On Applications of Bootstrap in Continuous Space Reinforcement Learning

Mohamad Kazem Shirani Faradonbeh, Ambuj Tewari, and George Michailidis

***Abstract*—In decision making problems for continuous state and action spaces, linear dynamical models are widely employed. Specifically, policies for stochastic linear systems subject to quadratic cost functions capture a large number of applications in reinforcement learning. Selected randomized policies have been studied in the literature recently that address the trade-off between identification and control. However, little is known about policies based on bootstrapping observed states and control inputs. In this work, we show that bootstrap-based policies achieve a square root scaling of regret with respect to time. We also obtain results on the accuracy of learning the model's dynamics. Corroborative numerical analysis that illustrates the technical results is also provided.**

***Index Terms*—Residual Bootstrap; Randomized Policies; Regret Analysis; Continuous State-Space; Identification for Control; Sequential Decision-making under Uncertainty.**

I. Introduction

In the theory of reinforcement learning, efficient algorithms with provable theoretical guarantees are established for two canonical settings. The first is *finite state* Markov decision processes (MDPs) with state spaces of small cardinalities [1]. The second is the continuous space setting of *linear quadratic (LQ)* systems [2]. In the latter one, the control action and the state both are multidimensional real vectors, and the state evolves according to stochastic linear dynamics determined by the control action. Further, the cost (or negative reward) has a quadratic form in both the state and the control input. Besides being theoretically amenable, LQ models capture a wide range of applications from air conditioning control [3] to portfolio optimization [4]. LQ models also arise when studying the behavior of nonlinear systems around the working equilibrium [5], [6].

In applications where the true system model is not known, data-driven strategies are required for decision making under uncertainty [7]. Then, the learning algorithm has to select actions among infinitely many options in order to steer the system toward minimizing the costs incurred. Note that, unlike the finite state MDP case, in LQ systems there is a possible danger of the state vector becoming unbounded [8], [9], [10]. Therefore, the design and analysis of reinforcement learning algorithms for LQ systems involve significantly different conceptual and technical issues to balance exploration (identification) and exploitation (control). For this purpose, one might consider to use upper-confidence bound (UCB) approaches [11], [12], [13], [14] that rely on the optimism in the face of uncertainty (OFU) principle. The UCB approach was historically first developed for finite action bandit problems [15]. While being efficient in the finite action setting, UCB-based approaches have been found to be computationally intractable in more general problems [16].

Recently, various methods for reinforcement learning have been proposed that leverage *randomization* strategies to guide the learning process. Randomized policy search methods have been studied both empirically, as well as theoretically (see e.g. [17], [18]). For the problem of stabilizing an unknown LQ system, an algorithm leveraging random feedback gains is proposed [19]. There is also work showing the efficiency of achieving the exploration-exploitation trade-off by randomizing the learned model through both posterior sampling [20] and additive randomization [21]. Finally, finite time analysis of Certainty Equivalent policies utilizing input perturbation has led to performance guarantees for both learning [22] and planning [16].

In this paper, we study randomized algorithms that leverage the *statistical bootstrap* [23] for reinforcement learning in LQ systems. Bootstrap-based exploration has been analyzed in simpler settings, such as bandit problems [24], [25]. There has been a lot of interest recently in using bootstrap-based exploration strategies especially along with deep neural networks [26], [27], [28], [29]. However, results on bootstrap-based reinforcement learning algorithms for LQ models have been limited to primarily numerical analyses for learning the model-misspecification error [30], while *rigorous performance guarantees* are not currently available.

Further, bootstrap methods are also of practical interest because of their *robustness* to misspecified models. The amount of exploration in bootstrap-based adaptive control policies is endogenously determined by the history of the system to date. Therefore, the policy adapts its decision-making strategy with possible systematic and/or latent "biases" occurring due to lack of accurate information regarding the system's dynamics[1]. Examples of such biases include structural breaks [31], system resets [30], and misspecification of the model dimension [32], [33], [34].

The focus of this work is on the performance of reinforcement learning policies that use the *residual bootstrap* to balance exploration and exploitation. We show that model-based strategies that use linear regression for learning the model and bootstrapping for policy design, provide a regret

M.K. Shirani Faradonbeh and G. Michailidis are with the Department of Statistics and the Informatics Institute, University of Florida, Gainesville, FL, 32611-5585 USA (e-mail: mfaradonbeh@ufl.edu, gmichail@ufl.edu)

Ambuj Tewari is with the Department of Statistics and the Department of Electrical Engineering and Computer Science (by courtesy), University of Michigan, Ann Arbor, MI 48109-1107 USA (e-mail: tewaria@umich.edu)

[1]See the discussion at the end of Section IV for more details.

that scales as the square root of the total time of interacting with the system. Further, the accuracy of learning the unknown dynamics parameter will be specified. To establish the results, we carefully examine the effect of different converging and diverging quantities involved in the problem, such as the errors in learning the model, the distributions induced by the regression residuals, the correlation within and between the observed input-state signals, and the ongoing learning-planning interactions, that are of independent interest. At the technical level, we leverage results in the literature on the bootstrap [35], as well as those on the martingale convergence [36] and central limit theorems [37], [38].

The remainder of the paper is organized as follows: Section II introduces the mathematical model under consideration, discusses the rigorous formulation of the problem, and also provides some necessary preliminaries. Section III describes the bootstrap procedure and the resulting reinforcement learning algorithm to design the policy. Subsequently, the main result on the performance of the proposed algorithm is presented, together with numerical work showcasing the performance of the algorithm, in Section IV.

a) Notation: The following notation will be employed throughout the paper. A' is the transpose of matrix or vector A. The largest and the smallest eigenvalues of square Hermitian matrix A are denoted by $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$, respectively. If A is not Hermitian, the ordering of the eigenvalues is determined by their magnitudes. The norm of the d dimensional vector v is denoted by $\|v\| = \left(\sum_{i=1}^{d} |v_i|^2\right)^{1/2}$, and $|||\cdot|||$ is used for the operator norm of matrices: $|||A||| = \sup_{\|v\|=1} \|Av\|$. For atomic measures on Euclidean spaces we use Dirac function; that is, $\boldsymbol{\delta}[v]$ denotes a unit point mass at $v \in \mathbb{R}^d$. Finally, the letters $\boldsymbol{\pi}$ and $\boldsymbol{\theta}$ (or $\widetilde{\boldsymbol{\theta}}, \widehat{\boldsymbol{\theta}}$) are being used for generic reinforcement learning policies and model parameters, respectively, and will be rigorously defined later on.

II. Setting and Problem Formulation

The model, denoted by $\mathcal{M}$, consists of multidimensional state and control vectors, parameters that specify its dynamical evolution over time, and cost matrices, as defined next. The p dimensional state process $\{x(t)\}_{t=0}^{\infty}$ evolves according to an *unknown* stochastic linear dynamic equation governed by the r dimensional control action $u(t)$, and the random disturbance (or noise) process $\{\xi(t)\}_{t=1}^{\infty}$:

$$x(t+1) \quad = \quad A_0 x(t) + B_0 u(t) + \xi(t+1). \qquad (1)$$

That is, the current state $x(t)$ and the input $u(t)$ determine the next state $x(t+1)$ through the state transition matrix $A_0 \in \mathbb{R}^{p\times p}$, and the input influence matrix $B_0 \in \mathbb{R}^{p\times r}$, respectively.

