, at time t; J is a cost based on x_t and u_t; $\gamma \in (0,1)$ is a discount factor; and f describes the discrete-time evolution of the system. The sets $\mathbf{X} \subset \mathbb{R}^n$ and $\mathbf{U} \subset \mathbb{R}^m$ represent constraints on the state and action, respectively. We assume the distribution of the disturbances is not known, as this is the case for most power systems with multiple renewable energy resources. Instead, we assume that d_t lies in a bounded set $\mathbf{D}$ and require that x_{t+1} must land in $\mathbf{X}$ for any $d_t \in \mathbf{D}$.

In general, simple constraint sets $\mathbf{X}$ that may make engineering sense could lead to (1c) becoming infeasible. This requires the introduction of *invariant sets* as defined next.

Definition 1 (Robust controlled-invariant set (Blanchini & Miani, 2015))**.** The set $\mathbf{S}$ is a *robust controlled-invariant set* (RCI) for the system with dynamics (1b) and constraint (1d) if there exists a feedback controller $u_t = \pi(x_t)$ such that if $x_0 \in \mathbf{S}$, then for all $t \geq 0$ and all disturbance sequences $d_t \in \mathbf{D}$, it holds that $x_t \in \mathbf{S}$.

If $\mathbf{S} \subset \mathbf{X}$, it is easy to see that ensuring safe operation amounts to ensuring that if $x_t \in \mathbf{S}$, then $x_{t+1} \in \mathbf{S}$. By definition, each state in the RCI can be associated with a nonempty set of control actions that guarantee this property. This set is called the *regulation map*:

Definition 2 (Regulation map (Blanchini & Miani, 2015))**.** The *regulation map* $\Omega : \mathbb{R}^n \to \mathbb{R}^m$ for a system with dynamics (1b), constraints (1c) and (1d), and RCI $\mathbf{S} \subset \mathbf{X}$ is a set-valued map from states to control actions defined as follows.

$$\Omega(x_t) = \{u_t \in \mathbf{U} : f(x_t,u_t,d_t) \in \mathbf{S},\ \forall\, d_t \in \mathbf{D}\} \tag{2}$$

$$= \{u_t \in \mathbf{U} : f(x_t,u_t,0) \in \mathbf{T} \subset \mathbf{S}\} \tag{3}$$

where $\mathbf{T} \subset \mathbf{S}$ is the target set, equal to the RCI "eroded" by the set of possible disturbances.

RCIs are typically described by polytopes, ellipsoids, or zonotopes, a special class of polytopes (Maidens et al., 2013; Zhang et al., 2020). Our approach is agnostic to the algorithm used to devise the RCI, but we do require that it can be represented as a polytope.

Let $\mathbf{S} \subset \mathbf{X}$ be an RCI for a system with dynamics f, and let $\pi_\theta : \mathbb{R}^n \to \mathbb{R}^m$ be a function parameterized by θ that maps states to actions. Then a feasible safe control problem is:

$$\min_{\theta} \mathbb{E}_{d_{(\cdot)}} \sum_{t=1}^{\infty} \gamma^t J(x_t,u_t) \tag{4a}$$

$$\text{subject to: } x_{t+1} = f(x_t,u_t,d_t),\ \forall\, t; x_0 \in \mathbf{S} \tag{4b}$$

$$u_t = \pi_\theta(x_t) \in \Omega(x_t),\ \forall\, x_t \in \mathbf{S},\ \forall\, t. \tag{4c}$$

where $\mathbf{X}$ in (1) is replaced by $\mathbf{S}$. Once $\mathbf{S}$ is computed from either a conservative model or from data, the goal is to find the controller π_θ.

The key question is how to address (4c) in a way that can be seamlessly integrated with standard RL techniques. To this end, we pose two questions: First, what is an appropriate function class for π_θ? Second, how can we constrain $\pi_\theta(x)$ to the set $\Omega(x)$ for all $x \in \mathbf{S}$ in a manner that is computationally efficient and differentiable? To answer the first question, we choose neural networks with a custom output layer. The design of this custom layer, which is the answer to the second question, is the focus of the remainder of this paper.

