RESEARCH ARTICLE

National Science Open
3: 20230054, 2024
https://doi.org/10.1360/nso/20230054

Information Sciences

Learning the continuous-time optimal decision law from discrete-time rewards

Ci Chen[1,2,*], Lihua Xie[3,*], Kan Xie[1,4], Frank Leroy Lewis[5], Yilu Liu[6,7] & Shengli Xie[1,8,*]

[1]*School of Automation, Guangdong University of Technology, Guangdong Key Laboratory of IoT Information Technology, Guangzhou 510006, China;*
[2]*Key Laboratory of Intelligent Information Processing and System Integration of IoT, Ministry of Education, Guangzhou 510006, China;*
[3]*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore;*
[4]*111 Center for Intelligent Batch Manufacturing Based on IoT Technology, Guangzhou 510006, China;*
[5]*UTA Research Institute, the University of Texas at Arlington, Fort Worth 76118, USA;*
[6]*Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville 37996, USA;*
[7]*Oak Ridge National Laboratory, Oak Ridge 37830, USA;*
[8]*Guangdong-HongKong-Macao Joint Laboratory for Smart Discrete Manufacturing, Guangzhou 510006, China*

*Corresponding authors (emails: ci.chen@gdut.edu.cn (Ci Chen); elhxie@ntu.edu.sg (Lihua Xie); shlxie@gdut.edu.cn (Shengli Xie))

Received 6 September 2023; Revised 18 January 2024; Accepted 18 March 2024; Published online 22 March 2024

Abstract: The concept of reward is fundamental in reinforcement learning with a wide range of applications in natural and social sciences. Seeking an interpretable reward for decision-making that largely shapes the system's behavior has always been a challenge in reinforcement learning. In this work, we explore a discrete-time reward for reinforcement learning in continuous time and action spaces that represent many phenomena captured by applying physical laws. We find that the discrete-time reward leads to the extraction of the unique continuous-time decision law and improved computational efficiency by dropping the integrator operator that appears in classical results with integral rewards. We apply this finding to solve output-feedback design problems in power systems. The results reveal that our approach removes an intermediate stage of identifying dynamical models. Our work suggests that the discrete-time reward is efficient in search of the desired decision law, which provides a computational tool to understand and modify the behavior of large-scale engineering systems using the optimal learned decision.

Keywords: continuous-time state and action, decision law learning, discrete-time reward, dynamical systems, reinforcement learning

INTRODUCTION

Reinforcement learning (RL) refers to action-based learning [1], which is responsible for the modification of control policies based on rewards received from the natural and man-made systems under study. RL can be interpreted as an invisible hand introduced by Adam Smith as it describes the social benefits of actions governed by certain interests. Biologically, living organisms learn to act by interacting with the environment and observing the resulting reward stimulus. In cognitive sciences, RL has been used by Burrhus Skinner to study behavior pattern learning based on reinforcement and punishment stimuli.

Recent advances in RL [2–5] have revealed its advantage in understanding complex dynamical systems and extracting decision laws for an underlying system. A seminal breakthrough by Google Deepmind [6, 7]

has resulted in a promising approach to playing the game of Go by using evaluation-based action selections with the help of powerful algorithmic computing. The resulting action has achieved a long-standing goal of artificial intelligence to defeat a world champion in the game of Go, the complexity of which is far beyond the ability of humans to master or model.

Although remarkable, most of the RL results were developed for discrete-time systems. For example, in the game of Go, every board position is within the region of 19×19 blocks meaning that the state and action spaces that the RL algorithms are working on are discrete-time. However, the advantages of using a continuous-time model, rather than the discrete-time one, have become clear in a situation, wherein one aims to identify explicit analytical laws for underlying physical systems [8], such as Newton's laws of motion and Navier-Stokes equations in fluid dynamics [9]. In such cases, the discrete-time system-oriented RL such as the example of the Go game in refs. [6, 7] may not be applicable for searching decision laws for the underlying continuous-time system.

One classic method to search continuous-time decision laws is to compute them after system identification. A review of system identification techniques was documented in ref. [10]. Various techniques have been recently employed to discover the governing equations including symbolic regression-based modeling [8,11], sparse identification [12], empirical dynamic modeling [13, 14], and automated inference of dynamics [15]. In the system identification-based methods, the system model must be learned before the control design.

In the field of RL, there has been slow progress in distilling decision laws for continuous-time dynamic processes. Some early attempts have been made, including refs. [16–20]. However, there are severe limitations in these works, wherein solving a continuous-time Bellman equation is a must for obtaining the optimal decision law. In ref. [16], Euler's method was used to discretize the Bellman equation so that RL-based methods for discrete-time systems can be applied. The main concern for ref. [16] is that its result is only based on the discretized system and may not lead to the optimal control policy even if the sampling period becomes small. Instead of using discretization, an exact method for continuous-time RL was given in ref. [21], wherein an integral reward is designed for feedback. This reward feedback-based technique is later termed as integral reinforcement learning (IRL) [22] as it requires that the integral reward be available for feedback. It is interesting to seek a continuous-time optimal decision law via the integral reward, since the reward is one of the fundamental units in RL to shape the behavior of complex systems. Later, some learning-based studies relied on the assumption of feedbacking the integral reward [23–28].

However, it is not always desirable to use such an integral reward as the integral operation is computationally expensive and storage over-occupying, especially for dynamical systems with large-scale dimensions. As an illustrative example, the utility for learning is defined as a quadratic energy function of the system state and action. Given this kind of utility, calculating the integral reward requires taking the tensor product of two vector spaces (state and action) with the dimensions n and m. After canceling the same items in the integral operation, the total dimension of the stored data for computing the reward becomes $\frac{1}{2}n(n+1)+\frac{1}{2}m(m+1)$ for each sampling. More data storage and computation will be occupied when action and output data are collected over a long period. It thus becomes challenging and interesting to get rid of adverse facts from the integral reward, meanwhile, extracting the continuous-time optimal decision law.

In this work, we focus on a discrete-time reward, which starts at $r(\mathbf{u}(t_1), \mathbf{y}(t_1))$ and stacks $r(\mathbf{u}(t_2), \mathbf{y}(t_2))$ below, and then continues for all $r(\mathbf{u}(t_i), \mathbf{y}(t_i))$ until $r(\mathbf{u}(t_s), \mathbf{y}(t_s))$, which is in a sharp different to the IRL-based method using the integral reward, which starts at $\int_{t_1}^{t_2} r(\mathbf{u}(t), \mathbf{y}(t))\mathrm{d}t$ until $\int_{t_s-1}^{t_s} r(\mathbf{u}(t), \mathbf{y}(t))\mathrm{d}t$ for feedback. The

discrete-time reward has clear physical merits as it could be sampled over non-uniform time and represents a simple slice of the overall integral reward. However, the inner mechanism of learning a decision law for underlying continuous-time dynamical systems from a discrete-time reward remains unstudied. In this work, we aim to study such a mechanism and propose an analytical reinforcement learning framework using the discrete-time reward to capture the optimal decision law for continuous-time dynamical systems. This technical innovation comes from the introduction of state derivative feedback into the learning process, which is in sharp contrast to the existing works based on IRL. We apply this framework to solve output-feedback design problems in power systems. Note that an output-feedback decision law design was given in ref. [28], which, however, required computing integrals of system input and output, wherein one integral is used to formulate rewards and the other for system state reconstruction. Compared to ref. [28], we remove the computation for integral rewards and only need one integral operator for system state reconstruction for output-feedback design. It is seen that the presented framework is a data-driven approach that removes an intermediate stage of identifying dynamical models in model-based control design methods. Our result suggests an analytical framework for achieving desired performance for complex dynamical systems.

CONTINUOUS-TIME OPTIMAL DECISION LAW LEARNING FROM DISCRETE-TIME REWARD

In this work, we revisit the problem of optimal decision law learning for continuous-time dynamical systems. We notice that an analytical dynamical system model is necessary to extract its explicit decision law. Here, we consider the following linear time-invariant continuous-time dynamical system, which is extensively used to capture a large number of physical phenomena in different communities, ranging from the control science [23], the neuroscience [29, 30], to the complex network science [31–33]:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t), \end{cases} \tag{1}$$

where the notation t denotes the time for system evolution, $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^{\mathrm{T}} \in \mathbb{R}^n$ represents the stacked state at the time t with the dimension being $n \times 1$, like operating states of organs within a human digestive system, $\mathbf{y}(t) = [y_1(t), y_2(t), \ldots, y_p(t)]^{\mathrm{T}} \in \mathbb{R}^p$ denotes the output measurement, like the mouth condition among organs of the digestive system, and $\mathbf{u}(t) = [u_1(t), u_2(t), \ldots, u_m(t)]^{\mathrm{T}} \in \mathbb{R}^m$ is the system action for applying the decision law to transform the system state, as the action of eating to stimulate the digestive system. The matrix $\mathbf{A}$ is called drift dynamics denoting how the system state evolves without any action. The action matrix $\mathbf{B}$ describes a mapping between the system state and the controller through which we attempt to change the behavior of the system. The matrix $\mathbf{C}$ denotes a mapping from the state to the output measurement. The system in eq. (1) is assumed to satisfy the controllability of $(\mathbf{A}, \mathbf{B})$, which evaluates the capability of control in manipulating the state. Its dual concept is the observability of $(\mathbf{A}, \mathbf{C})$ for evaluating the ability to observe the state from the output. The controllability and observability condition is standard and essential for the system design and control, and has been widely considered in recent works such as refs. [31–36]. The decision law is also termed as a control policy that aims to take an initial state $\mathbf{x}(0)$ to a state with a prescribed performance using the output of $\mathbf{y}(t)$.

