Lyapunov-Regularized Reinforcement Learning for Power System Transient Stability

Wenqi Cui and Baosen Zhang, *Member, IEEE*

***Abstract*—Transient stability of power systems is becoming increasingly important because of the growing integration of renewable resources. These resources lead to a reduction in mechanical inertia but also provide increased flexibility in frequency responses. Namely, their power electronic interfaces can implement almost arbitrary control laws. To design these controllers, reinforcement learning (RL) has emerged as a powerful method in searching for optimal non-linear control policy parameterized by neural networks. A key challenge is to enforce that a learned controller must be stabilizing. This letter proposes a Lyapunov regularized RL approach for optimal frequency control for transient stability in lossy networks. Because the lack of an analytical Lyapunov function, we learn a Lyapunov function parameterized by a neural network. The losses are specially designed with respect to the physical power system. The learned neural Lyapunov function is then utilized as a regularization to train the neural network controller by penalizing actions that violate the Lyapunov conditions. Case study shows that introducing the Lyapunov regularization enables the controller to be stabilizing and achieve smaller losses.**

***Index Terms*—Power system, frequency stability, reinforcement learning, stability.**

I. Introduction

TRANSIENT stability in power systems refers to the ability of a system to converge to an acceptable steady-state after a disturbance [1], [2]. With the increased penetration of renewable energy sources (RES), power systems have reduced inertia and transient stability is becoming increasingly important [3]. Meanwhile, RES are connected to the grid via electronic interfaces and can be controlled freely by inverters to implement almost arbitrary control laws. Therefore, instead of linear droop frequency response found in conventional generators, the response of the inverter-based RES can be optimized [4].

Transient stability describes how frequency changes in a system with a large deviation of operating states, and use the full nonlinear AC power flow equations [1]. Two challenges emerge in controller design. Firstly, we are searching over an infinite dimensional function space. Secondly, the controllers should be stabilizing, which is a nontrivial constraint to enforce algorithmically for nonlinear systems.

A popular way to address the first challenge is to parameterize the controllers (e.g., using a neural network) and training them using reinforcement learning (RL) [5]. Abundant algorithms, including Q-learning, deep direct reinforcement learning (DDPG), actor-critic, have been proposed for optimal control (see, e.g., [6] and the reference within). References [7]–[10] apply these algorithms for power system frequency regulation. However, the stabilizing requirement of the controllers is not considered in these works.

The challenge of ensuring controllers are stable is more difficult to address. If a Lyapunov function is available, it can potentially provide analytical constraints on the controller. For lossless power systems, using a well-known energy function [2], [11], our previous work in [12] showed how to impose structural constraints on the neural network controllers such that they are guaranteed to be stabilizing. Unfortunately, for lossy networks, there are no known analytic energy functions [1].

If analytical Lyapunov functions are not available, a natural approach would be to learn a Lyapunov function to facilitate controller design. For example, given input/output data and the assumption that the underlying system is stable, [13] learns a Lyapunov function jointly with learning the system model to find stable system dynamics. The work in [14] uses satisfiability modulo theories solvers to formally verify a function satisfies the Lyapunov conditions. However, it is only currently computational tractable for small systems. Reference [15] applies this method to distribution system by aggregating networked microgrids as a single node. Moreover, the above works focus on verifying a system is stable and do not include controller design.

This letter proposes a Lyapunov regularization approach to guide the training of neural network controller for primary frequency response in lossy power systems. We learn a Lyapunov function parameterized by a neural network. The loss function for training the neural Lyapunov function is designed to satisfy the positive definiteness of its value and the negative definiteness of its Lie derivative. Existing methods in [13]–[15] weigh all the states equally in the loss function, but this will cause the sub-optimum of Lyapunov function near the equilibrium since the magnitude of states' time derivative shrink quickly when approaching the equilibrium. Considering

Manuscript received March 4, 2021; revised May 7, 2021; accepted May 26, 2021. Date of publication June 9, 2021; date of current version June 30, 2021. The work of Wenqi Cui and Baosen Zhang was supported in part by the National Science Foundation under Grant ECCS-1930605 and Grant ECCS-1942326, and in part by the Washington Clean Energy Institute. Recommended by Senior Editor R. S. Smith. *(Corresponding author: Wenqi Cui.)*

The authors are with the Department of Electrical and Computer Engineering, University of Washington Seattle, Seattle, WA 98195 USA (e-mail: wenqicui@uw.edu; zhangbao@uw.edu).

Digital Object Identifier 10.1109/LCSYS.2021.3088068

that the states near the equilibrium are more important for control, we specially design the loss function such that the area around the equilibrium is emphasized.