3. RCI Policy Network

We propose a policy network architecture that guarantees safe operation by selecting actions from the regulation map. In (Zheng et al., 2020), the authors propose a policy network that selects actions by choosing convex combinations of the vertices of the "one-step admissible" action set $\{u \in \mathbf{U} : f(x,u) \in \mathbf{X}\}$. When $\mathbf{X}$ is replaced by $\mathbf{S}$, the one-step safety guarantee is replaced by an infinite-horizon safety guarantee, but the geometry of the regulation map (feasible set of control actions) becomes more complex. Our contribution is to accommodate sets described by arbitrary polytopes into the policy network architecture.

3.1. RCI Policy Network Architecture

The design of the policy network relies on several loose assumptions about the dynamics and constraints of the system.

Assumption 1. *The dynamics of the system can be described by the inclusion $f(x_t,u_t,d_t) = Ax_t + Bu_t + Ed_t, d_t \in \boldsymbol{D}$ where d_t is a noise term that can include (bounded) linearization error if the true system is nonlinear (Boyd et al., 1994).*

The case of uncertainty in A, B, and E is readily handled, and is not considered here (Blanchini & Miani, 2015).

Assumption 2. *The target set $\boldsymbol{T} \subset \boldsymbol{S}$ and action set $\boldsymbol{U}$ are polytopes given by $\boldsymbol{T} = \{x \in \mathbb{R}^n : Fx \leq g\}$ and $\boldsymbol{U} = \{u \in \mathbb{R}^m : Hu \leq \bar{u}\}$.*

Under these assumptions, the regulation map $\Omega(x)$ is a polytope described as

$$\Omega(x) = \{u \in \mathbb{R}^m : F(Ax+Bu) \leq g, u \in \mathbf{U}\} \quad (5)$$

$$= \left\{u \in \mathbb{R}^m : \begin{bmatrix} FB \\ H \end{bmatrix} u \leq \begin{bmatrix} g - FAx \\ \bar{u} \end{bmatrix}\right\}. \quad (6)$$

As stated earlier, the set $\Omega(x)$ is nonempty for all $x \in \mathbf{S}$. While it is possible to add a projection layer in a neural network (Agrawal et al., 2019), we seek a faster implementation. Moreover, methods that project directly onto $\Omega(x)$ run the risk of over-exploring the boundary of the set.

We generate a safe control action by taking a safe combination of base control actions, as follows. All $x \in \mathbf{S}$ can be described as a convex combination of the vertices of $\mathbf{S}$. That same convex combination can be used to generate a safe control action when the weights are applied to a set of carefully chosen base control actions. The base control actions must be safe actions associated with each vertex of $\mathbf{S}$, as described in Proposition 1.

Proposition 1. *Suppose $x \in \boldsymbol{S}$. Let $\{s_i\}_{i=1}^q$ be the set of vertices of $\boldsymbol{S}$, and let $V_i \in \mathbb{R}^{m \times p_i}$ be the matrix whose columns correspond to the p_i vertices of $\Omega(s_i)$. Let $Y_i \in \mathbb{R}^{n \times p_i}$ be the matrix whose p_i columns are identically s_i. Let $V = \begin{bmatrix} V_1 & \cdots & V_q \end{bmatrix}$, $Y = \begin{bmatrix} Y_1 & \cdots & Y_q \end{bmatrix}$, and $p = \sum_{i=1}^q p_i$. Define the affine set $\boldsymbol{a}(x) = \{\alpha \in \mathbb{R}^p : Y\alpha = x\}$ and its intersection with the unit simplex Δ_p as $\boldsymbol{A}(x) = \{\alpha \in \boldsymbol{a}(x) : \alpha \geq 0, \sum_{i=1}^p \alpha_i = 1\}$. Then the set $\boldsymbol{V}(x) = \{V\alpha : \alpha \in \boldsymbol{A}(x)\}$ is contained in the regulation map $\Omega(x)$.*

Although $\mathbf{V}(x)$ is not, in general, equal to the entirety of $\Omega(x)$, we demonstrate through simulations that the set $\mathbf{V}(x)$ provides an adequate set of control actions. The proof of Proposition 1 is straightforward and is provided in Appendix A. Figure 1 displays examples of $\mathbf{X}$ and $\mathbf{S}$ and illustrates the relationship between $\Omega(x_i)$ and $\mathbf{V}(x_i)$ for some sample points $x_i \in \mathbf{S}$.