To determine the decision law $\mathbf{u}(t)$, we collect input-output data $\mathbf{U} \in \mathbb{R}^{s\times n}$ and $\mathbf{Y} \in \mathbb{R}^{s\times p}$ over time for the system evolution as

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}^{\mathrm{T}}(t_1) \\ \vdots \\ \mathbf{u}^{\mathrm{T}}(t_i) \\ \vdots \\ \mathbf{u}^{\mathrm{T}}(t_s) \end{bmatrix} = \begin{bmatrix} u_1(t_1) & u_2(t_1) & \dots & u_m(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(t_i) & u_2(t_i) & \dots & u_m(t_i) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(t_s) & u_2(t_s) & \dots & u_m(t_s) \end{bmatrix},$$

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}^{\mathrm{T}}(t_1) \\ \vdots \\ \mathbf{y}^{\mathrm{T}}(t_i) \\ \vdots \\ \mathbf{y}^{\mathrm{T}}(t_s) \end{bmatrix} = \begin{bmatrix} y_1(t_1) & y_2(t_1) & \dots & y_p(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ y_1(t_i) & y_2(t_i) & \dots & y_p(t_i) \\ \vdots & \vdots & \ddots & \vdots \\ y_1(t_s) & y_2(t_s) & \dots & y_p(t_s) \end{bmatrix},$$

where the non-uniform sampling time satisfies $t_1 < t_2 < \ldots < t_i < \ldots < t_{s-1} < t_s$ with t_1 and t_s being, respectively, the points of time when the data collection starts and ends. Given the discrete-time data samples $\mathbf{U}$ and $\mathbf{Y}$, we now formulate a reward function as

$$\mathbf{\Theta}_r(\mathbf{U}, \mathbf{Y}) = \begin{bmatrix} | \\ r(\mathbf{u}(t_i), \mathbf{y}(t_i)) \\ | \end{bmatrix}, \tag{2}$$

where the utility $r(\mathbf{u}(t_i), \mathbf{y}(t_i))$ is defined as $\mathbf{u}^{\mathrm{T}}(t_i)\mathbf{R}\mathbf{u}(t_i) + \mathbf{y}^{\mathrm{T}}(t_i)\mathbf{Q}\mathbf{y}(t_i)$ with weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ being symmetric positive definite for tuning the outputs and actions; and the lines in the equation denote that $\mathbf{\Theta}_r(\mathbf{U}, \mathbf{Y})$ is a collection of the utility data that starts $r(\mathbf{u}(t_1), \mathbf{y}(t_1))$, and then continues for all $r(\mathbf{u}(t_i), \mathbf{y}(t_i))$ until $r(\mathbf{u}(t_s), \mathbf{y}(t_s))$. The notation $\mathbf{\Theta}_r(\mathbf{U}, \mathbf{Y})$ is a vector containing rewards observed by non-uniform sampling. Eq. (2) is called a discrete-time reward, which is in contrast to refs. [21,22] with an integral reward consisting of the integration $\int_{t_i}^{t_{i+1}} \mathbf{u}^{\mathrm{T}}(t)\mathbf{R}\mathbf{u}(t) + \mathbf{y}^{\mathrm{T}}(t)\mathbf{Q}\mathbf{y}(t)\mathrm{d}t$.

Then, we ask: can we design the decision law $\mathbf{u}(t)$ based on the discrete-time reward to solve the optimization problem

$$J(\mathbf{Q}, \mathbf{R}, \mathbf{u}(t), \mathbf{y}(t)) = \min_{\mathbf{u}(t)} \int_0^{\infty} r(\mathbf{u}(t), \mathbf{y}(t))\mathrm{d}t \tag{3}$$

without prior knowledge of the system dynamics $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and the system state $\mathbf{x}(t)$?

The optimization criterion in eq. (3) concerns minimizing the energy in the outputs with the least possible supplied action energy over the infinite continuous-time horizon. The available information for the design of a decision law is only the input and output data. This is called an output-feedback design, meaning that only the output $\mathbf{y}(t)$, rather than the state $\mathbf{x}(t)$, is available. The output-feedback design is more challenging than the state-feedback one, since the system output $\mathbf{y}(t)$ denotes only a part of the full state $\mathbf{x}(t)$. A key issue that needs addressing is how to use the discrete-time reward for decision law learning within the output-feedback design.

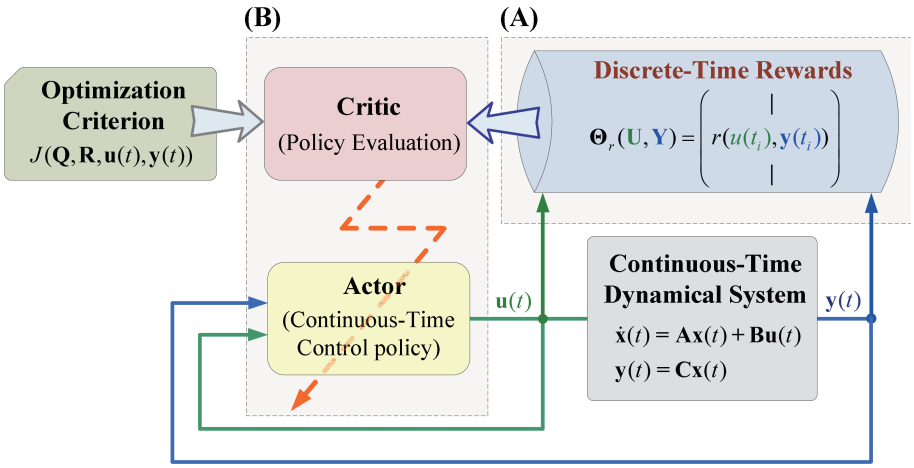

Figure 1 Schematic framework of the reinforcement learning algorithm using policy iteration for continuous-time dynamical systems. (A) At each time $t = t_i$, for $i = 1, 2, \ldots$, one observes the current output $\mathbf{y}(t)$ and action $\mathbf{u}(t)$. The sampled input-output data are collected along the trajectory of the dynamical system in real-time, and are stacked over the time interval $[t_1, t_s]$ as the discrete-time input-output data $\mathbf{U}$ and $\mathbf{Y}$. (B) The input-output data of $\mathbf{U}$ and $\mathbf{Y}$, associated with the prescribed optimization criterion, are used for updating the value estimate given in the critic module, based on which the control policy in the actor module is updated. The ultimate goal of this framework is to use the input-output data $\mathbf{U}$ and $\mathbf{Y}$ for learning the optimal decision law that minimizes the user-defined optimization criterion $J(\mathbf{Q}, \mathbf{R}, \mathbf{u}(t), \mathbf{y}(t))$.

We shall introduce an analytical framework as illustrated in Figure 1 for extracting the optimal decision law that minimizes the criterion in eq. (3) based on the discrete-time reward in eq. (2). A distinguishing feature of the presented framework is that the discrete-time reward is in a central place for learning. The information flow in Figure 1 illustrates that the input-output data are first collected as discrete-time data samples, based on which the discrete-time reward is constructed. Then, the discrete-time reward is fedback to a critic module for updating the value estimate. Next, this updated value estimate is used for control policy improvement, which finally leads to the optimal decision law.

One question for the framework in Figure 1 is the solvability. This was partially answered by the control system community (Supplementary information, Section 1A). Assuming that the system dynamics $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and the system state $\mathbf{x}(t)$ are available for the design, a model-based offline decision law $\mathbf{u}(t)$ that solves the optimization of eq. (3) is given by [23]

$$\mathbf{u}(t) = -\mathbf{K}^*\mathbf{x}(t), \tag{4}$$

where $\mathbf{K}^*$ is an optimal decision gain determined by $\mathbf{K}^* = \mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{P}^*$ with the matrix $\mathbf{P}^*$ obtained from solving an algebraic Riccati equation involving the full system dynamics $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ (Supplementary information, Section 1B).

Considering that only the system output $\mathbf{y}(t)$, rather than the system state $\mathbf{x}(t)$, is available, we may turn to set up an output-feedback design to approximate the optimal decision law in eq. (4) through

$$\hat{\mathbf{u}}(t) = -\mathbf{K}_{\mathbf{o}}^*\boldsymbol{\Phi}(\mathbf{u}(t), \mathbf{y}(t)), \tag{5}$$

where $\mathbf{K}_{\mathbf{o}}^*$ is a feedback decision gain and $\boldsymbol{\Phi}(\mathbf{u}(t), \mathbf{y}(t))$ is a feedforward signal from the data learning point of view. Here, eq. (5) provides a way of transforming the design problem of $\mathbf{u}(t)$ into two subproblems by searching $\mathbf{K}_{\mathbf{o}}^*$ and $\boldsymbol{\Phi}(\mathbf{u}(t), \mathbf{y}(t))$. Indeed, the design of $\boldsymbol{\Phi}(\mathbf{u}(t), \mathbf{y}(t))$ is feedforward depending on the control and output only. One realization form for $\boldsymbol{\Phi}(\mathbf{u}(t), \mathbf{y}(t))$ will be later specified as $\eta(t)$ in this study. We thus

shift the focus to resolving the feedback gain $\mathbf{K}_\mathbf{0}^*$ from the discrete-time reward in eq. (2), under a key premise that the model of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{x}(t)$ defined in eq. (1) exists but the accurate model information is not available for design beforehand.

