Reinforcement Learning for Optimal Frequency Control: A Lyapunov Approach

Wenqi Cui and Baosen Zhang

arXiv:2009.05654v3 [eess.SY] 11 Mar 2021

***Abstract*—The increase in penetration of inverter-based resources provides us with more flexibility in frequency regulation compared to conventional linear droop controllers. Using their power electronic interfaces, inverter-based resources can realize almost arbitrary responses to frequency changes. Reinforcement learning (RL) has emerged as a popular method to design these nonlinear controllers to optimize a host of objective functions.**

The key challenge with RL based approaches is how to enforce the constraint that the learned controller should be stabilizing. In addition, training these controllers is nontrivial because of the time-coupled dynamics of power systems. In this paper, we propose to explicitly engineer the structure of neural network-based controllers such that they guarantee system stability by design. This is done by constraining the network structures using a well-known Lyapunov function. A recurrent RL architecture is used to efficiently train the controllers. The resulting controllers only use local information and outperform linear droop as well as controllers trained using existing state-of-the-art learning approaches.

***Index Terms*—Frequency control, reinforcement learning, Lyapunov stability, recurrent neural network.**

I. Introduction

DROOP responses of synchronous machines are of fundamental importance to power system frequency stability. Because of the inertial characteristic of conventional generators, droop controls are typically linear in the frequency deviations [1]. However, for the common performance metrics adopted in practice for frequency deviation and control costs [2]–[4], the linear controller may not be optimal. With the increase in penetration of inverter-based resources such as solar PV and storage, there are more flexibility in control because of the ability of these devices to quickly adjust their power output setpoints [5]. These resources offer the potential to implement more complex control strategies with better performances compared to linear droop response.

Inverters, being solid state electronic devices, can implement almost arbitrary control laws, subject to some actuation limits [6]. Hence the bottleneck has shifted from hardware capability to algorithm design [7]. Even though inverters can implement almost any control strategy, designing nonlinear optimal controllers is not trivial. A controller is a function that maps frequency deviation to active power injection, but optimizing over this function space is an infinite dimensional problem. Importantly, the controller need to be stabilizing, introducing a nonlinear constraint. Model predictive control (MPC) has been used to iteratively solve the frequency control problem with time-coupled state variables and constraints [4], [5]. However, MPC requires real-time communication and computation capabilities, which is not available for much of the current system.

If the linearized small signal model is considered, distributed controllers can be designed to guarantee asymptotic stability. For example, [2] and [3] propose a distributed primal-dual algorithm to minimize the control cost expressed as the quadratic function of control action. However, optimal controller design becomes more crucial when deviations in the systems are larger, that is, under transient stability considerations. But because of the nonlinearity of power flow equations, most previous approaches are restricted to tuning the slopes of the linear droop controllers [8]. In Virtual Synchronous Machines (VSMs), both the damping and the inertial coefficients are tuned to optimize performance [9]. However, these strategies are limited to replicating the characteristics of traditional generators and do not make full use of the capability of inverter-based resources.

To break the unenviable position of not fully utilizing the control capabilities of inverters for transient stability, reinforcement learning (RL) approaches have been proposed [10]–[13]. Specifically, (deep) neural networks are often used to parameterize the controllers and RL is used to train them. The key challenge is to guarantee these learned controllers are stabilizing, that is, the system is stable under control. To this end, existing approaches typically use soft penalties. In [12], a high penalty is added to the reward function when the power flow diverges. For the specific system with an infinite size generator, a safety domain is pre-defined in [13] where the action is also penalized when leaving the domain. However, these approaches are ad hoc, and stability should be treated as a hard constraint rather than through penalties. This especially important since training can only be done on a limited number of samples, where the controller should be stabilizing over a large set in the state space.

Another important challenge comes the controller training process. Generated trajectories are normally used to train the neural networks. Time-coupled state variables over long time horizons make direct back-propagation computations difficult. A standard approach is to first find the approximate value function (or the Q function) [10], [11], [14]–[16]. However, this approach assumes that the system is in steady state, which is generally not true for transient stability where the dynamics and the state change rapidly. Direct policy methods, such as REINFORCE adopts the log probability trick and

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

The authors are supported in part by the National Science Foundation grant ECCS-1930605, ECCS-1942326 and the Washington Clean Energy Institute.

eliminates the requirement in learning the value function [14], [17]. However, the control policies need to be stochastic to encourage exploration. This contradicts the requirement of most operators for deterministic control policies. Moreover, once the data is generated, the physical system is disregarded and the learning process is agnostic to the fact that the data came from a power system. As a result, the physical model cannot be utilized efficiently to constrain the neural network controller.

This paper proposes a recurrent neural network (RNN) based reinforcement learning framework to solve optimal frequency control problem with stability guarantee. Lyapunov condition is enforced in the controller to guarantee asymptotic stability for all system parameters and topologies. We also prove that a monotonic control function can further contribute to exponential stability and enforce the monotone property in controller accordingly. The structure property of controller is realized through a stacked ReLU neural network and therefore avoid explicit non-linear constraints. In order to train the neural network controller efficiently, we design a RNN framework where the time-coupled variables in the power system form the cell component of RNN. This way, state transition dynamics will be directly utilized in back-propagation to guide weight updates efficiently. Simulation results show that the proposed method can learn a static non-linear controller that performs better than traditional linear droop control. All of the code and data described in this paper are publicly available at https://github.com/Wenqi-Cui/RNN-RL-Frequency-Lyapunov.

The main contributions are as follows:

1) Lyapunov condition for transient stability dynamics of power systems is integrated in the structure property of controller, which guarantee asymptotic stability for all system parameters and topologies.
2) The controller is parameterized with a stacked ReLU neural network to guarantee exponential stability without explicitly adding constraints on training process.
3) A RNN-based reinforcement learning framework is proposed to train the neural network controller with time-coupled state dynamics efficiently.

The paper is organized as the following. In Section II we introduce the dynamical system model and the optimal control problem. In Section III we give the main theorems governing the structure of a stabilizing controller and how it can be achieved via neural networks. In Section IV, we show how to train the weights of these controllers efficiently. Section V shows the simulation results and Section VI concludes the paper.

II. Model and Problem Formulation

A. Dynamic Network Model

Let N be the number of buses and $\mathcal{B}$ be the set of buses. The set of transmission lines connecting the buses is denoted as $\mathcal{E}$. The susceptance of the line $(i,j) \in \mathcal{E}$ is $B_{ij} = B_{ji} > 0$. The susceptance $B_{kl} = B_{lk} = 0$ when $(k,l) \notin \mathcal{E}$. The inertia constant of bus i is denoted as M_i. Let R_i and L_i be the droop response coefficient and the damping coefficient for all bus $i = 1, \cdots, N$. Then $D_i = R_i + L_i$ defines the combined frequency response coefficient from synchronous generators and load. Denote the generator power and load of bus i as $P_{g,i}$ and $P_{l,i}$, respectively. The net power injection of bus i is $P_i = P_{g,i} - P_{l,i}$. The angle and frequency of bus i are δ_i and ω_i, respectively. We make the following assumptions:

1) Bus voltage magnitudes are 1 p.u. for $i \in \mathcal{B}$, then $B_{ij}\sin(\delta_i - \delta_j)$ is the active power flow from bus i to bus j.
2) Reactive power injections on the buses and reactive power flows on the lines are ignored.
3) Each bus only has its local frequency measurement. The controller at bus i is a function of ω_i, writen as $u_i(\omega_i)$. It controls the changes in active power to provide primary frequency response, and mainly come from inverter-based resources (e.g., solar PV, EV and storage) located in the same bus as the generators..

The transient stability dynamics of the power systems is represented by the swing equation [1]

$$\dot{\delta}_i = \omega_i, \forall i \in \mathcal{B} \tag{1a}$$

$$M_i\dot{\omega}_i = P_i - D_i\omega_i - u_i(\omega_i) - \sum_{j=1}^{N} B_{ij}\sin(\delta_i - \delta_j), \forall i \in \mathcal{B} \tag{1b}$$

The goal of the u_i's is to decrease frequency deviations while avoiding a high control cost. The control cost incurred from the action of dispatching active power from different resources, for example, solar PV or battery storage. A quadratic cost is typically used in the literature [4], [18]–[20]. The space of feasible u_i's are constrained by the fact that they should be stabilizing, that is, the system in (1) should be asymptotically stable within some region. We envision that our controller would run at a faster timescale than AGC. Namely, we would call resources to respond in the timescale from 100ms to few seconds, while AGC is ran at timescales of 10s of seconds. Therefore the primary frequency regulation provided by inverter-based resources will not affect the optimality when co-exists with AGC [21]. Moreover, since we are concerned with primary frequency response that operates on the timescale of hundreds of milliseconds to seconds, we assume the resources would have enough energy buffer and the main limitation on actuation comes being power injection constraints. This is similar to the assumptions made in [22]–[24] which assume that storage would operate without being energy limited, and a slower control loop (e.g., AGC) would reset the energy in storage units.