3.2. Cyclic projections algorithm

What should be the outputs of a policy parameterized by a neural network? The policy should output α, an element of $\mathbf{A}(x)$, which provides a safe combination of base control actions. This takes a difficult problem (map the outputs of a neural network's last hidden layer to the set $\Omega(x)$) and reduces it to an easier problem (map the outputs of a neural network's last hidden layer to the set $\mathbf{A}(x)$). The set $\mathbf{A}(x)$ has a simple geometry: it is the (nonempty) intersection of an affine set with the unit simplex. We need a differentiable and efficient way to find a point in this intersection, since

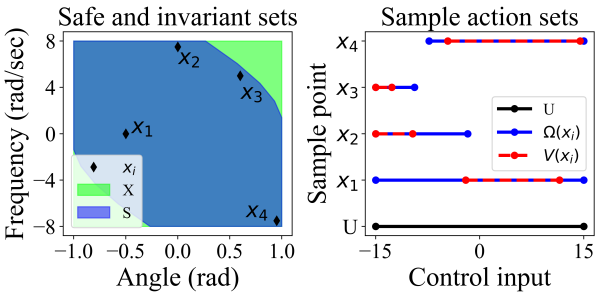

Figure 1. Left: Safe set $\mathbf{X}$ and RCI set $\mathbf{S}$ for an example system and the positions of four points inside $\mathbf{S}$. *Right:* the set of safe actions $\Omega(x_i)$ at x_i and the subset $\mathbf{V}(x_i) \subset \Omega(x_i)$ that is used by our algorithm, as well as the full action set $\mathbf{U}$.

the mapping is attached to the output of a neural network and must be implemented in real time. Since $\mathbf{A}(x)$ is an intersection of convex sets, we can efficiently find a point in the intersection of these sets using cyclic projections (Bregman, 1967) as described in Algorithm 1.

We initialize from $\beta \in \mathbb{R}^p$, which is the output from the last hidden layer of π_θ and which represents a set of weights that may be infeasible ($Y\beta \neq x$ or $\beta \notin \Delta_p$). Projection onto each of the convex sets is computationally efficient and differentiable. Solving this problem in the output layer of the policy network allows the network to choose a safe action $u = V\alpha, \alpha \in \mathbf{A}(x)$, just by specifying β.

Algorithm 1 Efficient cyclic projections to $\mathbf{A}(x_t)$

Require: State x, initial weights β, tolerance $\varepsilon > 0$
Ensure: $\alpha \in \mathbf{A}(x)$
1: $\alpha^0 \leftarrow \text{softmax}(\beta)$ {Initialize in Δ_p}
2: **for** $k = 0 : k_{\max}$ **do**
3: Iteratively project: $\Delta_p \to \mathbf{a}(x) \to \mathbb{R}^p_+ \to \Delta_p$ until convergence
4: **end for**

The cyclic projections algorithm will converge since all sets are convex (Bregman, 1967). When the initial distance $\|P(Y\alpha^0 - x)\|_2$ is small, Algorithm 1 converges in a few ($<$ 10) iterations. In order to incentivize efficiency, this distance is added as a penalty in the reward function. Simulation results show that this penalty is effective in encouraging efficiency.

Since each projection step is differentiable and can be performed quickly, Algorithm 1 can be integrated seamlessly into the output layer of the policy network. During training, it is possible to back-propagate through Algorithm 1 in order to tune not just the choice of initial weights β but also the choice of final weights α with respect to the policy parameters θ. Details of this algorithm are provided in Appendix B.

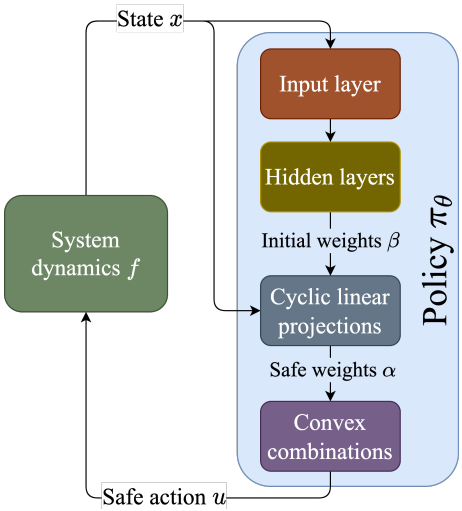

Figure 2. Illustration of the policy network architecture and its relationship to the environment.

