Efficient Reinforcement Learning Through Trajectory Generation

Wenqi Cui WENQICUI@UW.EDU
Department of Electrical and Computer Engineering, University of Washington, WA, USA

Linbin Huang LINHUANG@ETHZ.CH
Department of Information Technology and Electrical Engineering, ETH Zurich, Switzerland

Weiwei Yang WEIWYA@MICROSOFT.COM
Microsoft Research, Redmond, WA, USA

Baosen Zhang ZHANGBAO@UW.EDU
Department of Electrical and Computer Engineering, University of Washington, WA, USA

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

A key barrier to using reinforcement learning (RL) in many real-world applications is the requirement of a large number of system interactions to learn a good control policy. Off-policy and Offline RL methods have been proposed to reduce the number of interactions with the physical environment by learning control policies from historical data. However, their performances suffer from the lack of exploration and the distributional shifts in trajectories once controllers are updated. Moreover, most RL methods require that all states are directly observed, which is difficult to be attained in many settings.

To overcome these challenges, we propose a trajectory generation algorithm, which adaptively generates new trajectories as if the system is being operated and explored under the updated control policies. Motivated by the fundamental lemma for linear systems, assuming sufficient excitation, we generate trajectories from linear combinations of historical trajectories. For linear feedback control, we prove that the algorithm generates trajectories with the exact distribution as if they were sampled from the real system using the updated control policy. In particular, the algorithm extends to systems where the states are not directly observed. Experiments show that the proposed method significantly reduces the number of sampled data needed for RL algorithms.

Keywords: Reinforcement learning, trajectory generation, linear systems, distributional shifts.

1. Introduction

Reinforcement learning (RL) is becoming increasingly popular for the controller design of dynamical systems, especially when the exact system model or parameters are not available (Zheng et al., 2021; Hu et al., 2022). Much of the success in RL has relied on sampling-based algorithms such as the policy gradient algorithm (Sutton and Barto, 2018; Fazel et al., 2018), which typically requires repeated online interactions with the system. Moreover, the control actions need to incorporate sufficient exploration for the learning algorithm to search for better policies (Jin et al., 2021). However, sampling large batch of trajectories is expensive in many real-world problems (e.g., in energy systems, robotics or healthcare), and the exploration requirement for safety-critical systems may be dangerous (Levine et al., 2020; Fujimoto and Gu, 2021).

When the online interactions with the system are limited, two categories of RL methods are designed: off-policy RL and offline RL. Off-policy RL methods (e.g., Q-learning and its variants) typically learn a quality function (i.e, Q function) leveraging past experience, but online interactions with explorations are still required after the update of the control policies (Ghasemipour et al., 2021). Offline RL seeks to learn from a fixed dataset without interactions with environments (Gulcehre et al., 2020; Jin et al., 2021). The fundamental challenge is that once the control policies have been updated, the trajectories of the system under the new policies would not have the same distribution as the historical data (Fujimoto and Gu, 2021; Ghasemipour et al., 2021). As a result, existing algorithms typically constrain the control policy to be close with the policy utilized in the fixed dataset (Ostrovski et al., 2021; Fujimoto and Gu, 2021). Since most algorithms need to do some exploration, it is believed that past data is not helpful if high-reward regions are not covered in the collected trajectories (Levine et al., 2020; Jin et al., 2021).

A fundamental reason behind the above challenges is that the training process is restricted to fixed trajectories in the historical data, hence RL algorithms need to be restricted to historical control policies. We look at the problem from the other direction: *Using only historical data, can we generate trajectories that follow the same distribution induced by a new control policy?*

This paper proposes a trajectory generation algorithm for linear systems, which adaptively generates new trajectories as if the system were being operated and explored under the updated control policies. The key insights come from the fundamental lemma for linear systems, which shows that any set of persistently exciting trajectories can be used to represent the input-output behavior of the system (Willems et al., 2005; De Persis and Tesi, 2019; Markovsky and Dörfler, 2022). Inspired by this, we generate trajectories from linear combinations of historical trajectories, which can come from routine operations of the system. The set of linear combinations is derived from the updated control policy with perturbations on actions, such that the generated trajectory is the same as the trajectory sampled on the real system. This adaptive approach overcomes the challenges in distributional shift and lack of exploration. This is complementary to recent advances in learning linear feedback controllers for linear systems (See, for example, (Fazel et al., 2018; Zheng et al., 2021; Tang et al., 2021; Hu et al., 2022) and references within), where trajectories are sampled through online interactions. Experiments show that the proposed method significantly reduces the number of sampled data needed for RL algorithms. We summarize the contributions as follows:

1) We propose a simple end-to-end approach to generate input-output trajectories for linear systems, which significantly reduces the burden of sample collection in RL methods. In Theorem 2, we prove that the generated trajectories is adaptive to the distribution shift of any linear state-feedback controller with perturbations on actions for explorations. When the states are not directly observed, Theorem 4 shows that this framework also applies to output-feedback control by defining an extended state from the observations.
2) The proposed trajectory generation algorithm is compatible with any RL methods that learns from trajectories. The number of samples needed to learn is independent to the batch size (the number of trajectories in each episode) and the number of training episodes.

2. Preliminaries and problem formulation

2.1. Notations

Throughout this manuscript, vectors are denoted in lower case bold and matrices are denoted in upper case bold, unless otherwise specified. Vectors of all ones and zeros are denoted by $\mathbb{1}_n, \mathbb{0}_n \in \mathbb{R}^n$,

respectively. The identity matrix is denoted by $\boldsymbol{I}_n \in \mathbb{R}^{n\times n}$. We use $\mathcal{N}(\boldsymbol{A})$ to denote the null space of matrix $\boldsymbol{A}$. Given n matrices $\boldsymbol{M}_i, i = 1, \cdots, n$, we denote $[\boldsymbol{M}_1; \cdots; \boldsymbol{M}_n] := [\boldsymbol{M}_1^\top \cdots \boldsymbol{M}_n^\top]^\top$.

Given a discrete-time signal $\boldsymbol{z}(t) \in \mathbb{R}^d$ for $t = 0, 1, \cdots$, we use $\boldsymbol{z}_{[k,k+T]} \in \mathbb{R}^{Td}$ to denote the vector form of the sequence $\{\boldsymbol{z}(k), \ldots, \boldsymbol{z}(k+T)\}$, and the Hankel matrix $\boldsymbol{Z}_{i,t,N} \in \mathbb{R}^{td\times N}$ as

$$\boldsymbol{Z}_{[k,k+T]} = \begin{bmatrix} \boldsymbol{z}(k) \\ \vdots \\ \boldsymbol{z}(k+T) \end{bmatrix}, \boldsymbol{Z}_{i,t,N} = \begin{bmatrix} \boldsymbol{z}(i) & \boldsymbol{z}(i+1) & \cdots & \boldsymbol{z}(i+N-1) \\ \boldsymbol{z}(i+1) & \boldsymbol{z}(i+2) & \cdots & \boldsymbol{z}(i+N) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{z}(i+t-1) & \boldsymbol{z}(i+t) & \cdots & \boldsymbol{z}(i+t+N-2) \end{bmatrix},$$

where k, i and are integers, and t, N, T are natural numbers. The first subscript of $\boldsymbol{Z}_{i,t,N}$ denotes the time at which the first sample of the signal is taken, the second one the number of samples per each column, and the last one the number of signal samples per each row.