The neural Lyapunov function is utilized as a regularization to train the neural network controller by penalizing actions that violate the Lyapunov conditions. The regularized RL is integrated in the recurrent neural network (RNN) based framework in our previous work to increase its training efficiency [12]. Simulation results show that the learned function satisfies the Lyapunov conditions for almost all points in the state space, thus making it a good tool for regularization. Case study shows that introducing the Lyapunov regularization enables the controller to achieve smaller loss. More importantly, a controller designed without regularization can lead to unstable behaviors. Code and data described in this letter are available at https://github.com/Wenqi-Cui/Lyapunov-Regularized-RL. An important future work is to verify whether the learned function satisfies the Lyapunov conditions for all points in a region.

II. Model and Problem Formulation

A. Frequency Dynamics

Let N be the number of buses and $\mathcal{E}$ be the set of transmission lines connecting the buses. The susceptance and conductance of the line $(i, j) \in \mathcal{E}$ are $B_{ij} = B_{ji}$ and $G_{ij} = G_{ji}$, respectively; and 0 if the buses are not connected. We use the Kron reduced model to aggregate load buses into generator buses [16]. We assume that each bus i has the conventional inertia M_i and the damping from synchronous generator and loads is denoted as D_i [17], [18]. Denote the generator power and load of bus i as $P_{g,i}$ and $P_{l,i}$, respectively. Then, $P_i = P_{g,i} - P_{l,i}$ represents the net power injection of bus i. We assume that the synchronous generation are set to their nominal operating points and have some inertia and damping values. Our control comes from the inverter-connected resources such as storage and wind turbines [19]. Without loss of generality, we assume that each bus has an inverter-connected resources (the actuation bounds can be set to zero if a resource is not present).

The angle and frequency deviation of bus i are δ_i and ω_i, respectively. We assume that the bus voltage magnitudes are 1 p.u. and the reactive power flows are ignored. The dynamics of the power system is represented by the swing equation [20]

$$\dot{\delta}_i = \omega_i \quad , \forall i = 1, \ldots, N \tag{1a}$$

$$M_i\dot{\omega}_i = P_i - D_i\omega_i - u_i(\omega_i) - \sum_{j=1, j\neq i}^{N} B_{ij}\sin(\delta_i - \delta_j) - \sum_{j=1, j\neq i}^{N} G_{ij}\cos(\delta_i - \delta_j), \quad \forall i = 1, \ldots, N \tag{1b}$$

where $u_i(\omega_i)$ is the controller that changes active power to provide primary frequency response. Because power systems do not have real-time communication infrastructure, we restrict u_i to be a static feedback controller where only its local frequency measurement ω_i is available. We envision the control is provided by renewable energy resources such as batteries and solar PV. In the primary frequency regulation timescale from 100ms to few seconds for primary frequency regulation, the main limitation on actuation comes from power injection constraints.

B. Optimization Problem Formulation

The objective is to minimize the cost on frequency deviations and the control effort. Here we use frequency nadir, which is the maximum of $|\omega_i(t)|$ over the time horizon from 0 to T defined as $\|\boldsymbol{\omega_i}\|_\infty = \sup_{0\leq t\leq T}|\omega_i(t)|$ [21]. We use a quadratic cost for the control actions defined by $\|\boldsymbol{u_i}\|_2^2 = \frac{1}{T}\int_{t=0}^{T}(u_i(t))^2 dt$ [16], [22]. We aim to find an optimal stabilizing controller $\boldsymbol{u}(\cdot)$ by solving (2).

$$\min_{\mathbf{u}} \sum_{i=1}^{N} \|\boldsymbol{\omega_i}\|_\infty + \gamma\|\boldsymbol{u_i}\|_2^2 \tag{2a}$$

$$\text{s.t.} \quad (1a) - (1b) \tag{2b}$$

$$\underline{u}_i \leq u_i(\omega_i) \leq \overline{u}_i \tag{2c}$$

$$u_i(\omega_i) \text{ is stabilizing} \tag{2d}$$

where γ is a tradeoff parameter between cost of frequency deviation and action. The swing equations are in (2b). The controllers are power limited within as shown in (2c). We impose the condition that the controller should be stabilizing in (2d). Constraints (2b)-(2d) hold for the time t from 0 to T. Other objective functions can also be used (e.g., l_1 penalty on total frequency deviation and the rate of change of frequency) without changing the framework.

Problem (2) is challenging to solve by conventional control techniques and we will use RL to find $\boldsymbol{u}(\cdot)$. The key difficulty is to quantify the stability requirement in (2d). We mitigate this difficulty by using a Lyapunov function, which provides algebraic conditions for (2d). Since a Lyapunov function is not known for lossy systems [1], we show how one can be learned in the next section.

III. Learning a Lyapunov Function

A. Lyapunov Conditions

From standard system theory, the Lyapunov function need to satisfy conditions on its value and its Lie derivatives [23]. Let the state space be $\mathcal{D} = \{(\delta, \omega)|\delta = (\delta_1, \ldots, \delta_N), \omega = (\omega_1, \ldots, \omega_N)\}$. The state transition dynamics (1) is written as $(\dot{\delta}, \dot{\omega}) = f_u(\delta, \omega)$, where f_u stands for the state transition function with respect to the controller u. Using the notation from [14], we have

Definition 1 (Lie Derivatives): The Lie derivative of the continuously differentiable scalar function $V : \mathcal{D} \to \mathbb{R}$ over the vector field f_u is defined as

$$\nabla_{f_u} V(\delta, \omega) = \sum_{i=1}^{N} \frac{\partial V(\delta, \omega)}{\partial \delta_i}\dot{\delta}_i + \frac{\partial V(\delta, \omega)}{\partial \omega_i}\dot{\omega}_i. \tag{3}$$

It measures the rate of change of V along the direction of system dynamics. The next proposition is standard.

