Learning from similar systems and online data-driven LQR using iterative randomised data compression

Vatsal Kedia, Sneha Susan George and Debraj Chakraborty

***Abstract*— The problem of data driven recursive computation of receding horizon LQR control through a randomized combination of online/current and historical/recorded data, is considered. It is assumed that large amounts of historical input-output data from a system, which is similar but not identical to the current system under consideration, is available. This (possibly large) data set is compressed through a novel randomized subspace algorithm to directly synthesize an initial solution of the standard LQR problem, which however is sub-optimal due to the inaccuracy of the historical model. The first instance of this input is used to actuate the current system and the corresponding instantaneous output is used to iteratively re-solve the LQR problem through a computationally inexpensive randomized rank-one update of the old compressed data. The first instance of the re-computed input is applied to the system at the next instant, output recorded and the entire procedure is repeated at each subsequent instant. As more current data becomes available, the algorithm learns automatically from the new data while simultaneously controlling the system in near optimal manner. The proposed algorithm is computationally inexpensive due to the initial and repeated compression of old and newly available data. Moreover, the simultaneous learning and control makes this algorithm particularly suited for adapting to unknown, poorly modeled and time varying systems without any explicit exploration stage. Simulations demonstrate the effectiveness of the proposed algorithm vs popular exploration/exploitation approaches to LQR control.**

I. INTRODUCTION

Conventional system identification involves collecting input-output (I/O) data through controlled experiments and thereafter estimating model parameters from this data using any one among a rich variety of available algorithms (e.g. see [1], [2] and references therein). Any model based control algorithm can only be started after this required phase of parametric identification is completed successfully. However, the model thus formed is traditionally linear time invariant while the actual system might be more complex, non-linear, time varying or simply might have changed marginally due to wear and tear from the time it was last identified. In such a situation, re-identification is the recommended procedure, which however implies extra cost, effort and postponement and/or stoppage of the actual controlled operation. Moreover conventional re-identification of model parameters is computationally expensive and may only be done after sufficient new data is collected. All these issues predicate that even when re-identification is absolutely essential, that it is done at a rate which is several orders of magnitude slower than the time constants of the controlled system. This leads to accumulation of errors in the control sequences, sub-optimal performances and in extreme cases, might lead to instability of the closed loop. To address the above issues, we propose a compressed data driven iterative technique to directly compute the optimal control at each instant by seamlessly combining past recorded data with the "online" I/O data as and when it becomes available at the current instant of time. The inbuilt data compression combined with the iterative nature of the control sequence computation makes our technique computationally inexpensive enough to be implemented in real time on current control hardware.

A recently proposed paradigm at the boundary of system identification and transfer learning theory [3] involves the estimation of system models using a combination of historical/recorded data from a similar/auxiliary model as well as the current model [4]–[9]. This method effectively addresses the common problem of shortage/unavailability of data and/or difficulties in I/O data collection for the current, to-be-controlled system. The primary advantage of such a procedure is the increased robustness of the parameter estimation process to noise due to increased data size, while increased errors might result due to the mismatch between the auxiliary and the current system. Estimates of finite sample errors in system identification problems in the context of LQR have been studied recently in [10]. However the only available techniques to handle such errors remain in the domain of robust and/or adaptive control [11], [12].

Recursive corrections to the model with streaming data and simultaneous control updates has also been investigated [13]–[16]. Recently, machine learning methods such as exploration and exploitation to optimally combine the identification and control stages in the context of LQR control has been studied rigorously [17]–[19]. However, these methods are predominantly model based and computationally expensive.

On the other hand, another recent paradigm in control (and model predictive control in particular) is the use of I/O data directly to design a controller without explicitly identifying the underlying model [20]–[24]. These techniques have been widely studied for the implementation of predictive control techniques such as receding horizon, iterative or infinite horizon LQR [25], [26].

While both transfer learning and data driven controller synthesis are especially well suited for complex and time varying environments, their implementations with limited computational resources, such as in robotics, cyber physical systems, networked control systems, automotive control etc,

The authors are with the Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai, Maharashtra, India. {vatsalkedia,snehasg,dc}@ee.iitb.ac.in

where small onboard computers are common, are difficult due to the requirement of relatively large computing power. Hence in this paper we propose a computationally efficient amalgamation of the above two ideas. We use randomized (Gaussian) data compression matrices to efficiently learn from "large" historical I/O databases [27], [28] and simultaneously update this compressed data with online or current data as they become available. The updating of the data is done using a pure iteration with negligible computational cost. Thereafter the updated data is used to directly compute a receding horizon LQR control sequence without identifying any model. It turns out that the LQR control sequence computation can also be implemented as a pseudo-iteration based on the underlying iterative computation of the compressed data matrices and a rank one update of an intermediate QR factorization [29]. Due to the compression of the historical (long) database to conveniently small sizes, the entire operation is computationally extremely efficient. Moreover the iterative nature of all the major computations makes it fast enough for real time implementations in situations with limited computing resources.

The proposed algorithm is compared with a basic exploration-exploitation scheme, where the exploration stage consists of a minimum variance white noise excitation to identify the system. This experimental phase is conducted for the minimum duration required by popular subspace identification algorithms [2]. The identified model is used to construct a standard model based receding horizon LQR controller, whose performance cost is compared with that of the proposed algorithm. It turns out that the cost saved by the proposed method due to its inbuilt knowledge of the "similar" system, especially during the initial stages of controlled operation, enables it to vastly outperform such online identification approaches. At the same time, except for the initial offline compression of similar system data, all other online updates are extremely fast.

II. Preliminaries and Problem Formulation

Consider the following discrete LTI system

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + e(t) \\ y(t) &= Cx(t) \end{aligned} \tag{1}$$

Here $e(t) \in \mathbb{R}^n$ is the process noise and is assumed to be a white noise sequence with zero mean and finite covariance i.e. $\mathbb{E}\{e(t_1)e^T(t_2)\} = \eta\delta_{t_1 t_2}$ for all time instants t_1 and t_2, where, $\eta \in \mathbb{R}^{n\times n} > 0$ and δ is the Kronecker delta function. The system parameters $\{A, B, C\}$ are of appropriate dimensions: $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and $C \in \mathbb{R}^{p\times n}$. We assume that the state information can be measured directly [25] (i.e. $C = I_n$ and $p = n$). In the next subsection we briefly review the batch and receding horizon LQR formulations based on [30].