2.2. Problem formulation

We consider a discrete-time linear time-invariant system

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) \end{aligned} \tag{1}$$

with state $\boldsymbol{x} \in \mathbb{R}^n$, action $\boldsymbol{u} \in \mathbb{R}^m$, and output $\boldsymbol{y} \in \mathbb{R}^q$ (sometimes called observations in RL literature). If $\boldsymbol{C}$ is not of full (column) rank, we say that the states are not directly observed. Otherwise, we assume $\boldsymbol{C} = \boldsymbol{I}_n$ and the states are directly observed. We assume that $\boldsymbol{A}$ and $\boldsymbol{B}$ are not known. The matrix $\boldsymbol{C}$ is also unknown if the state is not directly observed. We also make the standard assumption that $(\boldsymbol{A}, \boldsymbol{B})$ is stabilizable and $(\boldsymbol{A}, \boldsymbol{C})$ observable (Hespanha, 2009).

A trajectory is a sequence of observations and actions of length T, given by $\boldsymbol{\tau} = \{\boldsymbol{y}(0), \boldsymbol{u}(0), \cdots, \boldsymbol{y}(T-1), \boldsymbol{u}(T-1)\}$. The control action $\boldsymbol{u}(k)$ most commonly comes from the control policy conditioned on the observation at time k, written as $\boldsymbol{\pi_\theta}(\boldsymbol{u}(k) \mid \boldsymbol{y}(k))$ with $\boldsymbol{\theta}$ being the parameter for the control policy. Let $c(\boldsymbol{\tau})$ be the cost defined over the trajectory $\boldsymbol{\tau}$. The goal is to optimize the control paramater $\boldsymbol{\theta}$ to minimize the expected cost over trajectories, written as:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau}\sim p_{\boldsymbol{\pi_\theta}}} c(\boldsymbol{\tau}), \tag{2}$$

where $p_{\boldsymbol{\pi_\theta}}$ is the probability distribution of trajectory subject to the policy $\boldsymbol{\pi_\theta}$. The definition of $c(\boldsymbol{\tau})$ varies for different problems. For linear control policy, quadratic costs are most commonly utilized to convert the optimization into classical linear quadratic problems (Fazel et al., 2018; Zheng et al., 2021). There are typically not closed-form solutions for (2) for other cost functions, e.g., $c(\boldsymbol{\tau}) = \sum_{i=1}^n \max_{k=0,\cdots,K-1} |x_i(k)|$ for the frequency control problem in power systems (Cui et al., 2022b). In this case, gradient-based methods can be utilized to update $\boldsymbol{\theta}$, but the lack of system parameters makes it difficult to compute the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$.

Direct policy gradient methods. Reinforcement learning (RL) is proposed to update $\boldsymbol{\theta}$ through gradient descent, where the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is approximated from trajectories of the system under the control policy $\boldsymbol{\pi_\theta}$. For example, the policy gradient methods in (Sutton and Barto, 2018) shows that the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ can be equivalently computed by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau}\sim p_{\boldsymbol{\pi_\theta}}(\boldsymbol{\tau})} \left[c(\boldsymbol{\tau}) \sum_{k=0}^{K-1} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\pi_\theta}}(\boldsymbol{u}(k) \mid \boldsymbol{y}(k)) \right], \tag{3}$$

where K is the length of the trajectory. The terms inside the expectation can be computed purely from observations and actions in the trajectory.

If $\boldsymbol{C} = \boldsymbol{I}_n$ in (1), the states are directly measured and the control law is called state-feedback control. Otherwise, the control law is called output-feedback control. Note that if $\boldsymbol{C}$ is not full column rank, then $(\boldsymbol{y}(t), \boldsymbol{u}(t))$ cannot uniquely determine $\boldsymbol{y}(t+1)$. Namely, the system is not a Markov decision process with respect to $(\boldsymbol{y}(t), \boldsymbol{u}(t))$ and it is difficult to use generic RL algorithms based on the quality function $Q((\boldsymbol{y}(t), \boldsymbol{u}(t))$ (Jin et al., 2020). For illustration purpose, this paper focus on the policy gradient algorithm (3), and we consider both state and output feedback control.

2.3. Approximate gradients with sampled trajectories

When online interactions with the system are limited, computing the expectation in (3) is not trivial even when the states are directly observed. In training, the expectation in (3) is approximated by the sample average over a large number of trajectories $\boldsymbol{\tau}$ collected from the system under the control policy $\boldsymbol{\pi_\theta}$. Since the distribution $p_{\boldsymbol{\pi_\theta}}(\boldsymbol{\tau})$ depends on the parameter $\boldsymbol{\theta}$, a new set of trajectories need to be collected after each iteration of updating $\boldsymbol{\theta}$. Thus, the number of samples increases with the batchsize (i.e., the number of trajectories in each episode) and the number of training episodes.

We seek to update the control policy using historical trajectories and thus do not need to interact with the system during training. Two challenges arise: *(i) Distribution Shift.* If the control policy changes, the distribution of the historical trajectories would be different from the true distribution $p_{\boldsymbol{\pi_\theta}}(\boldsymbol{\tau})$, potentially resulting in large errors when computing (3). *(ii) Exploration.* Most RL methods need to add (sometimes large) perturbations on actions to encourage exploration, but training with a fixed set of historical trajectories may limit exploration.

End-to-End Trajectory Generation. We propose to overcome the challenges of distribution shift and the lack of exploration through generating trajectories from historical data. In this paper, we focus on learning linear feedback control law $\boldsymbol{u}(k) = \boldsymbol{\theta}\boldsymbol{y}(k)$, with $\boldsymbol{\theta} \in \mathbb{R}^{m\times q}$ being the matrix of trainable parameters. For exploration, the action during training follows the control policy with perturbations $\boldsymbol{w}(k)$ as additive noise, written as

$$\boldsymbol{\pi_\theta}\left(\boldsymbol{u}(k) \mid \boldsymbol{y}(k)\right) := \left\{\boldsymbol{u}(k) = \boldsymbol{\theta}\boldsymbol{y}(k) + \boldsymbol{w}(k), \boldsymbol{w}(k) \sim \mathcal{D}\right\}, \tag{4}$$

where $\mathcal{D}$ is the prescribed distribution for the perturbations. The variance of $\mathcal{D}$ is typically initialized to be large and then shrink with the training episode to achieve exploration v.s. exploitation.

