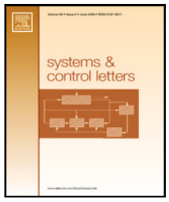
Systems & Control Letters 193 (2024) 105932

Contents lists available at ScienceDirect

Systems & Control Letters

journal homepage: www.elsevier.com/locate/sysconle



Data-driven control of nonlinear systems: An online sequential approach

Minh Vu [*,1], Yunshen Huang[1], Shen Zeng

Department of Electrical and Systems Engineering, Washington University in St. Louis, MO, USA

ARTICLE INFO

Keywords:
Model-based learning control
Iterative synthesis
Nonlinear dynamics
Neural networks
Physical experiments

ABSTRACT

While data-driven control has shown its potential for solving complex tasks, current algorithms such as reinforcement learning are still data-intensive and often limited to simulated environments. Model-based learning is a promising approach to reducing the amount of data required in practical implementations, yet it suffers from a critical issue known as model exploitation. In this paper, we present a sequential approach to model-based learning that avoids model exploitation and achieves stable system behaviors during learning with minimal exploration. The advocated control design utilizes estimates of the system's local dynamics to step-by-step improve the control. During the process, when additional data is required, the program pauses the control synthesis to collect data in the surrounding area and updates the model accordingly. The local and sequential nature of this approach is the key component to *regulating the system's exploration in the state–action space* and, at the same time, *avoiding the issue of model exploitation*, which are the main challenges in model-based learning control. Through simulated examples and physical experiments, we demonstrate that the proposed approach can quickly learn a desirable control from scratch, with just a small number of trials.

1. Introduction

In recent years, data-driven control approaches have shown great potential for solving complex tasks [1–6]. In this research area, model-based learning approaches are known to be more data-efficient compared to model-free learning [7–9]. Despite their benefits, model-based learning often suffers from an issue known as *model exploitation*, *i.e.*, an incorrect estimate of the system dynamics due to limited amounts of sample data, especially at the beginning of the learning process, would bias the control design toward an incorrect model and eventually make the algorithm diverge (see, *e.g.*, [10–12]). To this end, methods have been proposed, which in principle leverage probabilistic models such as Gaussian processes to mitigate overfitting and incorporate incorrect model estimates into long-term planning (see, *e.g.*, [13–15]). These probabilistic approaches, though effective in mitigating model exploitation, often lead to random and irregular behaviors of the system during (and even after) learning and thus may present challenges for practical implementations (see, *e.g.*, [16,17]).

In this paper, we present a sequential approach to model-based learning that avoids the above described model bias problem (without the use of a probabilistic model) and at the same time maintains well-regulated system behaviors during learning. To achieve that, we leverage an iterative control methodology that uses estimates of the system's *local* dynamics to step-by-step improve the control. This allows us to bypass the requirement of learning an accurate *"global"* model of the system and thus avoid the potential biases. To ensure the appropriateness of the currently estimated local dynamics, we evaluate the reliability of the model in terms of explored/unexplored regions (for which explicit conditions are provided). Using the model assessment metric, we subsequently alternate between model learning and the iterative control design. This sequential mechanism, *i.e.*, pausing the control and updating the model, along with the local nature of our control method, are the key components that allow us to avoid model exploitation.

In addition to preventing model exploitation, this sequential process will also tailor the acquisition of data to the purpose of control design. This integration, as one will see, naturally leads to more guided learning behaviors of the system and thus reduces excessive random exploration. We note that although similar terminologies, such as interleaving model training and data collection, have recently appeared in some learning papers from the robotics community [18–21], the question of the exact mechanism by which data collection and model learning should be combined with control design has not been investigated. In this paper, we study this issue in detail and provide a rigorous and systematic answer to this question.

The paper is organized as follows. In Section 2, we introduce the iterative control methodology and present its detailed theoretical analyses. In Section 3, we describe how to learn a system model from sampled data and integrate this online model learning into the

* Corresponding author.
E-mail address: minhvu@wustl.edu (M. Vu).
[1] The authors contributed equally to this work.

https://doi.org/10.1016/j.sysconle.2024.105932
Received 26 September 2023; Received in revised form 7 August 2024; Accepted 18 September 2024
Available online 10 October 2024

control design process. In Section 4, we illustrate the effectiveness of this sequential integrated control design through simulation of a benchmark control example and compare its efficiency with other learning-based techniques. To demonstrate its practicality, we also present experimental implementation of the proposed approach on a physical platform.