A. Linear Quadratic Regulator

Let us consider the deterministic version of the model (1) for simplicity. We define the following quadratic cost function over a finite horizon k_p as:

$$J(x_0, \mathcal{U}_0) = x(k_p)^T P x(k_p) + \sum_{i=0}^{k_p-1} x(i)^T Q x(i) + u(i)^T R u(i) \tag{2}$$

where $Q = Q^T \succcurlyeq 0 \in \mathbb{R}^{n\times n}$, $P = P^T \succcurlyeq 0 \in \mathbb{R}^{n\times n}$ and $R \succ 0 \in \mathbb{R}^{m\times m}$, $x(0) = x_0$ and $\mathcal{U}_0 = [u(0)\ u(1)\ ...u(k_p - 1)]^T$.

Consider the finite time optimal control problem:

$$\begin{aligned} J_0^*(x_0) &= \min_{\mathcal{U}_0} J(x_0, \mathcal{U}_0) \\ &\text{s.t. } x(t+1) = Ax(t) + Bu(t) \ \ \forall t = 0, 1, \ldots, k_p - 1 \end{aligned} \tag{3}$$

Next, we write the system evolution in terms of inputs and initial condition.

$$\underbrace{\begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(k_p) \end{bmatrix}}_{\mathcal{X}_0} = \underbrace{\begin{bmatrix} I \\ A \\ \vdots \\ A^{k_p} \end{bmatrix}}_{\mathcal{S}_*^x} x(0) + \underbrace{\begin{bmatrix} 0 & 0 & \ldots & 0 \\ B & 0 & \ddots & 0 \\ AB & B & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ A^{k_p-1}B & \ldots & \ldots & B \end{bmatrix}}_{\mathcal{S}_*^u} \underbrace{\begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(k_p-1) \end{bmatrix}}_{\mathcal{U}_0} \tag{4}$$

where, $\mathcal{X}_0 \in \mathbb{R}^{n(k_p+1)}$, $\mathcal{S}_*^x \in \mathbb{R}^{n(k_p+1)\times n}$, $\mathcal{S}_*^u \in \mathbb{R}^{n(k_p+1)\times k_p m}$ and $\mathcal{U}_0 \in \mathbb{R}^{k_p m}$. The cost function $J(x_0, \mathcal{U}_0)$ can be rewritten as:

$$J(x(0), \mathcal{U}_0) = \mathcal{X}_0^T \bar{Q} \mathcal{X}_0 + \mathcal{U}_0^T \bar{R} \mathcal{U}_0 \tag{5}$$

where $\bar{Q} = blockdiag\{Q, Q, \ldots, Q, P\} \in \mathbb{R}^{n(k_p+1)\times n(k_p+1)}$ and $\bar{R} = blockdiag\{R, R, \ldots, R\} \in \mathbb{R}^{k_p m\times k_p m}$. The optimal input sequence is given by:

$$\mathcal{U}_0^*(x(0)) = -(\mathcal{S}_*^{u^T} \bar{Q} \mathcal{S}_*^u + \bar{R})^{-1} \mathcal{S}_*^{u^T} \bar{Q} \mathcal{S}_*^x x(0) \tag{6}$$

The optimal cost $J_0^*(x(0))$ can be easily calculated by substituting the above result in (5).

As opposed to the open loop formulation described above, a similar but closed loop version of (6) is used in the receding horizon framework. Here, an open-loop optimal control problem is solved over a finite horizon of k_p steps at each sampling instant. Only the first instance of the optimal input sequence ($\mathcal{U}^*$) is applied to the process. At the next sampling instant, a new optimal control problem is solved over a shifted horizon of k_p-steps based on new measurements of the current states. The optimal input sequence at i^{th}- sampling instant is given by:

$$\mathcal{U}_i^*(x(i)) = -(\mathcal{S}_*^{u^T} \bar{Q} \mathcal{S}_*^u + \bar{R})^{-1} \mathcal{S}_*^{u^T} \bar{Q} \mathcal{S}_*^x x(i) \tag{7}$$

The above optimal input sequence works perfectly under the scenario where we know the system dynamics, and hence $\mathcal{S}_*^x$ and $\mathcal{S}_*^u$, exactly.

B. Problem Formulation

Let the dynamics of a similar system be given by

$$\begin{aligned} x(t+1) &= \tilde{A}x(t) + \tilde{B}u(t) + e(t) \\ y(t) &= x(t) \end{aligned} \tag{8}$$

where, $\tilde{A} = A + \Delta A$, $\tilde{B} = B + \Delta B$, the pair (A, B) is our actual system while the pair $(\tilde{A}, \tilde{B})$ represents the similar system with $(\Delta A, \Delta B)$ being the unknown perturbation matrices. The other variables in the equation are identical to those defined in the original system (1). We assume that we have access to the I/O data generated previously by the above system, either through separate identification experiments or through regular controlled use. The generated I/O data set is denoted by $\{\tilde{u}(i), \tilde{y}(i)\}_{i=0}^{N_t-1}$.

Problem 1: Design a data-driven "efficient" method to iteratively compute the receding LQR solution given in (7) by leveraging the data set $\{\tilde{u}(i), \tilde{y}(i)\}_{i=0}^{N_t-1}$ recorded from the similar system (8) and online data available at each iteration on application of (7) on the actual system (1).

Some assumptions standard in the system identification literature are listed below.

Assumption 1: [1], [2] The following are assumed:

1) The input $u(t)$ is persistently exciting.
2) The input $u(t)$ is uncorrelated with $e(t)$.
3) The signal to noise ratio is high.

III. Compressed Data Driven LQR Synthesis

In this section we present the two basic building blocks for the proposed method, namely the computation of $\mathcal{S}^x$ and $\mathcal{S}^u$ directly from I/O data and the same computation based on compressed I/O data.

A. Data driven learning of $\mathcal{S}^x$ and $\mathcal{S}^u$: Subspace approach

Given input-output time-series data sequence $\{u(i), y(i)\}$ $\forall i \in \{0, 1, \ldots, N_t - 1\}$. First, a horizon $k = k_p + 1$ is chosen where k_p is the prediction horizon for the LQR problem and the block size $N := N_t - 2k + 2$ is defined. Let us denote a block Hankel matrix based on input sequence $\{u(i)\}$ as follows:

$$U_{i|i+k-1} := \begin{bmatrix} u(i) & u(i+1) & \ldots & u(i+N-1) \\ u(i+1) & u(i+2) & \ldots & u(i+N) \\ \vdots & \vdots & \ddots & \vdots \\ u(i+k-1) & u(i+k) & \ldots & u(i+k+N-2) \end{bmatrix} \tag{9}$$