We provide a simple end-to-end approach to generate input-output trajectories without system identification, allowing it to extend to systems where the states are not directly observed (i.e., when $\boldsymbol{C}$ is not full column rank). The generated trajectories are guaranteed to have the same distribution as $p_{\boldsymbol{\pi_\theta}}(\boldsymbol{\tau})$ for all $\boldsymbol{\theta}$ and $\mathcal{D}$. We first show the trajectory generation algorithm for state-feedback control in Section 3, then generalize the results to output-feedback control in Section 4.

3. Trajectory Generation for State-Feedback Control

In this section, we show the conditions when the linear combination of historic trajectories spans all possible trajectories. On this basis, we derive the algorithm to generate trajectories corresponding to updated control policies with perturbations on actions for explorations.

3.1. Span of historic trajectories

Let $\boldsymbol{u}_{d,[0,L-1]}$ and $\boldsymbol{x}_{d,[0,L-1]}$ be the a length-L input and state from past trajectory, and let the corresponding Hankel matrix $\mathcal{H} \in \mathbb{R}^{(Tm+Tn)\times(L-T+1)}$ defined as

$$\underbrace{\begin{bmatrix}\boldsymbol{U}_{0,T,L-T+1}\\ \boldsymbol{X}_{0,T,L-T+1}\end{bmatrix}}_{\mathcal{H}} := \begin{bmatrix}\boldsymbol{u}_d(0) & \boldsymbol{u}_d(1) & \cdots & \boldsymbol{u}_d(L-T)\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{u}_d(T-1) & \boldsymbol{u}_d(T) & \cdots & \boldsymbol{u}_d(L-1)\\ \boldsymbol{x}_d(0) & \boldsymbol{x}_d(1) & \cdots & \boldsymbol{x}_d(L-T)\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{x}_d(T-1) & \boldsymbol{x}_d(T) & \cdots & \boldsymbol{x}_d(L-1)\end{bmatrix}. \tag{5}$$

By the state-space version of Fundamental Lemma (De Persis and Tesi, 2019) shown below, any linear combination of the columns of the Hankel matrix is a length-T input-state trajectory of (1). The proof is provided in (De Persis and Tesi, 2019) and we supplement it in Appendix 7.1 of the longer online version of the paper (Cui et al., 2022a) for completeness.

Lemma 1 (Fundamental Lemma) *If rank* $\begin{bmatrix}\boldsymbol{U}_{0,T,L-T+1}\\ \boldsymbol{X}_{0,L-T+1}\end{bmatrix} = n+Tm$*, then any length-*$T$ *input/state trajectory of system* (1) *can be expressed as* $\begin{bmatrix}\boldsymbol{u}_{[0,T-1]}\\ \boldsymbol{x}_{[0,T-1]}\end{bmatrix} = \begin{bmatrix}\boldsymbol{U}_{0,T,L-T+1}\\ \boldsymbol{X}_{0,T,L-T+1}\end{bmatrix}\boldsymbol{g}$*, where* $\boldsymbol{g} \in \mathbb{R}^{L-T+1}$.

For the rank condition in Lemma 1 to hold, the minimum requirement on the length of the collected trajectory is $L - T + 1 = n + Tm$, namely, $L = (m + 1)T - 1 + n$. When the rank condition holds, linear combination of the columns of the Hankel matrix is also a length-T trajectory of the system. Thus, we generate a trajectory of length T using

$$\big[\tilde{\boldsymbol{u}}(0); \cdots; \tilde{\boldsymbol{u}}(T-1); \tilde{\boldsymbol{x}}(0); \cdots; \tilde{\boldsymbol{x}}(T-1)\big] = \underbrace{\begin{bmatrix}\boldsymbol{U}_{0,T,L-T+1}\\ \boldsymbol{X}_{0,T,L-T+1}\end{bmatrix}}_{\mathcal{H}}\boldsymbol{g}. \tag{6}$$

For convenience, we adopt the notation $\mathcal{H}_u = \boldsymbol{U}_{0,T,L-T+1}$, $\mathcal{H}_x = \boldsymbol{X}_{0,T,L-T+1}$ in the following sections. To represent the rows of blocks starting from the time $k = 0, \cdots, T-1$, we denote

$$\mathcal{H}_x^k := [\boldsymbol{x}_d(k)\ \boldsymbol{x}_d(k+1)\ \cdots\ \boldsymbol{x}_d(L-T+k)],\ \mathcal{H}_u^k := [\boldsymbol{u}_d(k)\ \boldsymbol{u}_d(k+1)\ \cdots\ \boldsymbol{u}_d(L-T+k)]. \tag{7}$$

3.2. Trajectory generation

For generic RL algorithms, a trajectory is sampled from the system that starts from an initial state $\boldsymbol{x}(0)$ and subsequently implements the control policy $\boldsymbol{u}(k) = \boldsymbol{\theta}\boldsymbol{x}(k) + \boldsymbol{w}(k), \boldsymbol{w}(k) \sim \mathcal{D}$ for $k = 0, \cdots, T-1$. Given $\boldsymbol{\theta}$, the probability density function of a trajectory is

$$p_{\pi_{\boldsymbol{\theta}}}(\boldsymbol{\tau}) = p\left(\boldsymbol{x}(0)\right)\prod_{k=0}^{T-1} p\left(\boldsymbol{\theta}\boldsymbol{x}(k) + \boldsymbol{w}(k)|\boldsymbol{x}(k)\right) p\left(\boldsymbol{x}(k+1) \mid \boldsymbol{x}(k), \boldsymbol{\theta}\boldsymbol{x}(k) + \boldsymbol{w}(k)\right), \tag{8}$$

which is uniquely determined by $\boldsymbol{x}(0)$ and the sequence of perturbations $\boldsymbol{w}(0), \boldsymbol{w}(1), \cdots, \boldsymbol{w}(T-1)$.

In the following, we generate trajectories for each updated $\boldsymbol{\theta}$ as if they truly come from the system starting from $\boldsymbol{x}(0)$ under perturbations on actions given by $\boldsymbol{w}(0), \cdots, \boldsymbol{w}(T-1)$. Importantly, we use fixed historic trajectories $\{\boldsymbol{u}_d, \boldsymbol{y}_d\}$ where the Hankel matrix $\mathcal{H}$ in (5) satisfies $\text{rank}(\mathcal{H}) = n + Tm$, and $\boldsymbol{u}_d$ can come from controllers different from the control policy in (4).