Proposition 1 (Lyapunov Function and Asymptotic Stability): Consider a controlled system described by (1) with equilibrium at (δ^*, ω^*). Suppose there exists a continuously differentiable function $V : \mathcal{D} \to \mathbb{R}$ that satisfies:

$$V(\delta, \omega) > V(\delta^*, \omega^*) \quad \forall(\delta, \omega) \in \mathcal{D}\backslash\{(\delta^*, \omega^*)\} \tag{4a}$$

$$\nabla_{f_u} V(\delta,\omega) < 0 \quad \forall (\delta,\omega) \in \mathcal{D}\backslash\{(\delta^*,\omega^*)\} \tag{4b}$$
$$\nabla_{f_u} V(\delta^*,\omega^*) = 0. \tag{4c}$$

Then the system is asymptotically stable at the equilibrium.

In this letter, Lyapunov function is parameterized using neural network with weights ϕ, and written as $V_\phi(\delta,\omega)$. For differentiability, we use Exponential Linear Unit (ELU) activation functions. Note that $V_\phi(\delta,\omega)$ is purely a function of the state variable (δ,ω), while $\nabla_{f_u} V(\delta,\omega)$ will be affected by the controller u_i through the term $\dot{\omega}_i$ in (3). Therefore, only $\nabla_{f_u} V(\delta,\omega)$ will be utilized to regularize controller once it is learned.

B. Learning the Lyapunov Function

The condition (4a) is easy to be satisfied if we explicitly engineer the structure of $V_\phi(\delta,\omega)$. To name a few, $V_\phi(\delta,\omega)$ can be formulated using a convex function achieving the minimum at the equilibrium. Or, given an arbitrary function $g(\delta,\omega)$ and positive scalar ϵ, (4a) can be enforced by taking $V_\phi(\delta,\omega) = (g(\delta,\omega) - g(\delta^*,\omega^*))^2 + \epsilon\|(\delta,\omega) - (\delta^*,\omega^*)\|_2$. However, such parameterization may be too restrictive and make it hard to satisfy (4b). Therefore, we do not explicitly engineer the structure in satisfying condition (4a).

In this letter we use loss functions to penalize violations of (4a)-(4c). Training is implemented in a batch updating style where the number of batch is H and the state of the h-th batch is randomly generated $(\delta^h,\omega^h) \in \mathcal{D}$ for $h = 1,\ldots,H$. The losses are designed with respect to the following considerations:

1) *Avoid overfitting when $\dot{\delta}$ and $\dot{\omega}$ are large:* To satisfy (4b), the loss term need to encourage $\nabla_{f_u} V(\delta,\omega)$ to be negative and penalize its positive values. A loss that weighs all points in the space equally leads $\nabla_{f_u} V(\delta,\omega)$ to have very negative values when δ and ω are far away from the equilibrium, and may violate (4b) for points close to the equilibrium. This contradicts the premise that the small region around the equilibrium should be stabilizing. Therefore, we design the loss term with $\nabla_{f_u} V(\delta,\omega)$ to be

$$l_1(\phi) = \frac{1}{H}\sum_{h=1}^{H} \tanh\Big(\nabla_{f_u} V_\phi(\delta^h,\omega^h)\Big) \times \exp\left(-\frac{\|(\delta^h,\omega^h) - (\delta^*,\omega^*)\|_2}{\mu}\right) \tag{5}$$

where the term $\tanh(\nabla_{f_u} V_\phi(\delta^h,\omega^h))$ avoid the overfit of $\nabla_{f_u} V(\delta^h,\omega^h)$ to be extremely negative We use tanh function to make $l_1(\phi)$ to have the same sign with as $\nabla_{f_u} V_\phi(\delta^h,\omega^h)$. The term $\exp(-\frac{\|(\delta^h,\omega^h)-(\delta^*,\omega^*)\|_2}{\mu})$ emphasis the importance of (δ^h,ω^h) closer to the equilibrium. The hyper-parameter μ controls rate of decay.