B. Optimization Problem Formulation

We consider two costs in the objective function of optimal frequency control problem: the cost on frequency deviations [4], [24], [25] and the cost of controllers [3], [4], [26] . Firstly, considering that the power system operators restrict the maximum frequency deviation (typically referred to as frequency Nadir [27]), the cost on frequency deviation is represented by the infinity norm of $\omega_i(t)$ over the time horizon from 0 to the time T defined by $||\boldsymbol{\omega_i}||_\infty = \sup_{0\le t\le T}|\omega_i(t)|$ [28]. Secondly, the cost on controller is the quadratic function of action defined

by its two-norm $||\boldsymbol{u_i}||_2^2 = \frac{1}{T}\int_{t=0}^{T}(u_i(t))^2 dt$ [18]–[20], [24], [29]. The optimization problem is formulated as

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

$$\text{s.t. } \dot{\delta}_i = \omega_i \tag{2b}$$

$$M_i\dot{\omega}_i = P_i - D_i\omega_i - u_i(\omega_i) - \sum_{j=1}^{N} B_{ij}\sin(\delta_i - \delta_j) \tag{2c}$$

$$\underline{u}_i \le u_i(\omega_i) \le \overline{u}_i \tag{2d}$$

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

where γ is the coefficient that stands for the relative cost of action with respect to frequency deviation, similar to the cost coefficients used in LQR controllers [30]. The lower and upper bound for the control action at bus i are $\underline{u}_i$ and $\overline{u}_i$, respectively. Constraints (2b)-(2e) hold for the time t from 0 to T. We chose to use a softer cost for the frequency nadir to avoid operating too close to the maximum allowed frequency nadir, since the system may be pushed over the limit. A log-barrier based on the frequency nadir could also be used to closely approximate a hard constraint [31]. Note that we can assign the individual control action using distinct weights, that is, each u_i is weighted by γ_i. For simplicity, we choose a single γ in the manuscript. In practice, these coefficients are often tuned to achieve a desirable frequency performance at an acceptable level of control action. Other costs (e.g., total frequency deviation and the rate of change of frequency) can also be accommodated in formulation (2).

Considering that not all the generators have the computation ability to conduct real-time optimization, $u_i(\omega_i)$ is formulated as a static controller where the coefficients are determined before real-time operations. In (2), we are optimizing the function $\boldsymbol{u}(\cdot)$, which is an infinite dimensional problem. The stabilizing constraint and hard limits are also nonlinear. Therefore, problem (2) is challenging to solve by conventional optimization techniques and thus we try to find an optimal controller through reinforcement learning.

C. Reinforcement Learning for Optimal Frequency Control

The controller $u_i(\omega_i)$ is flexibly parameterized as a neural network with weight θ_i, written as $u_{\theta_i}(\omega_i)$. Reinforcement learning trains neural networks to map states of the environment to control action so as to minimize the numerical loss function [14]. By defining the loss function as the objective function (2a), the training of RL will gradually drive the decrease of the loss function and equivalently optimize the objective function.

General RL techniques, including Q-learning, policy gradient and DDPG, usually update weights θ_i through observable numerical loss from trajectories that comply with Bellman equation [14]. However, since the objective function (2) depends on the largest frequency deviation over the finite time horizon, we cannot obtain the instant reward of the current action for iterating the value function from the Bellman equation. Moreover, the definition of the Q function for a steady state also does not hold in the transient stability scenarios.

Apart from the dependence on Bellman equation, one major challenge for these methods comes from the hard constraint on stability of the system. Although we can add a high penalty to the large magnitude of ω_i, such a penalty may not guarantee that the constraints are always satisfied. The learned controllers that leads to reasonably looking and low cost trajectories in training may destabilize the system for different starting points. This paper directly uses the physical model (1) to derive the structure of the stabilizing controller according to Lyapunov theory. As illustrated in Fig. 1(b), stability is guaranteed by enforcing the structure property on designing neural network controller $u_{\theta_i}(\omega_i(t))$. Details are elaborated in Section III. Another challenge is the interconnected dependence of the weights θ_i on the time-coupled variables $(\delta_i(t), \omega_i(t), u_{\theta_i}(t))$ over the whole time stages shown by (2b) and (2c). For gradient-based method to search for optimal θ_i, that means the subsequent application of chain rule to compute gradient all the way through the time steps. This becomes hard to compute when there are a large number of time steps. Therefore, we formulate the state transition dynamics and the neural network controller implicitly in the framework of RNN to increase training efficiency, as illustrated in Fig. 1(c). Details are elaborated in Section IV.

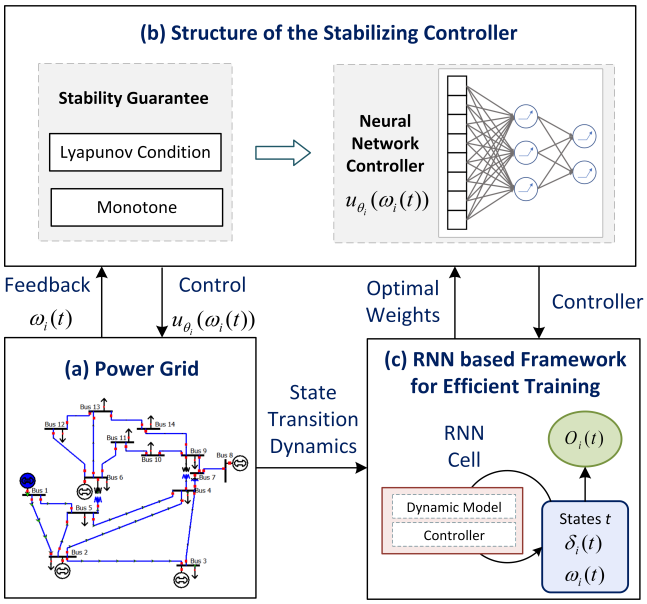


Fig. 1. Reinforcement learning for the frequency control problem

III. Structure Properties of the Controller

A. Lyapunov Stability

To constrain the search space of the set of stabilizing controllers, we investigate the reasonable structure property of controllers from Lyapunov stability theory. The Lyapunov approach provides the advantage to ascertain the stability of the equilibrium point without numerical integration [32]. The general structure property of the controllers are obtained through the following two theorems.

Theorem 1. *The equilibrium point* $\{(\delta_i, \omega_i)|w_i = 0, P_i = \sum_{j=1}^{N} B_{ij}\sin(\delta_i - \delta_j), i = 1, \ldots, N\}$ *of the dynamic system*

in (1) *is locally asymptotic stable if* $u_i(\omega_i)$ *is the same sign as* ω_i *for all* $i = 1, \cdots, N$

The qualifier "local" in the theorem is important since we need to assume that the trajectories start within a bounded region of the equilibrium point. This region is derived based on well understood power system energy (or Lyapunov) functions [33]–[36] and can be explicitly characterized (see Appendix A). The next theorem shows that if controllers have more structure, a desirable rate of convergence can be obtained.

Theorem 2. *If the control function* $u_i(\omega_i)$ *monotonically increases with* ω_i *for all* $i = 1, \cdots, N$*, then the equilibrium point is locally exponentially stable.*

Theorem 1 and 2 give structure properties for controllers that guarantee asymptotic and exponential stability for *all system parameter and topologies*. Namely, if the controllers satisfy the conditions in the theorems, they would be locally stabilizing for any network. Therefore, the optimal performance comes from training on a particular network, but the stability guarantees do not. This robustness to changes and uncertainties is a key advantage of defining the structure of networks compared to purely model-free RL approaches.

The design of neural networks that satisfy these properties is given in the next section (Section III-B). The proof of Theorem 1 is given below and the proof of Theorem 2 is given in the Appendix B.

Proof. A local Lyapunov function $V(\delta, \omega)$ for the dynamic system represented by (1) is [33]–[36]:

$$\begin{aligned} V(\delta,\omega) =& \frac{1}{2}\sum_{i=1}^{N} M_i\omega_i^2 - \sum_{i=1}^{N} P_i\delta_i \\ & - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} B_{ij}\cos(\delta_i - \delta_j) \end{aligned} \tag{3}$$

where the term $\frac{1}{2}\sum_{i=1}^{N} M_i\omega_i^2$ represents the kinetic energy of the turbines and the term $-\sum_{i=1}^{N} P_i\delta_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} B_{ij}\cos(\delta_i - \delta_j)$ is the potential energy of the system stored in the inductive lines in the power grid network. This function is not a "true" Lyapunov function since it is not non-negative on the entire state space (it is not bounded from below), and it is used to show that trajectories that start not too far away from the equilibrium point would converge to it.

Partial Derivative of the Lyapunov function is

$$\frac{\partial V(\delta,\omega)}{\partial \delta_i} = -P_i + \sum_{j=1}^{N} B_{ij}\sin(\delta_i - \delta_j) \tag{4a}$$

$$\frac{\partial V(\delta,\omega)}{\partial \omega_i} = M_i\omega_i \tag{4b}$$

An equilibrium point of the dynamic system is a state (δ, ω) where $w_i = 0$ for $i \in \mathcal{B}$ and $P_i = \sum_{j=1}^{N} B_{ij}\sin(\delta_i - \delta_j)$ for $(i, j) \in \mathcal{E}$, i.e.,where all frequency deviations and branch power deviations are constant over time.