The key is to use $\boldsymbol{x}(0)$ and $(\boldsymbol{w}(0), \cdots, \boldsymbol{w}(T-1))$ as extra constraints to find the $\boldsymbol{g}$ in (6) that generates the trajectory which follows the same distribution as (8). From the control policy

$\boldsymbol{u}(k) = \boldsymbol{\theta}\boldsymbol{x}(k) + \boldsymbol{w}(k)$, the trajectory subject to the perturbations $\boldsymbol{w}(0), \boldsymbol{w}(1), \cdots, \boldsymbol{w}(T-1)$ should satisfy

$$\begin{bmatrix} \tilde{\boldsymbol{u}}(0) \\ \vdots \\ \tilde{\boldsymbol{u}}(T-1) \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{\theta} & & \\ & \ddots & \\ & & \boldsymbol{\theta} \end{bmatrix}}_{\boldsymbol{I}_T \otimes \boldsymbol{\theta}} \begin{bmatrix} \tilde{\boldsymbol{x}}(0) \\ \vdots \\ \tilde{\boldsymbol{x}}(T-1) \end{bmatrix} + \begin{bmatrix} \boldsymbol{w}(0) \\ \vdots \\ \boldsymbol{w}(T-1) \end{bmatrix}. \tag{9}$$

Note that the generated initial state is given by $\tilde{\boldsymbol{x}}(0) = \begin{bmatrix} \mathcal{H}_x^0 \end{bmatrix} \boldsymbol{g}$. Combining with (9) gives

$$\underbrace{\begin{bmatrix} \mathcal{H}_u - (\boldsymbol{I}_T \otimes \boldsymbol{\theta})\mathcal{H}_x \\ \mathcal{H}_x^0 \end{bmatrix}}_{\boldsymbol{G}_{\boldsymbol{\theta}}} \boldsymbol{g} = \begin{bmatrix} \boldsymbol{w}(0) \\ \vdots \\ \boldsymbol{w}(T-1) \\ \boldsymbol{x}(0) \end{bmatrix}. \tag{10}$$

Note that the matrix $\boldsymbol{G}_{\boldsymbol{\theta}} \in \mathbb{R}^{(Tm+n)\times(L-T+1)}$ is not a square matrix and its rank is determined by the length of historic trajectory L. When the trajectory is sufficiently long and $\text{rank}(\mathcal{H}) = n + Tm$, we have $L - T + 1 > Tm + n$ and thus there might be multiple $\boldsymbol{g}$ where (10) holds. We use the minimum-norm solution of (10) given by

$$\boldsymbol{g}^* = \boldsymbol{G}_{\boldsymbol{\theta}}^\top \left(\boldsymbol{G}_{\boldsymbol{\theta}} \boldsymbol{G}_{\boldsymbol{\theta}}^\top\right)^{-1} \begin{bmatrix} \boldsymbol{w}(0); \cdots; \boldsymbol{w}(T-1); \boldsymbol{x}(0) \end{bmatrix}. \tag{11}$$

In the next Theorem, we prove the existence and uniqueness of the trajectory generated by (10). The goal is to show that given $(\boldsymbol{w}(0), \cdots, \boldsymbol{w}(T-1), \boldsymbol{x}(0))$, any $\boldsymbol{g}$ that satisfies (10) will generate the same trajectory using $\mathcal{H}\boldsymbol{g}$. So it is suffice to choose the closed-form solution in (11).

Theorem 2 *If* $\text{rank}(\mathcal{H}) = n + Tm$*, there exists at least one solution* $\boldsymbol{g}$ *such that* (10) *holds. Given* $(\boldsymbol{w}(0), \cdots, \boldsymbol{w}(T-1), \boldsymbol{x}(0))$ *and any* $\boldsymbol{g}$ *that solves* (10)*,* $\mathcal{H}\boldsymbol{g}^*$ *generates the same unique trajectory under the control policy* (4) *parameterized by* $\boldsymbol{\theta}$*.*

Theorem 2 shows that $\mathcal{H}\boldsymbol{g}^*$ generates the unique trajectory that starts from an initial state $\boldsymbol{x}(0)$ and subsequently implements the control policy $\boldsymbol{u}(k) = \boldsymbol{\theta}\boldsymbol{x}(k) + \boldsymbol{w}(k), \boldsymbol{w}(k) \sim \mathcal{D}$. Hence, if we fix $\boldsymbol{\theta}$ in (11) and sample $(\boldsymbol{x}(0), \boldsymbol{w}(0), \boldsymbol{w}(1), \cdots, \boldsymbol{w}(T-1))$, then $\mathcal{H}\boldsymbol{g}^*$ will generate a batch of trajectories following the distribution (8) corresponding to the parameter $\boldsymbol{\theta}$. After the update of $\boldsymbol{\theta}$ in each episode of training, we generate a new batch of trajectories by updating $\boldsymbol{G}_{\boldsymbol{\theta}}$ in (10) and sampling new $(\boldsymbol{x}(0), \boldsymbol{w}(0), \boldsymbol{w}(1), \cdots, \boldsymbol{w}(T-1))$. Thus, the generated trajectories adaptively follow the shifted distribution after updating $\boldsymbol{\theta}$. Importantly, it overcomes the negative perception in the field that there is no possibility to improve explorations beyond past trajectories (Levine et al., 2020). By sampling the noises $\boldsymbol{w}(k) \sim \mathcal{D}$, explorations can also be achieved through the generated trajectories. Hence, new trajectories are adaptively generated as if the system were being operated and explored under the updated control policies.

To prove Theorem 2, we first show that the null space of $\boldsymbol{G}_{\boldsymbol{\theta}}$ is exactly the same as that of the Hankel matrix $\mathcal{H}$. Then, $\text{rank}(\mathcal{H}) = n + Tm$ yields $\text{rank}(\boldsymbol{G}_{\boldsymbol{\theta}}) = n + Tm$. The full row rank of $\boldsymbol{G}_{\boldsymbol{\theta}}$ implies that there exist at least one $\boldsymbol{g}$ where (10) holds. The uniqueness of trajectory generated by $\boldsymbol{g}$ follows from the fact that $\boldsymbol{G}_{\boldsymbol{\theta}}$ and $\mathcal{H}$ have the same null space. Details of the proof is given below.

Proof We first prove that the null space $\mathcal{N}(\boldsymbol{G}_{\boldsymbol{\theta}})$ is the same as $\mathcal{N}(\mathcal{H})$ from *(i)* and *(ii)* :

(i) For all $\boldsymbol{q} \in \mathcal{N}(\mathcal{H})$, we have $[\mathcal{H}_x]\boldsymbol{q} = \mathbb{0}_{Tn}$ and $[\mathcal{H}_u]\boldsymbol{q} = \mathbb{0}_{Tm}$. Plugging in $\boldsymbol{G}_{\boldsymbol{\theta}}$ in (10) yields $\boldsymbol{G}_{\boldsymbol{\theta}}\boldsymbol{q} = \mathbb{0}_{Tm+n}$. Namely, $\boldsymbol{q} \in \mathcal{N}(\boldsymbol{G}_{\boldsymbol{\theta}})$.