Definition 1. *Henceforth, we will denote the true parameter tuple $[A_0, B_0] \in \mathbb{R}^{p\times(p+r)}$ by the $p \times q$ dynamics matrix $\boldsymbol{\theta}_0$, with $q \equiv p + r$. Similarly, we use the parameter $\boldsymbol{\theta} \in \mathbb{R}^{p\times q}$ to denote generic dynamics matrices.*

The additive noise in the stochastic dynamics (1) satisfies $\mathbb{E}[\xi(t)] = 0$. For the sake of simplicity, we assume that the sequence of noise vectors are independent, and have a stationary covariance structure: $\mathbb{E}\left[\xi(t)\,\xi(t)'\right] = \Sigma$. Further, Σ is assumed to be positive definite, and $\sup_{t\geq 1} \mathbb{E}\left[\|\xi(t)\|^{\alpha}\right] < \infty$, for some fixed $\alpha > 4$. As a matter of fact, extensions to more general technical settings such as non-stationary [16] or singular covariance matrices (assuming reachability [9]), as well as conditionally independent processes [13], can be accommodated in a similar manner. Note though that the assumed noise process is not necessarily stationary in the strict sense.

We are interested in finding reinforcement learning policies to minimize the long-term average cost as formally defined next. First, suppose that Q_x and Q_u are the regulation weight matrices reflecting the effect of the state and the input vectors in the cost function, respectively. Specifically, letting $\boldsymbol{\pi}$ be the decision making law (policy) determining the control input $u(t)$ at every time t, define the quadratic instantaneous cost of $\boldsymbol{\pi}$ according to

$$c_t(\boldsymbol{\pi}) \quad = \quad \left\|Q_x^{1/2} x(t)\right\|^2 + \left\|Q_u^{1/2} u(t)\right\|^2, \qquad (2)$$

where $Q_x \in \mathbb{R}^{p\times p}$, and $Q_u \in \mathbb{R}^{r\times r}$ are symmetric positive definite matrices. Thus, (2) reflects the desire to regulate the state of the system through control actions of small magnitude.

When the dynamics follow (1), and the instantaneous cost is given by (2), we denote the model by $\mathcal{M}(\boldsymbol{\theta}_0) = (\boldsymbol{\theta}_0, Q_x, Q_u)$. Further, the history of the system at time t, denoted as $\mathbb{H}_t$, consists of the sequence of the control inputs applied so far, and the resulting state vectors:

$$\mathbb{H}_t = (x(0), \cdots, x(t), u(0), \cdots, u(t-1)).$$

A reinforcement learning policy observes the history $\mathbb{H}_t$ at time t aiming to control the cost incurred. That is, the policy $\boldsymbol{\pi}$ is a (possibly random) mapping which designs the input sequence $\{u(t)\}_{t=0}^{\infty}$ according to the history available up to that time;

$$u(t) = \boldsymbol{\pi}(\mathbb{H}_t, Q_x, Q_u), \qquad (3)$$

so that the average cost is minimized. Thus, the objective is summarized in the following regulation problem:

Problem 1. *Find $\boldsymbol{\pi}$ to minimize the average cost below, subject to* (1)*,* (2)*, and* (3)*;*

$$\limsup_{n\to\infty} \frac{1}{n} \sum_{t=0}^{n-1} c_t(\boldsymbol{\pi}). \qquad (4)$$

Importantly, according to (3) the true dynamics parameter $\boldsymbol{\theta}_0$ in (1) is unknown. Therefore, the policy must also employ an exploration procedure to accurately learn the model parameters, thus addressing the following identification problem:

Problem 2. *Using* (1) *and* (3)*, design $\boldsymbol{\pi}$ to learn $\boldsymbol{\theta}_0$, as accurately as possible.*

Note that in the above formulation, the true system dynamics are unknown, while the cost matrices are known. It gives rise to a realistic setting, since the decision making algorithm does not know the actual evolution of the underlying system (i.e., $\boldsymbol{\theta}_0$), but is aware of the criteria according to which a policy to achieve the goal is being assessed (i.e., Q_x, Q_u).

Subsequently, we define the *regret* of a policy, which is the amount of sub-optimality it incurs due to lack of knowledge of the parameters of the model (1). To do so, we need to introduce an optimal policy $\boldsymbol{\pi}^{\star}$ that minimizes the average cost, given full knowledge of the system model $\mathcal{M}(\boldsymbol{\theta}_0)$. Then, $\boldsymbol{\pi}^{\star}$ will be the baseline for assessing the exploitation performance of the arbitrary reinforcement learning policy $\boldsymbol{\pi}$. It is well known that in order to find $\boldsymbol{\pi}^{\star}$, an algebraic Riccati equation needs to be solved [39], [40].

To proceed, we introduce some additional notation. First, for an arbitrary $\boldsymbol{\theta} = [A, B]$ define the matrix valued mapping

$$\Phi_{\boldsymbol{\theta}}(P) = Q_x + A'PA - A'PB\left(B'PB + Q_u\right)^{-1} B'PA.$$

Both the domain and the range of $\Phi_{\boldsymbol{\theta}}(\cdot)$ are the set of $p \times p$ matrices. Next, if there is a positive semidefinite matrix $P(\boldsymbol{\theta})$ satisfying the algebraic Riccati equation $P(\boldsymbol{\theta}) = \Phi_{\boldsymbol{\theta}}(P(\boldsymbol{\theta}))$, let the feedback gain matrix $G(\boldsymbol{\theta})$ be

$$G(\boldsymbol{\theta}) = -\left(B'P(\boldsymbol{\theta})B + Q_u\right)^{-1} B'P(\boldsymbol{\theta})A. \tag{5}$$

Furthermore, for $\boldsymbol{\theta}_0 = [A_0, B_0]$ in (1), define the linear time-invariant (LTI) policy

$$\boldsymbol{\pi}^{\star}: \quad u(t) = G(\boldsymbol{\theta}_0)\, x(t), \quad t = 0, 1, 2, \cdots. \tag{6}$$

Finally, using $\boldsymbol{\pi}^{\star}$, the regret of $\boldsymbol{\pi}$ is naturally defined by:

$$\mathcal{R}_n(\boldsymbol{\pi}) = \sum_{t=0}^{n-1} \left[c_t(\boldsymbol{\pi}) - c_t(\boldsymbol{\pi}^{\star})\right]. \tag{7}$$

It remains to specify settings for which $\boldsymbol{\pi}^{\star}$ is well-defined. To that end, the following closed-loop stabilizability condition for the model (1) is necessary and sufficient [39], [40].