To search for the gain $\mathbf{K}_\mathbf{0}^*$ that meets the optimization criterion eq. (3) without prior knowledge of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, one may turn to machine learning for the solution. In the setting of machine learning, an unknown system is referred to as an unknown environment. Thus, through interactions with the environment, the control policy design of maximizing a reward, equivalent to a cost to be minimized given in eq. (3), is termed as RL [4]. Recent advances have revealed that RL is a promising method across various disciplines for searching for a decision law that gives rise to satisfactory system performance.

Although great success has been achieved, RL-based results typically assume the state and action constrained in a discrete-time space, while it is not readily feasible to learn the decision law in eq. (5) for continuous-time systems in eq. (1). The framework of continuous-time systems is more suitable for modeling most physical phenomena, as the models of physical systems obtained from the application of physical laws are naturally in continuous-time forms such as refs. [23, 29–33]. Note that the discretization technique may not be applicable for transforming continuous-time systems into discrete-time ones. The reason is rooted in different structures of the optimal decision law between the continuous-time and discrete-time systems.

Another key observation for the framework in Figure 1 is that the dynamical systems are indeed continuous-time in terms of the state $\mathbf{x}(t)$, while the rewards for feedback are sampled over the discrete-time time series. The discrete-time data principle is the cornerstone of parameter learning having numerous applications ranging from control, signal processing, to astrophysics and economics. Although it is now possible to utilize IRL for learning a continuous-time optimal decision law, the method of IRL violates such a principle for data collection and processing. The direct aftermath is that IRL requires measuring the integral of the tensor product of two vector spaces over the time interval $[t_i, t_{i+1}]$, including the output-action data $\int_{t_i}^{t_{i+1}} \mathbf{y}(\tau) \otimes \mathbf{u}(\tau)\mathrm{d}\tau$ (or state-action data $\int_{t_i}^{t_{i+1}} \mathbf{x}(\tau) \otimes \mathbf{u}(\tau)\mathrm{d}\tau$), action-action data $\int_{t_i}^{t_{i+1}} \mathbf{u}(\tau) \otimes \mathbf{u}(\tau)\mathrm{d}\tau$, and output-output data $\int_{t_i}^{t_{i+1}} \mathbf{y}(\tau) \otimes \mathbf{y}(\tau)\mathrm{d}\tau$ (or state-state data $\int_{t_i}^{t_{i+1}} \mathbf{x}(\tau) \otimes \mathbf{x}(\tau)\mathrm{d}\tau$), wherein the symbol $\otimes$ denotes the Kronecker product operator. These integral tensored data are required in IRL as the smallest unit for formulating the integral rewards. Recent advances in adaptive optimal control support this view of the decision law design [22–28], where the continuous-time integration operator has to be applied over the tensor product.

Here, we advocate using discrete-time data samples as the smallest unit, from which the discrete-time reward is constructed for learning the feedback gain $\mathbf{K}_\mathbf{0}^*$. We shall explore the inner mechanism of learning a decision law for the underlying continuous-time dynamical system from the discrete-time reward, and provide rigorous mathematical reasoning for the decision law learning.

The schematic of the presented RL-based framework is illustrated in Figure 2 with a focus on constructing a suitable discrete-time reward for feedback learning.

In Figure 2A, the input-output signals, $\mathbf{u}(t)$ and $\mathbf{y}(t)$, determine the data sets of $\mathbf{U}$ and $\mathbf{Y}$ and also the feedforward signals of $\eta(t) = [\eta_\mathbf{u}^\mathrm{T}(t), \eta_\mathbf{y}^\mathrm{T}(t)]^\mathrm{T} \in \mathbb{R}^{n(m+p)}$ and $\theta(t) = [\dot{\eta}_\mathbf{u}^\mathrm{T}(t), \dot{\eta}_\mathbf{y}^\mathrm{T}(t)]^\mathrm{T} \in \mathbb{R}^{n(m+p)}$ satisfying

$$\dot{\eta}_\mathbf{u}(t) = (\mathbf{I}_m \otimes \mathbf{D}_\eta)\eta_\mathbf{u}(t) + \mathbf{u}(t) \otimes \mathbf{b}, \tag{6}$$

$$\dot{\eta}_\mathbf{y}(t) = (\mathbf{I}_p \otimes \mathbf{D}_\eta)\eta_\mathbf{y}(t) + \mathbf{y}(t) \otimes \mathbf{b}, \tag{7}$$

where the companion matrix $\mathbf{D}_\eta$ and the vector $\mathbf{b}$ are user-defined variables as detailed in Supplementary information, Section 2A. Matrix $\mathbf{D}_\eta$ should be made Hurwitz by choosing the entries on its last low to be

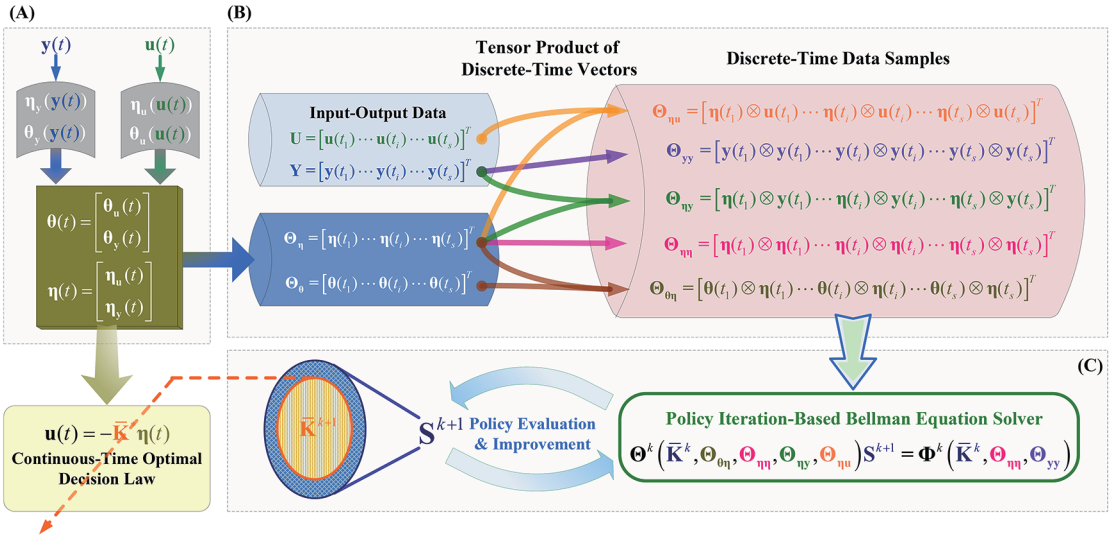

Figure 2 Computational approach for deriving optimal design laws from the data. (A) Pre-process the actions and outputs of the dynamical system and construct the feedforward signals that will be used for the feedback gain learning and the design of an online real-time control loop (Supplementary information, Section 2A). (B) Measure the input-output data, as well as the feedforward signals, over discrete-time series, based on which the discrete-time data samples are assembled using the tensor product (Supplementary information, Section 2B). (C) This part is central for learning the feedback gain $\mathbf{K}_{\mathbf{o}}^{*}$ from discrete-time data. First, calculate the Bellman equation for optimality via policy iterations. Then, through policy evaluation and improvement, the optimal feedback gain is obtained from the discrete-time data samples with rigorous mathematical operations and convergence deduction (Supplementary information, Section 2C). Finally, both the feedforward signal in (A) and the feedback gain $\mathbf{K}_{\mathbf{o}}^{*}$ contribute to the optimal decision law in eq. (5).

negative. Let the feedforward signal $\mathbf{\Phi}(\mathbf{u}(t), \mathbf{y}(t))$ in eq. (5) be realized as $\eta(t)$, which denotes the change of the state after the parametrization [37]. This further generates the following data sets collected over several time instants as

$$\mathbf{\Theta}_{\eta} = \begin{bmatrix} | \\ \eta^{\mathrm{T}}(t_i) \\ | \end{bmatrix} \in \mathbb{R}^{s\times n(p+m)}, \tag{8}$$

$$\mathbf{\Theta}_{\theta} = \begin{bmatrix} | \\ \theta^{\mathrm{T}}(t_i) \\ | \end{bmatrix} \in \mathbb{R}^{s\times n(p+m)}, \tag{9}$$

where n, p, and m are the row dimensions of $\mathbf{x}(t)$, $\mathbf{y}(t)$, and $\mathbf{u}(t)$, respectively, and s denotes the number of time samples. Note that $\eta(t)$ and $\theta(t)$ are vectors in the continuous-time space, as opposed to data matrices $\mathbf{\Theta}_{\eta}$ and $\mathbf{\Theta}_{\theta}$. The results in Figure 2A and B reveal how to learn the feedback gain $\mathbf{K}_{\mathbf{o}}^{*}$ from the discrete-time reward.