(ii) For all $\boldsymbol{v} \in \mathcal{N}(\boldsymbol{G_\theta})$, $\boldsymbol{G_\theta v} = \mathbb{0}_{Tm+n}$ yields $\mathcal{H}_x^0 \boldsymbol{v} = \mathbb{0}_n$ and $\mathcal{H}_u^k \boldsymbol{v} = \hat{\boldsymbol{\theta}} \mathcal{H}_x^k \boldsymbol{v}$ for $k = 0, \cdots, T-1$. Thus, $\mathcal{H}_u^0 \boldsymbol{v} = \hat{\boldsymbol{\theta}} \mathcal{H}_x^0 \boldsymbol{v} = \mathbb{0}_m$. From $\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k)$, we have $\mathcal{H}_x^{k+1} = A\mathcal{H}_x^k + B\mathcal{H}_u^k$. From $\mathcal{H}_x^0 = \mathbb{0}_n$ and $\mathcal{H}_u^0 = \mathbb{0}_m$, we apply $\mathcal{H}_x^{k+1} = A\mathcal{H}_x^k + B\mathcal{H}_u^k$ and $\mathcal{H}_u^k \boldsymbol{v} = \hat{\boldsymbol{\theta}} \mathcal{H}_x^k \boldsymbol{v}$ alternately. This induces $\mathcal{H}_x^k \boldsymbol{v} = \mathbb{0}_n$ and $\mathcal{H}_u^k \boldsymbol{v} = \mathbb{0}_m$ for $k = 0, \cdots, T-1$. Hence, $\mathcal{H}\boldsymbol{v} = \mathbb{0}_{Tm+Tn}$.

Next, we prove the existence of the solution in (10). Note that $\mathcal{H} \in \mathbb{R}^{(Tm+Tn)\times(L-T+1)}$. If $\text{rank}(\mathcal{H}) = n + Tm$, then $\text{rank}(\mathcal{N}(\mathcal{H})) = (L-T+1) - (n+Tm)$. Since $\mathcal{N}(\boldsymbol{G_\theta})$ is the same as $\mathcal{N}(\mathcal{H})$, then $\text{rank}(\mathcal{N}(\boldsymbol{G_\theta})) = (L-T+1) - (n+Tm)$. It follows directly that $\text{rank}(\boldsymbol{G_\theta}) = (L-T+1) - \text{rank}(\mathcal{N}(\boldsymbol{G_\theta})) = n + Tm$. Note that the number of rows of $\boldsymbol{G_\theta}$ is $n+Tm$, then the full row-rank of $\text{rank}(\boldsymbol{G_\theta})$ shows that there exists at least one solution such that (10) holds.

Lastly, we show the uniqueness of the generated trajectory. Suppose there exists $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$, which are both solutions of (10) and $\mathcal{H}\boldsymbol{g}_1 \neq \mathcal{H}\boldsymbol{g}_2$. Since $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ are both solution of (10), then $\boldsymbol{G_\theta g}_1 = \boldsymbol{G_\theta g}_1$ and thus $(\boldsymbol{g}_1 - \boldsymbol{g}_2) \in \mathcal{N}(\boldsymbol{G_\theta})$. On the other hand, $\mathcal{H}\boldsymbol{g}_1 \neq \mathcal{H}\boldsymbol{g}_2$ yields $\mathcal{H}(\boldsymbol{g}_1 - \boldsymbol{g}_2) \neq 0$ and thus $(\boldsymbol{g}_1 - \boldsymbol{g}_2) \notin \mathcal{N}(\mathcal{H})$. This contradicts that $\mathcal{N}(\boldsymbol{G_\theta})$ is the same as $\mathcal{N}(\mathcal{H})$. Hence, $\mathcal{H}\boldsymbol{g}_1 = \mathcal{H}\boldsymbol{g}_2$, namely, the generated trajectories are identical. ■

3.3. Algorithm

By Theorem 2, using historical data, given $\boldsymbol{x}(0)$ and perturbations on actions $\boldsymbol{w}(0), \cdots, \boldsymbol{w}(T-1)$, a trajectory can be generated as if it comes from sampling the system with the current control policy. The details of the algorithm is illustrated in Algorithm 1. We also use REINFORCE policy gradient (Sutton and Barto, 2018) in Appendix 7.2 of the online version (Cui et al., 2022a) as an example to show how to use the trajectory generation algorithm in RL methods. The key benefit of Algorithm 1 is that it is adaptive to the updates of parameter $\boldsymbol{\theta}$ and $\boldsymbol{w}(t)$ for exploration. In particular, $\boldsymbol{w}(t)$ can be sampled from any distribution $\mathcal{D}$, making it versatile for different applications.

Algorithm 1 Trajectory generation for state-feedback control

Data collection: Collect historic measurement of the system and stack each T-length input-output trajectory as Hankel matrix $\mathcal{H}$ shown in (5) until $\text{rank}(\mathcal{H}) = n + Tm$

Data generation: *Input* :Hankel matrix $\mathcal{H}$, weights $\boldsymbol{\theta}$ and the distribution $\mathcal{D}$ for the control policy, the batchsize Q (number of the generated trajectories), the distribution $\mathcal{S}_x$ of the initial states[1]

Function `TrajectoryGen` $(\mathcal{H}, \boldsymbol{\theta}, \mathcal{D}, Q, \mathcal{S}_x)$**:**

Plug in $\boldsymbol{\theta}$ to compute $\boldsymbol{G_\theta} = \left[\mathcal{H}_u - (\boldsymbol{I}_T \otimes \boldsymbol{\theta})\mathcal{H}_x; \mathcal{H}_x^0\right]$

for $i = 1$ *to* Q **do**

Sample $\boldsymbol{x}_i(0)$ from $\mathcal{S}_x$. Sample $\{\boldsymbol{w}_i(0), \cdots, \boldsymbol{w}_i(T-1)\}$ from $\mathcal{D}$.

Compute the coefficient $\boldsymbol{g}_i^* = \boldsymbol{G}_{\boldsymbol{\theta}}^\top \left(\boldsymbol{G_\theta G}_{\boldsymbol{\theta}}^\top\right)^{-1} [\boldsymbol{w}_i(0); \cdots; \boldsymbol{w}_i(T-1); \boldsymbol{x}_i(0)]$.