2) *Penalty term with $(V_\phi(\delta^*,\omega^*)) - V_\phi(\delta,\omega))$:* In order to satisfying condition (4a), $V_\phi(\delta^h,\omega^h)$ that is smaller than $V_\phi(\delta^*,\omega^*)$ need to be penalized. For points that satisfy $V_\phi(\delta^h,\omega^h) > V_\phi(\delta^*,\omega^*)$, we do not consider the magnitude of the difference. Therefore, we use ReLU function (written as $\sigma(\cdot)$) to penalize positive $(V_\phi(\delta^*,\omega^*)) - V_\phi(\delta,\omega))$. Define the loss term as:

$$l_2(\phi) = \frac{1}{H}\sum_{h=1}^{H} \sigma\Big(-V_\phi(\delta^h,\omega^h) + V_\phi\big(\delta^*,\omega^*\big)\Big) \tag{6}$$

3) *Penalty term with $\nabla_{f_u} V_\phi(\delta^*,\omega^*)$:* This term is employed to mitigate numerical errors. We design a extra loss term to penalize on the value of $\nabla_{f_u} V_\phi(\delta^*,\omega^*)$ as:

$$l_3(\phi) = \big(\nabla_{f_u} V_\phi(\delta^*,\omega^*)\big)^2 + \sigma\big(\nabla_{f_u} V_\phi(\delta^*,\omega^*)\big) \tag{7}$$

where $(\nabla_{f_u} V_\phi(\delta^*,\omega^*))^2$ guarantee the small magnitude of $\nabla_{f_u} V_\phi(\delta^*,\omega^*)$. Considering that $\nabla_{f_u} V_\phi(\delta^h,\omega^h)$ should never be positive, we use ReLU function $\sigma(\nabla_{f_u} V_\phi(\delta^*,\omega^*))$ to guarantee that $\nabla_{f_u} V_\phi(\delta^*,\omega^*)$ is negative close zero. This way, the zero action at the equilibrium is guaranteed to satisfy Lyapunov conditions.

Combining (5)-(7), the total loss function is

$$L_q(\boldsymbol{\phi}) = q_1 l_1(\phi) + q_2 l_2(\phi) + q_3 l_3(\phi) \tag{8}$$

where q_1, q_2, q_3 are hyperparameters balancing the loss terms, with q_3 tuned to be much larger than the others. For the specific problem in this letter, we found that letting magnitude of q_1 to be slight larger than q_2 (e.g., q_1 to be 1.5 times of q_2) leads to most samples satisfy Lyapunov conditions. Note that the equilibrium (δ^*,ω^*) is obtained from the steady state in (1) and we fix the equilibrium in training. Of course the equilibrium changes if the load or the parameters changes. More specifically, $\omega^* = 0$ always while δ^* varies. Since we only use the learned function as a regularization to train a controller, we are robust to changes in the equilibrium point. If the learned function is used to certify stability, then the changes in equilibrium should be carefully accounted for.

Algorithm 1 Learning Neural Lyapunov Function

Require: Learning rate α, number of episodes I, state transfer function (1), hyperparameters in (5)-(8)
Input: Droop coefficient l_i for the i-th bus, $i = 1,\cdots,N$
Initialisation :Initial weights ϕ for neural network
for *episode* $= 1$ to I **do**
  Generate batch state samples δ^h, ω^h for the h-th batch, $h = 1,\cdots,H$
  If $\varrho > \bar{\varrho}$, add the samples violates Lyapunov condition $\{(\delta,\omega)\} \leftarrow \{(\delta,\omega),(\hat{\delta},\hat{\omega})\}$
  Compute $f_u(\delta,\omega)$ for the sample states with linear droop control using (1)
  Calculate $V_\phi(\delta,\omega)$ and $\nabla_{f_u} V_\phi(\delta,\omega)$
  Identify the states $\left(\hat{\delta},\hat{\omega}\right)$ that does not satisfy Lyapunov condition and its percentage ϱ
  Calculate total loss of all the batches using (5)-(8)
  Update weights in the neural network by passing *Loss* to Adam optimizer: $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha\,\mathrm{Adam}(Loss)$
end for

C. Algorithm With Active Sampling

The goal for training the neural Lyapunov function is to make larger proportional of the batch samples satisfy the conditions (4). The pseudo-code for our proposed method is given

in Algorithm 1. A linear controller is used to initialize training. Let ϱ be the proportion of samples that satisfy the conditions (4). After most of the samples (e.g., $\varrho > 95\%$) have already satisfied the conditions, it would be difficult to improve the neural Lyapunov function further since the loss function will remain almost unchanged even though ϱ increases slightly. We augment the training performance by collecting samples violate (4) and add them to the next batch of training. Moreover, since we care more about the region with smaller frequency deviation, we also let 50% of the batch states to be sampled from regions close to equilibrium. This way, the neural Lyapunov can improve efficiently and ϱ can reach 99.9% in the end for both the region close and away from the equilibrium. Adam algorithm is adopted to update weights $\boldsymbol{\phi}$ in each episode.

IV. Learning Neural Network Controller With Lyapunov Regularization

We propose to use the learned neural Lyapunov function to guide the training of neural network controller. We adopt the neural Lyapunov function as an additional regularization that is used during the training process of the neural network controller. The real-time control policy is computed through the feedforward neural networks where the input is the local frequency deviation and the weights are trained offline. Note that we may be able to achieve better performance through a projection if the Lyapunov conditions are violated. However, such a projection requires information of all the state variables in real-time, which is unrealistic for the power system with large numbers of nodes and limited communication.