The total derivative of the Lyapunov function with respect to t is

$$\begin{aligned} \dot{V}(\delta,\omega) &= \sum_{i=1}^{N}\left(\frac{\partial V(\delta,\omega)}{\partial \delta_i}\dot{\delta}_i + \frac{\partial V(\delta,\omega)}{\partial \omega_i}\dot{\omega}_i\right) \\ &= \sum_{i=1}^{N}\left(-\omega_i u_i(\omega_i) - D_i\omega_i^2\right) \end{aligned} \tag{5}$$

The equilibrium is locally asymptotically stable when $\dot{V}(\delta, \omega) \leq 0$ with equality only reached at the equilibrium. From (5), $\dot{V}(\delta, \omega) < 0$ can be satisfied if $\omega_i u_i(\omega_i) > 0$ for $\omega_i \neq 0$. This is equivalent to the constraint that the sign of action $u_i(\omega_i)$ is the same as the sign of ω_i. Note this condition is distributed in the sense that it needs to be satisfied at each of the buses.

By the Krasovski-Lasalle principle [37], the constraint on control action can be further relaxed to include equality, i.e., $\omega_i u_i(\omega_i) \geq 0$. Define the set $S = \left\{(\delta, \omega) \in \Omega_r : \dot{V}(\delta, \omega) = 0\right\}$ where the derivative of Lyapunov equals to zero. For a trajectory to remain in this set we must have $\omega_i = 0$ or $u_i(\omega_i) = 0$. Consider one point where $\omega_i \neq 0$ while $u_i(\omega_i) = 0$. Using the dynamics (1a) we have $\dot{\delta}_i \neq 0$, this implies that $\dot{\omega}_i \neq 0$ if $u_i(\omega_i) = 0$. Therefore, points with $\omega_i \neq 0$ are not in the invariant set inside S. In other words, S contains no invariant sets other than the states that $w_i = 0$. Then from the Krasovski-Lasalle principle, we can conclude that the state (δ, ω) where $w_i = 0$ and $P_i = \sum_{j=1}^{N} B_{ij}\sin(\delta_i - \delta_j)$ for $(i, j) \in \mathcal{E}$, is locally asymptotically stable. □

B. Design of Neural Network Controllers

In this paper, we parametrize the controllers u_i by a single hidden layer neural network. By Theorems 1 and 2, we design the weights and the biases of the neural networks to have the following structures such that the controller will be locally exponentially stabilizing:

1) $u_{\theta_i}(\omega_i)$ has the same sign as ω_i
2) $u_{\theta_i}(\omega_i)$ is monotonically increasing
3) $\underline{u}_i \leq u_{\theta_i}(\omega_i) \leq \overline{u}_i$

The first two requirements are equivalent to designing a monotonic increasing function through the origin. This is constructed by decomposing the function into a positive and a negative part as $f_i(\omega_i) = f_i^+(\omega_i) + f_i^-(\omega_i)$, where $f_i^+(\omega_i)$ is monotonic increasing for $\omega_i > 0$ and zero when $\omega_i \leq 0$; $f_i^-(\omega_i)$ is monotonic increasing for $\omega_i < 0$ and zero when $\omega_i \geq 0$. The saturation constraints can be satisfied by hard thresholding the output of the neural network.

The function $f_i^+(\omega_i)$ and $f_i^-(\omega_i)$ are constructed using a single-layer neural network designed by stacking the ReLU function. Let m be the number of hidden units. For $f_i^+(\omega_i)$, let $q_i = [q_i^1 \quad q_i^2 \quad \cdots \quad q_i^m]$ be the weight vector of bus i; $b_i = [b_i^1 \quad b_i^2 \quad \cdots \quad b_i^m]^\intercal$ be the corresponding bias vector. For $f_i^-(\omega_i)$, let $z_i = [z_i^1 \quad z_i^2 \quad \cdots \quad z_i^m]$ be the weights vector and $c_i = [c_i^1 \quad c_i^2 \quad \cdots \quad c_i^m]^\intercal$ be the bias vector. Denote $\mathbf{1} \in \mathbb{R}^m$ as the all 1's column vector. Let $\sigma(\cdot)$ be the ReLU function. The detailed construction of $f_i^+(\omega_i)$ and $f_i^-(\omega_i)$ is given in Lemma 3.

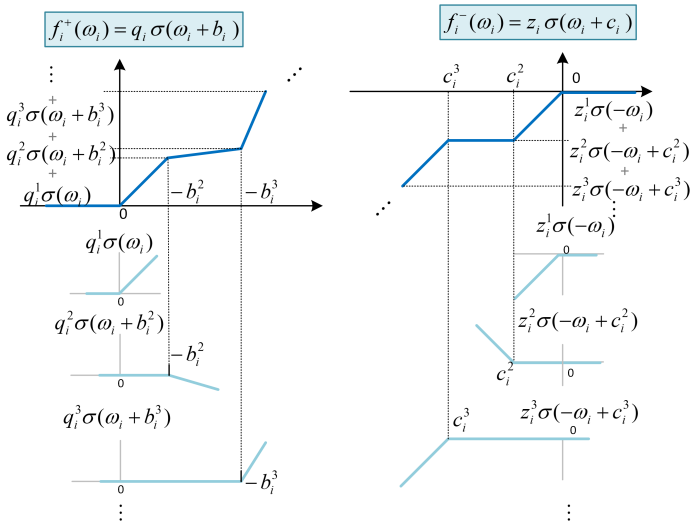


Fig. 2. Stacked ReLU neural network to formulate a monotonic increasing function through the origin

Lemma 3. *The stacked ReLU function constructed by* (6) *is monotonic increasing for* $\omega_i > 0$ *and zero when* $\omega_i \le 0$.

$$f_i^+(\omega_i) = q_i\sigma(\mathbf{1}\omega_i + b_i) \tag{6a}$$

$$\textit{where} \sum_{j=1}^{l} q_i^j \ge 0, \quad \forall l = 1, 2, \cdots, m \tag{6b}$$

$$b_i^1 = 0, b_i^l \le b_i^{(l-1)}, \quad \forall l = 2, 3, \cdots, m \tag{6c}$$

The stacked ReLU function constructed by (7) *is monotonic increasing for* $\omega_i < 0$ *and zero when* $\omega_i \ge 0$.

$$f_i^-(\omega_i) = z_i\sigma(-\mathbf{1}\omega_i + c_i) \tag{7a}$$

$$\textit{where} \sum_{j=1}^{l} z_i^j \le 0, \quad \forall l = 1, 2, \cdots, m \tag{7b}$$

$$c_i^1 = 0, c_i^l \le c_i^{(l-1)}, \quad \forall l = 2, 3, \cdots, m \tag{7c}$$

Proof. Note that the ReLU function $\sigma(x)$ is linear with x when activated ($x > 0$) and equals to zero when deactivated ($x \le 0$), we construct the monotonic increasing function $f_i^+(\omega_i)$ by stacking the function $g_i^l(\omega_i) = q_i^l\sigma(\omega_i + b_i^l)$ as illustrated by Fig. 2. Since $b_i^{1+} = 0$ and $b_i^{l+} \le b_i^{(l-1)+}, \forall 1 \le l \le m$, $g_i^l(\omega_i)$ is activated in sequence from $g_i^1(\omega_i)$ to $g_i^m(\omega_i)$ with the increase of ω_i. In this way, the stacked function is a piece-wise linear function and the slope for each piece is $\sum_{j=1}^{l} q_i^j$. Monotonic property can be satisfied as long as the slope of all the pieces are positive, i.e., $\sum_{j=1}^{l} q_i^l \ge 0, \forall 1 \le l \le m$. Similarly, $f_i^-(\omega_i)$ also construct by ReLU function activated for negative w_i in sequence corresponding to c_i^l for $l = 1, \cdots, m$. $\sum_{j=1}^{l} z_i^j \le 0$ means that all the slop of the piece-wise linear function is positive and therefore guarantees monotone. □

Note that there still exists inequality constraints in (6) and (7), which makes the training of the neural networks cumbersome. We can reformulate the weights to get a equivalent representation that is easier to deal with in training. Define the non-negative vectors $\hat{q}_i = \begin{bmatrix}\hat{q}_i^1 & \cdots & \hat{q}_i^m\end{bmatrix}$ and $\hat{b}_i = \begin{bmatrix}\hat{b}_i^1 & \cdots & \hat{b}_i^m\end{bmatrix}^\intercal$. Then, (6b) is satisfied if $q_i^1 = \hat{q}_i^1$, $q_i^l = \hat{q}_i^l - \hat{q}_i^{(l-1)}$ for $l = 2, \cdots, m$. (6c) is satisfied if $b_i^1 = 0$, $b_i^l = -\sum_{j=2}^{l} \hat{b}_i^j$ for $l = 2, \cdots, m$. Similarly, define $\hat{z}_i = \begin{bmatrix}\hat{z}_i^1 & \cdots & \hat{z}_i^m\end{bmatrix} \ge 0$ and $\hat{c}_i = \begin{bmatrix}\hat{c}_i^1 & \cdots & \hat{c}_i^m\end{bmatrix}^\intercal \ge 0$. Then, (7b) is satisfied if $z_i^1 = -\hat{z}_i^1$, $z_i^l = -\hat{z}_i^l + \hat{z}_i^{(l-1)}$ for $l = 2, \cdots, m$. (7c) is satisfied if $c_i^1 = 0$, $c_i^l = -\sum_{j=2}^{l} \hat{c}_i^l$ for $l = 2, \cdots, m$. If the dead-band of the frequency deviation within the range $[-d, d]$ is required, it can be easily satisfied by setting $b_i^2 = -d, q_i^1 = 0$ and $c_i^2 = -d, z_i^1 = 0$ in (6) and (7).