2. Iterative control design

In this section, we first give an overview of the iterative control methodology which was introduced in [22] to compute point-to-point steering of nonlinear control systems. We then provide detailed theoretical analyses of the iterative control that were not introduced in the initial work and discuss the key features which make the approach suitable for data-driven control settings.

2.1. Principles of iterative control synthesis

Consider steering a nonlinear control system

$$x_{k+1} = F(x_k, u_k) \tag{1}$$

from x_0 to x_{target}, where $x_k \in \mathbb{R}^n, u_k \in \mathbb{R}^m$, $F \in C^2(\mathbb{R}^{n+m}, \mathbb{R}^n)$, and $k \in \mathbb{N}_0$ is time index. We shall solve this problem through a sequence of iterations. Each iteration i begins with a nominal control $U^{(i)} = \begin{bmatrix} u_0^{(i)\top} & u_1^{(i)\top} & \dots & u_{N-1}^{(i)\top} \end{bmatrix}^\top$ and the resulting state trajectory $X^{(i)} = [x_1^{(i)\top} \quad x_2^{(i)\top} \quad \dots \quad x_N^{(i)\top}]^\top$ obtained from Eq. (1). For the first iteration, an arbitrary input can be used as a nominal control. We consider linearizing the system along the input-state trajectory, *i.e.*,

$$\delta x_{k+1}^{(i)} = A_k^{(i)} \delta x_k^{(i)} + B_k^{(i)} \delta u_k^{(i)}, \qquad \delta x_0^{(i)} = 0 \tag{2}$$

where $A_k^{(i)} := \frac{\partial F}{\partial x}(x_k^{(i)}, u_k^{(i)})$, $B_k^{(i)} := \frac{\partial F}{\partial u}(x_k^{(i)}, u_k^{(i)})$ are the derivatives of F with respect to the state and the input, respectively. By iterating (2), we have

$$\begin{aligned}
\delta x_1^{(i)} &= B_0^{(i)} \delta u_0^{(i)} \\
\delta x_2^{(i)} &= A_1^{(i)} B_0^{(i)} \delta u_0^{(i)} + B_1^{(i)} \delta u_1^{(i)} \\
&\vdots \\
\delta x_N^{(i)} &= A_{N-1}^{(i)} \dots A_1^{(i)} B_0^{(i)} \delta u_0^{(i)} + \dots + B_{N-1}^{(i)} \delta u_{N-1}^{(i)} \\
&= \underbrace{\begin{bmatrix} A_{N-1}^{(i)} \dots A_1^{(i)} B_0^{(i)} & \cdots & B_{N-1}^{(i)} \end{bmatrix}}_{=:\, H^{(i)}} \underbrace{\begin{bmatrix} \delta u_0^{(i)} \\ \vdots \\ \delta u_{N-1}^{(i)} \end{bmatrix}}_{=:\, \Delta U^{(i)}}.
\end{aligned}$$

Now, our approach is to utilize the above expression to determine a small suitable $\Delta U^{(i)}$ such that $U^{(i+1)} = U^{(i)} + \Delta U^{(i)}$ will steer $x_N^{(i+1)}$ a bit closer to x_{target}. This incremental steering process is repeated for a number of iterations until x_N approaches x_{target}. More specifically, in each iteration i, we consider the following optimization problem

$$\underset{\Delta U^{(i)}}{\text{minimize}} \quad \|x_N^{(i)} + H^{(i)} \Delta U^{(i)} - x_{\text{target}}\|^2 + \lambda^{(i)} \|\Delta U^{(i)}\|^2 \tag{3}$$

where $\|\cdot\|$ denotes the 2-norm (which will be used throughout this paper), and $\lambda^{(i)} \geq 0$ is a regularization parameter that enforces a penalty on the magnitude of $\Delta U^{(i)}$ to ensure a sufficiently incremental update in each iteration, which guarantees the appropriateness of the above linearization of the system. For the quadratic program (3), we have the following explicit solution

$$\Delta U_*^{(i)} = -(H^{(i)\top} H^{(i)} + \lambda^{(i)} I)^{-1} H^{(i)\top} (x_N^{(i)} - x_{\text{target}}). \tag{4}$$

Regarding the choice of $\lambda^{(i)}$, our approach is to keep computing $\Delta U_*^{(i)}$ with increasing values of $\lambda^{(i)}$ until $J(U^{(i)} + \Delta U_*^{(i)}) \leq J(U^{(i)})$, where $J(U^{(i)}) := \|x_N(U^{(i)}) - x_{\text{target}}\|^2$. When the condition is satisfied, we update the input by $U^{(i+1)} = U^{(i)} + \Delta U_*^{(i)}$ and proceed to the next iteration with a slightly reduced $\lambda^{(i+1)}$.