Now, the past input block Hankel matrix is defined as $U_p := U_{0|k-1} \in \mathbb{R}^{km \times N}$ by substituting $i = 0$ above, while the corresponding future input matrix is defined as $U_f := U_{k|2k-1} \in \mathbb{R}^{km \times N}$ (substituting $i = k$). Similarly we define the output block Hankel matrices $Y_p, Y_f \in \mathbb{R}^{kn \times N}$ using the past/future output data $\{y(i)\}$. Note that the nomenclature "past" and "future" is merely a mnemonic used in the subspace identification literature and has little to do with actual time instants. Although no recordings of noise are assumed to be available, for the sake of notational convenience, similar matrices are also defined for the corresponding past and future innovations processes: $E_p, E_f \in \mathbb{R}^{kn \times N}$. The past input and output data is combined into $W_p := \begin{bmatrix} U_p^T & Y_p^T \end{bmatrix}^T \in \mathbb{R}^{k(m+n) \times N}$. Let $X_f \in \mathbb{R}^{n \times N}$ denote the future state sequence defined as $X_f := \begin{bmatrix} x(k) & x(k+1) & \ldots & x(k+N-1) \end{bmatrix} \in \mathbb{R}^{n \times N}$. We further denote, $\Phi_k \in \mathbb{R}^{kn \times kn}$ as noise impulse response Toeplitz matrix as shown below.

$$\Phi_k = \begin{bmatrix} 0 & 0 & \ldots & 0 \\ I & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ A^{k-2} & \ldots & I & 0 \end{bmatrix}$$

Recursively using (1) and data matrices defined above we get,

$$Y_f = \mathcal{S}^x X_f + \begin{bmatrix} \mathcal{S}^u & 0_m \end{bmatrix} U_f + \Phi_k E_f. \tag{10}$$

where 0_m is zero matrix of size $km \times m$ and $\begin{bmatrix} \mathcal{S}^u & 0_m \end{bmatrix} \in \mathbb{R}^{kn \times km}$. Let $\bar{A} := A - C$, $\bar{B} := B$, $\Upsilon_k := \begin{bmatrix} \bar{A}^{k-1}\bar{B} & \bar{A}^{k-2}\bar{B} & \ldots & \bar{B} \end{bmatrix} \in \mathbb{R}^{n \times km}$ be the modified reversed extended controllability matrix and $\Upsilon_k^e := \begin{bmatrix} \bar{A}^{k-1} & \bar{A}^{k-2} & \ldots & I \end{bmatrix} \in \mathbb{R}^{n \times kn}$ be the modified reversed extended stochastic controllability matrix. Then, under the assumptions listed above and for large prediction horizons k it can be shown [31] that $X_f = L_p W_p$ for $L_p := \begin{bmatrix} \Upsilon_k & \Upsilon_k^e \end{bmatrix} \in \mathbb{R}^{n \times k(m+n)}$. Thereby (10) reduces to

$$Y_f = \mathcal{S}^x L_p W_p + \begin{bmatrix} \mathcal{S}^u & 0 \end{bmatrix} U_f + \Phi_k E_f. \tag{11}$$

Under assumption 1, the projection Y_f onto the joint span of W_p and U_f becomes:

$$Y_f / \begin{bmatrix} W_p \\ U_f \end{bmatrix} = \mathcal{S}^x L_p W_p + \begin{bmatrix} \mathcal{S}^u & 0 \end{bmatrix} U_f \tag{12}$$

Now Y_f orthogonally projected onto the joint span of W_p and U_f can also be written as,

$$\begin{aligned} Y_f / \begin{bmatrix} W_p \\ U_f \end{bmatrix} &= Y_f /_{U_f} W_p + Y_f /_{W_p} U_f \\ &= \underbrace{\bar{L}_p W_p}_{:=\zeta} + L_{U_f} U_f \end{aligned} \tag{13}$$

where $\zeta := Y_f /_{U_f} W_p \in \mathbb{R}^{kp \times N}$ is known as the oblique projection of Y_f onto W_p along U_f. On comparing (12) and (13) we get $\bar{L}_p = \mathcal{S}^x L_p$, $L_{U_f} = \begin{bmatrix} \mathcal{S}^u & 0 \end{bmatrix}$. Define $\zeta = \bar{L}_p W_p \in \mathbb{R}^{kp \times N}$. An efficient way to calculate the oblique projection is by using the QR decomposition.

1) QR step: Perform LQ decomposition on $H := \begin{bmatrix} U_f^T & W_p^T & Y_f^T \end{bmatrix}^T \in \mathbb{R}^{2k(m+n) \times N}$ to obtain the decomposition of Y_f as shown in (11).

$$H = \begin{bmatrix} U_f \\ W_p \\ Y_f \end{bmatrix} = \underbrace{\begin{bmatrix} R_{11} & 0 & 0 \\ R_{21} & R_{22} & 0 \\ R_{31} & R_{32} & 0 \end{bmatrix}}_{L} \begin{bmatrix} Q_1^T \\ Q_2^T \\ Q_3^T \end{bmatrix} \tag{14}$$

From (14),

$$Y_f / \begin{bmatrix} W_p \\ U_f \end{bmatrix} = R_{32} R_{22}^{\dagger} W_p + (R_{31} - R_{32} R_{22}^{\dagger} R_{21}) R_{11}^{-1} U_f \tag{15}$$

On comparing (12) and (15),

$$\begin{aligned} \bar{L}_p &= R_{32} R_{22}^{\dagger} \\ \begin{bmatrix} \mathcal{S}^u & 0 \end{bmatrix} &= (R_{31} - R_{32} R_{22}^{\dagger} R_{21}) R_{11}^{-1} \end{aligned} \tag{16}$$

Now it can be shown using (10), (12) and (13) that

$$\underbrace{\mathcal{S}^x X_f}_{theoretical} = \bar{L}_p W_p = \underbrace{R_{32} R_{22}^{\dagger} W_p}_{data} =: \zeta. \tag{17}$$

2) SVD step: Next we calculate the SVD of ζ as follows:

$$\begin{aligned} \zeta &= \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \\ &= U_1 \Sigma_1 V_1^T + \underbrace{U_2 \Sigma_2 V_2^T}_{noise} \\ &\approx U_1 \Sigma_1 V_1^T = \underbrace{U_1 \Sigma_1^{1/2} T}_{\mathcal{S}^x} T^{-1} \Sigma_1^{1/2} V_1^T \end{aligned} \tag{18}$$

where $T \in \mathbb{R}^{n \times n}$ is an arbitrary similarity transformation matrix. The second term is ignored assuming that the noise component is negligible as compared to the system contribution. Therefore, $\mathcal{S}^x$ and $\mathcal{S}^u$ can be computed directly from input-output data up to similarity transforms. Consequently, the optimal input $\mathcal{U}_*$ using (7) becomes:

$$\mathcal{U}_i^*(x(i)) = -(\mathcal{S}^{u^T} \bar{Q} \mathcal{S}^u + \bar{R})^{-1} \mathcal{S}^{u^T} \bar{Q} \mathcal{S}^x T^{-1} x(i) \tag{19}$$

In order to use the above feedback law, we need to compute T. It is easy to see that on comparing $\mathcal{S}_*^x$ (see (4)) and $\mathcal{S}^x$, the first n-rows of $\mathcal{S}^x$ forms T i.e. $T = \mathcal{S}^x(1:n, :)$.