Assumption 1. *There is a LTI feedback gain $u(t) = G_s x(t)$, such that $G_s \in \mathbb{R}^{r\times p}$ satisfies $|\boldsymbol{\lambda}_{\max}(A_0 + B_0 G_s)| < 1$.*

Note that in general, the stabilizing gain G_s mentioned above is only required to exist, and does not need to be known to the decision maker. In other words, to verify that the stabilizability Assumption 1 holds, it suffices to show that a hypothetical omniscient decision maker (who knows the true model $\mathcal{M}(\boldsymbol{\theta}_0)$) possessing an omnipotent computational power is able to stabilize the system. However, we briefly outline the available constructive methods to compute G_s (as well as $\boldsymbol{\pi}^{\star}$). It is shown (see for example [39], [40], [19]) that under Assumption 1 the following statements hold;

1) The positive definite matrix $P(\boldsymbol{\theta}_0)$ uniquely exists. So, both the feedback $G(\boldsymbol{\theta}_0)$ and the optimal policy $\boldsymbol{\pi}^{\star}$ are well defined.
2) Letting P_0 be an arbitrary positive semidefinite $p\times p$ matrix, the recursive formula $P_{k+1} = \Phi_{\boldsymbol{\theta}_0}(P_k)$ converges exponentially fast to $P(\boldsymbol{\theta}_0)$ as k grows.
3) The feedback matrix $G(\boldsymbol{\theta}_0)$ stabilizes the system:
$$|\boldsymbol{\lambda}_{\max}(A_0 + B_0 G(\boldsymbol{\theta}_0))| < 1.$$
4) The minimum of the average cost (4) is achieved by $\boldsymbol{\pi}^{\star}$.
5) In the class of LTI policies (i.e. of the form $u(t) = Gx(t)$), the policy $\boldsymbol{\pi}^{\star}$ is the only optimal one.

In the remainder of the paper, we employ reinforcement learning algorithms to address Problem 1, studying the growth rates of $\mathcal{R}_n(\boldsymbol{\pi})$. Similarly, letting $\widetilde{\boldsymbol{\theta}}_n$ be the learned/estimated parameter at time n (the sample size is n as well), we consider the exploration performance in Problem 2 through the rates of the learning error $\left|\!\left|\!\left|\widetilde{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0\right|\!\right|\!\right|$. Bootstrap is the cornerstone of the proposed algorithms to efficiently randomize the design of the control inputs, and address the trade-off between the learning accuracy and the regret.

III. Algorithms

An algorithm needs to address the common dilemma of decision making under uncertainty, as follows. First, if the algorithm makes decisions naively according to the estimated (learned) dynamics parameter, it can incur high regret. Intuitively, the state $x(t)$ and the action $u(t)$ are required to be highly correlated in order to remain close to the optimal strategy $\boldsymbol{\pi}^{\star}$ in (6). Because of this correlation, history $\mathbb{H}_t$ may fail to accurately learn $\boldsymbol{\theta}_0$, which can lead to drastically large regret values. Technically, if $u(t) = Gx(t)$ for some $r \times p$ feedback gain matrix G, then the dimension of the observed history is effectively p, while the rows of the matrices $\boldsymbol{\theta}$ in the parameter space belong to $\mathbb{R}^q$. Therefore, learning can be dramatically misleading. This phenomenon of *failing to falsify* the imprecise approximations of the true model is extensively discussed in the adaptive control literature [41], [42], [12], [21].

In other words, if the policy fails to sufficiently explore the parameter space, an *inaccurate* approximation $\widetilde{\boldsymbol{\theta}}_t$ can falsely be treated as an accurate one. This necessitates an efficient exploration strategy to decrease the aforementioned correlation between the state and the action. Moreover, the above argument reveals the reasoning leading to UCB approaches [13], [14], or statistically independent dither schemes [16], [22], as useful prescriptions to overcome the exploration-exploitation dilemma.

In order to explore, the decision maker needs to deviate from the learned model $\widetilde{\boldsymbol{\theta}}_t$ prior to using $\mathcal{M}\left(\widetilde{\boldsymbol{\theta}}_t\right)$ to design the reinforcement learning policy. On the other hand though, the above deviations must be sufficiently small in order to avoid significant deterioration in the exploitation performance (i.e., increase in the regret). The solution we discuss here is to utilize the bootstrap to provide the necessary balance between these two competing objectives.

To this end, the policy $\boldsymbol{\pi}$ applies the supposedly optimal control action treating $\mathcal{M}\left(\widehat{\boldsymbol{\theta}}_t\right)$ as the true model, where $\widehat{\boldsymbol{\theta}}_t$ is provided by the bootstrap algorithm. It computes the regression residuals for the learned parameter $\widetilde{\boldsymbol{\theta}}_t$, and bootstraps (i.e., resamples) them to reconstruct a *surrogate* system. Then, the history of the surrogate system will be the data being used to compute $\widehat{\boldsymbol{\theta}}_t$. In the first subsection, we explain the least squares estimator for learning the model parameter, as well as the above residual bootstrap procedure.

Subsequently, in the second subsection we present an episodic algorithm which updates the model-based policy at the end of every episode, while the lengths of the episodes grow exponentially fast. Therefore, as the duration of the interaction with the system grows, the number of policy

updates scales logarithmically. In fact, this leads to a significant reduction in the computation of the policy, by avoiding unnecessary updates before collecting sufficient data, due to the fact that the solution of the algebraic Riccati equation (5) for a hypothetical model is not instantly available. The latter would impose a substantial computational burden, especially for systems whose dimension is fairly large. Note that in this case, additional considerations might be needed since the memory required by the bootstrap procedure scales with the dimension q.

A. Residual Bootstrap

According to the linear dynamical model in (1), a natural procedure to learn $\boldsymbol{\theta}_0$ through the control input $u(t)$ and the observed states $x(t), x(t+1)$ is based on least squares. In the sequel, we discuss the residual bootstrap method for the least squares learning procedure. Further, we will present the corresponding algorithm which will be used as a subroutine in the reinforcement learning algorithm in the next subsection.

Recall that the LTI policy $\boldsymbol{\pi}^{\star}$ in (6) is optimal. Thus, a natural form of the adaptive policies that a reinforcement learning algorithm is expected to provide through planing according to the learned model, is $u(t) = G_t x(t)$. Assuming so for $t < n$, now the algorithm needs to decide about the control action at time n. Thus, plugging $u(t) = G_t x(t)$ in the dynamical model (1), and denoting

$$F_t = [I_p, G_t']' \in \mathbb{R}^{q \times p},$$

we get the so-called closed-loop evolution of the system by the (possibly time-varying) autoregressive dynamics

$$x(t+1) = \boldsymbol{\theta}_0 F_t x(t) + \xi(t+1),$$

for $0 \leq t < n$. Then, Algorithm 1 returns the bootstrapped parameter $\widehat{\boldsymbol{\theta}}_n$ based on the matrices $\{F_t\}_{t=0}^{n-1}$, as well as the available state observations $\{x(t)\}_{t=0}^{n}$. The details are provided below.

First, based on the collected history $\{F_t\}_{t=0}^{n-1}, \{x(t)\}_{t=0}^{n}$, define the following least square estimate of $\boldsymbol{\theta}_0$:

$$\widetilde{\boldsymbol{\theta}}_n = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^{p \times q}} \sum_{t=0}^{n-1} \|x(t+1) - \boldsymbol{\theta} F_t x(t)\|^2. \tag{8}$$