Now, all the data sets required in learning the gain $\mathbf{K}_{\mathbf{o}}^{*}$ have been determined, specified as $\mathbf{U}$, $\mathbf{Y}$, $\mathbf{\Theta}_{\eta}$, and $\mathbf{\Theta}_{\theta}$. Based on these four data sets, it is ready to construct the following discrete-time data samples:

$$\mathbf{\Theta}_{\eta\mathbf{u}} = \begin{bmatrix} | \\ \eta^{\mathrm{T}}(t_i) \otimes \mathbf{u}^{\mathrm{T}}(t_i) \\ | \end{bmatrix} \in \mathbb{R}^{s\times nm(p+m)},$$

$$\mathbf{\Theta}_{\mathbf{yy}} = \begin{bmatrix} | \\ \mathbf{y}^{\mathrm{T}}(t_i) \otimes \mathbf{y}^{\mathrm{T}}(t_i) \\ | \end{bmatrix} \in \mathbb{R}^{s\times p^2},$$

$$\mathbf{\Theta}_{\eta\mathbf{y}} = \begin{bmatrix} | \\ \eta^{\mathrm{T}}(t_i) \otimes \mathbf{y}^{\mathrm{T}}(t_i) \\ | \end{bmatrix} \in \mathbb{R}^{s\times np(p+m)},$$

$$\mathbf{\Theta}_{\eta\eta} = \begin{bmatrix} | \\ \eta^{\mathrm{T}}(t_i) \otimes \eta^{\mathrm{T}}(t_i) \\ | \end{bmatrix} \in \mathbb{R}^{s\times n^2(p+m)^2},$$

$$\mathbf{\Theta}_{\theta\eta} = \begin{bmatrix} | \\ \theta^{\mathrm{T}}(t_i) \otimes \eta^{\mathrm{T}}(t_i) \\ | \end{bmatrix} \in \mathbb{R}^{s\times n^2(p+m)^2},$$

using the tensor product of discrete-time spaces, the data flow of which is given in Figure 2B. Take $\mathbf{\Theta}_{\eta\mathbf{u}}$ for example. Each row of $\mathbf{\Theta}_{\eta\mathbf{u}}$ is obtained by the tensor product of the vectors $\eta(t_i)$ and $\mathbf{u}(t_i)$, while each column is collected over different time series ranging from $t = t_1$ to $t = t_s$.

The learning philosophy is now evolved from the integral reward to the discrete-time one, resulting in extra design benefits. The main advantage of using the discrete-time reward is that the computational efficiency is significantly improved when compared to that in the method of IRL. For example, in the considered output-feedback design, the data space for storage equals the sum of the dimension of $\mathbf{U}$, $\mathbf{Y}$, $\mathbf{\Theta}_{\eta}$, and $\mathbf{\Theta}_{\theta}$, which is labeled as $T_{\mathrm{tal},1} = s \times [p + m + 2n(p + m)]$. If a system with the same dimension is applied in the setting of IRL, the data space for storage would become $T_{\mathrm{tal},2} = s \times [nm(p + m) + \frac{p(p+1)}{2} + np(p + m) + \frac{(np+nm)(np+nm+1)}{2}]$, which is obtained by summing the dimensions of $\mathbf{\Theta}_{\eta\mathbf{u}}$, $\mathbf{\Theta}_{\mathbf{yy}}$, $\mathbf{\Theta}_{\eta\mathbf{y}}$, and $\mathbf{\Theta}_{\eta\eta}$ after eliminating same elements in $\mathbf{\Theta}_{\mathbf{yy}}$ and $\mathbf{\Theta}_{\eta\eta}$. The column length of $T_{\mathrm{tal},1}$ is much less than that of $T_{\mathrm{tal},2}$, especially for a large magnitude of s, p, m, or n. Besides, the integral operator has to be imposed for all the $T_{\mathrm{tal},2}$ samples in IRL, while it is ruled out for the $T_{\mathrm{tal},1}$ samples. This reveals that fewer data space for storage and less computational time is consumed in discrete-time reward-based learning when compared to that in IRL.

We seek the feedback gain $\mathbf{K}_{\mathbf{o}}^{*}$ by employing the policy iteration. Let the gain matrix at the kth iteration be $\bar{\mathbf{K}}^{k}$, and let the vector $\mathbf{S}^{k}$ satisfy

$$\mathbf{S}^{k} = \begin{bmatrix} | \\ \mathbf{vec}(\bar{\mathbf{K}}^{k}) \end{bmatrix}, \tag{10}$$

where $\mathbf{vec}(\cdot)$ is the vectorization operator and $\bar{\mathbf{K}}^{0}$ denotes an initial stabilizing gain obtained by trial and error. Construct libraries $\mathbf{\Theta}^{k}(\bar{\mathbf{K}}^{k}, \mathbf{\Theta}_{\theta\eta}, \mathbf{\Theta}_{\eta\eta}, \mathbf{\Theta}_{\eta\mathbf{y}}, \mathbf{\Theta}_{\eta\mathbf{u}})$ and $\mathbf{\Phi}^{k}(\bar{\mathbf{K}}^{k}, \mathbf{\Theta}_{\eta\eta}, \mathbf{\Theta}_{\mathbf{yy}})$ consisting of the discrete-time data samples $\mathbf{\Theta}_{\theta\eta}$, $\mathbf{\Theta}_{\eta\mathbf{y}}$, $\mathbf{\Theta}_{\eta\mathbf{u}}$, $\mathbf{\Theta}_{\eta\eta}$, and $\mathbf{\Theta}_{\mathbf{yy}}$.

As a variant reward of eq. (2), the discrete-time reward used in the policy iteration is defined as follows:

$$\mathbf{\Theta}_r(\mathbf{U}_k, \mathbf{Y}) = \begin{bmatrix} | \\ r(\mathbf{u}_k(t_i), \mathbf{y}(t_i)) \\ | \end{bmatrix}, \tag{11}$$

where $\mathbf{u}_k(t) = -\bar{\mathbf{K}}^k \eta(t)$ denotes an iterative decision law at the kth iteration step. Considering that the optimal decision law is unknown, one needs to use the reward of eq. (11) in the policy iteration, rather than eq. (2), for the algorithmic stability purpose. Based on eq. (11), straightforward manipulation leads to $r(\mathbf{u}_k(t_i), \mathbf{y}(t_i)) = \mathbf{\Theta}_{\eta\eta}\mathbf{vec}((\bar{\mathbf{K}}^k)^{\mathrm{T}}\mathbf{R}\bar{\mathbf{K}}^k) + \mathbf{\Theta}_{\mathbf{yy}}\mathbf{vec}(\mathbf{Q})$. The discrete-time reward in eq. (11) is non-uniformly sampled and can be stacked into the form of $\mathbf{\Phi}^k(\bar{\mathbf{K}}^k, \mathbf{\Theta}_{\eta\eta}, \mathbf{\Theta}_{\mathbf{yy}})$.

With the collected data, we construct a policy iteration-based Bellman equation for solving $\mathbf{S}^{k+1}$:

$$\mathbf{\Theta}^k(\bar{\mathbf{K}}^k, \mathbf{\Theta}_{\theta\eta}, \mathbf{\Theta}_{\eta\eta}, \mathbf{\Theta}_{\eta\mathbf{y}}, \mathbf{\Theta}_{\eta\mathbf{u}})\mathbf{S}^{k+1} = \mathbf{\Phi}^k(\bar{\mathbf{K}}^k, \mathbf{\Theta}_{\eta\eta}, \mathbf{\Theta}_{\mathbf{yy}}), \tag{12}$$

where the matrix $\bar{\mathbf{K}}^{k+1}$, contained in $\mathbf{S}^{k+1}$, is the feedback gain that we want to learn (Supplementary information, Section 2B). A verifiable condition

$$\mathrm{rank}([\mathbf{\Theta}_{\eta\eta}, \mathbf{\Theta}_{\eta\mathbf{u}}]) = (nm + np)\left(\frac{nm + np + 1}{2} + m\right) \tag{13}$$

is proposed for evaluating the richness of the collected data samples that uniquely solve the iterative gain $\bar{\mathbf{K}}^{k+1}$ from eq. (12) (Supplementary information, Section 2C). This rank condition is related to persistent excitation which is well-known in parameter estimation and adaptive control [37–39].

As illustrated in Figure 2C, the computation in eq. (12) is carried out iteratively by replacing $\bar{\mathbf{K}}^{k+1}$ from a previous step k with that from the current one until the convergence criteria are met. Such an iterative procedure ultimately leads to the unique optimal feedback gain. With the converged $\bar{\mathbf{K}}^{k+1}$, label $\bar{\mathbf{K}}^{k+1}$ as $\mathbf{K}_{\mathbf{o}}^*$, and the decision law is then given by eq. (5) which solves the optimization in eq. (3) (Supplementary information, Section 2C).