B. Efficient learning: Randomized approach

We propose randomized data compression based computation of $\mathcal{S}^x$ and $\mathcal{S}^u$.

1) Data Compression:: Recall that the matrix $H := \begin{bmatrix} U_f^T & W_p^T & Y_f^T \end{bmatrix}^T \in \mathbb{R}^{2k(m+n) \times N}$ and define $N_c := 2k(m+n) + l$ where, $l > 0$ is commonly known as the oversampling parameter [32]. Define $\mathcal{C} \in \mathbb{R}^{N \times N_c}$ to be a random matrix whose elements are iid gaussian with $(\mathcal{C})_{ij} \sim \mathcal{N}(0, \frac{1}{N_c})$. We further define, $\bar{U}_f := U_f \mathcal{C}$, $\bar{U}_p := U_p \mathcal{C}$, $\bar{Y}_p := Y_p \mathcal{C}$, $\bar{Y}_f := Y_f \mathcal{C}$, $\bar{W}_p := W_p \mathcal{C} = \begin{bmatrix} \bar{U}_p^T & \bar{Y}_p^T \end{bmatrix}^T$ and $\bar{H} := H\mathcal{C} = \begin{bmatrix} \bar{U}_f^T & \bar{W}_p^T & \bar{Y}_f^T \end{bmatrix}^T \in \mathbb{R}^{2k(m+n) \times N_c}$.

2) LQ Decomposition: In the uncompressed case, the first step to compute $\mathcal{S}^x$ and $\mathcal{S}^u$ is to perform QR on H^T to obtain $\mathcal{S}^x$ and $\mathcal{S}^u$ (see (14)). Since, we need only the R-factor from the QR step, hence instead, we propose to perform QR on the compressed matrix $\bar{H}^T \in \mathbb{R}^{N_c \times 2k(m+n)}$. This step reduces the QR computation cost significantly since $N_c << N$. However for this method to work, we must show that the projection can still be used to extract the desired subspace even after data compression.

First, note that an equation similar to (15) can be obtained for the compressed case using QR decomposition of $\bar{H}^T$:

$$\bar{Y}_f / \begin{bmatrix} \bar{W}_p \\ \bar{U}_f \end{bmatrix} = \bar{R}_{32} \bar{R}_{22}^{\dagger} \bar{W}_p + (\bar{R}_{31} - \bar{R}_{32} \bar{R}_{22}^{\dagger} \bar{R}_{21}) \bar{R}_{11}^{-1} \bar{U}_f \tag{20}$$

where $\bar{(.)}$ denote the equivalent matrices for the compressed case. Now, we right multiply (11) by $\mathcal{C}$

$$\bar{Y}_f = \bar{L}_p \bar{W}_p + \begin{bmatrix} \mathcal{S}^u & 0 \end{bmatrix} \bar{U}_f + \Phi_k \bar{E}_f \tag{21}$$

Note that, the $\bar{L}_p$ remains the same as in the uncompressed case due to multiplication from the right by $\mathcal{C}$ on (11). Under assumption 1, the orthogonal projection of $\bar{Y}_f$ onto the joint span of $\bar{W}_p$ and $\bar{U}_f$ is

$$\bar{Y}_f / \begin{bmatrix} \bar{W}_p \\ \bar{U}_f \end{bmatrix} \approx \bar{L}_p \bar{W}_p + \begin{bmatrix} \mathcal{S}^u & 0 \end{bmatrix} \bar{U}_f \tag{22}$$

The above result ensures that we can use the projection to extract the desired subspace under assumption 1 (similar to the uncompressed case). Recall that the oblique projection $\zeta := Y_f /_{U_f} W_p \in \mathbb{R}^{kp \times N}$ and define the compressed version $\bar{\zeta} := \bar{Y}_f /_{\bar{U}_f} \bar{W}_p \in \mathbb{R}^{kp \times N_c}$. Then the following result can be proved along the lines of [27], [28] and is not included here in the interest of space.

Lemma 1: $\mathcal{R}(\bar{\zeta}) = \mathcal{R}(\zeta)$ almost surely.

Therefore, various projections of $\bar{Y}_f$ onto $\bar{U}_f$ and $\bar{W}_p$ as shown in (22) can be calculated from the compressed QR factors $\big(\bar{R}_{ij} \ \forall i, j \in \{1, 2, 3\}$, see (20)$\big)$. Using the above Lemma and comparing (15) and (20), we get

$$\bar{\zeta} = \underbrace{\bar{R}_{32} \bar{R}_{22}^{\dagger}}_{\bar{L}_p} \bar{W}_p \tag{23}$$

3) Projection: SVD Step: As mentioned in previous subsection, after computing the oblique projection $\zeta \in \mathbb{R}^{kp \times N}$ (see (17)), the next step is to perform SVD according to (18). Since, we are interested to compute $\mathcal{S}^x$ using only the left singular vectors of ζ, we show that an equivalent operation can be performed on $\bar{\zeta}$.

Theorem 1: There exists a decomposition of $\bar{\zeta} = \bar{L}_p \bar{W}_p = \bar{\mathcal{S}}^x X \in \mathbb{R}^{kp \times N_c}$ such that $\mathcal{R}(\bar{\mathcal{S}}^x) = \mathcal{R}(\mathcal{S}^x)$ a.s. where $\bar{\mathcal{S}}^x \in \mathbb{R}^{kp \times n}$ and $X \in \mathbb{R}^{n \times N_c}$.

Proof: The proof is similar to Theorem 2 in [28] ■

Therefore SVD can be performed on $\bar{\zeta}$ to compute $\mathcal{S}^x$ upto similarity transform. This step reduces the computation-cost as we use $\bar{\zeta} \in \mathbb{R}^{kp \times N_c}$ to compute $\bar{\mathcal{S}}^x$ instead of $\zeta \in \mathbb{R}^{kp \times N}$. The matrix $\mathcal{S}^u$ can be easily estimated using (20)

$$\begin{bmatrix} \mathcal{S}^u & 0 \end{bmatrix} = (\bar{R}_{31} - \bar{R}_{32} \bar{R}_{22}^{\dagger} \bar{R}_{21}) \bar{R}_{11}^{-1} \tag{24}$$

The above Theorem proves that even after data compression we can use appropriate projections to learn $\mathcal{S}^x$ and $\mathcal{S}^u$ up to similarity transforms.