The learning procedure (8) treats the noise vectors $\xi(t)$ as the errors of a linear regression procedure, based on the dynamical model (1). Therefore, the residuals of the least squares estimate are defined by the difference between the observed response $x(t+1)$, and the fitted response $\widetilde{\boldsymbol{\theta}}_n F_t x(t)$. That is,

$$\zeta(t+1) = x(t+1) - \widetilde{\boldsymbol{\theta}}_n F_t x(t), \tag{9}$$

for $0 \leq t < n$. The residuals $\{\zeta(t)\}_{t=1}^{n}$ can conceptually be considered as approximations of the actual regression errors $\{\xi(t)\}_{t=1}^{n}$. Using the residuals $\{\zeta(t)\}_{t=1}^{n}$, we define the centered empirical distribution

$$\widehat{\mathbb{P}}_n = \frac{1}{n} \sum_{t=1}^{n} \boldsymbol{\delta}\big[\zeta(t) - \overline{\zeta}_n\big], \tag{10}$$

where $\overline{\zeta}_n$, the average of the residuals given by

$$\overline{\zeta}_n = \frac{1}{n} \sum_{t=1}^{n} \zeta(t), \tag{11}$$

is being used for centering the empirical distribution. In fact, $\widehat{\mathbb{P}}_n$ is the sample probability measure for the population distribution of the noise process $\{\xi(t)\}_{t=1}^{\infty}$. Note that $\widehat{\mathbb{P}}_n$ is defined on $\mathbb{R}^p$. We then use $\widetilde{\boldsymbol{\theta}}_n$ and $\widehat{\mathbb{P}}_n$ to generate the surrogate state vectors $\{\widehat{x}(t)\}_{t=0}^{n}$ by the dynamical model

$$\widehat{x}(t+1) = \widetilde{\boldsymbol{\theta}}_n F_t \widehat{x}(t) + \widehat{\xi}(t+1),$$

where the bootstrap noise vectors $\widehat{\xi}(t+1)$ are drawn independently from $\widehat{\mathbb{P}}_n$. Hence, letting $\widehat{\mathbb{E}}_n$ be the expectation with respect to $\widehat{\mathbb{P}}_n$, clearly we have $\widehat{\mathbb{E}}_n\left[\widehat{\xi}(t)\right] = 0$. Also note that the actual dynamics parameter for the surrogate system $\{\widehat{x}(t)\}_{t=0}^{n}$ is the learned parameter $\widetilde{\boldsymbol{\theta}}_n$ defined in (8). Finally, the algorithm applies the least squares estimator to the generated surrogate states to obtain $\widehat{\boldsymbol{\theta}}_n$:

$$\widehat{\boldsymbol{\theta}}_n = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^{p \times q}} \sum_{t=0}^{n-1} \|\widehat{x}(t+1) - \boldsymbol{\theta} F_t \widehat{x}(t)\|^2. \tag{12}$$

The pseudo-code for the above residual bootstrap procedure is given in Algorithm 1. It will be used later at the heart of Algorithm 2 to design reinforcement learning policies.

Algorithm 1 : **BOOTSTRAP**

Inputs: data $\{x(t)\}_{t=0}^{n}, \{F_t\}_{t=0}^{n-1}$
Output: bootstrapped estimate $\widehat{\boldsymbol{\theta}}_n$
Define $\widetilde{\boldsymbol{\theta}}_n, \{\zeta(t)\}_{t=1}^{n}, \overline{\zeta}_n$, and $\widehat{\mathbb{P}}_n$ according to (8), (9), (11), and (10), respectively
Let $\widehat{x}(0) = x(0)$
for $t = 0, 1, 2, \cdots, n-1$ **do**
  Draw $\widehat{\xi}(t+1)$ from $\widehat{\mathbb{P}}_n$, independently
  Let $\widehat{x}(t+1) = \widetilde{\boldsymbol{\theta}}_n F_t \widehat{x}(t) + \widehat{\xi}(t+1)$
end for
Return $\widehat{\boldsymbol{\theta}}_n$ given by (12)

Remark 1. *If the noise process is parameterized, one can accordingly draw $\widehat{\xi}(t)$ from the corresponding parametric sample distribution.*

To see that, assume we know that the noise vectors belong to a parametric family of stochastic processes. Then, instead of using the nonparametric empirical distribution in (10), one can use the the residuals $\zeta(t)$ to estimate the parameters of interest. So, letting $\widehat{\mathbb{P}}_n$ be the parametric distribution provided by the obtained estimate, the bootstrap noise $\widehat{\xi}(t)$ can be sampled independently from $\widehat{\mathbb{P}}_n$. For example, if we know that $\xi(t)$ are i.i.d. Gaussian vectors, we can find the sample covariance matrix $\widehat{\Sigma}_n = n^{-1} \sum_{t=1}^{n} \zeta(t) \zeta(t)' - \overline{\zeta}_n \overline{\zeta}_n'$, and draw $\widehat{\xi}(t)$ independently from the centered Gaussian distribution with covariance matrix $\widehat{\Sigma}_n$.

Remark 2. *In the original version of bootstrap [23], the covariates (i.e., the state vectors) are fixed, and only the*

residuals are being bootstrapped. In the time series models such as (1)*, every state vector comprises of the previous noise vectors. Therefore, bootstrapping the residuals automatically leads to new state sequence* $\{\widehat{x}(t)\}_{t=0}^{\infty}$ *for the surrogate system [30].*

B. Policy Design

Next, Algorithm 2 for decision making under uncertainty based on bootstrapping the residuals (Algorithm 1) is discussed. For this purpose, we first define the extended gain matrix $F(\boldsymbol{\theta})$ based on the optimal feedback $G(\boldsymbol{\theta})$.

Definition 2. *For parameter* $\boldsymbol{\theta} \in \mathbb{R}^{p\times q}$*, using the matrix* $G(\boldsymbol{\theta})$ *in* (5)*, define the* $q \times p$ *matrix* $F(\boldsymbol{\theta}) = \left[I_p, G(\boldsymbol{\theta})'\right]'$.

The matrix $F(\boldsymbol{\theta})$ can be interpreted as an extension of the original feedback gain; applying $u(t) = G(\boldsymbol{\theta})x(t)$, the closed-loop transition matrix takes the form $A_0 + B_0 G(\boldsymbol{\theta}) = \boldsymbol{\theta}_0 F(\boldsymbol{\theta})$.

Recall that the true model is not known, and a reinforcement learning policy needs to simultaneously learn the dynamics parameter, and design the control input. To do so, we present an episodic decision making strategy outlined in Algorithm 2. That is, the policy applies control actions during each episode, assuming that the approximation of the model available at the time *coincides* with the true model. Then, at the end of every episode, the algorithm updates the learned model based on the history collected so far, and continues making decisions as if the new approximate model is the truth. The learning mentioned above is through a linear regression for the dynamics (1), and the approximation consists of bootstrapping (by Algorithm 1) the model estimate obtained by the regression. In the sequel, we explain the details of the above alternating steps of the algorithm.

Algorithm 2 : POLICY DESIGN

Let $\mathbb{H}_0 = \{x(0)\}$
Choose stabilizable $\widehat{\boldsymbol{\theta}}_0$ arbitrarily
for $m = 1, 2, \cdots$ **do**
  while $t < \beta^m$ **do**
    Apply feedback gain $u(t) = G\left(\widehat{\boldsymbol{\theta}}_t\right)x(t)$
    Update history $\mathbb{H}_{t+1} = \mathbb{H}_t \cup \left\{x(t+1), F\left(\widehat{\boldsymbol{\theta}}_t\right)\right\}$
    $\widehat{\boldsymbol{\theta}}_{t+1} = \widehat{\boldsymbol{\theta}}_t$
  end while
  Update parameter $\widehat{\boldsymbol{\theta}}_{t+1} = \textbf{BOOTSTRAP}(\mathbb{H}_{t+1})$
end for

The reinforcement learning policy is initiated with the history $\mathbb{H}_0$ in the first line of Algorithm 2. Then, it chooses an arbitrary stabilizable approximation of $\boldsymbol{\theta}_0$, denoted by $\widehat{\boldsymbol{\theta}}_0$, and starts the system by applying the action prescribed by the model $\mathcal{M}\left(\widehat{\boldsymbol{\theta}}_0\right)$; that is, $u(t) = G\left(\widehat{\boldsymbol{\theta}}_0\right)x(t)$. Note that selection of $\widehat{\boldsymbol{\theta}}_0$ is straightforward, since almost all (w.r.t. Lebesgue measure) parameter matrices are stabilizable [20].