The next Theorem states the converse of Lemma 3, that is, the constructions in (6) and (7) suffice to approximate all functions of interest.

Theorem 4. *Let* $r(x)$ *be any continuous, bounded monotonic function through the origin with bounded derivatives, mapping compact set* $\mathbb{X}$ *to* $\mathbb{R}$. *Then there exists a function* $f(x) = f^+(x) + f^-(x)$ *constructed by* (6) *and* (7) *such that, for any* ϵ *and any* $x \in \mathbb{X}$, $|r(x) - f(x)| < \epsilon$.

The proof is given in Appendix C. Note that $f(x)$ is a single-layer neural network. When approximating an arbitrary function, the number of neurons and the height will depend on ϵ. Since the controller in this paper is bounded, the stacked-ReLU neural network with limited number of neurons is sufficient for parameterization. The last step is to bound the output of the neural networks, which can be done easily using ReLU activation functions.

Lemma 5. *The neural network controller* u_i *given below is a monotonic increasing function through the origin and bounded in* $[\underline{u}_i, \overline{u}_i]$ *for all* $i = 1, \cdots, N$:

$$\begin{aligned} u_i(\omega_i) =& \overline{u}_i - \sigma(\overline{u}_i - f_i^+(\omega_i) - f_i^-(\omega_i)) \\ &+ \sigma(\underline{u}_i - f_i^+(\omega_i) - f_i^-(\omega_i)) \end{aligned} \tag{8}$$

The proof of this lemma is by inspection.

IV. Learning Control Policies Using RNNs

The structure of the controllers are decided by the constructions in (6), (7) and (8). In this section we develop a RNN based RL algorithm to learn the weights and biases of these networks.

A. Discretize Time System

To learn the controller and simulate the trajectories of the system, we discretize the dynamics (1) with step size Δt. We use k and K to represent the discrete time and the total number of stages, respectively. The states (δ_i, w_i) at bus i evolves along the trajectory are represented as $\boldsymbol{\delta_i} = (\delta_i(0), \delta_i(1), \cdots, \delta_i(K))$ and $\boldsymbol{\omega_i} = (\omega_i(0), \omega_i(1), \cdots, \omega_i(K))$ over K stages, with the control sequence $\boldsymbol{u_{\theta_i}} = (u_{\theta_i}(\omega_i(0)), \cdots, u_{\theta_i}(\omega_i(K)))$. The infinity norm of the sequence of $\omega_i(k)$ is then defined by $||\boldsymbol{\omega_i}||_\infty = \max_{k=0,\cdots,K} |w_i(k)|$. The cost on controller is

the quadratic function of action defined by its two-norm $||\boldsymbol{u_{\theta_i}}||_2^2 = \frac{1}{K}\sum_{k=1}^{K}(u_{\theta_i}(k))^2$. The optimization problem is

$$\min_{\boldsymbol{\theta}} \quad \sum_{i=1}^{N}\left(||\boldsymbol{\omega_i}||_\infty + \gamma||\boldsymbol{u_{\theta_i}}||_2^2\right) \tag{9a}$$

$$\text{s.t. } \delta_i(k) = \delta_i(k-1) + \omega_i(k-1)\Delta t \tag{9b}$$

$$\omega_i(k) = -\frac{\Delta t}{M_i}\sum_{j=1}^{|\mathcal{B}|}B_{ij}\sin(\delta_{ij}(k-1)) + \frac{\Delta t}{M_i}P_i + \left(1-\frac{D_i\Delta t}{M_i}\right)\omega_i(k-1) - \frac{\Delta t}{M_i}u_{\theta_i}(\omega_i(k-1)) \tag{9c}$$

$$\underline{u}_i \le u_{\theta_i}(\omega_i(k)) \le \overline{u}_i \tag{9d}$$

$$\omega_i(k)u_{\theta_i}(\omega_i(k)) \ge 0 \tag{9e}$$

$$\frac{\mathrm{d}u_{\theta_i}(\omega_i(k))}{\mathrm{d}\omega_i(k)} \ge 0 \tag{9f}$$

and all equations hold for $i = 1, \ldots, N$. The constraints (9e) and (9f) guarantee exponentially stability.

Note that the optimization variable $\boldsymbol{\theta}$ exists in all the time steps in (9). A straightforward gradient-based training approach is challenging since we need to calculate the gradient all the way to the first time step for all time steps $k = 0, \cdots, K$. To mitigate this challenge, we propose a RNN-based framework to integrate the state transition dynamics (9b) and (9c) implicitly. This way, the gradient of the optimization objective with respect to $\boldsymbol{\theta}$ can be computed efficiently through back-propagation.

B. RNN for control

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. By defining the cell state as the time-coupled states δ_i and ω_i, the state transition dynamics of the power system is integrated as illustrated in Fig. 3

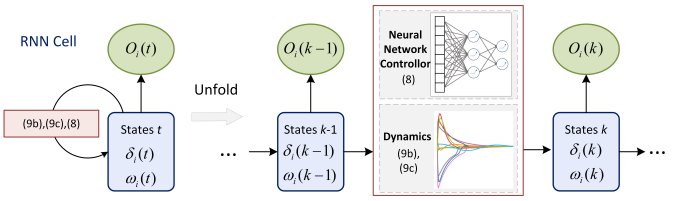


Fig. 3. Structure of RNN for the frequency control problem

The operation process of RNN is shown by the left side of Fig. 3. The cell unit of RNN will remember its current state at the stage k and pass it as an input to the next stage. Unfolding the cell unit through time will give the right side of Fig. 3. In this way, RNN can be utilized in the frequency control problem to deal with time-coupled state variables. Specifically, the state $(\delta_i(k-1), \omega_i(k-1))$ for all $i = 1, \cdots, N$ at the stage $k-1$ is taken as an input in the state transition function (9b) (9c) and thus the state $(\delta_i(k), \omega_i(k))$ for all $i = 1, \cdots, N$ at the stage k is obtained. The control function $u_{\theta_i}(\omega_i(k))$ in the state transition function is formatted through (8) to satisfy inequality constraints. The output $O_i(k) = \begin{bmatrix} O_i^1(k) & O_i^2(k) \end{bmatrix}$ at stage k is a vector with two components computed by $O_i^1(k) = \omega_i(k)$ and $O_i^2(k) = \left(u_{\theta_i}(\omega_i(k))\right)^2$. The loss function is formulated to be equivalent with the objective function (9a) as:

$$Loss = \sum_{i=1}^{N}\max_{k=0,\cdots,K}|O_i^1(k)| + \gamma\frac{1}{K}\sum_{k=1}^{K}O_i^2(k) \tag{10}$$

The trainable variables $\boldsymbol{\theta}$ is specified in the neural network controller (8) and updated by gradient descent through the Loss function (10). The unfolded structure of RNN form a directed graph along a temporal sequence where the gradient of Loss function can be efficiently computed by auto-differentiation mechanisms [38].

C. Algorithm

The pseudo-code for our proposed method is given in Algorithm 1. The variables to be trained are weights $\boldsymbol{\theta} = \{\hat{q}, \hat{b}, \hat{z}, \hat{c}\}$ for control network represented by (6)-(8) . The $i-th$ row of $\hat{q}$ and $\hat{z}$ are the vector $\hat{q}_i$ and $\hat{z}_i$ in (6) and (7), respectively. The i-th column of $\hat{b}$ and $\hat{c}$ are the vector $\hat{b}_i$ and $\hat{c}_i$ in (6) and (7), respectively. Training is implemented in a batch updating style where the h-th batch initialized with randomly generated initial states $\{\delta_i^h(0), \omega_i^h(0)\}$ for all $i = 1, \cdots, N$. The evolution of states in K stages will be computed through the structure of RNN as shown by Fig. 3. Adam algorithm is adopted to update weights $\boldsymbol{\theta}$ in each episode.

Algorithm 1 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 (9)

Input: The bound of $\overline{\delta}_i$ and $\overline{\omega}_i$ to generate the initial states *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 the output $\{O_{h,i}(0), O_{h,i}(1), \cdots, O_{h,i}(K)\}$
5: Calculate total loss of all the batches $Loss = \frac{1}{H}\sum_{h=1}^{H}\sum_{i=1}^{N}\max_{k=0,\cdots,K}|O_{h,i}^1(k)| + \gamma\frac{1}{K}\sum_{k=1}^{K}O_{h,i}^2(k)$
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**

V. Case Study

Case studies are conducted on the IEEE New England 10-machine 39-bus (NE39) power network to illustrate the effectiveness of the proposed method. Firstly, we show that the proposed Lyapunov-based approaches for designing neural

network controller can guarantee stability, while unconstrained neural networks may result in unstable controllers. Then, we show that the proposed structure can learn a non-linear controller that performs better than linear droop control.