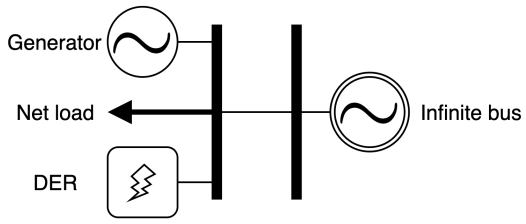

Figure 3. Illustration of the single machine-infinite bus power system under consideration.

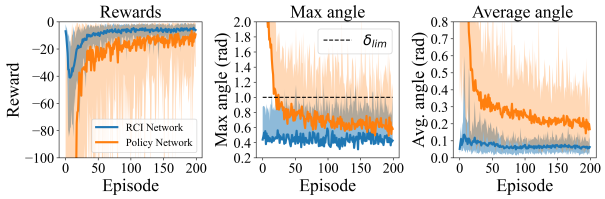

Figure 4. Training results per episode over 20 training trials (min/avg/max). *Left:* Episode reward. *Middle:* Maximum absolute rotor angle. *Right:* Average absolute rotor angle.

4. Simulations

4.1. Power System Example

Because of space constraints, we consider a simplified single machine-infinite bus power system (Machowski et al., 2008) where the generator bus includes a synchronous electric generator, a DER, and a fluctuating net load comprising a load and/or variable renewable energy supply (Fig. 3). The rotor angle dynamics of the generator are given by the swing equations in discrete time, where Δt is the time step:

$$\delta_{t+1} = \delta_t + \omega_{t+1}\Delta t \tag{7a}$$

$$M\omega_{t+1} = M\omega_t - (K\sin\delta_t + D\omega_t - u_t + d_t)\Delta t \tag{7b}$$

where δ is the rotor phase angle measured relative to the infinite bus, ω is the deviation of the generator frequency from 60 Hz, u is the control input (real power injection from DER), and d is an exogenous disturbance in the form of fluctuating net real power demand. The parameters M, D, and K are the inertia, damping, and synchronizing power coefficients, respectively. The internal dynamics of the DER occur on much faster timescales and are not modeled.

The objective of the DER is to modulate its real power output u_t to balance transient stability enhancement with control effort, subject to the constraints of the system. Transient stability is aided by keeping δ and ω close to their nominal values (taken to be zero without loss of generality). The state at time t is $x_t = \begin{bmatrix}\delta_t & \omega_t\end{bmatrix}^T$. The constraint $x_t \in \mathbf{X}$ is a box constraint which includes an interval constraint on δ_t to enforce transient stability margins and an interval constraint on ω_t to protect generators and other equipment. The interval constraints $u_t \in \mathbf{U}$ and $d_t \in \mathbf{D}$ represent the power capacity limits of the DER and net load variations, respectively. The sets $\mathbf{X}$ and $\mathbf{S}$ for the system are shown in the left pane of Figure 1.

4.2. RL Algorithm

To train the policy network, we use the Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al., 2016). Implementation details are provided in Appendix C. For comparison, we also train a policy network that does not have built-in safety guarantees. Instead, the cost function for training this policy network includes a penalty term for state constraint violations. The results show that even with a soft penalty, constraint violations persist throughout both training and testing. Figure 4 compares the training performance of the two networks in terms of rewards, maximum rotor angle per episode, and average rotor angle per episode. Figure 5 compares some time-domain test trajectories resulting from the control policies of the RCI and generic policy networks. We found that the average number of Algorithm 1 iterations decreased from 10 to 7 over the course of training, demonstrating (increasing) efficiency throughout. All code used to generate these results is available at `github.com/dtabas/safe-rl`.