The starting time-points of the episodes are determined by the exponents of a fixed constant $\beta > 1$. That is, at every time $t = \lceil \beta^m \rceil$, the approximation $\widehat{\boldsymbol{\theta}}_t$ will be updated, while for $\beta^m \le t < \beta^{m+1}$ the algorithm freezes $\widehat{\boldsymbol{\theta}}_t$. In other words, whenever $t = \lceil \beta^m \rceil$, Algorithm 2 calls the residual bootstrap Algorithm 1 to get $\widehat{\boldsymbol{\theta}}_t$ according to the collected history of the control actions and the states. So, for all $\beta^m \le t < \beta^{m+1}$, the matrices $F\left(\widehat{\boldsymbol{\theta}}_t\right)$ are exactly the same. The efficiency of the policy relies on the idea that the sequence $\left\{\widehat{\boldsymbol{\theta}}_t\right\}_{t=0}^{\infty}$ will provide finer approximations of the truth $\boldsymbol{\theta}_0$, as the algorithm proceeds (or more precisely, as m grows).

IV. Theoretical Results and Simulations

We start by establishing performance guarantees on the regret and the learning accuracy for bootstrap-based policies, supplemented by numerical examples that illustrate the behavior of Algorithm 2 for both identification and regulation. The following result specifies the growth rate of the regret $\mathcal{R}_n(\boldsymbol{\pi})$ for the policy $\boldsymbol{\pi}$ designed by Algorithm 2, as well as the decay rate of the identification error $\left\|\widetilde{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0\right\|$.

Theorem 1. *Letting* $\boldsymbol{\pi}$ *be the policy given by Algorithm 2, define the learned parameter* $\widetilde{\boldsymbol{\theta}}_n$ *by* (8)*. Then, we have*

$$\begin{aligned}\limsup_{n\to\infty}\left(n^{-1/2}\log^{-2}n\right)\mathcal{R}_n(\boldsymbol{\pi}) &< \infty,\\ \limsup_{n\to\infty}\left(n^{1/2}\log^{-2}n\right)\left\|\widetilde{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0\right\|^2 &< \infty.\end{aligned}$$

Due to space limitations, the proof of Theorem 1 is delegated to the longer version of the paper, which is available online [43]. Technically, it relies on the careful examination of the effect of Algorithm 1 on the randomization of the feedback gains $G\left(\widehat{\boldsymbol{\theta}}_{\lceil\beta^i\rceil}\right)$. This randomization in turn diversifies the extended gain matrices $\left\{F\left(\widehat{\boldsymbol{\theta}}_{\lceil\beta^i\rceil}\right)\right\}_{i=1}^{m}$, so that their superposition efficiently explores the whole parameter space $\mathbb{R}^{p\times q}$, as m grows. To this end, we utilize the state-of-the-art results on the behavior of the algebraic Riccati equation [14], properties of the optimality manifold [44], [45], [21], results from martingale theory [46], [47], [48], limit distributions of dependent sequences [37], [38], and the bootstrap [35].

The regulation and identification rates of Theorem 1 are (modulo logarithmic factors) similar to the corresponding rates of the reinforcement learning policies utilizing OFU [13], [14], additive randomization [21], posterior sampling [20], and input perturbation [16]. Moreover, the square root scaling of the regret is efficient for adaptive regulation of LQ systems as discussed next.

Recalling the discussion at the beginning of Section III, an adaptive control policy needs to sufficiently explore the parameter space in order to balance the trade-off of identification and regulation. For falsifying the imprecise approximation $\widetilde{\boldsymbol{\theta}}_n$ through an exploration procedure, the control signals $\{u(t)\}_{t=0}^{n-1}$ need to deviate from the optimal feedback gain $G(\boldsymbol{\theta}_0)$. More precisely, for a policy $\boldsymbol{\pi}$, let the deviations from the optimal feedback be $\epsilon_t = \|u(t) - G(\boldsymbol{\theta}_0)x(t)\|$, for $0 \le t < n$. Then, observing the history $\mathbb{H}_n$, the error of estimating the true dynamics parameter $\boldsymbol{\theta}_0$ is at least σ_n (modulo a constant factor), where $\sigma_n^{-2} = \sum_{t=0}^{n-1}\epsilon_t^2$ [7], [10].

$$A_0 = \begin{bmatrix} 1.07 & 0 & -0.37 \\ 0.48 & -0.89 & 0.85 \\ 0 & 0.04 & -0.93 \end{bmatrix}, \quad B_0 = \begin{bmatrix} -0.48 & 0.44 & -0.30 \\ -0.52 & 0.59 & 0.26 \\ 0.30 & 0 & -0.74 \end{bmatrix}, \quad Q_x = \begin{bmatrix} 0.65 & -0.08 & -0.14 \\ -0.08 & 0.57 & 0.26 \\ -0.14 & 0.26 & 1.00 \end{bmatrix}, \tag{13}$$

$$Q_u = \begin{bmatrix} 0.20 & 0.05 & 0.09 \\ 0.05 & 0.14 & 0.04 \\ 0.08 & 0.04 & 0.28 \end{bmatrix}, \quad P(\boldsymbol{\theta}_0) = \begin{bmatrix} 0.94 & 0.06 & -0.32 \\ 0.06 & 0.88 & 0.02 \\ -0.32 & 0.02 & 1.37 \end{bmatrix}, \quad G(\boldsymbol{\theta}_0) = \begin{bmatrix} 0.64 & -0.13 & 0.44 \\ -0.71 & 0.63 & -0.11 \\ 0.22 & 0.08 & -0.91 \end{bmatrix}. \tag{14}$$

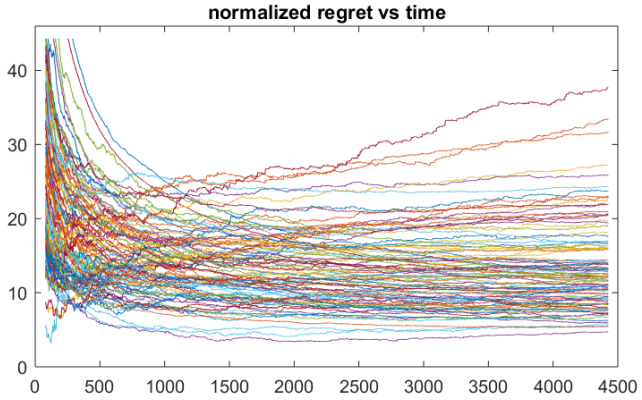

Figure 1: Normalized regret $n^{-1/2}\mathcal{R}_n(\boldsymbol{\pi})$ vs n, for Algorithm 2 with $\beta = 1.2$.