As for solving $\bar{\mathbf{K}}^{k+1}$ from eq. (12), the computation error, termed as $e_k(t_i)$, consists of the multiplication of a matrix exponential and an initial system state $e^{(\mathbf{A}-\mathbf{LC})t_i}\mathbf{x}(0)$ (Supplementary information, Section 2B). This is indeed an error in the basis function approximation for resolving the state [28], which happens in the pre-processing period for the feedforward signals $\eta(t)$ and $\theta(t)$ as illustrated in Figure 2A. This error is ruled out if $\mathbf{x}(0) = \mathbf{0}$. Despite that the output-feedback design does not allow manipulating the initial state $\mathbf{x}(0)$, one can decrease the computation error $e_k(t_i)$ by executing the pre-processing period for a long enough time. This allows us to handle the unknown non-zero initial state $\mathbf{x}(0)$ and to remove the impacts of the computation error $e_k(t_i)$ by decaying the matrix exponential $e^{(\mathbf{A}-\mathbf{LC})t_i}$.

Even though one way for reducing the computation error $e_k(t_i)$ caused by $\mathbf{x}(0)$ is found, the design for the basis function approximation still needs the dimension of system state $\mathbf{x}(t)$. Fortunately, one may deduce the state dimension, from either the model structure of the physical system or from the system data. From the perspective of physics, it may provide a reasonable estimation of the dimension of the controlled system with the application of physical laws. For example, the relationship between the action force and the resulting mass displacement in the mass-spring-damper system is clear after the application of physical laws. From the perspective of the data, the dimension of the state vector that we are seeking relates the collected input

to output data. Take subspace analysis in biological learning [30] for example. The dimension of the motor learning system state depends on the number of singular values associated with a matrix consisting of input-output data. Also, the System Identification Toolbox built-in commercial software such as Matlab offers graphical user interfaces to assist in the task of model order estimation.

The initial stabilizing gain $\bar{\mathbf{K}}^0$ is required in this work, as well as in all the policy iteration-based RL including [23–28]. The procedure of finding the stabilizing gain $\bar{\mathbf{K}}^0$ is verifiable as we feedback it to the controlled system using $\mathbf{u}(t) = -\bar{\mathbf{K}}^0\boldsymbol{\Phi}(\mathbf{u}(t), \mathbf{y}(t))$. The time for finding such a gain $\bar{\mathbf{K}}^0$ can be used for decaying $e^{(\mathbf{A}-\mathbf{LC})t_i}\mathbf{x}(0)$ in the computation error $e_k(t_i)$.

Robustness to noise has been a vital issue of an algorithm for extracting the decision law. The robustness of the control policy obtained using the proposed framework is analyzed in (Supplementary information, Section 2D). Depending on the noise, it may be necessary to filter the system output $\mathbf{y}(t)$ and action $\mathbf{u}(t)$ before the sampling. For the removal of high-frequency components from the signal, the Nyquist frequency, which is equal to half the sampling rate, is required. To counteract noised signals of $\mathbf{y}(t)$ and $\mathbf{u}(t)$, one feasible solution is to learn a decision law directly for an augmented system that integrates the original control system and the extra filter dynamics. For example, as illustrated in the engineering control system [40], the presented framework works for the augmented system with a filter of system output.

RESULTS

In what follows, we will use the algorithm in Figure 2 to search for the optimal decision law based on the data captured from dynamical power systems. In power systems, it is important to sustain feedback control regulators of the prescribed structure for stability, meanwhile, possessing some desired performance. The algorithm in Figure 2 is responsible for collecting the discrete-time data samples from the power system, defining the discrete-time reward, and learning the prescribed control policy. In the learning process, no additional prior knowledge about the power system's model is used for seeking the optimal decision law.

The design task is now specified as designing an output-feedback continuous-time control policy via discrete-time rewards for solving the load-frequency regulator problem of power systems [23, 41]. The power system dynamics considered consists of governor, turbine, and generator models, whose outputs are governor position change, generator output change, and frequency change. An additional model is introduced for integrating the frequency change to supply the governor model. Linearization is utilized to obtain the power system dynamics operating on the normal condition. The frequency change in the power system, denoted as $\mathbf{y}(t) = \Delta f(t)$, is allowed for the measurement, rather than the whole power system's states. The system dynamics are shown in Figure 3A, wherein the integral action of the frequency deviation is the control policy $\mathbf{u}(t)$ to be designed; the state for the load-frequency regulation system is stacked as $\mathbf{x}(t) = [x_1(t), x_2(t), x_3(t), x_4(t)]^{\mathrm{T}}$ with $x_1(t) = \Delta f(t)$ incremental frequency change (Hz), $x_2(t) = \Delta P_g(t)$ incremental generator output change (p.u. MW), $x_3(t) = \Delta X_g(t)$ incremental governor position change (p.u. MW), and $x_4(t) = \Delta E(t)$ incremental change in voltage angle in radians. The physical parameters for the system dynamics shown in Figure 3A can be found in refs. [23, 41]. The desired control policy is to minimize the following utility over an infinity horizon

$$J(\mathbf{Q} = \mathbf{1}, \mathbf{R} = \mathbf{1}, \mathbf{u}(t), \mathbf{y}(t)) = \min_{\mathbf{u}(t)} \int_0^\infty \mathbf{y}^{\mathrm{T}}(t)\mathbf{y}(t) + \mathbf{u}^{\mathrm{T}}(t)\mathbf{u}(t)\mathrm{d}t \tag{14}$$

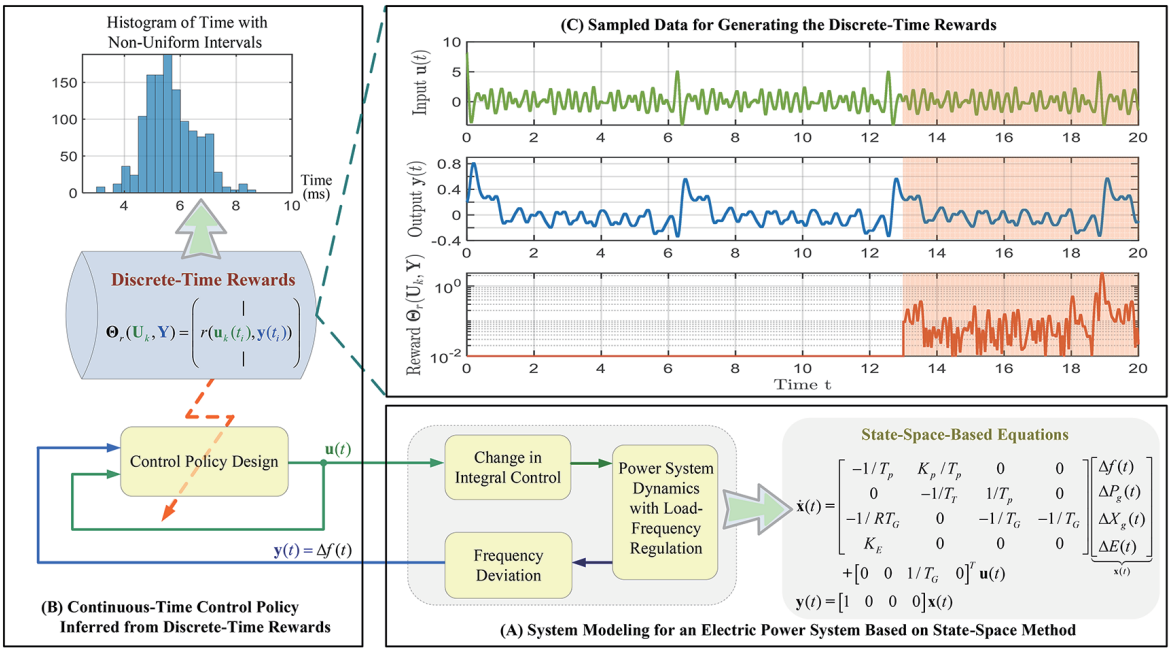

Figure 3 System modeling for an electric power system and the discrete-time data for learning control policy. (A) Load frequency control of an electric power system is modeled by considering its nominal continuous-time system around an operating point specified by a constant load value. Only partial system state $\Delta f(t)$ is available for measurement, requiring the policy design to follow the output-feedback principle. The state-space equation of the power system is highlighted in the figure, but the system dynamics are unknown to the control policy designer. (B) The continuous-time control policy is inferred from the discrete-time reward. The time for the power system's evolution is sampled non-uniformly. (C) Sampled input-output data are collected for generating the discrete-time reward.

using the input-output data from the power system, together with the discrete-time reward, as shown in Figure 3B. In eq. (14), the choice of $\mathbf{Q}$, $\mathbf{R} = \mathbf{1}$ is for the example illustration only. Such a choice of $\mathbf{Q}$, $\mathbf{R}$ works means that the proposed method can also be applied under other feasible choices of $\mathbf{Q}, \mathbf{R}$ for different trials.

It is checkable that the considered power system is stabilizable and detectable, which guarantees that the decision law to be learned exists and is unique [23]. The unique decision law will be solved offline for comparison purposes only. A stabilizing feedback gain $\bar{\mathbf{K}}^0$ is required in our framework and the existing policy iteration-based methods [23–28]. In the power system control design, such a stabilizing gain $\bar{\mathbf{K}}^0$ can be obtained through trial and error as it can be fedback into the system for testing, while the optimal policy associated with $\mathbf{Q}$ and $\mathbf{R}$ is the target for pursuing from the data. Some alternative methods for obtaining the stabilizing gain $\bar{\mathbf{K}}^0$ can be found in the literature such as refs. [23, 24, 42, 43].