Generate the i-th trajectory $\boldsymbol{\tau}_i := [\tilde{\boldsymbol{u}}_i(0); \cdots; \tilde{\boldsymbol{u}}_i(T-1); \tilde{\boldsymbol{x}}_i(0); \cdots; \tilde{\boldsymbol{x}}_i(T-1);] = \mathcal{H}\boldsymbol{g}_i^*$.

end

return $[\boldsymbol{\tau}_1, \cdots, \boldsymbol{\tau}_Q]$

4. Trajectory Generation for Output-Feedback Control

In this section, we show how to obtain a Markov decision process by defining an extended state using input-output trajectory. The key difference to Section 3 is that $\boldsymbol{G_\theta}$ may not be full row rank even when the rank condition on the Hankel matrix holds. Thus, the coefficients for generated trajectories and associated proofs are more nuanced.

1. The set of historic initial states in past trajectories can be used to estimate $\mathcal{S}_x$.

4.1. Extended states for constructing Markov decision process

Let $\mathcal{O}_{[0,\ell]}(\boldsymbol{A},\boldsymbol{C}) := \operatorname{col}\left(\boldsymbol{C},\boldsymbol{CA},\ldots,\boldsymbol{CA}^{\ell-1}\right)$ be the extended observability matrix. The lag of the system (1) is defined by the smallest integer $\ell \in \mathbb{Z}_{\geq 0}$ such that the observability matrix $\mathcal{O}_{[0,\ell]}(\boldsymbol{A},\boldsymbol{C})$ has rank n, i.e., the state can be reconstructed from ℓ measurements (Huang et al., 2021).

Let $T_0 \geq \ell$ be the length of a trajectory. Define the extended states as

$$\mathcal{X}(k-1) := \left[\ \boldsymbol{y}(k-T_0);\cdots;\boldsymbol{y}(k-1);\boldsymbol{u}(k-T_0);\cdots;\boldsymbol{u}(k-2)\ \right]. \tag{12}$$

Then extending the system transition from time step 0 to T_0 gives

$$\mathcal{X}(k) = \tilde{\boldsymbol{A}}\mathcal{X}(k-1) + \tilde{\boldsymbol{B}}\boldsymbol{u}(k-1) \text{ for } k \geq T_0, k \in \mathbb{Z}, \tag{13}$$

which is a Markov decision process in terms of the extended states. Detailed proof and the definition of system transition matrix $(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{B}})$ is given in Appendix 7.3 in the longer online version of the paper (Cui et al., 2022a). For the output-feedback control law in (4), we have $p\left(\boldsymbol{u}(k)|\mathcal{X}(k)\right) = p\left(\boldsymbol{u}(k)|\boldsymbol{y}(k)\right)$ and it is straightforward to show that policy gradient algorithm using (3) still works. The proof is given in Appendix 7.4 of the longer online version (Cui et al., 2022a).

By defining the the Hankel matrix as $\mathcal{H} := \begin{bmatrix}\boldsymbol{U}_{0,T,L-T+1}\\ \boldsymbol{Y}_{0,T,L-T+1}\end{bmatrix} \in \mathbb{R}^{(Tm+Tq)\times(L-T+1)}$, the following fundamental Lemma in terms of input-output trajectories holds.

Lemma 3 (Fundamental Lemma Willems et al. (2005); Markovsky and Dörfler (2022)) *If rank* $\begin{bmatrix}\boldsymbol{U}_{0,T,L-T+1}\\ \boldsymbol{Y}_{0,L-T+1}\end{bmatrix} = n+Tm$*, then any length-*$T$ *input/output trajectory of system* (1) *can be expressed as* $\begin{bmatrix}\boldsymbol{u}_{[0,T-1]}\\ \boldsymbol{y}_{[0,T-1]}\end{bmatrix} = \begin{bmatrix}\boldsymbol{U}_{0,T,L-T+1}\\ \boldsymbol{Y}_{0,T,L-T+1}\end{bmatrix}\boldsymbol{g}$ *where* $\boldsymbol{g} \in \mathbb{R}^{L-T+1}$.

When the rank condition in Lemma 3 holds, linear combination of the columns of the Hankel matrix is an input/output trajectory of the system. We then generate trajectory of length T using

$$\left[\ \tilde{\boldsymbol{u}}(0);\cdots;\tilde{\boldsymbol{u}}(T-1);\tilde{\boldsymbol{y}}(0);\cdots;\tilde{\boldsymbol{y}}(T-1)\ \right] = \mathcal{H}\boldsymbol{g}. \tag{14}$$

For convenience, we adopt the notation $\mathcal{H}_u = U_{0,T,L-T+1}$, $\mathcal{H}_u = Y_{0,T,L-T+1}$ in the following sections. To represent the lines of blocks starting from the time $k = 0,\cdots,T-1$, $\mathcal{H}_u^k$ the same as in (7) and $\mathcal{H}_y^k := [y_d(k)\ \cdots\ y_d(L-T+k)]$. We also define the stacked blocks starting from $k_1,\cdots,k_2$ as $\mathcal{H}_y^{k_1:k_2} := \left[H_y^{k_1};H_y^{k_1+1};\cdots;H_y^{k_2}\right]$ and $\mathcal{H}_u^{k_1:k_2} := \left[H_u^{k_1};H_u^{k_1+1};\cdots;H_u^{k_2}\right]$.

4.2. Trajectory generation

Using the transition dynamics (13), the probability density function of a length-T trajectory is

$$p_{\pi_{\boldsymbol{\theta}}}(\boldsymbol{\tau}) = p\left(\mathcal{X}\left(T_0-1\right)\right)\prod_{k=T_0-1}^{T-1} p\left(\boldsymbol{\theta}\boldsymbol{y}(k)+\boldsymbol{w}(k)|\mathcal{X}(k)\right)p\left(\mathcal{X}(k+1)|\mathcal{X}(k),\boldsymbol{\theta}\boldsymbol{y}(k)+\boldsymbol{w}(k)\right),$$

which is uniquely determined by $\mathcal{X}\left(T_0-1\right)$ and the sequences $\boldsymbol{w}(T_0-1),\cdots,\boldsymbol{w}(T-1)$.