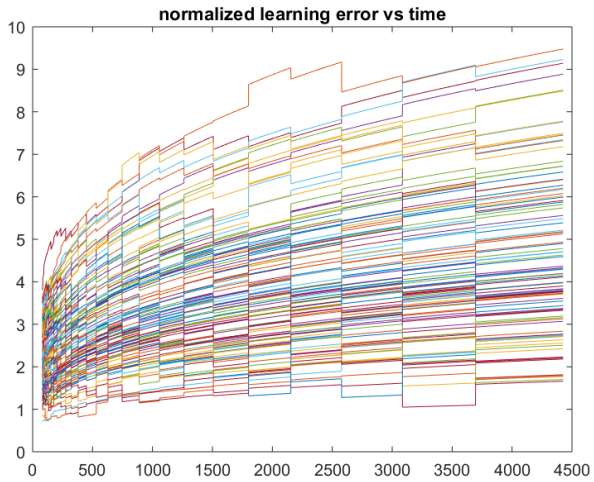

Figure 2: Normalized identification error $n^{1/4}\left\|\widetilde{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0\right\|$ vs n, for Algorithm 2 with $\beta = 1.2$.

Hence, if $\boldsymbol{\pi}$ aims to falsify $\widetilde{\boldsymbol{\theta}}_n$, the difference $\left\|\widetilde{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0\right\|$ needs to be in the order of magnitude at least σ_n [12], [21]. Whenever $\boldsymbol{\pi}$ employs $\widehat{\boldsymbol{\theta}}_n$ for designing control inputs, $\liminf_{n\to\infty} \sigma_n^{-1}\left\|\widehat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0\right\| > 0$ holds, since $\widehat{\boldsymbol{\theta}}_n$ needs to be found through $\widetilde{\boldsymbol{\theta}}_n$.

On the other hand, for the above deviations we have

$$\liminf_{n\to\infty} \sigma_n^2 \mathcal{R}_n(\boldsymbol{\pi}) > 0, \tag{15}$$

according to the regret specification recently established [21]. Further, applying the adaptive feedback $u(n) = G\left(\widehat{\boldsymbol{\theta}}_n\right) x(n)$ at time n, the increase $\mathcal{R}_{n+1}(\boldsymbol{\pi}) - \mathcal{R}_n(\boldsymbol{\pi})$ in the regret is approximately $\left\|G\left(\widehat{\boldsymbol{\theta}}_n\right) - G(\boldsymbol{\theta}_0)\right\|^2$ [16], which is up to a constant factor at least $\left\|\widehat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0\right\|^2$ [14], [21]. Thus, the lower bound σ_n for $\left\|\widehat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0\right\|$ implies that $\liminf_{n\to\infty} \sigma_n^{-2}\left(\mathcal{R}_{n+1}(\boldsymbol{\pi}) - \mathcal{R}_n(\boldsymbol{\pi})\right) > 0$. Putting the latter result and (15) together, we obtain $\liminf_{n\to\infty}\left(\mathcal{R}_{n+1}(\boldsymbol{\pi})^2 - \mathcal{R}_n(\boldsymbol{\pi})^2\right) > 0$, which provides the lower bound $\liminf_{n\to\infty} n^{-1/2}\mathcal{R}_n(\boldsymbol{\pi}) > 0$. Note that a rigorous proof of the above lower bound argument is beyond the scope of this paper. For more detailed discussions, we refer the reader to the aforementioned references.

A. Numerical Illustration

Next, we present numerical analyses employing Algorithm 2 for decision-making under uncertainty. Henceforth, let $\boldsymbol{\pi}$ be the reinforcement learning policy provided by Algorithm 2, with rate $\beta = 1.2$. The true dynamical model and cost matrices are provided in (13), (14). Solving the algebraic Riccati equation, we get $P(\boldsymbol{\theta}_0), G(\boldsymbol{\theta}_0)$ given in (14), which lead to a closed-loop matrix of the spectral radius $|\boldsymbol{\lambda}_{\max}(\boldsymbol{\theta}_0 F(\boldsymbol{\theta}_0))| = 0.26$.

Figure 1 depicts the normalized regret as a function of n, for 100 replicates of the stochastic linear system in (1). The corresponding normalized identification errors are plotted in Figure 2. These figures are in a full agreement with the theoretical result of Theorem 1; that is, both normalized rates $n^{-1/2}\mathcal{R}_n(\boldsymbol{\pi})$ and $n^{1/4}\left\|\widetilde{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0\right\|$ are dominated by logarithmic factors of the time index n. In Figure 3, we plot the resulting spectral radius of the reinforcement learning policy $\boldsymbol{\pi}$ for the actual system of the dynamics parameter $\boldsymbol{\theta}_0$, as well as that of the surrogate system of $\widetilde{\boldsymbol{\theta}}_n$. According to Figure 3, Algorithm 2 fully stabilizes the system, even though in the first few episodes the system is unstable. The ensuing figures indicate the robustness of Algorithm 2 to structural breaks. Figure 4 shows the curves of $n^{-1/2}\mathcal{R}_n(\boldsymbol{\pi})$ and $n^{1/4}\left\|\widetilde{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0\right\|$ for a single break in the model, wherein at time $t = 400$ the dynamics matrices suddenly become

$$A_0 = \begin{bmatrix} 1.07 & 0 & -0.37 \\ 0.48 & -0.89 & 0.85 \\ \mathbf{0.44} & 0.04 & \mathbf{0} \end{bmatrix}, B_0 = \begin{bmatrix} -0.48 & 0.44 & -0.30 \\ -0.52 & 0.59 & 0.26 \\ 0.30 & \mathbf{-0.44} & \mathbf{0} \end{bmatrix}.$$

A similar analysis while the system incurs two breaks is provided in Figure 5. The first break is similar to the one mentioned above, and occurs at $t = 200$. Then, for the second break at $t = 700$, the true dynamics matrices change to

$$A_0 = \begin{bmatrix} 1.07 & 0 & \mathbf{-1.04} \\ 0.48 & -0.89 & 0.85 \\ 0.44 & \mathbf{0.81} & 0 \end{bmatrix}, B_0 = \begin{bmatrix} -0.48 & 0.44 & -0.30 \\ -0.52 & 0.59 & \mathbf{-0.26} \\ 0.30 & \mathbf{-0.30} & 0 \end{bmatrix}.$$

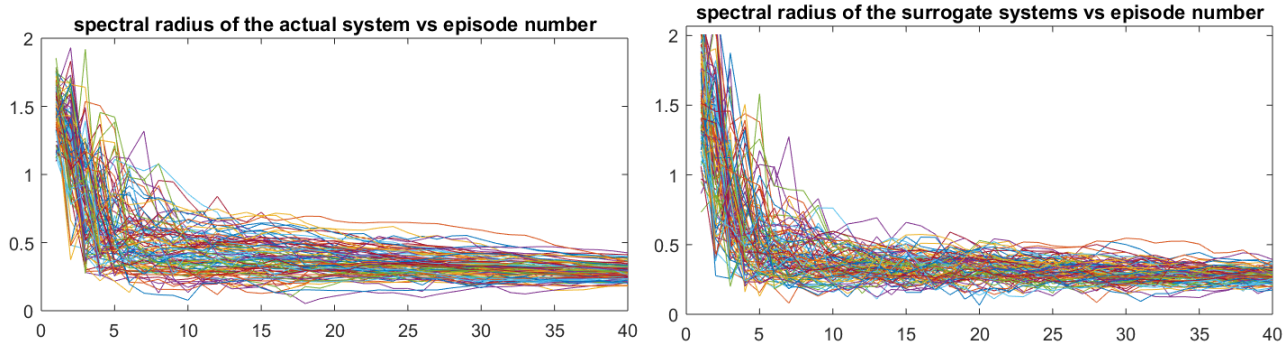

Figure 3: Stability of the closed-loop matrices for Algorithm 2 with $\beta = 1.2$: the spectral radius of the actual system $\left|\boldsymbol{\lambda}_{\max}\left(\boldsymbol{\theta}_0 F\left(\widehat{\boldsymbol{\theta}}_{\lceil\beta^m\rceil}\right)\right)\right|$, and the surrogate system $\left|\boldsymbol{\lambda}_{\max}\left(\widetilde{\boldsymbol{\theta}}_{\lceil\beta^m\rceil} F\left(\widehat{\boldsymbol{\theta}}_{\lceil\beta^m\rceil}\right)\right)\right|$, are reported as functions of m.