A. Lyapunov Regularization

Given a Lyapunov function, Proposition 2 illustrates the condition for locally exponentially stability [24].

Proposition 2 (Locally Exponentially Stable Condition): For the function $V : \mathcal{D} \to \mathbb{R}$ satisfying (4), if there is constant $\beta > 0$ such that for all $(\delta, \omega) \in \mathcal{D}$ we have

$$\nabla_{f_u} V(\delta, \omega) \leq -\beta\big(V(\delta, \omega) - V(\delta^*, \omega^*)\big) \tag{9}$$

Then, the equilibrium is locally exponentially stable.

In order to satisfy (9) with the neural network controller, we propose a Lyapunov regularization approach that the action is penalized if this inequality does not hold. Compared with traditional regularization (e.g., lasso, ridge) or penalty term on large state magnitude, we do not add regularization uniformly to all the weights or actions. Instead, the action is only penalized when (9) is violated. The regularization term is

$$R_\phi(u_\theta) = \sigma\big(\nabla_{f_u} V_\phi(\delta, \omega) + \beta(V_\phi(\delta, \omega) - V_\phi(\delta^*, \omega^*))\big).$$

B. Controller and Architecture

The formulation of controller and the training architecture is from our previous work [12]. For completeness, we reiterate the key design in this subsection. The work in [12] showed that a controller mapping frequency to active power needs to be a function that is monotonic, increasing and goes through the origin. To this end, we explicitly engineer the

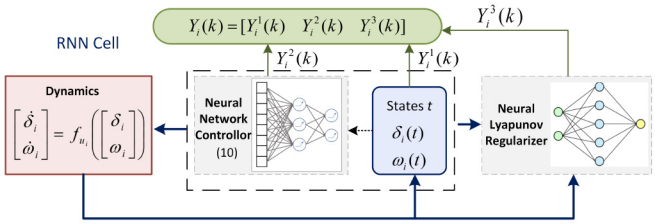


Fig. 1. Structure of RNN for frequency control problem.

neural network controller with a stacked-ReLU structure and represented as (10)

$$u_i(\omega_i) = s_i\sigma(\mathbf{1}\omega_i + b_i) + z_i\sigma(-\mathbf{1}\omega_i + c_i) \tag{10a}$$

$$\text{where} \quad \sum_{j=1}^{l} s_i^j \geq 0, \quad \sum_{j=1}^{l} z_i^j \leq 0, \quad \forall l = 1, 2, \ldots, m \tag{10b}$$

$$b_i^1 = 0, b_i^l \leq b_i^{(l-1)}, \quad \forall l = 2, 3, \ldots, m \tag{10c}$$

$$c_i^1 = 0, c_i^l \leq c_i^{(l-1)}, \quad \forall l = 2, 3, \ldots, m \tag{10d}$$

where m is the number of hidden units and $\mathbf{1} \in \mathbb{R}^m$ is the all 1's column vector. Variables $s_i = [s_i^1 \quad s_i^2 \quad \cdots \quad s_i^m]$ and $z_i = [z_i^1 \quad z_i^2 \quad \cdots \quad z_i^m]$ are the weight vector of bus i; $b_i = [b_i^1 \quad b_i^2 \quad \cdots \quad b_i^m]^\intercal$ and $c_i = [c_i^1 \quad c_i^2 \quad \cdots \quad c_i^m]^\intercal$ are the corresponding bias vector. The variables to be trained are weights $\boldsymbol{\theta} = \{s, b, z, c\}$ in (10).

To obtain the trajectory for training the controller, we discretize dynamics (1) with step size Δt. We use k and K to represent the discrete time and total number of stages, respectively. The neural network controller is then denoted as $u_{\theta_i}(\omega_i)$. From (1), $\omega_i(k)$ in each timestep k is a function of $\omega_i(k-1)$ and $u_{\theta_i}(\omega_i(k-1))$, which is then a function of $\omega_i(k-2)$ and $u_{\theta_i}(\omega_i(k-2)$. This means that computing gradient of $u_{\theta_i}(\omega_i(k)$ with respect to θ_i needs the chain-rule from the step k all the way to the first time step for all $k = 0, \ldots, K$. To mitigate the computation burden caused by the subsequent application of chain-rule, we proposed a RNN-based framework to integrate the state transition dynamics (1) implicitly.

As illustrated in Fig. 1, the state of RNN cell of bus i is set to be (δ_i, ω_i). The system dynamics (1) is set as the transition function of RNN cell. At each time k, state of RNN cell and the current action from neural network controller will go through the transition dynamics to calculate the state of the time $k+1$. The state ω and action u constitute the first two component of output where $Y_i^1(k) = \omega_i(k)$ and $Y_i^2(k) = (u_{\theta_i}(\omega_i(k)))^2$. The total state information and time derivative information are simultaneously send as input into Neural Lyapunov function to calculate the Lyapunov regularization term, written as $Y_i^3(k) = \sigma(\nabla_{f_u} V_\phi(\delta, \omega) + \beta(V_\phi(\delta, \omega) - V_\phi(\delta^*, \omega^*)))/N$.