IV. Randomized Iterative LQR (RiLQR)

The methods described in the previous section allows for efficient computation of the LQR control directly from large quantities of I/O data. This efficiency is essential to learn from the potentially large sized similar system data. However we would like to continuously update the optimal control based on the online data generated in real time from the actual system being controlled. This calls for a iterative method to update $\mathcal{S}^x$, $\mathcal{S}^u$ and ideally $\mathcal{U}_i^*(x(i))$ in (19) based on the newly available data at each time instant. We propose a method to partially achieve this objective in this section.

A. Learning control from similar system

Recall the dynamics of the similar system given in (8) which has previously generated the I/O data set denoted by $\{\tilde{u}(i), \tilde{y}(i)\}_{i=0}^{N_t - 1}$. We formulate data matrices from this I/O

data, denote them as $U_{p(-1)}$, $U_{f(-1)}$, $Y_{p(-1)}$, $Y_{f(-1)}$ and define $H_{-1} := \begin{bmatrix} U_{f(-1)}^T & W_{p(-1)}^T & Y_{f(-1)}^T \end{bmatrix}^T \in \mathbb{R}^{2k(m+n)\times N}$. The $U_{p(-1)}$ and $U_{f(-1)}$ using data from similar system can be represented as:

$$U_{p(-1)} := \begin{bmatrix} \tilde{u}(0) & \tilde{u}(1) & \dots & \tilde{u}(N-1) \\ \tilde{u}(1) & \tilde{u}(2) & \dots & \tilde{u}(N) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{u}(k-1) & \tilde{u}(k) & \dots & \tilde{u}(k+N-2) \end{bmatrix}$$

$$U_{f(-1)} := \begin{bmatrix} \tilde{u}(k) & \tilde{u}(k+1) & \dots & \tilde{u}(k+N-1) \\ \tilde{u}(k+1) & \tilde{u}(k+2) & \dots & \tilde{u}(k+N) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{u}(2k-1) & \tilde{u}(2k) & \dots & \tilde{u}(2k+N-2) \end{bmatrix}$$

Let us define $\tilde{u}_{p(-1)}$ as the last column of $U_{p(-1)}$ so that $\tilde{u}_{p(-1)} = \begin{bmatrix} \tilde{u}(N-1) \\ \tilde{u}(N) \\ \vdots \\ \tilde{u}(k+N-2) \end{bmatrix}$; and $\tilde{u}_{f(-1)}$ as the last column of $U_{f(-1)}$ so that $\tilde{u}_{f(-1)} = \begin{bmatrix} \tilde{u}(k+N-1) \\ \tilde{u}(k+N) \\ \vdots \\ \tilde{u}(2k+N-2) \end{bmatrix}$. Similarly, $\tilde{y}_{p(-1)}$ and $\tilde{y}_{f(-1)}$ can be defined. Therefore the last column of H_{-1} is defined as $h_{-1} := \begin{bmatrix} \tilde{u}_{f(-1)} \\ \tilde{u}_{p(-1)} \\ \tilde{y}_{p(-1)} \\ \tilde{y}_{f(-1)} \end{bmatrix} \in \mathbb{R}^{2k(m+n)}$.

Following the same notation as in section III-B, the compressed version of H_{-1} is denoted by $\bar{H}_{-1} = H_{-1}\mathcal{C}_{-1}$. Using this $\bar{H}_{-1}$ and our knowledge of the initial state $x(0)$ of the to-be-controlled system (1), the optimal control $\mathcal{U}_0^*(x(0))$ can be computed using $\mathcal{S}^x$ and $\mathcal{S}^u$ as shown above. The first instance of this control input sequence (say $u^*(0)$) is applied at the starting instant $t = 0$ of the controlled operation of system (1). The output $y(0)$ is recorded.

B. Update of H

Assume that we receive the new data $(u^*(0), y(0))$ generated at time $t = 0$ from the actual system. This new information can be used to augment the I/O data matrices as shown below:

$$U_{p(0)} := \begin{bmatrix} \tilde{u}(0) & \tilde{u}(1) & \dots & \tilde{u}(N-1) & \tilde{u}(N) \\ \tilde{u}(1) & \tilde{u}(2) & \dots & \tilde{u}(N) & \tilde{u}(N+1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \tilde{u}(k-1) & \tilde{u}(k) & \dots & \tilde{u}(k+N-2) & \tilde{u}(k+N-1) \end{bmatrix}$$

$$U_{f(0)} := \begin{bmatrix} \tilde{u}(k) & \tilde{u}(k+1) & \dots & \tilde{u}(k+N-1) & \tilde{u}(k+N) \\ \tilde{u}(k+1) & \tilde{u}(k+2) & \dots & \tilde{u}(k+N) & \tilde{u}(k+N+1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \tilde{u}(2k-2) & \tilde{u}(2k-1) & \dots & \tilde{u}(2k+N-3) & \tilde{u}(2k+N-2) \\ \tilde{u}(2k-1) & \tilde{u}(2k) & \dots & \tilde{u}(2k+N-2) & u^*(0) \end{bmatrix}$$

Similarly $Y_{p(0)}$ and $Y_{f(0)}$ can also be augmented. So, the updated new column to be added to H_{-1} can be written as $h_0 = \begin{bmatrix} u_{f0} \\ u_{p0} \\ y_{p0} \\ y_{f0} \end{bmatrix}$ where u_{f0} denotes the last column of $U_{f(0)}$ and so on. Therefore, $H_0 = \begin{bmatrix} H_{-1} & h_0 \end{bmatrix}$. Now following the same notation as in section III-B, we perform data compression on H_0 leading to:

$$\begin{aligned} \bar{H}_0 = H_0\mathcal{C}_0 &= \begin{bmatrix} H_{-1} & h_0 \end{bmatrix} \begin{bmatrix} \mathcal{C}_{-1} \\ c_0^T \end{bmatrix} \\ &= H_{-1}\mathcal{C}_{-1} + h_0 c_0^T \\ &= \bar{H}_{-1} + h_0 c_0^T \end{aligned} \tag{25}$$

From time instant $t = 1$ onwards the compressed data matrix at the t^{th} instant can be obtained from the data matrix at the $(t-1)^{th}$ instant through an iteration identical to (25) above.

$$\bar{H}_t = \bar{H}_{t-1} + h_t c_t^T \tag{26}$$

where h_t is the last column of H_t augmented with the current data point $(u^*(t), y(t))$ as shown above, and c_t is a iid Gaussian vector of appropriate size. Therefore the new information can be included by iterating with the previously compressed data matrix. Note that the size of the compressed matrix remains invariant even after addition of infinite data points. Though not pursued in this article, one may weigh new information differently than previously collected data by weighing the rank one update by a weighing factor $\gamma > 0$ as in

$$\bar{H}_t = \bar{H}_{t-1} + \gamma h_t c_t^T. \tag{27}$$

C. Update on QR decomposition: Iterative rank-1 update

Once we have computed $\bar{H}_t$, the next step would be to perform QR decomposition ($\bar{H}_t^T$) followed by SVD and the subsequent steps described in the previous section. In this section, we show that the QR decomposition can be iterated directly without computing $\bar{H}_t$ explicitly. Clearly (25) is a rank-1 update on $\bar{H}_{t-1} \in \mathbb{R}^{s\times N_c}$ where $s := 2k(m+n)$. Therefore we can directly iterate on the Q and R factors.