This (adaptive) mechanism allows us to keep increasing $\lambda^{(i)}$ so as to enforce a strict penalty on the magnitude of $\Delta U_*^{(i)}$ and recompute $\Delta U_*^{(i)}$ until the updated control steers the system closer to the target. Once the condition is satisfied, the control input is updated, and $\lambda^{(i+1)}$ is slightly decreased in the subsequent iteration, which gradually relaxes the penalty on the magnitude of $\Delta U_*^{(i+1)}$ and results in a larger control update and more progress toward the final control solution. The iterative scheme for steering the system to the desired target is implemented as follows.

Algorithm 1 Steering the system to a desired target

Require: A desired state x_{target}, an initial (arbitrary) input $U^{(i=0)}$, a regularizer $\lambda^{(i=0)} \geq 0$, an adaptive rate $1 > \alpha > 0$, and the cost $J(U^{(i)}) = \|x_N(U^{(i)}) - x_{\text{target}}\|^2$.

1: Apply input $U^{(i)}$ to the system and store $A_k^{(i)}, B_k^{(i)}$.
2: Calculate $H^{(i)}$.
3: Compute the control update $\Delta U_*^{(i)}$ using (4).
4: If $J(U^{(i)} + \Delta U_*^{(i)}) \leq J(U^{(i)})$, set $\lambda^{(i+1)} = (1-\alpha)\lambda^{(i)}$.
Else set $\lambda^{(i)} = (1+\alpha)\lambda^{(i)}$ and come back to step 3.
5: Update the control input via $U^{(i+1)} = U^{(i)} + \Delta U_*^{(i)}$ and proceed to the next iteration $i = i + 1$.
6: Repeat step $1-5$ until $\|x_N - x_{\text{target}}\|^2 \leq \epsilon_{\text{tol}}$.

In summary, the above approach strategically transforms the steering problem into an iterative sequence of quadratic programs (3). If the system model is available, we apply Algorithm 1 to steer the system to the desired target. If the model is not available, it can be learned from data and integrated to the iterative control (which will be presented in Section 3). We note that control constraints (*e.g.* actuation limits and state constraints) can also be incorporated in the above approach by including the constraints into the quadratic program [23,24]. The approach can also be considered for optimal control design [25,26].

2.2. Theoretical analysis of iterative control

This subsection provides theoretical analyses of the presented iterative control. Throughout this subsection, we make the following assumptions.

Assumption 2.1. (i) Let $\mathscr{D}_u$ be a set of all possible control $U^{(i)}$. Suppose $\mathscr{D}_u$ is closed and bounded. (ii) Suppose the local linearization (2) of the original control system is N-step controllable with bounded degrees of controllability. It is equivalent that the controllability matrix, which is the $H^{(i)}$ matrix, being full rank and $0 < \underline{\sigma} \leq \sigma_{\min}(H^{(i)}), \forall i \in \mathbb{N}_+$ [27].

Lemma 2.2. *For each iteration i, there exists a $\lambda^{(i)}$ such that for any $U^{(i)} \in \mathscr{D}_u$, $J(U^{(i)} + \Delta U_*^{(i)}) \leq J(U^{(i)})$ where $\Delta U_*^{(i)}$ is the solution of (3). Moreover, $\lambda^{(i)}$ can be uniformly bounded for all iterations.*

Proof. First, we have $J(U^{(i)}) = \|x_N(U^{(i)}) - x_{\text{target}}\|^2$ where $x_N(U^{(i)})$ is calculated by iterating (1), *i.e.*,

$$\begin{aligned}
x_N(U^{(i)}) &= F(x_{N-1}^{(i)}, u_{N-1}^{(i)}) = F(F(x_{N-2}^{(i)}, u_{N-2}^{(i)}), u_{N-1}^{(i)}) \\
&= \dots = F(\dots, F(F(x_0^{(i)}, u_0^{(i)}), u_1^{(i)}), \dots, u_{N-1}^{(i)}).
\end{aligned}$$