5. Conclusions and future work

Safe RL applied to power system operations has the potential to contribute to the widespread adoption of low-carbon energy sources. In this paper, we propose a novel policy network architecture for computationally efficient safe RL in power systems. Our approach leverages set-theoretic techniques to provide an efficient and differentiable means

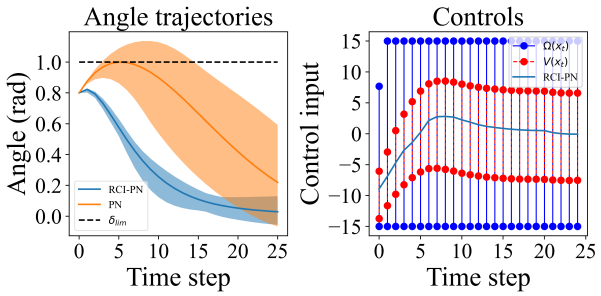

Figure 5. Test results. *Left:* rotor angle trajectories, min/avg/max over 20 trained networks. *Right:* Control actions for a single trained RCI policy network, along with the safe control set $\Omega(x_t)$ and subset $\mathbf{V}(x_t)$ for each state visited along the trajectory.

of guaranteeing that state trajectories will satisfy hard constraints. We demonstrate the proposed method on a simple power system model, and show that safe operation is maintained throughout training. The proposed policy network architecture outperforms conventional RL methods in terms of safety. In future work, we will investigate the robustness of the policies to topology changes. We will also propose an alternative closed-form safe output layer.

Acknowledgements

The authors would like to thank Liyuan Zheng for guidance, Sarah H.Q. Li for helpful discussions, and the reviewers for their useful comments. This work is partially supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1762114 and NSF grant ECCS-1930605. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Zico Kolter, J. Differentiable convex optimization layers. *Advances in Neural Information Processing Systems*, 32(NeurIPS), 2019.

Berkenkamp, F., Turchetta, M., Schoellig, A. P., and Krause, A. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pp. 909–919, 2017.

Blanchini, F. and Miani, S. *Set-theoretic methods in control*. Birkhauser, 2015.

Boyd, S., El Ghaoui, L., Feron, E., and Balakrishnan, V. *Linear Matrix Inequalities in System and Control Theory*, volume 15. Society for Industrial and Applied Mathematics, Philadelphia, 1994.

Bregman, L. M. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3): 200–217, 1967.

Cao, D., Hu, W., Zhao, J., Zhang, G., Zhang, B., Liu, Z., Chen, Z., and Blaabjerg, F. Reinforcement Learning and Its Applications in Modern Power and Energy Systems: A Review. *Journal of Modern Power Systems and Clean Energy*, 8(6):1029–1042, 2020.

Chen, X., Qu, G., Tang, Y., Low, S., and Li, N. Reinforcement Learning for Decision-Making and Control in Power Systems: Tutorial, Review, and Vision. *arXiv preprint: arXiv 2102.01168*, 2021.

Chen, Y., Peng, H., Grizzle, J., and Ozay, N. Data-Driven Computation of Minimal Robust Control Invariant Set. In *2018 IEEE Conference on Decision and Control*, volume 2018-Decem, pp. 4052–4058. IEEE, 2019.

Cheng, R., Orosz, G., Murray, R. M., and Burdick, J. W. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *33rd AAAI Conference on Artificial Intelligence*, pp. 3387–3395, 2019.

Cui, W. and Zhang, B. Reinforcement Learning for Optimal Frequency Control: A Lyapunov Approach. *arXiv preprint: 2009.05654v3*, 2021.

Donti, P. L., Roderick, M., Fazlyab, M., and Kolter, J. Z. Enforcing robust control guarantees within neural network policies. In *International Conference on Learning Representations*, pp. 1–26, 2021.

García, J. and Fernández, F. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research*, 16:1437–1480, 2015.

Glavic, M. (Deep) Reinforcement learning for electric power system control and related problems: A short review and perspectives. *Annual Reviews in Control*, 48: 22–35, 2019.

Kretchmar, R. M., Young, P. M., Anderson, C. W., Hittle, D. C., Anderson, M. L., and Delnero, C. C. Robust reinforcement learning control with static and dynamic stability. *International Journal of Robust and Nonlinear Control*, 11(15):1469–1500, 2001.

Kroposki, B., Johnson, B., Zhang, Y., Gevorgian, V., Denholm, P., Hodge, B. M., and Hannegan, B. Achieving

a 100% Renewable Grid: Operating Electric Power Systems with Extremely High Levels of Variable Renewable Energy. *IEEE Power and Energy Magazine*, 15(2):61–73, 2017.

Kundur, P., Balu, N., and Lauby, M. *Power system stability and control*. McGraw-Hill, New York, 7 edition, 1994.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

Machowski, J., Bialek, J. W., and Bumby, J. R. *Power System Dynamics: Stability and Control*. Wiley, 2 edition, 2008.