A. Simulation Setting

We use TensorFlow 2.0 framework to build the reinforcement learning environment and run the training process in Google Colab with GPU acceleration. Power System Toolbox (PST) in MATLAB is utilized to simulate the dynamic response with 6-order generator model and 2-order phase-locked-loop (PLL) block on the inverter-based resources [39], [40]. Parameters for the transient and sub-transient process of generators are obtained in [41]. The system is in the Kron reduced form [4], [42] and its dynamics is represented by (1). The bound on action $\overline{u}_i$ is generated to be uniformly distributed in $[0.8P_i, P_i]$. The initial states of angle and frequency are randomly generated such that $\delta_i(0)$ is uniformly distributed in $[-0.05, 0.05]$ rad, $\omega_i(0)$ is uniformly distributed in $[-0.1, 0.1]$ Hz. Other simulation setup parameters are shown in Table I .

TABLE I
SIMULATION PARAMETERS

Parameters	description	Values
Δt	Stepsize	0.01s
K	Time stages	200
γ	Cost coefficient	0.01
m	Neuron in hidden layer	20
I	Episode	600
H	Batchsize	800

We compare the performance of the proposed RNN based structure where the neural network controller is designed with and without the Lyapunov-based approach, and the drop control with optimized linear coefficient. The parameter settings are as follows:

1) RNN-Lyapunov: Neural network controller designed based on Algorithm 1, which satisfies Theorems 1 and 2. The episode number, batch size and the number of neurons are shown in Table I. Parameters of RNN are updated using Adam with learning rate initializes at 0.05 and decays every 30 steps with a base of 0.7.
2) RNN-Wo-Lyapunov: Controllers are learned without imposing any structures and purely optimizes the reward during training. The structure is the neural network with two dense layer where the activation function in the first layer is Tanh. All the other parameters are the same as RNN-Lyapunov.
3) Linear droop control: Let k_i be the droop coefficient for bus i, droop control policy is represented as $u_i(\omega_i) = k_i\omega_i$ for $i = 1, \cdots, N$, thresholded to their upper and lower bounds. The optimized droop coefficient is obtained by solving problem (9) using fmincon function of Matlab.
4) PG-Monotone: Neural network controller designed by the proposed stacked-ReLU structure and trained with REINFORCE Policy Gradient algorithm [14]. The neural networks for controller, the episode number, batch number and optimizer are the same as RNN-Lyapunov. The learning rate initializes at 0.01 and decays every 30 steps with a base of 0.7.
5) PG-Wo-Monotone: Neural network controller designed by multi-layer neural network and trained with REINFORCE Policy Gradient algorithm. The episode number, batch number and optimizer are the same as RNN-Lyapunov. In order to make the controller equals to zero at equilibrium and same sign with ω, the controller is parameterized as softsign function of ω, multiply with ReLU function of a dense neural network with two hidden layers and tanh activation. The learning rate initializes at 0.02 and decays every 30 steps with a base of 0.6.

B. Necessity of Lyapunov-based Approach

Theorems 1 and 2 ensures that the learned controller would be locally exponentially stable, but it's interesting to check the performance of an unconstrained controller. Intuitively, an unstable controller should lead to large costs since some states would be blowing up. Then maybe a controller that minimizes the cost would also be stabilizing.

We show that in this section the above intuition can be misleading. Figure 4 shows the training loss between controllers learned with and without the Lyapunov-based approach. Both losses converge, with the Lyapunov-based controller having better performances. However, when we implement the controllers, the one without Lyapunov-based approach is *unstable* and leads to very large state oscillations (Fig. 5b. In contrast, the controller constrained by the Lyapunov condition shows good performance in Fig. 5a. The reason for this dichotomy in performance is that we can only check a finite number of starting points in training, and good training performance does not in itself guarantee good generalization. Therefore, explicitly constraining the controller structure is necessary.

Note that the dynamics above are obtained with the controller where the frequency deviation ω is inferred from PLL block. We further compare the dynamics in Fig. 5a with the case that the exact frequency deviation is available for controller. The results in Appendix D shows that the inclusion of PLL does not change our results.

C. Performance Results

This subsection shows that the proposed method can learn a static non-linear controller that outperforms the optimal linear droop control. Fig. 6 illustrates the control policy learnt from RNN-Lyapunov, Policy Gradient and the linear droop control with optimized droop coefficient for four generators. Compared with the traditional droop control, the proposed stacked ReLU neural network learns a non-linear controller with different shapes for RNN-Lyapunov and PG-Monotone. Despite of parameterizing using a potentially non-linear neural network, PG-Wo-monotone learns the shape of a linear policy. Fig. 7 illustrates the dynamics of ω and corresponding control action u with a loss of load at bus 2 at two seconds.

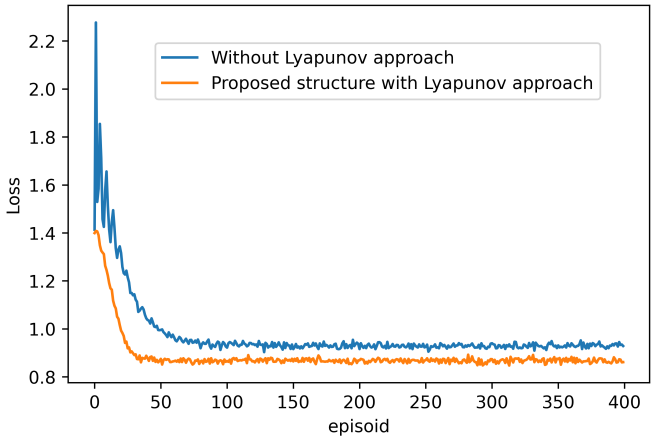
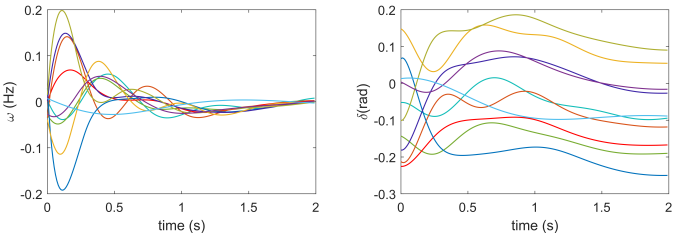
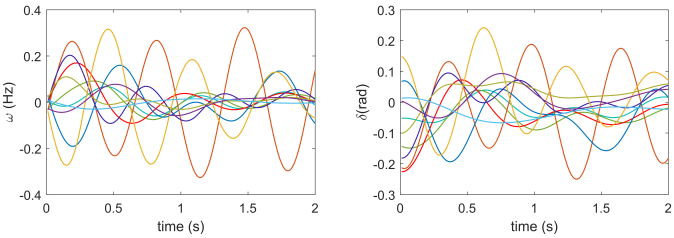


Fig. 4. Average batch loss along episodes for neural network controller designed with and without the Lyapunov-based approach. Both converges quickly, with the former having better performance than the latter.



(a) Dynamics of ω (left) and δ (right) for RNN-Lyapunov



(b) Dynamics of ω (left) and δ (right) for RNN-Wo-Lyapunov

Fig. 5. Dynamics of angle δ and frequency deviation ω in 10 generator buses corresponding to (a) the neural network controller designed with the Lyapunov-based approach and (b) the neural network controller designed without the Lyapunov-based approach. The two controllers exhibit qualitatively different behavior even though they both achieve finite training losses in Fig. 4. The controller designed without the Lyapunov-based approach leads to unstable trajectories of the system.

After the step load change, the largest frequency deviation of RNN-Lyapunov is 0.1413, which is lower than that of linear droop control, PG-Monotone, PG-Wo-Monotone (equas to 0.1531, 0.1528, 0.1535, respectively). The computational time of the proposed RNN based method is 1080.38s, while the computational time of REINFORCE policy gradient takes 4206.43s. Therefore, the proposed RNN based structure reduces computational time by approximate 74.32% compared with the general reinforcement learning structure.