By following the chain rule, we can verify that the formulation of $H^{(i)}$ is indeed the first derivative of $x_N(U^{(i)})$ with respect to the control input, *i.e.*,

$$\begin{aligned}
\left.\frac{dx_N}{dU}\right|_{U^{(i)}} &= \begin{bmatrix} \left.\frac{\partial x_N}{\partial u_0}\right|_{U^{(i)}} & \left.\frac{\partial x_N}{\partial u_1}\right|_{U^{(i)}} & \cdots & \left.\frac{\partial x_N}{\partial u_{N-1}}\right|_{U^{(i)}} \end{bmatrix} \\
&= \begin{bmatrix} A_{N-1}^{(i)} \dots A_1^{(i)} B_0^{(i)} & A_{N-1}^{(i)} \dots A_2^{(i)} B_1^{(i)} & \cdots & B_{N-1}^{(i)} \end{bmatrix} \\
&= H^{(i)}
\end{aligned}$$

with $A_k^{(i)} = \frac{\partial F}{\partial x}(x_k^{(i)}, u_k^{(i)})$, $B_k^{(i)} = \frac{\partial F}{\partial u}(x_k^{(i)}, u_k^{(i)})$, $k = 0, \dots, N-1$. Additionally, since $F \in C^2(\mathbb{R}^{n+m}, \mathbb{R}^n)$, $x_N(\cdot)$ is twice continuously differentiable in U. Since $\mathscr{D}_u$ is closed and bounded (Assumption 2.1), the first and second derivative of x_N is bounded on $\mathscr{D}_u$. Thus, for any $U^{(i)} \in \mathscr{D}_u$, we have $\|H^{(i)}\| \le \bar{H}$ and $|\partial_j^2 x_N(U^{(i)})| \le M$ where $\partial_j^2 x_N(U^{(i)})$ denote the coordinate-wise second derivatives of x_N.

Note that from a standard analysis of ridge regression, *e.g.*, see Section 3.4 in [28], $\Delta U_*^{(i)} \to 0$ as $\lambda^{(i)} \to \infty$. Now, by applying the first-order Taylor expansion to $x_N(U^{(i)})$, we have

$$\begin{aligned} J(U^{(i)} + \Delta U_*^{(i)}) &= \|x_N(U^{(i)}) + H^{(i)}\Delta U_*^{(i)} + R(\Delta U_*^{(i)}) - x_{\text{target}}\|^2 \\ &\le \|x_N(U^{(i)}) + H^{(i)}\Delta U_*^{(i)} - x_{\text{target}}\|^2 + \|R(\Delta U_*^{(i)})\|^2 \\ &\quad + 2\|x_N(U^{(i)}) + H^{(i)}\Delta U_*^{(i)} - x_{\text{target}}\|\,\|R(\Delta U_*^{(i)})\| \end{aligned} \tag{5}$$

where $R(\Delta U_*^{(i)}) := [r_1(\Delta U_*^{(i)}), \dots, r_n(\Delta U_*^{(i)})]^\top$, r_j denotes the coordinate-wise residuals of the Taylor expansion. From multivariate Taylor's theorem (Section 2.4 in [29]), each residual can be upper bounded by $|r_j(\Delta U_*^{(i)})| \le M\|\Delta U_*^{(i)}\|^2$. Thus, $\|R(\Delta U_*^{(i)})\|$ can also be upper bounded, *i.e.*,

$$\|R(\Delta U_*^{(i)})\|^2 \le \sum_{j=1}^{n}\left(M\|\Delta U_*^{(i)}\|^2\right)^2 = nM^2\|\Delta U_*^{(i)}\|^4. \tag{6}$$

Incorporating (6) into (5), we have

$$\begin{aligned} J(U^{(i)} + \Delta U_*^{(i)}) \le \|x_N(U^{(i)}) + H^{(i)}\Delta U_*^{(i)} - x_{\text{target}}\|^2 + nM^2\|\Delta U_*^{(i)}\|^4 \\ + 2\|x_N(U^{(i)}) + H^{(i)}\Delta U_*^{(i)} - x_{\text{target}}\|\sqrt{n}M\|\Delta U_*^{(i)}\|^2. \end{aligned} \tag{7}$$