In analogy with the derivation in (9), we aim to generate the trajectory starting from $\mathcal{X}\left(T_0-1\right)$ under perturbations on actions given by $\boldsymbol{w}(T_0-1),\cdots,\boldsymbol{w}(T-1)$. From the control policy $\boldsymbol{u}(k) = \boldsymbol{\theta}\boldsymbol{y}(k)+\boldsymbol{w}(k)$, the trajectory subject to the perturbations $\boldsymbol{w}(T_0),\cdots,\boldsymbol{w}(T-1)$ should satisfy

$$\left[\mathcal{H}_u^{T_0-1:T-1}\right]\boldsymbol{g} = \left(\boldsymbol{I}_{T-T_0}\otimes\boldsymbol{\theta}\right)\left[\mathcal{H}_y^{T_0-1:T-1}\right]\boldsymbol{g} + \begin{bmatrix}\boldsymbol{w}(T_0-1)\\ \vdots\\ \boldsymbol{w}(T-1)\end{bmatrix}. \tag{15}$$

Note that the generated extended initial state is given by $\tilde{\mathcal{X}}(T_0-1) := \left[\mathcal{H}_y^{0:T_0-1}; \mathcal{H}_u^{0:T_0-2}\right] \boldsymbol{g}$. Together with the constraints in (15) gives

$$\underbrace{\begin{bmatrix} \mathcal{H}_u^{T_0-1:T-1} - (\boldsymbol{I}_{T-T_0} \otimes \boldsymbol{\theta})\, \mathcal{H}_y^{T_0-1:T-1} \\ \hline \mathcal{H}_y^{0:T_0-1} \\ \mathcal{H}_u^{0:T_0-2} \end{bmatrix}}_{G_\theta} \boldsymbol{g} = \underbrace{\begin{bmatrix} \boldsymbol{w}(T_0-1) \\ \vdots \\ \boldsymbol{w}(T-1) \\ \hline \mathcal{X}(T_0-1) \end{bmatrix}}_{\boldsymbol{R}} \tag{16}$$

Note that the matrix $\boldsymbol{G_\theta} \in \mathbb{R}^{(Tm+T_0 d)\times(L-T+1)}$ is not a square matrix and there might be multiple solutions to (16). Moreover, $\boldsymbol{G_\theta}$ is not full row rank and $\left(\boldsymbol{G_\theta}\boldsymbol{G_\theta}^\top\right)$ is not invertible. Here, we compute the eigenvalue decomposition of $\left(\boldsymbol{G_\theta}\boldsymbol{G_\theta}^\top\right)$. Let s be the number of nonzero eigenvalue of $\left(\boldsymbol{G_\theta}\boldsymbol{G_\theta}^\top\right)$. Let λ_i be the i-th non-zero eigenvalue and $\boldsymbol{p}_i$ be the associated eigenvector of $\left(\boldsymbol{G_\theta}\boldsymbol{G_\theta}^\top\right)$. Denote $\boldsymbol{P_\theta} := \begin{bmatrix}\boldsymbol{p}_1 & \boldsymbol{p}_2 & \cdots & \boldsymbol{p}_s\end{bmatrix}$ and $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_s)$. Then clearly $\left(\boldsymbol{G_\theta}\boldsymbol{G_\theta}^\top\right) = \boldsymbol{P_\theta}\boldsymbol{\Lambda}\boldsymbol{P_\theta}^\top$ and we compute the solution of (16) given by

$$\boldsymbol{g}^* = \boldsymbol{G_\theta}^\top \boldsymbol{P_\theta} \boldsymbol{\Lambda}^{-1} \boldsymbol{P_\theta}^\top \left[\ \boldsymbol{w}(T_0-1); \cdots; \boldsymbol{w}(T-1); \mathcal{X}(T_0-1)\ \right]. \tag{17}$$

Next, we prove the existence and uniqueness of the trajectory generated by (16). The goal is to show that given $(\boldsymbol{w}(T_0-1), \cdots, \boldsymbol{w}(T-1), \mathcal{X}(T_0-1))$, any $\boldsymbol{g}$ that satisfies (16) will generate the same trajectory using $\mathcal{H}\boldsymbol{g}$. So it is suffice to choose the closed-form solution in (17).

Theorem 4 *If rank$(\mathcal{H}) = n + Tm$, there exists at least one solution $\boldsymbol{g}^*$ such that (16) holds. Given $(\boldsymbol{w}(T_0-1), \cdots, \boldsymbol{w}(T-1), \mathcal{X}(T_0-1))$ and any $\boldsymbol{g}$ that solves (16), $\mathcal{H}\boldsymbol{g}^*$ generates the same unique trajectory under the control policy (4) parameterized by $\boldsymbol{\theta}$.*

The key is to show that rank$(\boldsymbol{G_\theta}) = n + Tm$ if rank$(\mathcal{H}) = n + Tm$ and the rank of $\boldsymbol{R}$ is at most $n + Tm$. Then the existence and uniqueness of the trajectory generated by $\mathcal{H}\boldsymbol{g}^*$ follows similar proof as Theorem 2. Hence, we can generate a trajectory $\mathcal{H}\boldsymbol{g}^*$ by randomly sampling $\mathcal{X}(T_0-1)$ and $\boldsymbol{w}(t) \in \mathcal{D}$ for $t = T_0-1, \cdots, T-1$. For the cost (2) calculated on the trajectory of the length K, we setup $T = K + T_0 - 1$, and using the generated trajectory $\mathcal{H}\boldsymbol{g}^*$ from T_0-1 to $T-1$ to train the controller. The detailed proof and algorithm are found in the online version (Cui et al., 2022a).

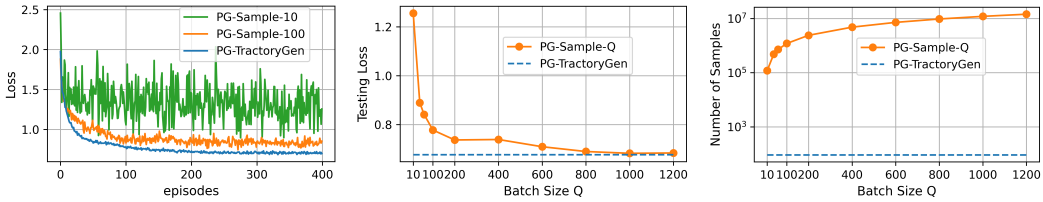
5. Experiment

We end the paper with case studies for the control of the batch reactor system in (De Persis and Tesi, 2019) and the power distribution system in (Cui et al., 2022c). Both state-feedback and output-feedback control are studied in these systems. Detailed simulation setting and results are provided in the online version (Cui et al., 2022a). Code is available at https://github.com/Wenqi-Cui/Trajectory-Generation. Major simulation results are summarized below.

Experiment Setup. We use REINFORCE policy gradient algorithm (Sutton and Barto, 2018) to train a linear feedback controller, with the goal to minimize the cost of trajectories with length K. Let E be the number of episode in training and Q be the batch size of trajectories collected for each episode, respectively. Standard policy gradient algorithms needs $Q \times K \times E$ samples. We compare the performance of the REINFORCE policy gradient algorithm using the generated trajectories (labeled as PG-TrajectoryGen) and the same algorithm using trajectories sampled by interacting with the system (labeled as PG-Sample-Q for the batchsize Q). For testing, we randomly sample 800 initial states and compare the cost on trajectories starting from these states.

State-feedback control in a batch reactor system. We use the model of a batch reactor system in (De Persis and Tesi, 2019) , where $\boldsymbol{x}(t) \in \mathbb{R}^4, \boldsymbol{u}(t) \in \mathbb{R}^2$. The time horizon is $K = 30$.