The loss function is formulated to be equivalent with the objective function (2a) plus the Lyapunov regularization as:

$$Loss = \sum_{i=1}^{N} \max_k |Y_i^1(k)| + \frac{1}{K}\sum_{k=1}^{K}\Big(\gamma Y_i^2(k) + \lambda Y_i^3(k)\Big).$$

C. Algorithm to Train Neural Network Controller

The pseudo-code for learning the neural network controller is given in Algorithm 2. Training is implemented in a batch

Algorithm 2 Reinforcement Learning With RNN

Require: Learning rate α, batch size H, total time stages K, number of episodes I, parameters in optimal frequency control problem (2)

Input: The neural Lyapunov function $V_\phi(\delta, \omega)$

Initialisation :Initial weights θ for control network

1: **for** *episode* $= 1$ to I **do**

2: Generate initial states $\delta_i^h(0), \omega_i^h(0)$ for the i-th bus in the h-th batch, $i = 1, \cdots, N$, $h = 1, \cdots, H$

3: Reset the state of cells in each batch as the initial value $x_i^h \leftarrow \{\delta_i^h(0), \omega_i^h(0)\}$.

4: RNN cells compute through K stages to obtain output $\{Y_{h,i}(0), Y_{h,i}(1), \cdots, Y_{h,i}(K)\}$

5: Calculate total loss of all the batches $Loss = \frac{1}{H}\sum_{h=1}^{H}\sum_{i=1}^{N}\max_{k=0,\cdots,K}|Y_{h,i}^1(k)| + \gamma\frac{1}{K}\sum_{k=1}^{K}Y_{h,i}^2(k) + \lambda\frac{1}{K}\sum_{k=1}^{K}Y_{h,i}^3(k)$.

6: Update weights in the neural network by passing *Loss* to Adam optimizer: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha\text{Adam}(Loss)$

7: **end for**

updating style where the h-th batch initialized with randomly generated initial states $\{\delta_i^h(0), \omega_i^h(0)\}$ for all $i = 1, \ldots, N$. The evolution of states in K stages will be computed through structure of RNN as shown by Fig. 1. Although Algorithms 1 and 2 can be iterated to make further update, we did not see an obvious improvement in simulation.

V. Case Study

Case studies are conducted on the IEEE New England 10-machine 39-bus (NE39) power network [25] to illustrate the effectiveness of the proposed method. We visualized the learned Lyapunov function and its Lie derivative. Then we show that regularization is necessary, in the sense that a controller learned without it can be unstable. Lastly, we show the training losses.

A. Simulation Setting

The step size for the discrete simulation is 0.02 (20ms) and K is 100. Power injection P_i are at their nominal values, the bound on action $\overline{u}_i$ is uniformly distributed in $[0.8P_i, P_i]$ and γ is 0.005. The base power unit is 100 MVA. The detailed parameters are given in [26].

B. Visualization Lyapunov Function and the Lie Derivative

To visualize the Lyapunov function with a large number of state variables, we fix all the states at their equilibrium value and vary the state variable for one generator bus. Fig. 2 illustrates the value of Lyapunov function and Lie derivative with the variation of δ and ω in generator bus 5. The Lyapunov function $V(\delta, \omega)$ achieves the minimum at the equilibrium point and the Lie derivative $\nabla_{f_u}V_\phi(\delta, \omega)$ is smaller than zero for most points. After training, $\nabla_{f_u}V_\phi(\delta, \omega)$ is slightly positive for about 0.1% of the samples.

C. Performance Comparison

Under the same hyperparameters and RNN structure, we train the neural network controller with Lyapunov regularization (labeled as RNN-Lyapunov) and without Lyapunov regularization (labeled as RNN-w.o.-Lyapunov), respectively. We test the effect of large deviation in initial operating points and sudden changes in topology.

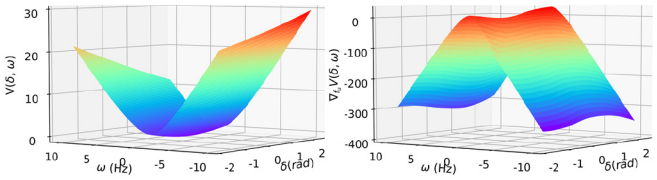
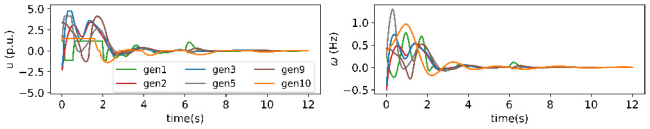
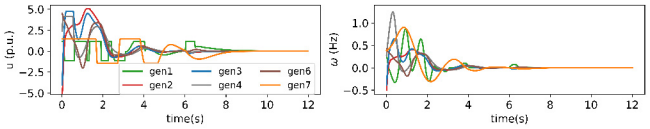


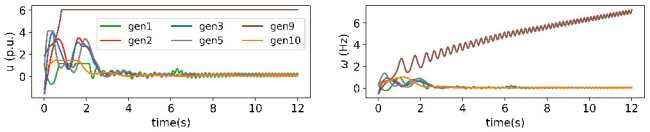
Fig. 2. Neural Lyapunov function (left) and Lie derivative (right) when changing (δ, ω) in generator 5 and keep state variable of other generators at the equilibrium value.