Now, we intend to employ the discrete-time reward for inferring the continuous-time control policy satisfying a form of eq. (5) and meeting the performance criterion of eq. (14). The histogram of the time under non-uniform sampling is given in Figure 3B, which indicates that the data only sampled at those instants of time are collected for learning. Such non-uniform sampling will cause difficulty in directly converting a discrete-time transfer function to a continuous-time one [44]. Thus, searching for the optimal decision law by the discretization technique may not be feasible.

From Figure 3C, the data of the input, output, and reward are presented alongside the non-uniform sam-

pling time. Their trajectories along with the time are divided into two regions to distinguish between the feedforward and feedback learning, which, respectively, correspond to the learning of $\mathbf{\Phi}(\mathbf{u}(t), \mathbf{y}(t))$, and $\mathbf{K}_{\mathbf{o}}^{*}$ as formulated in eq. (5).

For the period of feedforward learning, the priority is to reduce the computation error $e_k(t_i)$. To this end, the input and output data in the first 13 s in Figure 3C are used to learn the feedforward signal $\mathbf{\Phi}(\mathbf{u}(t), \mathbf{y}(t))$. Note that a long time for feedforward learning is preferred over a short one. A long time makes the computation error $e_k(t_i)$ small as it exponentially vanishes. During this period, it is not necessary to give the discrete-time reward, since the rewards are only used for the feedback-gain learning in the presented framework. Thus, for the feedforward learning in Figure 3C, the reward is set to 0.01 for using a logarithmic scale on the y-axis.

During the period of the feedback learning (indicated by the region in the orange-red color in Figure 3C, the key task is to collect the input-output data for calculating rewards and for iteratively learning the feedback gain $\mathbf{K}_{\mathbf{o}}^{*}$. Note that the reward is calculated only after the feedforward learning is accomplished. With the feedforward signal $\mathbf{\Phi}(\mathbf{u}(t), \mathbf{y}(t))$ and the stabilizing gain matrix $\bar{\mathbf{K}}^{0}$ ready, one thus constructs an iterative decision law $\mathbf{u}_k(t)$, where k denotes the iteration step in the policy iteration. The discrete-time reward is now calculated as in eq. (11) with the system output data $\mathbf{y}(t)$ and the iterative decision law $\mathbf{u}_k(t)$. At the end of the time for the data collection, namely at the 20th second, by eq. (12), the iterative gain $\bar{\mathbf{K}}^{k}$ is computed for $k = 1, 2, \ldots$ We set a stopping criterion when the norm of the error between two successive gains $\bar{\mathbf{K}}^{k}$ and $\bar{\mathbf{K}}^{k+1}$ is less than 10^{-5}. When the iteration arrives at the 8th step, it yields the feedback policy as

$$\begin{aligned}\bar{\mathbf{K}}^{8} =[\ &-5.0394 \cdot 10^{-8} \quad 136.0782 \quad 43.6079 \quad 3.9698\\ &-63.4867 \quad -57.9988 \quad -11.3416 \quad -0.8173\ ],\end{aligned}$$

which converges to the computed optimal feedback gain $\bar{\mathbf{K}}^{*}$ assuming that the offline state-space equation of the power system is known (see Figure 4D). Therefore, the desired feedback gain is well identified after using the presented framework. The identified gain $\bar{\mathbf{K}}^{8}$, together with the feedforward signal $\mathbf{\Phi}(\mathbf{u}(t), \mathbf{y}(t))$, results in the exact control policy that satisfies the prescribed performance eq. (14).

To illustrate the dynamical learning process, we present the discrete-time reward and the learned control policy gain at each iteration in Figure 4. The reward associated with the action and output at the eighth trial is given in Figure 4A. Rewards used in the first trial and the eighth trial are presented in Figure 4B, where an order of magnitude for the peak value of rewards is reduced from 10^2 in the first trial to 10^0 in the eighth trial. It reveals that the control effort, as well as the discrete-time reward, is reduced after learning. Such a reduction corresponds to the design goal of minimizing the infinity horizon utility in eq. (3). The discrete-time rewards for all eight iterations are given in Figure 4C, wherein the results within the time window from 18.33 to 20 s are now displayed. It further indicates that the rewards tend to decrease as the iteration step moves forward. Figure 4D reveals that the feedback gain $\bar{\mathbf{K}}^{k}$ converges to the predetermined gain as the iteration step increases. Let a matrix $\bar{\mathbf{P}}^{k}$ be an unspecified matrix in eq. (10), which corresponds to eq. (26) of Supplementary information, Section 2B. From Figure 4D, the learning ratio between $\bar{\mathbf{K}}^{k}$ and $\bar{\mathbf{P}}^{k}$ varies in the policy iteration. It implies that the components in eq. (10) differ from each other in terms of learning accuracy. Note that the feedback gain $\bar{\mathbf{K}}^{k}$ is computationally unique for each iteration (Supplementary information, Section 2C). The uniqueness of $\bar{\mathbf{K}}^{k}$ is equivalent to the uniqueness of the learned control policy. From the power system design, it is clear that our framework does not have an intermediate

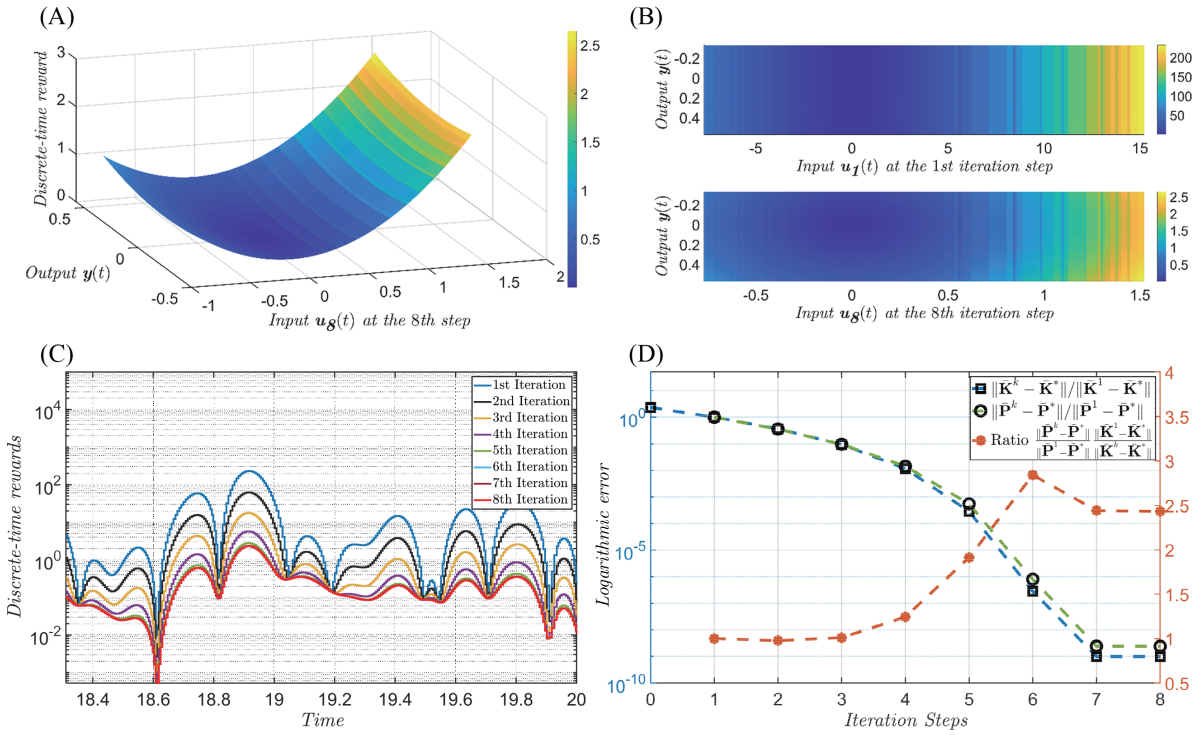

Figure 4 Learning from discrete-time rewards. (A, B) Rewards associated with the output $\mathbf{y}(t)$ and the action $\mathbf{u}_k(t)$ at the kth iteration step. The data here represent the rewards at the 1st and 8th iteration steps, which correspond to the learning results for the first and final trials. (C) Discrete-time rewards for the different iteration steps. It suggests that as the iteration steps go larger, the value of the discrete-time reward decreases. (D) The convergence of learned control policy. The convergence error decreases as the iteration step turns large. This ensures to increase in accuracy by simply setting the iteration step large. The ratio between the gain matrices $\bar{\mathbf{K}}^k$ and $\bar{\mathbf{P}}^k$ reveals the different learning capabilities in approximating the gain matrices from data.

stage of identifying dynamical models, but directly learns the optimal control policy from the data. This reveals that the presented framework provides a data-driven control policy design, meanwhile, the prescribed system performance requirement is incorporated into its design.