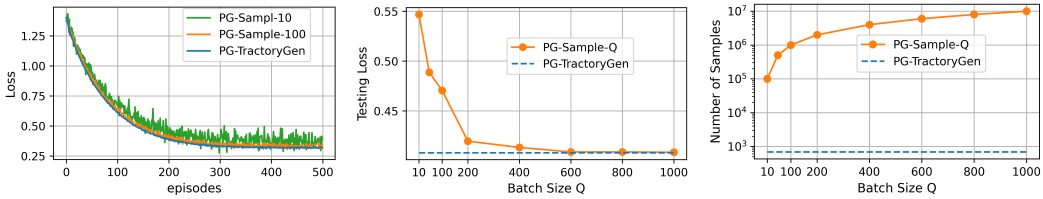
Since the state is observed, we setup $T = K = 30$ and collect historic trajectory of length $L = (m+1)T - 1 + n = 93$ such that $\text{rank}(\mathcal{H}) = n + Tm$. With a sampling period of 0.1s, the data collection takes 9.3s. We generate trajectories using Algorithm 1 with a batch size of 1200 in each episode. Figure 1(a) and Figure 1(b) shows the training and the testing loss, respectively. PG-TrajectoryGen attains the same performance as PG-Sample-1200, since Theorem 2 guarantees that we generate trajectories as if it were truly sampled on the system. Moreover, Figure 1(a)-(c) shows that the loss attained by PG-Sample-Q reduces with larger Q, but it comes at the expense of increased number of samples from the system. By contrast, the number of samples in PG-TrajectoryGen purely comes from the fixed historic trajectory of the length $L = 93$, which is much smaller than PG-Sample-10 that needs $10 \times 30 \times 400 = 120000$ online samples.



(*a*) Training Loss (*b*) Testing Loss (*c*) Number of samples

Figure 1: Performance of learning state-feedback controllers in the batch reactor system. PG-TrajectoryGen achieves the same loss as PG-Sample-1200 with much less samples.

Output-feedback control in a power distribution network. We further conduct experiments on the voltage control problem in IEEE 33bus test feeder (Baran and Wu, 1989; Cui et al., 2022c), where $\boldsymbol{x}(t) \in \mathbb{R}^{32}$. We assume only 20 buses are measured and controlled, so $\boldsymbol{y}(t), \boldsymbol{u}(t) \in \mathbb{R}^{20}$. The observability matrix $\mathcal{O}_{[0,T_0]}(\boldsymbol{A}, \boldsymbol{C})$ becomes full column rank when $T_0 = 3$. The time horizon of trajectory is $K = 20$. According to the trajectory generation algorithm developed in Subsection 4.2, we setup $T = K + T_0 - 1 = 22$ and collect historic trajectory of length $L = (m+1)T - 1 + n = 493$. With a sampling period of 1s, data collection takes 493s. Figure 2(a)-(c) compares the training loss, testing loss and the number of samples, respectively. PG-TrajectoryGen achieves the same training and testing loss as PG-Sample-1000 with much smaller number of samples.



(*a*) Training Loss (*b*) Testing Loss (*c*) Number of samples

Figure 2: Performance of learning output-feedback controllers in power distribution network. PG-TrajectoryGen achieves the same loss as PG-Sample-1000 with much less samples.

6. Conclusions

This paper proposes a trajectory generation algorithm for learning linear feedback controllers in linear systems. We prove that the algorithm generates trajectories with the exact distribution as if they are sampled by interacting with the real system using the updated control policy. In particular, the algorithm extends to systems where the states are not directly observed. This is done by equivalently defining system transition dynamics using input-output trajectories. Experiments show that the proposed method significantly reduces the number of sampled data needed for RL algorithms.

Acknowledgments

The authors W. Cui and B. Zhang are partly supported by NSF ECCS-2153937 and the State of Washington through the University of Washington Clean Energy Institute.

References

Mesut E Baran and Felix F Wu. Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Power Engineering Review*, 9(4):101–102, 1989.

Wenqi Cui, Linbin Huang, Weiwei Yang, and Baosen Zhang. Efficient reinforcement learning through trajectory generation. *arXiv preprint arXiv:2211.17249*, 2022a.

Wenqi Cui, Yan Jiang, and Baosen Zhang. Reinforcement learning for optimal primary frequency control: A Lyapunov approach. *IEEE Transactions on Power Systems*, 2022b.

Wenqi Cui, Jiayi Li, and Baosen Zhang. Decentralized safe reinforcement learning for inverter-based voltage control. *Electric Power Systems Research*, 211:108609, 2022c.

Claudio De Persis and Pietro Tesi. Formulas for data-driven control: Stabilization, optimality, and robustness. *IEEE Transactions on Automatic Control*, 65(3):909–924, 2019.

Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *International Conference on Machine Learning*, pages 1467–1476. PMLR, 2018.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, pages 3682–3691. PMLR, 2021.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Thomas Paine, Sergio Gómez, Konrad Zolna, Rishabh Agarwal, Josh S Merel, Daniel J Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7248–7259, 2020.

Joao P Hespanha. *Linear systems theory*. Princeton university press, 2009.

Bin Hu, Kaiqing Zhang, Na Li, Mehran Mesbahi, Maryam Fazel, and Tamer Başar. Towards a theoretical foundation of policy optimization for learning control policies. *arXiv preprint arXiv:2210.04810*, 2022.

Linbin Huang, Jianzhe Zhen, John Lygeros, and Florian Dörfler. Robust data-enabled predictive control: Tractable formulations and performance guarantees. *arXiv preprint arXiv:2105.07199*, 2021.

Chi Jin, Sham Kakade, Akshay Krishnamurthy, and Qinghua Liu. Sample-efficient reinforcement learning of undercomplete pomdps. *Advances in Neural Information Processing Systems*, 33: 18530–18539, 2020.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pages 5084–5096. PMLR, 2021.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Ivan Markovsky and Florian Dörfler. Identifiability in the behavioral setting. *IEEE Transactions on Automatic Control*, 2022.

Georg Ostrovski, Pablo Samuel Castro, and Will Dabney. The difficulty of passive learning in deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:23283–23295, 2021.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2nd edition, 2018.

Yujie Tang, Yang Zheng, and Na Li. Analysis of the optimization landscape of linear quadratic gaussian (lqg) control. In *Learning for Dynamics and Control*, pages 599–610. PMLR, 2021.

Jan C Willems, Paolo Rapisarda, Ivan Markovsky, and Bart LM De Moor. A note on persistency of excitation. *Systems & Control Letters*, 54(4):325–329, 2005.

Yang Zheng, Luca Furieri, Maryam Kamgarpour, and Na Li. Sample complexity of linear quadratic gaussian (lqg) control for output feedback systems. In *Learning for dynamics and control*, pages 559–570. PMLR, 2021.