Since $\Delta U_*^{(i)} \to 0$ as $\lambda^{(i)} \to \infty$, there exists $\lambda^{(i)}$ such that

$$J(U^{(i)} + \Delta U_*^{(i)}) \le \underbrace{\|x_N(U^{(i)}) + H^{(i)}\Delta U_*^{(i)} - x_{\text{target}}\|^2 + \lambda^{(i)}\|\Delta U_*^{(i)}\|^2}_{=:\, V_{\Delta U_*^{(i)}, \lambda^{(i)}}}$$

Since $V_{\Delta U_*^{(i)}, \lambda^{(i)}}$ is indeed the optimal objective of (3), it yields a lower value compared to that of $\Delta U^{(i)} = 0$, *i.e.*, $V_{\Delta U_*^{(i)}, \lambda^{(i)}} \le V_{\Delta U^{(i)}=0, \lambda^{(i)}} = J(U^{(i)})$. Thus, $J(U^{(i)} + \Delta U_*^{(i)}) \le J(U^{(i)})$.

Finally, we show that the required threshold for $\lambda^{(i)}$ can be upper bounded. Specifically, from (7), we have

$$\begin{aligned} \mathscr{R}^{(i)} &:= nM^2\|\Delta U_*^{(i)}\|^2 + 2\sqrt{n}M\|x_N(U^{(i)}) + H^{(i)}\Delta U_*^{(i)} - x_{\text{target}}\| \\ &\le nM^2\|\Delta U_*^{(i)}\|^2 + 2\sqrt{n}M(\|x_N(U^{(i)}) - x_{\text{target}}\| + \|H^{(i)}\Delta U_*^{(i)}\|) \\ &\le nM^2\sigma_{\min}^4(H^{(i)})\,\sigma_{\max}^2(H^{(i)})\|x_N(U^{(i)}) - x_{\text{target}}\|^2 \\ &\quad + 2\sqrt{n}M(1 + \sigma_{\min}^2(H^{(i)})\,\sigma_{\max}^2(H^{(i)}))\|x_N(U^{(i)}) - x_{\text{target}}\| \end{aligned}$$

where the last inequality is due to the fact that $\|\Delta U_*^{(i)}\| \le \sigma_{\min}^2(H^{(i)})\,\sigma_{\max}(H^{(i)})\,\|x_N(U^{(i)}) - x_{\text{target}}\|$ (see Lemma A.2 for more details). Note that $\sigma_{\min}(H^{(i)}) \le \sigma_{\max}(H^{(i)}) \le \bar{H}$. Also, since $\|x_N(U^{(i)}) - x_{\text{target}}\|$ decreases in each iteration, we have $\|x_N(U^{(i)}) - x_{\text{target}}\| \le d_0$ where d_0 denotes the distance in the first iteration. Thus, we can upper bound the required threshold for $\lambda^{(i)}$ for all iterations, *i.e.*, $\mathscr{R}^{(i)} \le nM^2\bar{H}^6 d_0^2 + 2\sqrt{n}M(1 + \bar{H}^4)d_0$. □

Lemma 2.3. *Let $A \in \mathbb{R}^{p\times p}$ be an invertible matrix. For $\beta > 0$, the two following conditions are equivalent: (i) $y^\top Ay \ge \beta\|Ay\|^2, \forall y \in \mathbb{R}^p$. (ii) $y^\top A^{-1}y \ge \beta\|y\|^2, \forall y \in \mathbb{R}^p$.*

Proof. See example 22.7 in [30] for details. □

Theorem 2.4. *The iterative control synthesis converges, and the system is steered to the target, i.e., $x_N(U^{(i)}) \to x_{target}$ as $i \to \infty$.*