The trajectories start from different initial points will lead to different loss values. To investigate the general performance of RNN-Lyapunov, Linear droop control and Policy Gradient with different initial conditions, we fix initial δ in $U[-0.1, 0.1]$ rad and let the initial ω to uniformly distributed

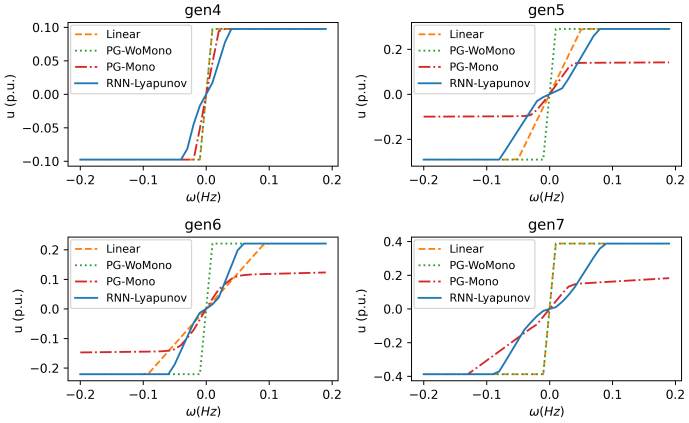


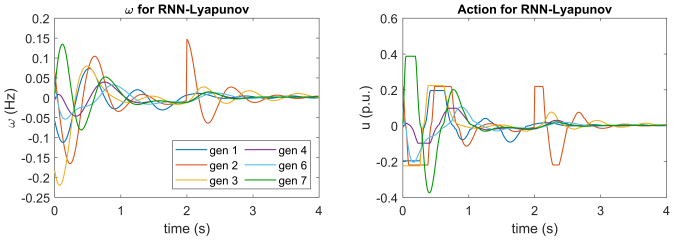
Fig. 6. Control action u of RNN-Lyapunov, Linear droop control and Policy Gradient for generator buses 4,5,6,7. The comparison shows that the proposed Stacked-ReLU neural network will learn a non-linear controller in flexible shapes.

in $U[-\bar{\omega}, \bar{\omega}]$ around the equilibrium. $\bar{\omega}$ denotes the variation bound of initial ω. The average loss corresponding to $\bar{\omega} = 0.0, 0.025, \cdots, 0.15$ Hz are illustrated in Fig. 8. $\bar{\omega} = 0$ is the case that no variation of ω exists in the initial condition. For this case, the average loss of RNN-Lyapunov is 0.70, while the average loss of linear droop control, PG-Monotone and PG-Wo-Monotone, are 0.79, 0.74, 0.85 respectively. Overall, RNN-Lyapunov remains approximate 11.39%, 5.41%, 17.65% lower in average loss than that of linear droop control, PG-Monotone and PG-Wo-Monotone, respectively. Therefore, the proposed method learn the non-linear control that leads to better average control performance under different initial conditions. Moreover, despite of parameterizing the controller using the same stacked-ReLU structure, RNN-Lyapunov is better than PG-Monotone because of more accurate gradient calculations.

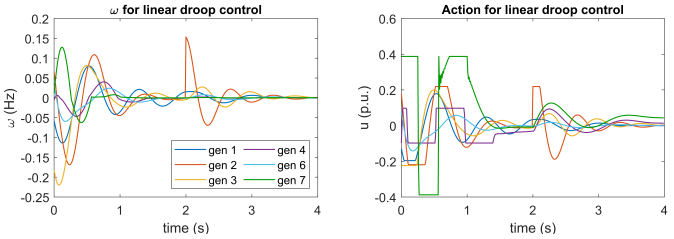
D. Test on 50-generator system

To validate the performance of the proposed method in a larger system, simulation is further conducted on the IEEE 145-bus, 50-generator dynamic stability test case [39]. The bound on action $\overline{u}_i$ is generated to be uniformly distributed in $[0.8P_{m,i}, P_{m,i}]$. γ is set as 0.005. The parameter setting for the neural network controller and the training process are the same as the 39-bus system.

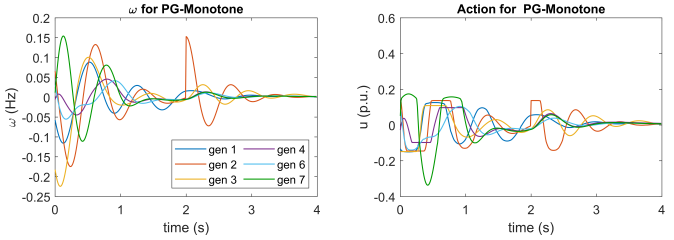
For RNN-Lyapunov and linear droop with different initial conditions, the distribution of loss corresponding to $\bar{\omega} = 0.0, 0.1, \cdots, 0.2$ Hz are illustrated as box plot in Fig. 9. $\bar{\omega} = 0$ is the case that no variation of ω exists in the initial condition. For this case, the average loss of RNN-Lyapunov is 25.79, which is approximate 24.89% lower than that of linear droop control (34.35). The average losses increase slightly with larger $\bar{\omega}$ for both RNN-Lyapunov and linear droop control. The variation of loss increases significantly with larger $\bar{\omega}$ for linear droop control, while the variation increase slightly for RNN-Lyapunov. Therefore, the proposed method learns the non-linear controller that leads to lower average loss and variation



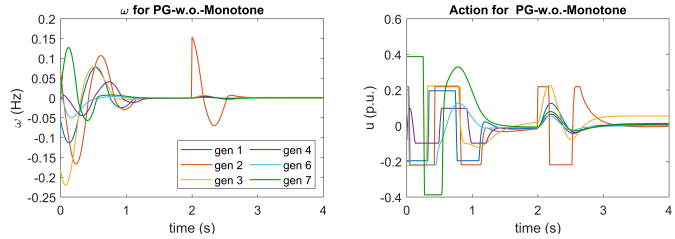
(a) Dynamics of ω (left) and u (right) for RNN-Lyapunov



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



(c) Dynamics of ω (left) and u (right) for controller obtained by policy gradient with Monotone condition



(d) Dynamics of ω (left) and u (right) for controller obtained by policy gradient without Monotone condition

Fig. 7. Dynamics of the frequency deviation w and the control action u in selected generator buses corresponding to (a) Lyapunov-guided neural network controller learned with RNN. (b) Linear droop control. (c) Lyapunov-guided neural network controller learned from Policy Gradient with Monotone structure design. (d) Neural network controller learned from Policy Gradient without Monotone structure design. The proposed RNN controller has the minimal cost.

of loss, which means improved average control performance and higher robustness under different initial conditions.

VI. Conclusion

This paper investigates the optimal frequency control problem using reinforcement learning with stability guarantees. Constraints obtained from Lyapunov stability theory and saturation limits are realized by parameterizing the controller with stacked ReLU neural network. In particular, we construct the controllers to be monotonically increasing through the origin, and prove they guarantee stability for all operating points in a region. These controllers are trained using a RNN-based method that allows for efficient back propagation

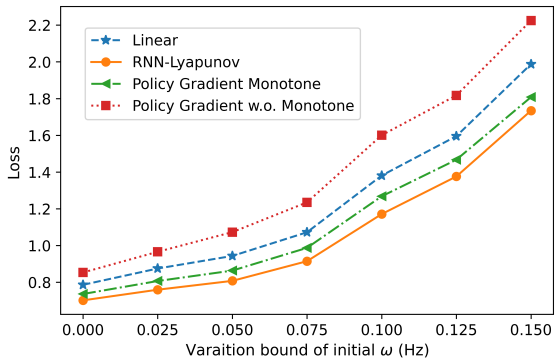


Fig. 8. Loss with different variation range of initial conditions for RNN-Lyapunov, Linear droop controller and Policy Gradient. Compared with Linear droop controller PG-Monotone and PG-Wo-Monotone, RNN-Lyapunov reduces the loss by approximate 11.39%, 5.41%, 17.65%, respectively.

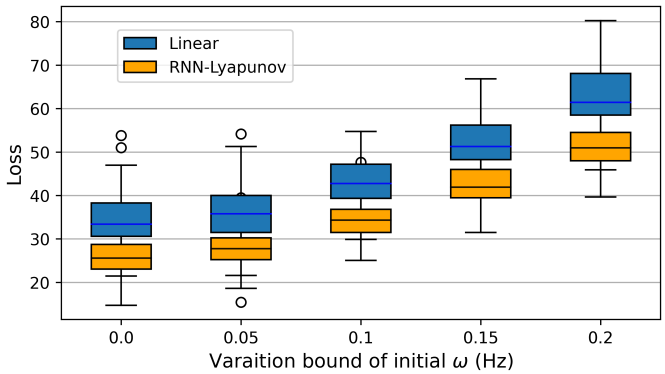


Fig. 9. Loss with different variation range of initial conditions for RNN-Lyapunov and Linear droop controller. Compared with Linear droop controller, RNN-Lyapunov reduces the loss by approximate 24.89%.

through time. The learned controllers are static piece-wise linear functions that do not need real-time computation and is practical for implementation. Through simulations, we show that they outperform optimal linear droop as well as purely unstructured controllers trained via reinforcement learning. In particular, controllers failing to consider stability constraints in learning may lead to unstable trajectories of the state variables, while our proposed controllers can achieve optimal performances in system frequency responses that use small control efforts. Further directions include energy constraints for high power/low energy devices, and the consideration of Lyapunov function with losses in the system.

References

[1] P. Kundur, N. J. Balu, and M. G. Lauby, *Power system stability and control*. McGraw-hill New York, 1994, vol. 7.

[2] C. Zhao, U. Topcu, N. Li, and S. Low, "Design and stability of load-side primary frequency control in power systems," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1177–1189, 2014.

[3] E. Mallada, C. Zhao, and S. Low, "Optimal load-side control for frequency regulation in smart grids," *IEEE Transactions on Automatic Control*, vol. 62, no. 12, pp. 6294–6309, 2017.

[4] A. Ademola-Idowu and B. Zhang, "Frequency stability using inverter power control in low-inertia power systems," *IEEE Transactions on Power Systems*, pp. 1–1, 2020.