Maidens, J. N., Kaynama, S., Mitchell, I. M., Oishi, M. M., and Dumont, G. A. Lagrangian methods for approximating the viability kernel in high-dimensional systems. *Automatica*, 49(7):2017–2029, 2013.

Perkins, T. J. and Barto, A. G. Lyapunov design for safe reinforcement learning control. *Journal of Machine Learning Research*, 3(4-5):803–832, 2002.

Steinberg, D., Bielen, D., Eichman, J., Eurek, K., Logan, J., Mai, T., McMillan, C., Parker, A., Vimmerstedt, L., and Wilson, E. Electrification and Decarbonization: Exploring U.S. Energy Use and Greenhouse Gas Emissions in Scenarios with Widespread Electrification and Power Sector Decarbonization. Technical Report July, National Renewable Energy Laboratory, 2017.

Wabersich, K. P. and Zeilinger, M. N. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129:109597, 2021.

Wachi, A. and Sui, Y. Safe reinforcement learning in constrained markov decision processes. *37th International Conference on Machine Learning, ICML 2020*, 2020.

Zhang, Y., Li, Y., Tomsovic, K., Djouadi, S., and Yue, M. Review on Set-Theoretic Methods for Safety Verification and Control of Power System. *IET Energy Systems Integration*, pp. 2–12, 2020.

Zheng, L., Shi, Y., Ratliff, L. J., and Zhang, B. Safe reinforcement learning of control-affine systems with vertex networks. *arXiv preprint: arXiv 2003.09488v1*, 2020.

Appendix

A. Proof of Proposition 1

First, we restate the definition of regulation map. The regulation map at a state $x \in \mathbf{S}$ is given by

$$\Omega(x) = \left\{ u \in \mathbb{R}^m : \begin{bmatrix} FB \\ H \end{bmatrix} u \leq \begin{bmatrix} g - FAx \\ \bar{u} \end{bmatrix} \right\} \tag{8}$$

where A is the state transition matrix, B is the input-to-state matrix, F and g define the target set $\mathbf{T} \subset \mathbf{S} : \mathbf{T} = \{x \in \mathbb{R}^n : Fx \leq g\}$, and H and $\bar{u}$ define the control set $\mathbf{U} := \{u \in \mathbb{R}^m : Hu \leq \bar{u}\}$. As stated earlier, the case of bounded uncertainty in A and B arising from noise or nonlinearities in the dynamics is readily handled as an extension and not considered here (Blanchini & Miani, 2015). The proof of Proposition 1 is stated next.

Proof. Fix $x \in \mathbf{S}$ and $u \in \mathbf{V}(x) := \{V\alpha : \alpha \in \mathbf{A}(x)\}$, so that $u = \sum_{i=1}^{p} \alpha_i v_i$, where v_i is the i^{th} column of V, for some $\alpha \in \mathbf{A}(x)$. We will show that $u \in \Omega(x)$. Evaluating the left-hand side of (8) yields

$$\begin{bmatrix} FB \\ H \end{bmatrix} u = \begin{bmatrix} FB \\ H \end{bmatrix} \sum_{i=1}^{p} \alpha_i v_i \tag{9}$$

$$= \sum_{i=1}^{p} \alpha_i \begin{bmatrix} FB \\ H \end{bmatrix} v_i \tag{10}$$

$$\leq \sum_{i=1}^{p} \alpha_i \begin{bmatrix} g - FAs_i \\ \bar{u} \end{bmatrix} \tag{11}$$

where (11) follows from the fact that for each $i = 1, \ldots, p$, (8) holds with $u = v_i$ and $x = s_i$, since v_i is a vertex of (and therefore an element of) $\Omega(s_i)$, the set of safe actions at s_i, a vertex of $\mathbf{S}$. Continuing from (11), we have

$$\sum_{i=1}^{p} \alpha_i \begin{bmatrix} g - FAs_i \\ \bar{u} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{p} \alpha_i (g - FAs_i) \\ \sum_{i=1}^{p} \alpha_i \bar{u} \end{bmatrix} \tag{12}$$

$$= \begin{bmatrix} g - FA \sum_{i=1}^{p} \alpha_i s_i \\ \bar{u} \end{bmatrix} \tag{13}$$

$$= \begin{bmatrix} g - FAx \\ \bar{u} \end{bmatrix} \tag{14}$$

where (13) follows from the fact that $\sum_{i=1}^{p} \alpha_i = 1$, and (14) follows from the fact that for $\alpha \in \mathbf{A}(x)$, $\sum_{i=1}^{p} \alpha_i s_i = x$. By the definition (8), we conclude $u \in \Omega(x)$. Since u was arbitrary, we conclude that $\mathbf{V}(x) \subset \Omega(x)$ for any $x \in \mathbf{S}$. □