Proof. From Lemma 2.2, we have $J(U^{(i)}+\Delta U_*^{(i)}) \le J(U^{(i)})$. Since $J(U^{(i)})$ is bounded below, the sequence of $\{J(U^{(i)})\}$ generated by Algorithm 1 is a convergent sequence. It implies that $J(U^{(i)}) - J(U^{(i)} + \Delta U_*^{(i)}) \to 0$. On the other hand, from $J(U^{(i)} + \Delta U_*^{(i)}) \le V_{\Delta U_*^{(i)}, \lambda^{(i)}}$, we have

$$0 \le J(U^{(i)}) - V_{\Delta U_*^{(i)}, \lambda^{(i)}} \le J(U^{(i)}) - J(U^{(i)} + \Delta U_*^{(i)}),$$

where the first inequality is because the optimal objective of (3), *i.e.*, $V_{\Delta U_*^{(i)}, \lambda^{(i)}}$, yields a lower value compared to that of $\Delta U^{(i)} = 0$, *i.e.*, $V_{\Delta U_*^{(i)}, \lambda^{(i)}} \le V_{\Delta U^{(i)}=0, \lambda^{(i)}} = J(U^{(i)})$. Thus, as $i \to \infty$, $J(U^{(i)}) - V_{\Delta U_*^{(i)}, \lambda^{(i)}} \to 0$. Since (3) has a unique solution because of its strict convexity, it implies that $\Delta U_*^{(i)} \to 0$ as $i \to \infty$.

Note that from the proof of Lemma 2.2 we have $\sigma_{\max}(H^{(i)}) = \|H\| \le \bar{H} =: \bar{\sigma}$. Then, for any $y \in \mathbb{R}^{mN}$, we have

$$y^\top(\lambda^{(i)}I + {H^{(i)}}^\top H^{(i)})y = \lambda^{(i)}\|y\|^2 + \|H^{(i)}y\|^2 \ge (\lambda^{(i)} + \underline{\sigma}^2)\|y\|^2$$

and

$$\|(\lambda^{(i)}I + {H^{(i)}}^\top H^{(i)})y\| \le \lambda^{(i)}\|y\| + \bar{\sigma}^2\|y\| = (\lambda^{(i)} + \bar{\sigma}^2)\|y\|.$$

Combining the two equations, we have

$$y^\top(\lambda^{(i)}I + {H^{(i)}}^\top H^{(i)})y \ge \underbrace{\frac{\lambda^{(i)} + \underline{\sigma}^2}{(\lambda^{(i)} + \bar{\sigma}^2)^2}}_{=:\, \beta^{(i)}}\|(\lambda^{(i)}I + {H^{(i)}}^\top H^{(i)})y\|^2.$$

From Lemma 2.3, we have $\beta^{(i)}\|y\|^2 \le y^\top(\lambda^{(i)}I + {H^{(i)}}^\top H^{(i)})^{-1}y$, for any $y \in \mathbb{R}^{mN}$. Thus, it is true for $y = {H^{(i)}}^\top(x_{\text{target}} - x_N(U^{(i)}))$, *i.e.*,

$$\beta^{(i)}\|y\|^2 \le y^\top(\lambda^{(i)}I + {H^{(i)}}^\top H^{(i)})^{-1}y = y^\top\Delta U_*^{(i)} \le \|y\|\,\|\Delta U_*^{(i)}\|$$

where the equality is due to (4). Thus, we have

$$\|\Delta U_*^{(i)}\| \ge \beta^{(i)}\|y\| \ge \beta^{(i)}\underline{\sigma}\|x_{\text{target}} - x_N(U^{(i)})\|. \tag{8}$$

Note that since $\lambda^{(i)}$ is upper bounded (as shown in the proof of Lemma 2.2), $\beta^{(i)}$ is lower bounded. Thus, as $i \to \infty$, $\Delta U_*^{(i)} \to 0$ and $x_N(U^{(i)}) \to x_{\text{target}}$. □

Remark. Theorem 2.4 relies on the controllability assumption of (2). Note that this assumption may not be satisfied in general. However, for certain problems where the model of the system is available, one could use the model to analyze the required condition analytically. For example, consider a simple control system

$$\begin{aligned} x_{k+1} &= x_k + \frac{1}{2}u_k^2 \\ y_{k+1} &= y_k + u_k. \end{aligned}$$

The linearization of the system at each trajectory pairs $(x_k^{(i)}, u_k^{(i)})$ is $A_k^{(i)} = I, B_k^{(i)} = [u_k^{(i)}, 1]^\top$. Therefore, the N-step controllability matrix

$$H^{(i)} = \begin{bmatrix} u_0^{(i)} & u_1^{(i)} & \cdots & u_{N-1}^{(i)} \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

could remain full rank as long as the control inputs are not identical (*i.e.*, $u_0^{(i)} = u_1^{(i)} = \cdots = u_{N-1}^{(i)}$). Note that the controllability assumption, in the context of data-driven control, could also be verified numerically, by calculating the rank (or singular values) of $H^{(i)}$. Since $H^{(i)}$ is a wide matrix with significantly more columns than rows, in our implementations, we observe that the condition is often satisfied.