The below shows the comparisons between this work and the existing IRL works based on continuous-time rewards such as ref. [28]. For comparison, we use the same parameters and conditions as shown in Figure 3A for learning. Due to the different principles in storage and computation, this work only requires about 0.8 s (the time consuming for CPU computing) to run the iterations based on the data collected over the time period of [13, 20] s, while the IRL-based work requires about 7 s to accomplish under the fixed sampling interval 0.01 s. The feature of high efficiency is ensured since this work now removes the computation of integrals required in the existing works of continuous-time reward. Moreover, from Figure 5, the proposed algorithm's accuracy differs from that of IRL [28] along with the iteration step k even under the same initial conditions including the same fixed sampling interval (0.01 s). The orders of the iterative errors for policy learning $\|\bar{\mathbf{K}}^k - \bar{\mathbf{K}}^*\|$ and value learning $\|\bar{\mathbf{P}}^k - \bar{\mathbf{P}}^*\|$ are, respectively, reduced to about 10^{-4} and 10^{-6} in the eighth trial, while they are about 10^0 and 10^{-3} only by IRL [28]. When increasing the data samples, the traditional design may also have high learning accuracy. Therefore, our design that leverages discrete-time reward admits an advantage at the given samples when compared to the continuous-time reward from the performance viewpoint.

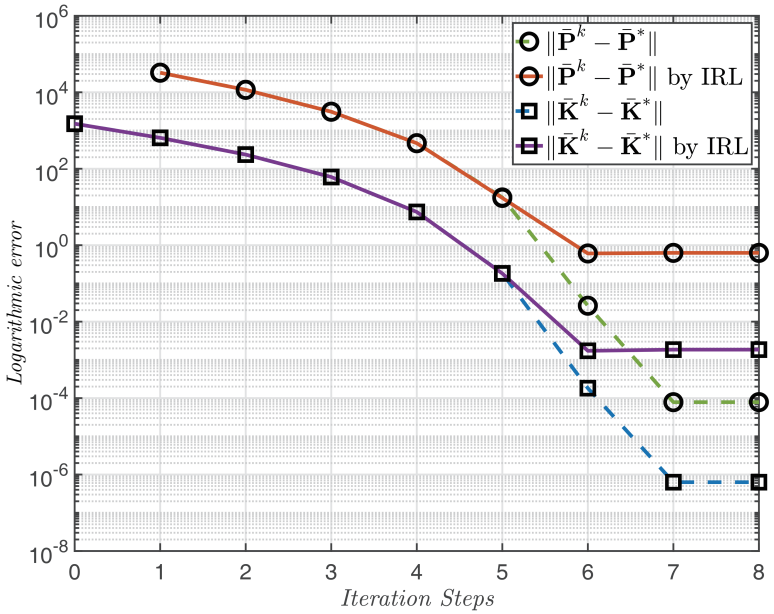

Figure 5 Policy and value learning results by the proposed method and by the method of IRL. The solid lines denote the results of iterations by IRL [28], and the dashed lines are for the proposed method. At the initial iteration, the same policy and value conditions for both two methods, while the convergent norms of policy and value learning errors show that the proposed method has the higher algorithm's accuracy over IRL [28].

We further consider applying the proposed method to a power-grid network with 341 generators, each of which has the following electro-mechanical dynamics [45]:

$$\dot{\delta}_i = \omega_i - \omega_s, \quad \dot{\omega}_i = \frac{1}{H_i}(D_i(\omega_s - \omega_i) + P_{mi} - P_{ei}), \tag{15}$$

where $i = 1, 2, \ldots, 341$; δ_i and ω_i are, respectively, the ith generator's rotor angle and frequency; ω_s denotes the nominal frequency; H_i and D_i are the inertia and damping constants; P_{ei} and P_{mi}, respectively, denote electrical power and mechanical power. The mathematical model of electrical power shall satisfy $P_{ei} = E_i \sum_{j=1}^{341} E_j[\mathrm{Re}(\mathcal{A}_{i,j})\cos(\delta_{ij}) + \mathrm{Im}(\mathcal{A}_{i,j})\sin(\delta_{ij})]$, where $\mathbf{E}$ is a column vector by stacking the internal voltage E_i; δ_{ij} denotes the relative angle between two generators i and j; $\mathcal{A} = [\mathcal{A}_{i,j}]$ is the effective admittance matrix of the grid network representing the coupling among generators; and operators $\mathrm{Im}(\cdot)$ and $\mathrm{Re}(\cdot)$ are, respectively, imaginary and real parts of a complex number.

For each local generator in the power grid, its frequency and frequencies of other generators (ω_i for $i = 1, 2, \ldots, 341$) are not measurable, which makes the policy design indeed an output-feedback regulation problem. The equilibrium, i.e., generation matches consumption, follows from the power-flow equation [46], whose numerical solution such as δ_s for the generator angle at the equilibrium can be offline computed by MATPOWER toolbox [47] for example. Parameters of the 341-machine power grid were given in ref. [36]. In addition to the frequency of the ith generator, the inertia and damping constants D_i and H_i are unknown to the proposed method.

On each generator, a governor is installed and the proposed learning method is applied to extract the optimal policy associated with the weighting matrices $\mathbf{Q}_i = 1$ and $\mathbf{R}_i = 1$. The angle of each generator in the power system is measurable and its trajectory is shown in Figure 6A, while frequencies are not measurable

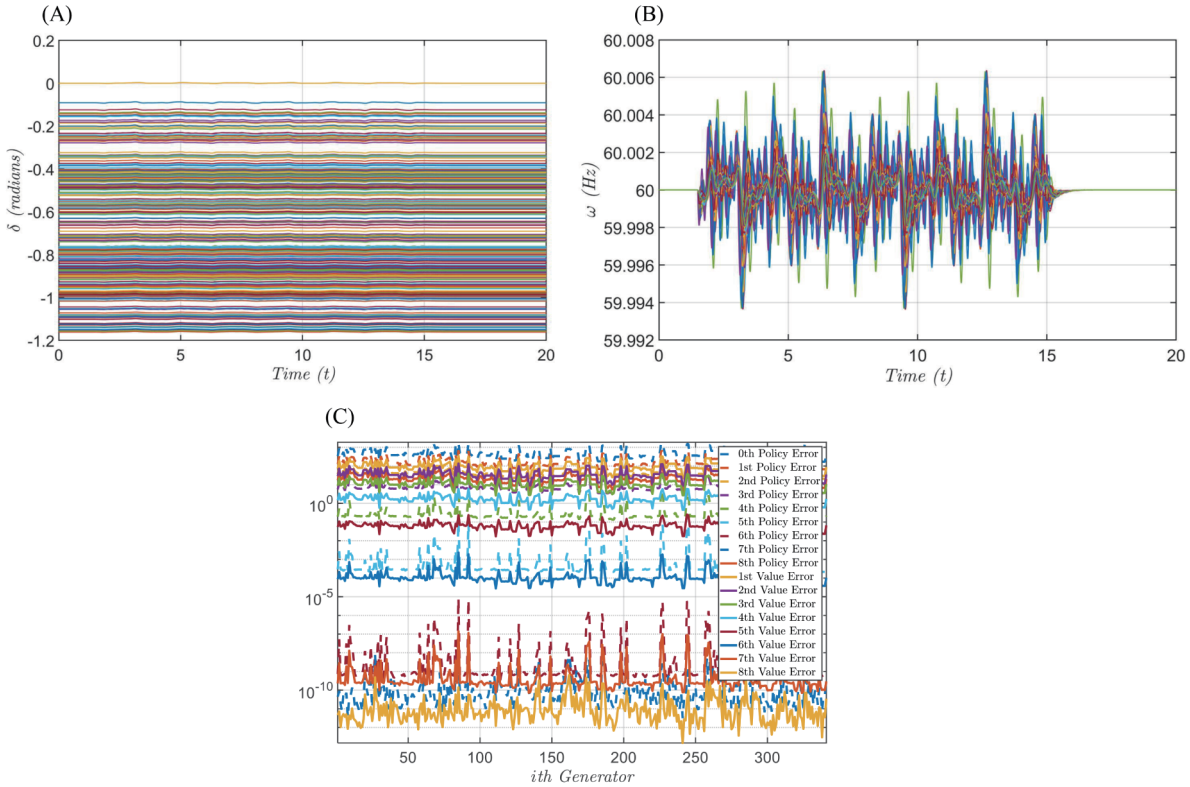

Figure 6 Power-grid network regulation. (A, B) From $t = 0$ s to $t = 1.5$ s, all 341 generators in the network were operating at the equilibrium with $\Delta\delta_i = \delta_i - \delta_s$ and Δw being zeros. At $t = 1.5$ s, exploration noise was added to the multi-machine power network, and the data over the time interval [10, 15] s are collected for policy and value learning. The learned output-feedback policy was installed on each generator governor over [15, 20] s for reaching the equilibrium of the power-grid network. (C) The convergence of the policy and value learning is shown for each local generator.

for feedback and their trajectories are given in Figure 6B. Both Figure 6A and B validate the effectiveness of regulating the power system state to the equilibrium. The norm errors of the policy learning and value learning are shown in Figure 6C with the order of the norm of the error could be driven to around 10^{-10}, which showcases the learning convergence for the power-grid network.