B. Cyclic projections algorithm

Algorithm 2 describes in more detail the cyclic projections algorithm that is employed at the output layer of the policy network. Step 4 is the Euclidean projection onto the affine set $\mathbf{a}(x_t)$. For efficiency, we divide the projection onto the simplex into two steps. Step 5 is the Euclidean projection onto the nonnegative orthant $\mathbb{R}^p_+$, while step 6 is the projection from $\mathbb{R}^p_+$ to the unit simplex Δ_p with respect to the relative entropy function $D(x,y) = \sum_{i=1}^{p} (y_i - x_i + x_i \ln \frac{x_i}{y_i})$

Algorithm 2 Efficient cyclic projections to $\mathbf{A}(x_t)$

Require: State x_t, initial infeasible weights β_t, tolerance $\varepsilon > 0$
Ensure: $\alpha_t \in \mathbf{A}(x_t)$
1: $P := Y^T(YY^T)^{-1}$ {Projection matrix}
2: $\alpha^0 \leftarrow \text{softmax}(\beta_t)$ {Initialize in Δ_p}
3: **for** $k = 0 : k_{\max}$ **do**
4: $\quad \xi^k \leftarrow (I - PY)\alpha_t^k + Px_t$ {Project onto $\mathbf{a}(x_t)$}
5: $\quad \eta^k \leftarrow \max(\xi^k, \mathbf{0})$ {Project onto $\mathbb{R}^p_+$}
6: $\quad \alpha_t^{k+1} \leftarrow \frac{\eta^k}{\|\eta^k\|_1}$ {Project from $\mathbb{R}^p_+$ to Δ_p (Bregman, 1967)}
7: $\quad$ **if** $\|\alpha_t^{k+1} - \xi^k\| < \varepsilon$ **then**
8: $\quad\quad$ Return α_t^{k+1}
9: $\quad$ **end if**
10: **end for**

(Bregman, 1967). The reason for this approach is that steps 5 and 6 are each computationally efficient, whereas projection from all of $\mathbb{R}^p$ to the unit simplex is a more difficult problem. Using the softmax operation in place of steps 5 and 6 displayed worse convergence in simulations, but the softmax is still used to generate an initial point inside one of the target sets (the unit simplex).

C. RL algorithm implementation details

The policy network architecture consists of 2 hidden layers each with 256 nodes, and an output layer generating safe actions as described in Section 3. We train the policy network for 200 episodes, where each episode consists of 100 time steps (5 second duration). Each episode is initailized to a random starting point in the interior of the invariant set, and during each episode, the system is subjected to persistent, randomly generated disturbances. To generate the range of results displayed, we perform 20 "trials" training the network from scratch. Simulations were built upon an existing DDPG implementation from the GitHub repo `github.com/higgsfield/RL-Adventure-2`, and the power system model was built upon the OpenAI gym environment `pendulum-v0`.

The reward function is given by $J(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t + c_t$, where Q and R are diagonal matrices representing costs on states and actions and c_t is a penalty term. For the RCI network, c_t incentivizes the network to produce intial points α_t^0 in Algorithm 1 that are close to the target affine set $\mathbf{a}(x_t)$, in order to speed up convergence of the cyclic projection algorithm. This penalty is given by the Euclidean distance to the affine set, $c_t = \|P(Y\alpha_t^0 - x_t)\|_2$, where P is defined in Algorithm 2.

We compare the performance of the RCI policy network to that of a policy network with a soft penalty on constraint violations in place of the set-based safety guarantee. For this baseline method, the penalty term is given by $c_t = 0$ if $x_t \in \mathbf{X}$ and $c_t = \mu(x_t)$ else, where $\mu(x_t)$ is constant or increasing with respect to the extent of the constraint violation.