2.3. Analysis of iterative control in data-driven control settings

In practice, we may not have the access to the system model to compute $A_k^{(i)}$, $B_k^{(i)}$, and $H^{(i)}$, and thus have to estimate $\hat{A}_k^{(i)}$, $\hat{B}_k^{(i)}$, and $\hat{H}^{(i)}$, from data. Therefore, it is important to evaluate the effect of estimation errors on the above presented results. To study this aspect, we use the "hat" notation to denote quantities associated with the estimation and consider the following assumptions.

Assumption 2.5. Suppose $\|\hat{H}^{(i)} - H^{(i)}\| \leq \frac{1}{4}\sigma_{\min}(\hat{H}^{(i)})$ and $0 < \underline{\hat{\sigma}} \leq \sigma_{\min}(\hat{H}^{(i)}) \leq \sigma_{\max}(\hat{H}^{(i)}) = \|\hat{H}^{(i)}\| \leq \bar{r}$ for all iterations.

Lemma 2.6. *Let $\hat{U}^{(i)}$ be the input at iteration i, computed from the estimated $\hat{A}_k^{(i)}$, $\hat{B}_k^{(i)}$, $\hat{H}^{(i)}$, and $\Delta\hat{U}_*^{(i)}$ be the solution of (3) where the estimated $\hat{H}^{(i)}$ is used instead of $H^{(i)}$. Then, for each iteration i, there exists $\hat{\lambda}^{(i)}$ such that $J(\hat{U}^{(i)} + \Delta\hat{U}_*^{(i)}) \leq J(\hat{U}^{(i)})$.*

Proof. Let $\Delta H^{(i)} := H^{(i)} - \hat{H}^{(i)}$. Similar to the proof of Lemma 2.2, we consider the first-order Taylor expansion of $x_N(\hat{U}^{(i)})$, *i.e.*,

$$\begin{aligned} J(\hat{U}^{(i)} + \Delta\hat{U}_*^{(i)}) &= \|x_N(\hat{U}^{(i)}) + H^{(i)}\Delta\hat{U}_*^{(i)} + R(\Delta\hat{U}_*^{(i)}) - x_{\text{target}}\|^2 \\ &= \|x_N(\hat{U}^{(i)}) + (\hat{H}^{(i)} + \Delta H^{(i)})\Delta\hat{U}_*^{(i)} + R(\Delta\hat{U}_*^{(i)}) - x_{\text{target}}\|^2 \\ &\leq \|x_N(\hat{U}^{(i)}) + \hat{H}^{(i)}\Delta\hat{U}_*^{(i)} - x_{\text{target}}\|^2 + \|\Delta H^{(i)}\Delta\hat{U}_*^{(i)} + R(\Delta\hat{U}_*^{(i)})\|^2 \\ &\quad + 2\|x_N(\hat{U}^{(i)}) + \hat{H}^{(i)}\Delta\hat{U}_*^{(i)} - x_{\text{target}}\|\,\|\Delta H^{(i)}\Delta\hat{U}_*^{(i)} + R(\Delta\hat{U}_*^{(i)})\|. \end{aligned} \tag{9}$$

From Assumption 2.5 and (6), we have

$$\|\Delta H^{(i)}\Delta\hat{U}_*^{(i)} + R(\Delta\hat{U}_*^{(i)})\| \leq \frac{1}{4}\sigma_{\min}(\hat{H}^{(i)})\|\Delta\hat{U}_*^{(i)}\| + \sqrt{n}M\|\Delta\hat{U}_*^{(i)}\|^2. \tag{10}$$