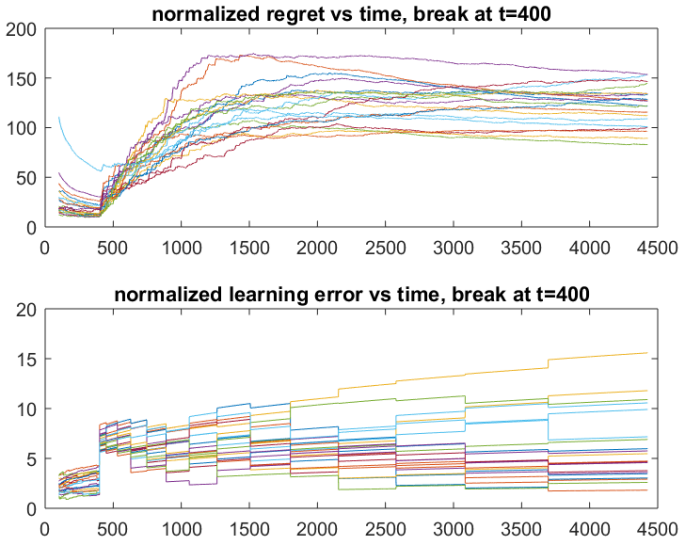

Figure 4: Normalized regret $n^{-1/2}\mathcal{R}_n(\boldsymbol{\pi})$ and normalized learning error $n^{1/4}\left\|\widetilde{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0\right\|$ vs n. Algorithm 2 is run with $\beta = 1.2$, while a break occurs at time $t = 400$.

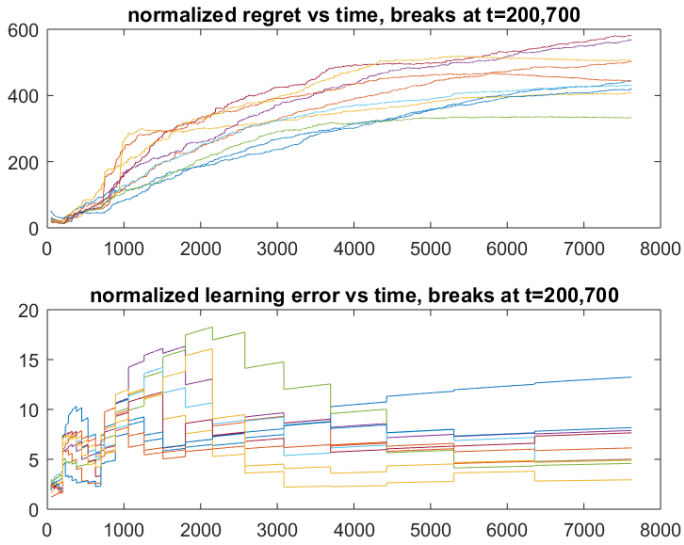

Figure 5: Normalized regret $n^{-1/2}\mathcal{R}_n(\boldsymbol{\pi})$ and normalized learning error $n^{1/4}\left\|\widetilde{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0\right\|$ vs n. Algorithm 2 is run with $\beta = 1.2$, while two breaks occur at times $t = 200$, $t = 700$.

According to figures 4 and 5, Algorithm 2 is robust to remarkably large values of model misspecifications. Note that since the reinforcement learning policy is fully ignorant of the breaks, Algorithm 2 adaptively adjusts its decision-making law toward the new optimal policies.

The rationale for the exhibited robustness is as follows: when a break occurs, the parameter estimate $\widetilde{\boldsymbol{\theta}}$ becomes an inaccurate approximation of the true dynamics matrix $\boldsymbol{\theta}_0$. This in turn leads the regression residuals $\{\zeta(t)\}_{t=1}^{n}$ becoming large. Therefore, the bootstrapped parameter $\widetilde{\boldsymbol{\theta}}$ computed by Algorithm 1 provides a large randomization, which in turn leads to an increase in the exploration phase. Then, after a few episodes, the resulting enhanced exploration provides more accurate estimates $\widetilde{\boldsymbol{\theta}}$, and the above *negative feedback* procedure proceeds. Thus, as time grows, Algorithm 2 *self-tunes* to the equilibrium of the suitable amount of exploration.

The above argument intuitively indicates that the aforementioned equilibrium is a stable one. Since the endogenous randomization of the bootstrap procedure consistently assesses the accuracy of the fitted model $\widetilde{\boldsymbol{\theta}}$, the resulting adaptive policy automatically adjusts the old decision-making strategy to the new environment. Hence, the algorithm accordingly addresses the unexpected flaw of the *sudden and unknown* changes in the true model $\mathcal{M}(\boldsymbol{\theta}_0)$, as well as the resulting unforeseen deviations in the trajectory of the state sequence $\{x(t)\}_{t=0}^{\infty}$.

V. Concluding Remarks

We proposed a reinforcement learning algorithm for sequential decision-making for an LQ system with unknown temporal dynamics. The presented model-based policy is based on residual bootstrap, and is shown to be efficient in terms of both identification and regulation. Namely, we establish the rates for the worst-case regret, as well as the learning accuracy. Moreover, we discussed the robustness of the bootstrap method for handling unexpected changes in the dynamical model.

As the first work on bootstrap-based policies for LQ models, it poses a number of interesting questions. For example, theoretical analysis for addressing the performance of bootstrap

method under *imperfect* observation is a natural direction for future work. Further subjects of interest include design and analysis of fully non-parametric randomization methods such as *covariate resampling*. Finally, extending the presented framework to *model-free* algorithms can be considered as another fruitful research direction to examine.

REFERENCES

[1] L. Li, "Sample complexity bounds of exploration," in *Reinforcement Learning*. Springer, 2012, pp. 175–204.

[2] P. Dorato, C. T. Abdallah, V. Cerone, and D. H. Jacobson, *Linear-quadratic control: an introduction*. Prentice Hall Englewood Cliffs, NJ, 1995.

[3] N. Lazic, C. Boutilier, T. Lu, E. Wong, B. Roy, M. Ryu, and G. Imwalle, "Data center cooling using model-predictive control," in *Advances in Neural Information Processing Systems*, 2018, pp. 3814–3823.

[4] M. Abeille, A. Lazaric, and X. Brokmann, "LQG for portfolio optimization," *arXiv preprints arXiv:1611.00997*, 2016.

[5] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *ICINCO (1)*, 2004, pp. 222–229.

[6] H. J. Kappen, "Linear theory for control of nonlinear stochastic systems," *Physical review letters*, vol. 95, no. 20, p. 200201, 2005.