Let the QR decomposition of $\bar{H}_{t-1}^T = Q_{t-1}R_{t-1}$ and $\bar{H}_t = \bar{H}_{t-1}^T + h_t c_t^T = Q_{t-1}(R_{t-1} + z_t c_t^T)$ where $z_t = Q_{t-1}^T h_t$. We further define $L_t^J := J_{1(i)}^T J_{2(i)}^T \dots J_{(s-1)(i)}^T$ where $J_{(q)(i)}$ $\forall q \in \{1, 2, \dots, s-1\}$ are rotation matrices in planes q and $q+1$ such that $L_t^J z_t = \pm||z_t||_2 e_1$. These rotation matrices can be easily computed using Algorithms 5.2.4 in [29] and requires $\mathcal{O}(s^2)$ computation. It is easy to see that $M := L_t R_{t-1}$ is upper Hessenberg. Consequently, $L_t^J(R_{t-1} + z_t c_t^T) = M_{t-1} \pm ||z_t||_2 e_1 c_t^T$ is also upper Hessenberg. Next we define, $L_t^G := G_{(s-1)(i)}^T G_{(s-2)(i)}^T \dots G_{1(i)}^T$ where $G_{(q)(i)}$ $\forall q \in \{1, 2, \dots, s-1\}$ are rotation matrices such that $L_t^G(M_{t-1} \pm ||z_t||_2 e_1 c_t^T)$ is upper triangular. Therefore the iterative update on QR decomposition is given by:

$$\begin{aligned} Q_t &= Q_{t-1}Z_t \\ R_t &= Z_t^T R_{t-1} + \Delta R_t \end{aligned} \tag{28}$$

where $Z_t = L_t^G L_t^J$ and $\Delta R_t = L_t^G ||z_t||_2 e_1 c_t^T$.

It is well known [29] that the above rank-1 QR update requires $\mathcal{O}(s^2)$ computation. Hence, the above algorithm requires $\mathcal{O}(s^2)$ computation at each iteration as compared to $\mathcal{O}(s^3 + s^2 l)$ in case of repeated QR decomposition on $\bar{H}_t$. It is important to note that if we had not compressed H_t, then the required QR on H_t^T would have required $\mathcal{O}(s^2 N)$ computation at each instant making it prohibitively large due to $N >> s$. Moreover, in such an uncompressed

scenario, the data size (N) would grow with time making this computation even more intractable. Therefore, there is significant saving in computation in this step due to iterative compression and the rank-1 update of the compressed QR factors.

The proposed pseudo-iterative control scheme (RiLQR) is summarized in two algorithms: Algorithm 1 describes the process of learning the initial optimal control by leveraging data from a similar system, while Algorithm 2 iteratively learns the modified control by augmenting old information from the similar system with online data from the actual system.

Algorithm 1: Learning control from similar system

Input: Load input-output data from similar system.
Formulate data matrices $U_{p(-1)}$, $U_{f(-1)}$, $Y_{p(-1)}$ and $Y_{f(-1)}$ from input-output data. Stack them to form $H_{-1} = \begin{bmatrix} U_{f(-1)}^T & U_{p(-1)}^T & Y_{p(-1)}^T & Y_{f(-1)}^T \end{bmatrix}^T$ matrix.
Store the last column of H_{-1} as $h_{-1} = H_{-1}(:, end)$.
Generate random Gaussian iid matrix $\mathcal{C}_{-1} \in \mathbb{R}^{N \times N_c}$.
Perform randomized data compression on H_{-1} using $\mathcal{C}_{-1}$ defined as $\bar{H}_{-1} := H_{-1}\mathcal{C}_{-1}$.
Perform QR decomposition on $\bar{H}_{-1}^T = Q_{-1}R_{-1}$.
Extract $\bar{L}_p$ and $\mathcal{S}_{-1}^u$ from R_{-1} obtained above using (23) and (24) respectively.
Perform SVD using $\bar{L}_p$ and $\bar{W}_p$ to estimate $\bar{\mathcal{S}}_{-1}^x$ (18).
Compute $\mathcal{U}_{-1}^*$ using (19). Store first input sequence from $\mathcal{U}_{-1}^*$ as $u^*(0)$.
Output: $\mathcal{S}_{-1}^x$, $\mathcal{S}_{-1}^u$, Q_{-1}, R_{-1}, h_{-1} and $u^*(0)$.

Algorithm 2: Learning control from similar system and actual system

Initialize: $t \leftarrow 0$
while $t \neq T_{iter}$ **do**
  Input: $\mathcal{S}_{t-1}^x$, $\mathcal{S}_{t-1}^u$, Q_{t-1}, R_{t-1}, h_{t-1} and $u^*(t)$.
  Generate output data $y(t)$ using $u^*(t)$ from actual system (A, B). Record the I/O pair as $(u^*(t), y(t))$.
  Update h_t based on h_{t-1} using current measurement from actual system $(u^*(t), y(t))$ (see section IV-B).
  Generate random Gaussian vector c_t.
  Iteration on QR decomposition to obtain R_t using Q_{t-1}, R_{t-1}, h_t and c_t using (28).
  From R_t obtained above compute $\mathcal{S}_t^x$ and $\mathcal{S}_t^u$ as performed in Algorithm 1.
  Compute $\mathcal{U}_t^*$ based on $\mathcal{S}_t^x$ and $\mathcal{S}_t^u$ using (19). Store first sequence from $\mathcal{U}_t^*$ denoted by $u^*(t+1)$
  Output:. $\mathcal{S}_t^x$, $\mathcal{S}_t^u$, Q_t, R_t, h_t and $u^*(t+1)$.
  $t \leftarrow t+1$;

D. Efficiency of the proposed algorithm

The computation complexity of the proposed algorithm is presented in Table I. The main computation burden of the proposed algorithm is due to initial data compression step i.e. matrix multiplication of H_{-1} and $\mathcal{C}_{-1}$ (see step 5 in Algorithm 1) that requires $\mathcal{O}(k^2N)$ computation. There are two ways to address this issue:

1) Offline computation of $\bar{H}_{-1}$: Since the data was generated from a similar system, hence the compression can be done before the start of real time operation on the actual system.
2) Matrix multiplication is a highly parallel operation hence can be implemented efficiently.