(a) u (left) and ω (right) for RNN with Lyapunov regularization



(b) u (left) and ω (right) for linear droop control



(c) u (left) and ω (right) for RNN without Lyapunov regularization

Fig. 3. Dynamics of control action u and frequency deviation w in selected generator buses corresponding to (a) RNN-Lyapunov (b) linear droop (c) RNN-w.o.-Lyapunov. The neural network controller trained with Lyapunov regularization achieve smaller control cost than linear droop control, and better stablizing performance than that without Lyapunov regularization.

At time t=0, the system starts from some initial conditions that deviates from the equilibrium. At time t=6s, the lines between buses 1 and 39, and 2 and 3 are disconnected. The dynamics of the system under the controller obtained by RNN-Lyapunov, linear droop control and RNN-w.o.-Lyapunov are shown in Fig. 3. Both RNN-Lyapunov and linear droop control stablize the system, while RNN-w.o.-Lyapunov leads to loss of synchronicity in gen 9 as shown in Fig. 3(c). Compared with dynamics of linear droop control in Fig. 3(b), RNN-Lyapunov in Fig. 3(a) achieve similar frequency deviation while using smaller control action.

After losing two lines at time t=6s, the system experience frequency deviation of approximate 0.1 Hz and return to stable state within 2s for RNN-Lyapunov (Fig. 3(a)). Therefore, the proposed RNN-Lyapunov approach is robust to topology changes. In the longer online version [12], we show that the weights for the previous topology can serve as warm start for training with the new topology.

We further compare RNN-Lyapunov and RNN-w.o.-Lyapunov with the benchmark of linear droop control, where the droop coefficient is obtained by solving problem (2) using

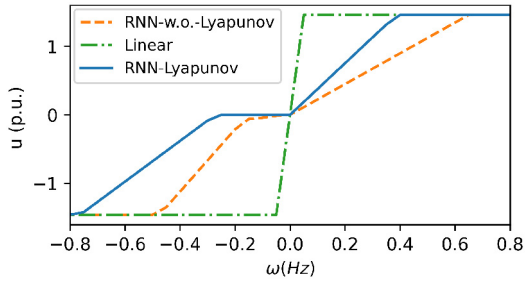


Fig. 4. Control action u of RNN-Lyapunov, RNN-w.o.-Lyapunov and Linear droop control for generator bus 10. Lyapunov regularization leads to different non-linear control law.

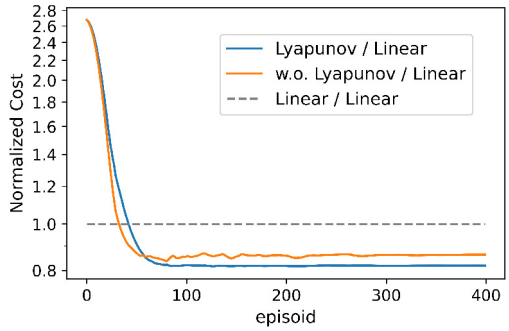


Fig. 5. Normalized cost along the episode during the training of neural network controllers. RNN-Lyapunov and RNN-w.o.-Lyapunov reduce the cost by 19% and 14% compared with linear droop control.

fmincon function of MATLAB [12]. Fig. 4 illustrates the control policy obtained from the three methods. Compared with linear droop control, the stacked-ReLU neural network learns a highly non-linear controller. The average cost normalized by the cost of linear droop control along episode is shown in Fig. 5. Both RNN Lyapunov and RNN-w.o.-Lyapunov converge in approximate 150 episodes. After convergence, RNN-Lyapunov reduces the cost by approximate 19% compared with linear droop control. Additional numerical validation with power disturbance and larger test system can be found in the longer online version [26].

VI. Conclusion

This letter proposes a Lyapunov regularization approach to guide the training of neural network controller for primary frequency response. A function paramertized as a neural network is learned to overcome the non-existence of analytical Laypunov functions for lossy power networks. By integrating the neural Lyapunov function as a regularization term for the training of neural network controller in RL, control actions that violate Lyapunov conditions are penalized. Case studies verify introducing Lyapunov regularization enable the controller to be stabilizing and achieve smaller losses, whereas controllers trained without regularization can fail to stabilize the system. An important future direction is to understand the region of attraction better in the context of learning controllers.