[5] O. Stanojev, U. Markovic, P. Aristidou, G. Hug, D. S. Callaway, and E. Vrettos, "Mpc-based fast frequency control of voltage source converters in low-inertia power systems," *IEEE Transactions on Power Systems*, pp. 1–1, 2020.

[6] B. B. Johnson, S. V. Dhople, A. O. Hamadeh, and P. T. Krein, "Synchronization of parallel single-phase inverters with virtual oscillator control," *IEEE Transactions on Power Electronics*, vol. 29, no. 11, pp. 6124–6138, 2013.

[7] B. Kroposki, B. Johnson, Y. Zhang, V. Gevorgian, P. Denholm, B.-M. Hodge, and B. Hannegan, "Achieving a 100% renewable grid: Operating electric power systems with extremely high levels of variable renewable energy," *IEEE Power and Energy Magazine*, vol. 15, no. 2, pp. 61–73, 2017.

[8] J. Van de Vyver, J. D. De Kooning, B. Meersman, L. Vandevelde, and T. L. Vandoorn, "Droop control as an alternative inertial response strategy for the synthetic inertia on wind turbines," *IEEE Transactions on Power Systems*, vol. 31, no. 2, pp. 1129–1138, 2015.

[9] Z. Zhang, E. Du, F. Teng, N. Zhang, and C. Kang, "Modeling frequency dynamics in unit commitment with a high share of renewable energy," *IEEE Transactions on Power Systems*, 2020.

[10] Z. Yan and Y. Xu, "Data-driven load frequency control for stochastic power systems: A deep reinforcement learning method with continuous action search," *IEEE Transactions on Power Systems*, vol. 34, no. 2, pp. 1653–1656, 2018.

[11] C. Chen, M. Cui, F. F. Li, S. Yin, and X. Wang, "Model-free emergency frequency control based on reinforcement learning," *IEEE Transactions on Industrial Informatics*, 2020.

[12] J. Duan, D. Shi, R. Diao, H. Li, Z. Wang, B. Zhang, D. Bian, and Z. Yi, "Deep-reinforcement-learning-based autonomous voltage control for power grid operations," *IEEE Transactions on Power Systems*, vol. 35, no. 1, pp. 814–817, 2019.

[13] D. Ernst, M. Glavic, F. Capitanescu, and L. Wehenkel, "Reinforcement learning versus model predictive control: a comparison on a power system problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 517–529, 2008.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[15] D. Cao, W. Hu, J. Zhao, G. Zhang, B. Zhang, Z. Liu, Z. Chen, and F. Blaabjerg, "Reinforcement learning and its applications in modern power and energy systems: A review," *Journal of Modern Power Systems and Clean Energy*, vol. 8, no. 6, pp. 1029–1042, 2020.

[16] 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," *arXiv preprint arXiv:2102.01168*, 2021.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[18] F. Dörfler, M. Chertkov, and F. Bullo, "Synchronization in complex oscillator networks and smart grids," *Proceedings of the National Academy of Sciences*, vol. 110, no. 6, pp. 2005–2010, 2013.

[19] B. K. Poolla, S. Bolognani, and F. Dorfler, "Optimal placement of virtual inertia in power grids," *IEEE Transactions on Automatic Control*, 2017.

[20] R. Ofir, U. Markovic, P. Aristidou, and G. Hug, "Droop vs. virtual inertia: Comparison from the perspective of converter operation mode," in *2018 IEEE International Energy Conference (ENERGYCON)*. IEEE, 2018.

[21] S. V. Dhople, Y. C. Chen, A. Al-Digs, and A. D. Domínguez-García, "Reexamining the distributed slack bus," *IEEE Transactions on Power Systems*, vol. 35, no. 6, pp. 4870–4879, 2020.

[22] B. Xu, Y. Dvorkin, D. S. Kirschen, C. A. Silva-Monroy, and J.-P. Watson, "A comparison of policies on the participation of storage in us frequency regulation markets," in *2016 IEEE Power and Energy Society General Meeting (PESGM)*. IEEE, 2016, pp. 1–5.

[23] H. Zhao, M. Hong, W. Lin, and K. A. Loparo, "Voltage and frequency regulation of microgrid with battery energy storage systems," *IEEE Transactions on smart grid*, vol. 10, no. 1, pp. 414–424, 2017.

[24] Y. Jiang, R. Pates, and E. Mallada, "Dynamic droop control in low-inertia power systems," *IEEE Transactions on Automatic Control*, 2020.

[25] Y.-z. Sun, Z.-s. Zhang, G.-j. Li, and J. Lin, "Review on frequency control of power systems with wind power penetration," in *2010 international conference on power system technology*. IEEE, 2010, pp. 1–8.

[26] H. Bevrani, *Robust power system frequency control*. Springer, 2009, vol. 85.

[27] S. Püschel-L, P. Mancarella *et al.*, "Mapping the frequency response adequacy of the australian national electricity market," in *2017 Australasian Universities Power Engineering Conference (AUPEC)*. IEEE, 2017, pp. 1–6.

[28] D. Tabas and B. Zhang, "Optimal l-infinity frequency control in microgrids considering actuator saturation," *arXiv preprint arXiv:1910.03720*, 2019.

[29] F. Dörfler, M. R. Jovanović, M. Chertkov, and F. Bullo, "Sparsity-promoting optimal wide-area control of power networks," *IEEE Transactions on Power Systems*, vol. 29, no. 5, pp. 2281–2291, 2014.

[30] U. Markovic, Z. Chu, P. Aristidou, and G. Hug, "Lqr-based adaptive virtual synchronous machine for power systems with high inverter penetration," *IEEE Transactions on Sustainable Energy*, vol. 10, no. 3, pp. 1501–1512, 2018.

[31] A. Conn, N. Gould, and P. Toint, "A globally convergent lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds," *Mathematics of Computation*, vol. 66, no. 217, pp. 261–288, 1997.

[32] P. W. Sauer, M. A. Pai, and J. H. Chow, *Power system dynamics and stability: with synchrophasor measurement and power system toolbox*. John Wiley & Sons, 2017.

[33] A. Arapostathis, S. Sastry, and P. Varaiya, "Global analysis of swing dynamics," *IEEE Transactions on Circuits and Systems*, vol. 29, no. 10, pp. 673–679, 1982.

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

[35] H.-D. Chang, C.-C. Chu, and G. Cauley, "Direct stability analysis of electric power systems using energy functions: theory, applications, and perspective," *Proceedings of the IEEE*, vol. 83, no. 11, pp. 1497–1529, 1995.

[36] T. L. Vu and K. Turitsyn, "Lyapunov functions family approach to transient stability assessment," *IEEE Transactions on Power Systems*, vol. 31, no. 2, pp. 1269–1277, 2015.

[37] K. J. Åström and R. M. Murray, *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2010.

[38] A. Griewank, "On automatic differentiation," *Mathematical Programming: recent developments and applications*, vol. 6, no. 6, pp. 83–107, 1989.

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

[40] A. Ortega and F. Milano, "Generalized model of vsc-based energy storage systems for transient stability analysis," *IEEE transactions on Power Systems*, vol. 31, no. 5, pp. 3369–3380, 2015.

[41] P. Demetriou, M. Asprou, J. Quiros-Tortos, and E. Kyriakides, "Dynamic ieee test systems for transient analysis," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2108–2117, 2015.

[42] T. Nishikawa and A. E. Motter, "Comparative analysis of existing models for power-grid synchronization," *New Journal of Physics*, vol. 17, no. 1, p. 015012, 2015.

[43] U. Tamrakar, D. Shrestha, M. Maharjan, B. P. Bhattarai, T. M. Hansen, and R. Tonkoski, "Virtual inertia: Current trends and future directions," *Applied Sciences*, vol. 7, no. 7, p. 654, 2017.

APPENDIX A
REGION OF ATTRACTION

The region of attraction (ROA) interested in this paper is the state (δ, ω) that satisfies $|\delta_i - \delta_j| \in [0, \pi/2]$ for all $(i,j) \in \mathcal{E}$ and $|\omega_i| \in [0, \overline{\omega}_i]$ for all $i = 1, \cdots, N$, where $\overline{\omega}_i$ is the upper bound of ω covers the variation range of frequency. Without loss of generosity, let the angle of bus N be zero and thus the angle of each bus will be determined from the angle difference.

From [18], the equilibrium point $\{(\delta_i^*, \omega_i^*) | w_i = 0, P_i = \sum_{j=1}^{N} B_{ij} \sin(\delta_i^* - \delta_j^*), i = 1, \ldots, N\}$ is unique in the defined ROA. According to (4a), $\frac{\partial V(\delta,\omega)}{\partial \delta_i}$ is monotonic increasing and zero at equilibrium δ_i^* for δ_i that satisfy $|\delta_i - \delta_j| \in [0, \pi/2]$. According to (4b), $\frac{\partial V(\delta,\omega)}{\partial \omega_i}$ is monotonic increasing and zero at equilibrium for ω_i in its boundary $|\omega_i| \in [0, \overline{\omega}_i]$ $\forall i = 1, \cdots, N$. Therefore, the Lyapunov function in ROA is bounded below at the equilibrium and bounded above by the neighbouring local maximal. This makes (3) a qualified Lyapunov function in the defined ROA.