Except the initial data compression step, all other steps of the proposed algorithm is $\mathcal{O}(k^2N_c) \approx \mathcal{O}(k^3)$.

TABLE I
COMPUTATION COMPLEXITY OF RILQR

	Steps	Proposed
1	Initial data compresssion $\bar{H}_{-1} = H_{-1}\mathcal{C}_{-1}$	$\mathcal{O}(k^2N)$
2	Initial QR decomposition $\bar{H}_{-1} = Q_{-1}R_{-1}$	$\mathcal{O}(k^3)$
3	Rank-1 QR update	$\mathcal{O}(k^2)$
4	$\mathcal{S}^x$	$\mathcal{O}(k^3)$
5	$\mathcal{S}^u$	$\mathcal{O}(k^3)$
6	$\mathcal{U}^*$	$\mathcal{O}(k^3)$

V. NUMERICAL EXAMPLE

We illustrate the advantage of using historical data set obtained from a similar system in the proposed method, over the standard exploration-exploitation approach, where the system is unknown. Consider a second order, discrete-time system whose matrices are given by

$$A = \begin{bmatrix} 1.0 & 0.40 \\ 0.005 & -0.99 \end{bmatrix}; B = \begin{bmatrix} 0.2 \\ 0.5 \end{bmatrix}; C = I_2$$

Firstly, as part of the exploratory phase in the standard approach, a Gaussian iid white noise sequence of variance $\sigma_{u_E}^2$ is used to drive the system forward in open loop through $T_{explore} = 50$ time-steps, and the generated I/O data set $\{\tilde{u}(i), \tilde{y}(i)\}_{i=0}^{50}$ is used to identify the system using the Multivariable Subspace Identification: MOESP [33] algorithm. This algorithm requires the creation of the Hankel matrices (see (9)) The width of these matrices depend on the length of the collected data. Since we would like to keep the exploration phase to be as short as possible to keep down the LQR cost, we choose $T_{explore} = 50$ to make the data matrix H approximately square (see (14)). The variance of the noise sequence $e(t)$ is kept around 0.01 in all the experiments. The model based LQR controller uses the estimated system matrices to compute the optimal feedback gain [30] with a prediction horizon $k_p = 4$ and is run in closed loop for $T_{exploit} = 150$ time-steps as part of the exploitation phase. The $T_{exploit} = 150$ is kept sufficiently long for the system to become stable. The total run time is: $T_{iter} = T_{explore} + T_{exploit}$. In order to apply the proposed methodology, we set

a parallel open-loop experiment to collect the historical data with a similar system with perturbed eigenvalues, given as

$$\tilde{A} = \begin{bmatrix} 0.80 & 0.30 \\ 0.105 & -0.89 \end{bmatrix} \quad \tilde{B} = \begin{bmatrix} 0.21 \\ 0.6 \end{bmatrix}$$

The other system matrices are kept the same as in the original system. The eigenvalues of the original system are 1.001 and -0.991, and of the similar system are are 0.818 and -0.908. For the similar system experiment, without loss of any generality, the input signal is chosen to be a white Gaussian noise sequence with zero mean, covariance $\sigma_{ud}^2 = 1$ with data length N_t =100,000. The I/O data is collected and stored for future use. Algorithm 1 is applied on the similar system data collected and then the actual system (A, B) is run in closed loop in conjunction with Algorithm 2 for the same $T_{iter} = 200$, where the RiLQR controller learns from the actual system outputs in real time and generates the optimal input sequence. The state and input penalty matrices are chosen as $Q = P = I_2$ and $R = 1$ respectively.

For the proposed algorithm, the total optimal cost (J_{RiLQR}) is computed using (3) over T_{iter} time-steps. The total cost for the model based LQR method (mLQR) is summed over the exploration and exploitation phases: $J_{mLQR} = J_{explore} + J_{exploit}$.

Clearly the model based control cost $J_{explore}$ is dependent on both the duration of the explore phase as well as the variance of the input sequence used during the explore phase. The choice of $T_{explore} = 50$ is explained above. The variance of the input noise during the explore phase, denoted by $\sigma_{u_E}^2$, also affects the accuracy of the estimated system parameters and hence the control. For very low $\sigma_{u_E}^2$, the system is misidentified by the MOESP algorithm owing to low excitation; which is reflected in the worsening control performance for very low values of $\sigma_{u_E}^2$. Hence a very low $\sigma_{u_E}^2$ may decrease the $J_{explore}$ but will increase the $J_{exploit}$. We repeat the model based experiment described above for several values of $\sigma_{u_E}^2$. It can be seen from Table II, that the variation of J_{mLQR} vs $\sigma_{u_E}^2$ is as expected. It is also clear that even the minimum value of J_{mLQR} ($= 55.27$) achieved approximately with $\sigma_{u_E}^2 =$ 3.2e-5, is far greater than that resulting from the proposed method. The input and the state trajectories for the model based control vs the proposed algorithm are shown for two different values of $\sigma_{u_E}^2$: $\sigma_{u_E}^2 = 0.83$ for Figure 1 and $\sigma_{u_E}^2 =$ 9.5e-8 for Figure 2.

TABLE II

COST AND EIGENVALUE VARIATION WITH RESPECT TO INPUT VARIANCE

Cases	$\sigma_{u_E}^2$	SNR	$J_{explore}$	$J_{exploit}$	J_{mLQR}	J_{RiLQR}
1	1.15	98.90	625.16	20.0	645.16	5.62
2	0.83	79.95	232.18	16.17	248.36	6.25
3	0.08	9.21	79.83	3.15	82.98	4.72
4	0.001	0.08	53.05	2.92	55.97	5.69
5	3.2e-5	3.7e-3	53.25	2.02	55.27	5.40
6	9.5e-8	8.4e-6	28.60	55.68	84.29	5.05
7	1.1e-10	1.1e-8	49.89	72.96	122.86	5.71

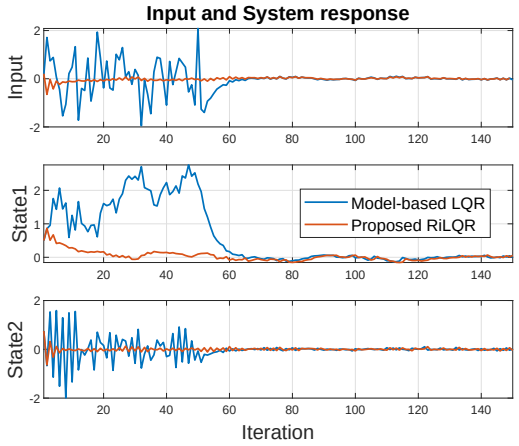

Fig. 1. State response comparison for mLQR and RiLQR for T_{iter} = 150 with SNR = 79.95