References

[1] H.-D. Chiang, "Study of the existence of energy functions for power systems with losses," *IEEE Trans. Circuits Syst.*, vol. CS-36, no. 11, pp. 1423–1429, Nov. 1989.

[2] A. Arapostathis, S. Sastry, and P. Varaiya, "Global analysis of swing dynamics," *IEEE Trans. Circuits Syst.*, vol. CS-29, no. 10, pp. 673–679, Oct. 1982.

[3] Y. Jiang, R. Pates, and E. Mallada, "Dynamic droop control in low-inertia power systems," *IEEE Trans. Autom. Control*, early access, Oct. 29, 2020, doi: 10.1109/TAC.2020.3034198.

[4] B. B. Johnson, S. V. Dhople, A. O. Hamadeh, and P. T. Krein, "Synchronization of parallel single-phase inverters with virtual oscillator control," *IEEE Trans. Power Electron.*, vol. 29, no. 11, pp. 6124–6138, Nov. 2014.

[5] X. Chen, G. Qu, Y. Tang, S. Low, and N. Li, "Reinforcement learning for decision-making and control in power systems: Tutorial, review, and vision," 2021. [Online]. Available: arXiv:2102.01168.

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA, USA: MIT Press, 2018.

[7] Z. Yan and Y. Xu, "Data-driven load frequency control for stochastic power systems: A deep reinforcement learning method with continuous action search," *IEEE Trans. Power Syst.*, vol. 34, no. 2, pp. 1653–1656, Mar. 2019.

[8] C. Chen, M. Cui, F. F. Li, S. Yin, and X. Wang, "Model-free emergency frequency control based on reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2336–2346, Apr. 2021.

[9] J. Duan *et al.*, "Deep-reinforcement-learning-based autonomous voltage control for power grid operations," *IEEE Trans. Power Syst.*, vol. 35, no. 1, pp. 814–817, Jan. 2019.

[10] D. Ernst, M. Glavic, F. Capitanescu, and L. Wehenkel, "Reinforcement learning versus model predictive control: A comparison on a power system problem," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 39, no. 2, pp. 517–529, Apr. 2009.

[11] I. Dobson and H.-D. Chiang, "Towards a theory of voltage collapse in electric power systems," *Syst. Control Lett.*, vol. 13, no. 3, pp. 253–262, 1989.

[12] W. Cui and B. Zhang, "Reinforcement learning for optimal frequency control: A Lyapunov approach," 2020. [Online]. Available: arXiv:2009.05654.

[13] G. Manek and J. Z. Kolter, "Learning stable deep dynamics models," 2020. [Online]. Available: arXiv:2001.06116.

[14] Y.-C. Chang, N. Roohi, and S. Gao, "Neural Lyapunov control," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 3240–3249.

[15] T. Huang, S. Gao, and L. Xie, "Transient stability assessment of networked microgrids using neural Lyapunov methods," 2020. [Online]. Available: arXiv:2012.01333.

[16] A. Ademola-Idowu and B. Zhang, "Frequency stability using inverter power control in low-inertia power systems," *IEEE Trans. Power Syst.*, vol. 36, no. 2, pp. 1628–1637, Mar. 2021.

[17] J. Machowski, Z. Lubosny, J. W. Bialek, and J. R. Bumby, *Power System Dynamics: Stability and Control*. Hoboken, NJ, USA: Wiley, 2020.

[18] C. Zhao, E. Mallada, S. Low, and J. Bialek, "A unified framework for frequency control and congestion management," in *Proc. IEEE Power Syst. Comput. Conf. (PSCC)*, 2016, pp. 1–7.

[19] E. Muljadi, V. Gevorgian, M. Singh, and S. Santoso, *Understanding Inertial and Frequency Response of Wind Power Plants*. Philadelphia, PA, USA: IEEE, 2012.

[20] P. Kundur, N. J. Balu, and M. G. Lauby, *Power System Stability and Control*, vol. 7. New York, NY, USA: McGraw-Hill, 1994.

[21] D. Tabas and B. Zhang, "Optimal L-infinity frequency control in microgrids considering actuator saturation," 2019. [Online]. Available: arXiv:1910.03720.

[22] F. Dörfler, M. Chertkov, and F. Bullo, "Synchronization in complex oscillator networks and smart grids," *Proc. Nat. Acad. Sci. USA*, vol. 110, no. 6, pp. 2005–2010, 2013.

[23] S. Sastry, *Nonlinear Systems: Analysis, Stability, and Control*. New York, NY, USA: Springer, 2013.

[24] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, NJ, USA: Princeton Univ. Press, 2010.

[25] J. H. Chow and K. W. Cheung, "A toolbox for power system dynamics and control engineering education and research," *IEEE Trans. Power Syst.*, vol. 7, no. 4, pp. 1559–1564, Nov. 1992.

[26] W. Cui and B. Zhang, "Lyapunov-regularized reinforcement learning for power system transient stability," 2021. [Online]. Available: arXiv:2103.03869.