[7] T. L. Lai, "Asymptotically efficient adaptive control in stochastic regression models," *Advances in Applied Mathematics*, vol. 7, no. 1, pp. 23–45, 1986.

[8] T. L. Lai and C. Z. Wei, "Asymptotic properties of multivariate weighted sums with applications to stochastic regression in linear dynamic systems," *Multivariate Analysis VI*, pp. 375–393, 1985.

[9] M. K. S. Faradonbeh, A. Tewari, and G. Michailidis, "Finite time identification in unstable linear systems," *Automatica*, vol. 96, pp. 342–353, 2018.

[10] T. Sarkar and A. Rakhlin, "Near optimal finite time identification of arbitrary linear dynamical systems," in *International Conference on Machine Learning*, 2019, pp. 5610–5618.

[11] M. C. Campi and P. Kumar, "Adaptive linear quadratic gaussian control: the cost-biased approach revisited," *SIAM Journal on Control and Optimization*, vol. 36, no. 6, pp. 1890–1907, 1998.

[12] S. Bittanti and M. C. Campi, "Adaptive control of linear time invariant systems: the bet on the best principle," *Communications in Information & Systems*, vol. 6, no. 4, pp. 299–320, 2006.

[13] Y. Abbasi-Yadkori and C. Szepesvári, "Regret bounds for the adaptive control of linear quadratic systems." in *COLT*, 2011, pp. 1–26.

[14] M. K. S. Faradonbeh, A. Tewari, and G. Michailidis, "Optimism-based adaptive regulation of linear-quadratic systems," *arXiv preprint arXiv:1711.07230*, 2017.

[15] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in applied mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[16] M. K. S. Faradonbeh, A. Tewari, and G. Michailidis, "Input perturbations for adaptive control and learning," *arXiv preprint arXiv:1811.04258*, 2018.

[17] H. Mania, A. Guy, and B. Recht, "Simple random search of static linear policies is competitive for reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 1800–1809.

[18] D. Malik, A. Pananjady, K. Bhatia, K. Khamaru, P. Bartlett, and M. Wainwright, "Derivative-free methods for policy optimization: Guarantees for linear quadratic systems," in *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019, pp. 2916–2925.

[19] M. K. S. Faradonbeh, A. Tewari, and G. Michailidis, "Finite time adaptive stabilization of linear systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 8, pp. 3498–3505, 2019.

[20] M. Abeille and A. Lazaric, "Improved regret bounds for thompson sampling in linear quadratic control problems," in *International Conference on Machine Learning*, 2018, pp. 1–9.

[21] M. K. S. Faradonbeh, A. Tewari, and G. Michailidis, "On optimality of adaptive linear-quadratic regulators," *arXiv preprint arXiv:1806.10749*, 2018.

[22] S. Dean, S. Tu, N. Matni, and B. Recht, "Safely learning to control the constrained linear quadratic regulator," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 5582–5588.

[23] B. Efron, "Bootstrap methods: Another look at the jackknife," *The Annals of Statistics*, vol. 7, no. 1, pp. 1–26, 1979.

[24] B. Kveton, C. Szepesvari, S. Vaswani, Z. Wen, T. Lattimore, and M. Ghavamzadeh, "Garbage in, reward out: Bootstrapping exploration in multi-armed bandits," in *International Conference on Machine Learning*, 2019, pp. 3601–3610.

[25] S. Vaswani, B. Kveton, Z. Wen, A. Rao, M. Schmidt, and Y. Abbasi-Yadkori, "New insights into bootstrapping for bandits," *arXiv preprint arXiv:1805.09793*, 2018.

[26] D. Eckles and M. Kaptein, "Thompson sampling with the online bootstrap," *arXiv preprint arXiv:1410.4009*, 2014.

[27] I. Osband and B. Van Roy, "Bootstrapped thompson sampling and deep exploration," *arXiv preprint arXiv:1507.00300*, 2015.

[28] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Advances in neural information processing systems*, 2016, pp. 4026–4034.

[29] K. Rajagopal, S. N. Balakrishnan, and J. R. Busemeyer, "Neural network-based solutions for stochastic optimal control using path integrals," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 534–545, 2017.

[30] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu, "On the sample complexity of the linear quadratic regulator," *arXiv preprint arXiv:1710.01688*, 2017.

[31] M. H. Pesaran and A. Timmermann, "Small sample properties of forecasts from autoregressive models under structural breaks," *Journal of Econometrics*, vol. 129, no. 1-2, pp. 183–217, 2005.

[32] C. I. Byrnes, A. Lindquist, and Y. Zhou, "On the nonlinear ddynamics of fast filtering algorithms," *SIAM Journal on Control and Optimization*, vol. 32, no. 3, pp. 744–789, 1994.

[33] L. Guo and C. Wei, "Global stability/instability of LS-based discrete-time adaptive nonlinear control," *IFAC Proceedings Volumes*, vol. 29, no. 1, pp. 5215–5220, 1996.

[34] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *American Control Conference, 2005. Proceedings of the 2005*. IEEE, 2005, pp. 300–306.

[35] P. Hall, *The bootstrap and Edgeworth expansion*. Springer Science & Business Media, 2013.

[36] P. Hall and C. C. Heyde, *Martingale limit theory and its application*. Academic press, 2014.

[37] B. M. Brown *et al.*, "Martingale central limit theorems," *The Annals of Mathematical Statistics*, vol. 42, no. 1, pp. 59–66, 1971.

[38] D. L. McLeish *et al.*, "Dependent central limit theorems and invariance principles," *the Annals of Probability*, vol. 2, no. 4, pp. 620–628, 1974.

[39] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[40] P. R. Kumar and P. Varaiya, *Stochastic systems: Estimation, identification, and adaptive control*. SIAM, 2015.

[41] A. Becker, P. Kumar, and C.-Z. Wei, "Adaptive control with the stochastic approximation algorithm: Geometry and convergence," *IEEE Transactions on Automatic Control*, vol. 30, no. 4, pp. 330–338, 1985.

[42] T. L. Lai and C.-Z. Wei, "Extended least squares and their applications to adaptive control and prediction in linear systems," *IEEE Transactions on Automatic Control*, vol. 31, no. 10, pp. 898–906, 1986.

[43] M. K. S. Faradonbeh, A. Tewari, and G. Michailidis, "On applications of bootstrap in continuous space reinforcement learning," *arXiv preprint arXiv:1903.05803*, 2019.

[44] J. W. Polderman, "On the necessity of identifying the true parameter in adaptive LQ control," *Systems & control letters*, vol. 8, no. 2, pp. 87–91, 1986.

[45] ——, "A note on the structure of two subsets of the parameter space in adaptive control problems," *Systems & control letters*, vol. 7, no. 1, pp. 25–34, 1986.

[46] T. L. Lai and C. Z. Wei, "Asymptotic properties of general autoregressive models and strong consistency of least-squares estimates of their parameters," *Journal of Multivariate Analysis*, vol. 13, no. 1, pp. 1–23, 1983.

[47] V. Dani, T. P. Hayes, and S. M. Kakade, "Stochastic linear optimization under bandit feedback," *21st Annual Conference on Learning Theory*, pp. 355–366, 2008.

[48] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári, "Improved algorithms for linear stochastic bandits," *Advances in Neural Information Processing Systems*, pp. 2312–2320, 2011.