APPENDIX B
PROOF THEOREM 2

For the system with N buses, we have $2N$ state variables and stack as $[\delta_1, \cdots, \delta_N, \omega_1, \cdots, \omega_N]$. Without loss of generosity, let bus N be the slack bus. The Jacobian of the state transition dynamics (1) is

$$\mathbf{J}(\delta,\omega) = \left[\begin{array}{c|c} \mathbf{0} & \mathbf{I} \\ \hline \mathbf{J}_\delta\omega & \mathbf{J}_\omega\omega \end{array}\right] \tag{11}$$

where $\mathbf{0} \in \mathbb{R}^{(N-1)\times(N-1)}$ is the Jacobian of δ with respect to δ and all the element equals to zero according to (1a). $\mathbf{I} \in \mathbb{R}^{(N-1)\times(N-1)}$ is the Jacobian of δ with respect to ω and all the diagonal element equals to one according to (1a). $\mathbf{J}_\delta\omega \in \mathbb{R}^{(N-1)\times(N-1)}$ and $\mathbf{J}_\omega\omega \in \mathbb{R}^{(N-1)\times(N-1)}$ is the jacobian of ω with respect to δ and ω, respectively.

From (1b), the Jacobian of ω with respect to δ is computed as

$$\mathbf{J}_\delta\omega = \left[\begin{array}{cccc} -\sum_{j=1}^N B_{1j}\cos\delta_{1j} & B_{12}\cos\delta_{12} & \cdots & B_{1(N-1)}\cos\delta_{1(N-1)} \\ B_{21}\cos\delta_{21} & -\sum_{j=1}^N B_{2j}\cos\delta_{2j} & \cdots & B_{2(N-1)}\cos\delta_{2(N-1)} \\ \vdots & & \ddots & \\ B_{(N-1)1}\cos\delta_{(N-1)1} & \cdots & & -\sum_{j=1}^N B_{(N-1)j}\cos\delta_{(N-1)j} \end{array}\right] \prec 0 \tag{12}$$

From (1b), the Jacobian of ω with respect to ω is

$$\mathbf{J}_\omega\omega = \left[\begin{array}{ccc} -\frac{\mathrm{d}u_1(\omega_1)}{\mathrm{d}\omega_1} - D_1 & & \\ & \ddots & \\ & & -\frac{\mathrm{d}u_{N-1}(\omega_{N-1})}{\mathrm{d}\omega_{N-1}} - D_{N-1} \end{array}\right] \tag{13}$$

To prove that the system is exponential stable under control, we show that all the real part of eigenvalues are negative for Jacobian $\mathbf{J}(\delta,\omega)$. To this end, we show that there exist a matrix $\mathbf{P} \in \mathbb{R}^{(N-1)\times(N-1)}, \mathbf{P} \succ 0$, such that $\mathbf{J}(\delta,\omega)\mathbf{P} + \mathbf{P}\mathbf{J}(\delta,\omega)^T \prec 0$. Define $\mathbf{P}$ as

$$\mathbf{P} = \left[\begin{array}{c|c} -\left(\mathbf{J}_\delta\omega\right)^{-1} & \beta\left(\mathbf{J}_\delta\omega\right)^{-1} \\ \hline \beta\left(\mathbf{J}_\delta\omega\right)^{-1} & \mathbf{I} \end{array}\right] \tag{14}$$

where $\beta \in \mathbb{R}$ is a positive number and we will specify its value in the following part.

From (12) and (14), we have

$$\mathbf{J}(\delta,\omega)\mathbf{P} + \mathbf{P}\mathbf{J}(\delta,\omega)^T = \left[\begin{array}{c|c} 2\beta\left(\mathbf{J}_\delta\omega\right)^{-1} & \beta\left(\mathbf{J}_\delta\omega\right)^{-1}\left(\mathbf{J}_\omega\omega\right) \\ \hline \beta\left(\mathbf{J}_\omega\omega\right)\left(\mathbf{J}_\delta\omega\right)^{-1} & 2\beta\mathbf{I} + 2\left(\mathbf{J}_\omega\omega\right) \end{array}\right] \tag{15}$$

where $2\beta\left(\mathbf{J}_\delta\omega\right)^{-1} \prec 0$ for positive β since we have shown that $\mathbf{J}_\delta\omega \prec 0$. The Schur complement of $\mathbf{J}(\delta,\omega)\mathbf{P}+\mathbf{P}\mathbf{J}(\delta,\omega)^T$ is

$$\begin{aligned} S =& 2\left(\beta\mathbf{I} + \mathbf{J}_\omega\omega\right) \\ & - \left(\beta\left(\mathbf{J}_\omega\omega\right)\left(\mathbf{J}_\delta\omega\right)^{-1}\right)\left(2\beta\left(\mathbf{J}_\delta\omega\right)^{-1}\right)^{-1}\left(\beta\left(\mathbf{J}_\delta\omega\right)^{-1}\left(\mathbf{J}_\omega\omega\right)\right) \\ =& 2\mathbf{J}_\omega\omega + \beta\left(2\mathbf{I} - \frac{1}{2}\mathbf{J}_\omega\omega\left(\mathbf{J}_\delta\omega\right)^{-1}\mathbf{J}_\omega\omega\right) \end{aligned} \tag{16}$$

From (13), we have $\mathbf{J}_\omega\omega \prec 0$ when the control function $u_i(\omega_i)$ is monotonic increasing with ω_i for all $i = 1, \cdots, N-1$. Take β as a sufficiently small positive number, we have (16)$\prec 0$. Therefore, $\mathbf{J}(\delta,\omega)\mathbf{P} + \mathbf{P}\mathbf{J}(\delta,\omega)^T \prec 0$ and we complete the proof.

APPENDIX C
PROOF OF THEOREM 4

Let α bound the magnitude of first derivative of r on $\mathbb{X}$. Define an equispaced grid of points on $\mathbb{X}$, where $\beta = \frac{1}{n}$ is the spacing between grid points along each dimension. Corresponding to each grid interval $[k\beta, (k+1)\beta]$, assign a linear function $y(x) = r(k\beta) + \frac{r((k+1)\beta)-r(k\beta)}{\beta}(x - k\beta)$, where $y(k\beta) = r(k\beta)$ and $y((k+1)\beta) = r((k+1)\beta)$. For all $x \in [k\beta, (k+1)\beta]$, from monotonic property, we have $r(k\beta) \le r(x) \le r((k+1)\beta)$ and $r(k\beta) \le y(x) \le r((k+1)\beta)$. Therefore, we can bound the approximation error by

$$|y(x) - r(x)| \le |r((k+1)\beta) - r(k\beta)| \tag{17}$$

By mean value theorem, we know that

$$r((k+1)\beta) - r(k\beta) = \beta\frac{\partial r(c)}{\partial x} \tag{18}$$

for some point c on the line segment between $k\beta$ and $(k+1)\beta$. Given the assumptions made at the outset, $|\frac{\partial r(c)}{\partial x}|$ is bounded by α and therefore $|y(x) - r(x)|$ can be bounded by $\beta\alpha$.

Further, we show that any piece-wise linear function of $y(x) = r(k\beta) + \frac{r((k+1)\beta)-r(k\beta)}{\beta}(x - k\beta)$ can be represented by the proposed construction (6)(7). Without loss of generosity, assume that $y(x)$ is the positive fraction and approximated by $f^+(x)$. Let $b_i^1 = 0$, $q^1 = r(\beta)$ and subsequently $b_i^k = (k-1)\beta$, $\sum_{j=1}^k q^j = \frac{r(k\beta)-r((k-1)\beta)}{\beta}$ for $k = 2, 3, \cdots, n$. Then the construction of $f^+(x)$ through (6) is exactly the same as $y(x)$. Therefore, $|f(x) - r(x)|$ can also be bounded by $\beta\alpha$. We take $\beta < \frac{\epsilon}{\alpha}$ to complete the proof.

APPENDIX D
SIMULATION WITHOUT PLL

This appendix illustrates that the inclusion of PLL do not change our results. Fig.10 is the dynamics of system when the controller directly obtain frequency information in simulation. Compared with the case where the frequency deviation is inferred from PLL block in Fig. 5(a), the dynamics in frequency and angles are almost identical. This also confirms other studies which report that the dynamics of PLLs and frequency response of the AC system is mostly decoupled (which is not necessarily true for voltage) [7], [43].

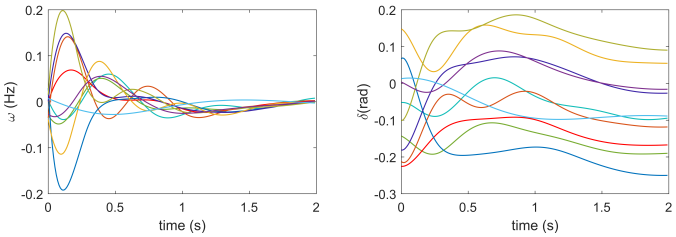


Fig. 10. Dynamics of frequency deviation w and phase angle δ in ten generator buses where the controller obtains accurate frequency deviation instead of inferring from PLLs. It almost identical with Fig.5(a) since PLLs operate at much faster timescales.