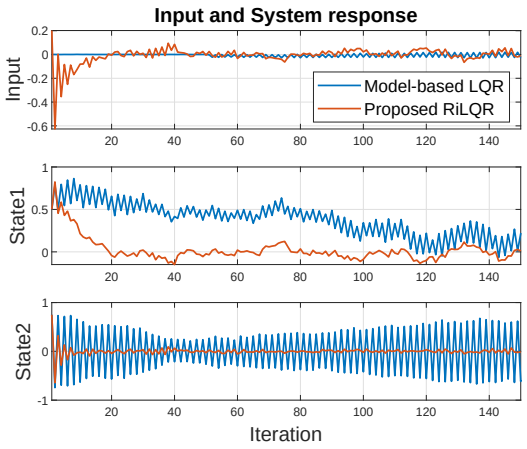

Fig. 2. State response comparison for mLQR and RiLQR for T_{iter} = 150 with SNR = 8.4e-6

VI. CONCLUSIONS

A data driven iterative LQR which seamlessly combine past and online data through randomized compression, is presented in this paper. While not included in this preliminary study, it is expected that discrepancies with the historical model and/or rapid real time variations in model parameters will adversely affect the efficacy of the computed control sequence. Convergence proofs and performance guarantees will require restrictions on the admissible model variations, while constraints on the states/inputs might be challenging to handle in a truly iterative framework. The performance of the proposed algorithm vs more sophisticated exploitation-exploration scheme should be investigated in the future.

REFERENCES

[1] L. Ljung, *System identification: Theory for the User (second edition)*. Springer, 1999.

[2] P. Van Overschee and B. De Moor, *Subspace identification for linear systems: Theory-Implementation-Applications*. Springer Science & Business Media, 2012.

[3] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[4] L. Xin, L. Ye, G. Chiu, and S. Sundaram, "Learning dynamical systems by leveraging data from similar systems," *arXiv preprint arXiv:2302.04344*, 2023.

[5] L. Xin, L. Ye, G. Chiu, and S. Sundaram, "Identifying the dynamics of a system by leveraging data from similar systems," in *2022 American Control Conference (ACC)*, pp. 818–824, 2022.

[6] E. Lau, V. Srivastava, and S. D. Bopardikar, "A multi-fidelity bayesian approach to safe controller design," *IEEE Control Systems Letters*, 2023.

[7] L. F. Toso, H. Wang, and J. Anderson, "Learning personalized models with clustered system identification," *arXiv preprint arXiv:2304.01395*, 2023.

[8] Y. Zheng and N. Li, "Non-asymptotic identification of linear dynamical systems using multiple trajectories," *IEEE Control Systems Letters*, vol. 5, no. 5, pp. 1693–1698, 2020.

[9] S. Tu, R. Frostig, and M. Soltanolkotabi, "Learning from many trajectories," *arXiv preprint arXiv:2203.17193*, 2022.

[10] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu, "On the sample complexity of the linear quadratic regulator," *Foundations of Computational Mathematics*, vol. 20, no. 4, pp. 633–679, 2020.

[11] K. Zhou and J. C. Doyle, *Essentials of robust control*, vol. 104. Prentice hall Upper Saddle River, NJ, 1998.

[12] K. J. Åström and B. Wittenmark, *Adaptive control*. Courier Corporation, 2013.

[13] T.-H. Kim and T. Sugie, "Adaptive receding horizon predictive control for constrained discrete-time linear systems with parameter uncertainties," *International Journal of Control*, vol. 81, no. 1, pp. 62–73, 2008.

[14] M. Tanaskovic, L. Fagiano, R. Smith, and M. Morari, "Adaptive receding horizon control for constrained mimo systems," *Automatica*, vol. 50, no. 12, pp. 3019–3029, 2014.

[15] M. Lorenzen, M. Cannon, and F. Allgöwer, "Robust mpc with recursive model update," *Automatica*, vol. 103, pp. 461–471, 2019.

[16] M. Bujarbaruah, X. Zhang, M. Tanaskovic, and F. Borrelli, "Adaptive stochastic mpc under time-varying uncertainty," *IEEE Transactions on Automatic Control*, vol. 66, no. 6, pp. 2840–2845, 2020.

[17] Y. Abbasi-Yadkori and C. Szepesvári, "Regret bounds for the adaptive control of linear quadratic systems," in *Proceedings of the 24th Annual Conference on Learning Theory*, pp. 1–26, JMLR Workshop and Conference Proceedings, 2011.

[18] M. Abeille and A. Lazaric, "Thompson sampling for linear-quadratic control problems," in *Artificial intelligence and statistics*, pp. 1246–1254, PMLR, 2017.

[19] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu, "Regret bounds for robust adaptive control of the linear quadratic regulator," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[20] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. De Moor, "A note on persistency of excitation," *Systems & Control Letters*, vol. 54, no. 4, pp. 325–329, 2005.

[21] C. De Persis and P. Tesi, "Formulas for data-driven control: Stabilization, optimality, and robustness," *IEEE Transactions on Automatic Control*, vol. 65, no. 3, pp. 909–924, 2019.

[22] A. Allibhoy and J. Cortés, "Data-based receding horizon control of linear network systems," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1207–1212, 2020.

[23] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Data-driven model predictive control with stability and robustness guarantees," *IEEE Transactions on Automatic Control*, vol. 66, no. 4, pp. 1702–1717, 2020.

[24] T. Dai and M. Sznaier, "Data-driven quadratic stabilization and lqr control of lti systems," *Automatica*, vol. 153, p. 111041, 2023.

[25] M. Rotulo, C. De Persis, and P. Tesi, "Data-driven linear quadratic regulation via semidefinite programming," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 3995–4000, 2020.

[26] G. R. G. da Silva, A. S. Bazanella, C. Lorenzini, and L. Campestrini, "Data-driven lqr control design," *IEEE control systems letters*, vol. 3, no. 1, pp. 180–185, 2018.

[27] V. Kedia and D. Chakraborty, "Randomized subspace identification for lti systems," in *2023 European Control Conference (ECC)*, pp. 1–6, IEEE, 2023.

[28] V. Kedia and D. Chakraborty, "Fast multivariable subspace identification (fmsid) of combined deterministic-stochastic/general lti systems for large input-output data," *arXiv preprint arXiv:2303.00994*, 2023.

[29] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013.

[30] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[31] S. J. Qin, W. Lin, and L. Ljung, "A novel subspace identification approach with enforced causal models," *Automatica*, vol. 41, no. 12, pp. 2043–2053, 2005.

[32] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[33] M. Verhaegen, "Identification of the deterministic part of mimo state space models given in innovations form from input-output data," *Automatica*, vol. 30, no. 1, pp. 61–74, 1994.