CONCLUSION

We have demonstrated a learning mechanism for extracting the continuous-time optimal decision law from the discrete-time reward through RL. Compared to the integral reward, the discrete-time reward in eq. (2) is computationally efficient as its discrete-time form is a slice of the ultimate reward in eq. (3) with the infinity horizon. We have used the discrete-time reward to build a new RL-based framework that guides the search for the decision law with the desired system performance. The search has been accomplished using the data collected directly from the real-time trajectories of the dynamical systems. Our framework extracts the decision law without an intermediate stage of identifying dynamical models, which is required in a system identification-based control policy design. The analytical RL framework that we proposed is interpretable and provable. This framework may help to better reveal the physics underlying the observed phenomenon

and to enable a system to behave in a desired manner.

In summary, we have revealed the use of the discrete-time-reward-based technique to search the optimal decision law for dynamical systems from data without prior knowledge of the exact model of the dynamical system. We proposed an idea of feedbacking the state derivative into the learning process, which makes our result unique among the previous results in the field. The power of exploiting the state derivative further allows us to establish an analytical RL framework using discrete-time rewards. To achieve this, we have divided the searching procedure into two stages. One stage is to learn a feedforward signal and the other one is to learn the feedback gain. The combination of the feedforward and feedback gains leads to the discovery of the desired control policy directly from the data. We have demonstrated this method in solving design problems in power systems. Within the presented framework, we now equivalently achieve the linear quadric control design using output-feedback control based on the action data and output data. Our framework provides a design tool to understand and transform a dynamical system, with potential applications in fields such as complex networks.

Funding

This work was supported by the Guangdong Basic and Applied Basic Research Foundation (2024A1515011936) and the National Natural Science Foundation of China (62320106008).

Author contributions

C.C., L.X., K.X., F.L.L., Y.L. and S.X. designed the research; C.C., L.X. and S.X. performed the research; C.C., L.X. and S.X. contributed new reagents/analytic tools; C.C., L.X., F.L.L. and S.X. analyzed the data; C.C. and L.X. wrote the supporting information; and C.C., L.X., K.X., F.L.L., Y.L. and S.X. wrote the paper.

Conflict of interest

The authors declare no conflict of interest.

Supplementary information

The supporting information is available online at https://doi.org/10.1360/nso/20230054. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Mendel JM, McLaren RW. Reinforcement-learning control and pattern recognition systems. In: Mendel JM, Fu KS (eds). *Adaptive, Learning and Pattern Recognition Systems: Theory and Applications*. New York: Academic Press, 1970, 287–318.

2 Jordan MI, Mitchell TM. Machine learning: Trends, perspectives, and prospects. *Science* 2015; **349**: 255–260.

3 LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature* 2015; **521**: 436–444.

4 Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*. Cambridge: MIT Press, 2018.

5 Botvinick M, Ritter S, Wang JX, *et al*. Reinforcement learning, fast and slow. *Trends Cogn Sci* 2019; **23**: 408–422.

6 Silver D, Huang A, Maddison CJ, *et al*. Mastering the game of go with deep neural networks and tree search. *Nature* 2016; **529**: 484–489.

7 Silver D, Schrittwieser J, Simonyan K, *et al*. Mastering the game of go without human knowledge. *Nature* 2017; **550**: 354–359.

8 Schmidt M, Lipson H. Distilling free-form natural laws from experimental data. *Science* 2009; **324**: 81–85.

9 Batchelor CK. *An Introduction to Fluid Dynamics*. Cambridge: Cambridge University Press, 2000.

10 Ljung L. *System Identification: Theory for the User*. Prentice-Hall: Upper Saddle River, 1999.

11 Bongard J, Lipson H. Automated reverse engineering of nonlinear dynamical systems. *Proc Natl Acad Sci USA* 2007; **104**: 9943–9948.

12 Brunton SL, Proctor JL, Kutz JN. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc Natl Acad Sci USA* 2016; **113**: 3932–3937.

13 Sugihara G, May R, Ye H, *et al*. Detecting causality in complex ecosystems. *Science* 2012; **338**: 496–500.

14 Ye H, Beamish RJ, Glaser SM, *et al*. Equation-free mechanistic ecosystem forecasting using empirical dynamic modeling. *Proc Natl Acad Sci USA* 2015; **112**: E1569–E1576.

15 Daniels BC, Nemenman I. Automated adaptive inference of phenomenological dynamical models. *Nat Commun* 2015; **6**: 8133.

16 Baird LC. Reinforcement learning in continuous time: Advantage updating. In: *Proceedings of the 1994 IEEE International Conference on Neural Networks (ICNN'94)*. Orlando: IEEE, 1994, 2448–2453.

17 Doya K. Reinforcement learning in continuous time and space. *Neural Comput* 2000; **12**: 219–245.

18 Hanselmann T, Noakes L, Zaknich A. Continuous-time adaptive critics. *IEEE Trans Neural Netw* 2007; **18**: 631–647.

19 Murray J, Cox C, Saeks R, *et al*. Globally convergent approximate dynamic programming applied to an autolander. In: *Proceedings of the 2001 American Control Conference*. Arlington: IEEE, 2001, 2901–2906.

20 Mehta P, Meyn S. Q-learning and pontryagin's minimum principle. In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. Shanghai: IEEE, 2009, 3598–3605.

21 Vrabie D, Pastravanu O, Abu-Khalaf M, *et al*. Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica* 2009; **45**: 477–484.

22 Lewis FL, Vrabie D, Vamvoudakis KG. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Contr Syst Mag* 2012; **32**: 76–105.

23 Lewis FL, Vrabie D, Syrmos VL. *Optimal Control*. Hoboken: John Wiley & Sons, 2012.

24 Jiang Y, Jiang ZP. *Robust Adaptive Dynamic Programming*. Hoboken: John Wiley & Sons, 2017.

25 Kamalapurkar R, Walters P, Rosenfeld J, *et al*. *Reinforcement Learning for Optimal Feedback Control: A Lyapunov-Based Approach*. Cham: Springer, 2018.

26 Chen C, Modares H, Xie K, *et al*. Reinforcement learning-based adaptive optimal exponential tracking control of linear systems with unknown dynamics. *IEEE Trans Automat Contr* 2019; **64**: 4423–4438.

27 Chen C, Lewis FL, Xie K, *et al*. Off-policy learning for adaptive optimal output synchronization of heterogeneous multi-agent systems. *Automatica* 2020; **119**: 109081.

28 Chen C, Xie L, Xie K, *et al*. Adaptive optimal output tracking of continuous-time systems via output-feedback-based reinforcement learning. *Automatica* 2022; **146**: 110581.

29 Todorov E, Jordan MI. Optimal feedback control as a theory of motor coordination. *Nat Neurosci* 2002; **5**: 1226–1235.

30 Shadmehr R, Mussa-Ivaldi S. *Biological Learning and Control: How the Brain Builds Representations, Predicts Events, and Makes Decisions*. Cambridge: MIT Press, 2012.

31 Liu YY, Slotine JJ, Barabási AL. Controllability of complex networks. *Nature* 2011; **473**: 167–173.

32 Ruths J, Ruths D. Control profiles of complex networks. *Science* 2014; **343**: 1373–1376.

33 Li A, Cornelius SP, Liu YY, *et al*. The fundamental advantages of temporal networks. *Science* 2017; **358**: 1042–1046.

34 Zañudo JGT, Yang G, Albert R. Structure-based control of complex networks with nonlinear dynamics. *Proc Natl Acad Sci USA* 2017; **114**: 7234–7239.

35 Yan G, Tsekenis G, Barzel B, *et al*. Spectrum of controlling and observing complex networks. *Nat Phys* 2015; **11**: 779–786.

36 Duan C, Nishikawa T, Motter AE. Prevalence and scalable control of localized networks. *Proc Natl Acad Sci USA* 2022; **119**: e2122566119.

37 Tao G. *Adaptive control design and analysis*. Hoboken: John Wiley & Sons, 2003.

38 Åström KJ, Wittenmark B. *Adaptive Control*. North Chelmsford, MA: Courier Corporation, 2013.

39 Ioannou PA, Sun J. *Robust Adaptive Control*. New York: Courier Corporation, 2012.

40 Stevens BL, Lewis FL, Johnson EN. *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*. Hoboken: John Wiley & Sons, 2015.

41 Wang Y, Zhou R, Wen C. Robust load-frequency controller design for power systems. In: *Proceedings of the First IEEE Conference on Control Applications*. Dayton: IEEE, 1993.

42 Chen C, Lewis FL, Li B. Homotopic policy iteration-based learning design for unknown linear continuous-time systems. *Automatica* 2022; **138**: 110153.

43 Franklin DW, Burdet E, Peng Tee K, *et al*. CNS learns stable, accurate, and efficient movements using a simple algorithm. *J Neurosci* 2008; **28**: 11165–11173.

44 Marvasti F. *Nonuniform Sampling: Theory and Practice*. New York: Springer, 2012.

45 Machowski J, Lubosny Z, Bialek JW, *et al*. *Power System Dynamics: Stability and Control*. Hoboken: John Wiley & Sons, 2020.

46 Kundur PS, Malik OP. *Power System Stability and Control*. New York: McGraw-Hill Education, 2022.

47 Zimmerman RD, Murillo-Sanchez CE, Thomas RJ. MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Trans Power Syst* 2010; **26**: 12–19.