Substituting (10) into (9) and simplifying the expression, we have

$$\begin{aligned} J(\hat{U}^{(i)} + \Delta\hat{U}_*^{(i)}) \leq \|x_N(\hat{U}^{(i)}) + \hat{H}^{(i)}\Delta\hat{U}_*^{(i)} - x_{\text{target}}\|^2 \\ + c_1^{(i)}\|\Delta\hat{U}_*^{(i)}\| + c_2^{(i)}\|\Delta\hat{U}_*^{(i)}\|^2 \end{aligned} \tag{11}$$

where $c_1^{(i)} = \frac{1}{2}\sigma_{\min}(\hat{H}^{(i)})\|x_N(\hat{U}^{(i)}) - x_{\text{target}}\|$, and $c_2^{(i)} = \frac{1}{2}\sigma_{\min}(\hat{H}^{(i)})\|\hat{H}^{(i)}\| + 2\sqrt{n}M\|x_N(\hat{U}^{(i)}) + \hat{H}^{(i)}\Delta\hat{U}_*^{(i)} - x_{\text{target}}\| + (\frac{1}{4}\sigma_{\min}(\hat{H}^{(i)}) + \sqrt{n}M\|\Delta\hat{U}_*^{(i)}\|)^2$. Now, since from (4)

$$\hat{\lambda}^{(i)}\|\Delta\hat{U}_*^{(i)}\| = \|(I + \hat{\lambda}^{(i)^{-1}}\hat{H}^{(i)^\top}\hat{H}^{(i)})^{-1}\hat{H}^{(i)^\top}(x_{\text{target}} - x_N(\hat{U}^{(i)}))\|,$$

we have $\hat{\lambda}^{(i)}\|\Delta\hat{U}_*^{(i)}\| \to \|\hat{H}^{(i)^\top}(x_{\text{target}} - x_N(\hat{U}^{(i)}))\|$ as $\hat{\lambda}^{(i)} \to \infty$. Thus, there exists $\hat{\lambda}_1^{(i)}$ such that

$$\begin{aligned} \hat{\lambda}_1^{(i)}\|\Delta\hat{U}_*^{(i)}\| &\geq \frac{1}{2}\|\hat{H}^{(i)^\top}(x_{\text{target}} - x_N(\hat{U}^{(i)}))\| \\ &\geq \frac{1}{2}\sigma_{\min}(\hat{H}^{(i)})\|x_{\text{target}} - x_N(\hat{U}^{(i)})\| = c_1^{(i)}. \end{aligned}$$

Then, for any $\hat{\lambda}^{(i)} \geq \hat{\lambda}_1^{(i)} + c_2^{(i)}$, we have $\hat{\lambda}^{(i)}\|\Delta\hat{U}_*^{(i)}\|^2 \geq \hat{\lambda}_1^{(i)}\|\Delta\hat{U}_*^{(i)}\|^2 + c_2^{(i)}\|\Delta\hat{U}_*^{(i)}\|^2 \geq c_1^{(i)}\|\Delta\hat{U}_*^{(i)}\| + c_2^{(i)}\|\Delta\hat{U}_*^{(i)}\|^2$, which is incorporated into (11) as

$$J(U^{(i)} + \Delta\hat{U}_*^{(i)}) \leq \|x_N(\hat{U}^{(i)}) + \hat{H}^{(i)}\Delta\hat{U}_*^{(i)} - x_{\text{target}}\|^2 + \hat{\lambda}^{(i)}\|\Delta\hat{U}_*^{(i)}\|.^2$$

Since the above right hand side is the optimal objective $V_{\Delta\hat{U}_*^{(i)},\hat{\lambda}^{(i)}}$ of (3), it yields a lower value compared to that of $\Delta\hat{U}^{(i)} = 0$, *i.e.*, $V_{\Delta\hat{U}_*^{(i)},\hat{\lambda}^{(i)}} \leq V_{\Delta\hat{U}^{(i)}=0,\hat{\lambda}^{(i)}} = J(\hat{U}^{(i)})$. Thus, $J(\hat{U}^{(i)} + \Delta\hat{U}_*^{(i)}) \leq J(\hat{U}^{(i)})$. □

Theorem 2.7. *The iterative control synthesis using the estimated dynamics $\hat{H}^{(i)}$ converges, and the system is steered to the desired target.*

Proof. Given Assumption 2.5 and Lemma 2.6, the proof is followed from the proof of Theorem 2.4. □

3. Integration of model learning and control

In this section, we first describe our approach to learning a system model from data. We then present the integration of model learning and control design.

3.1. Learning the system's dynamics from data

Consider a discrete-time nonlinear control system

$$x_{k+1} = F(x_k, u_k) \tag{12}$$

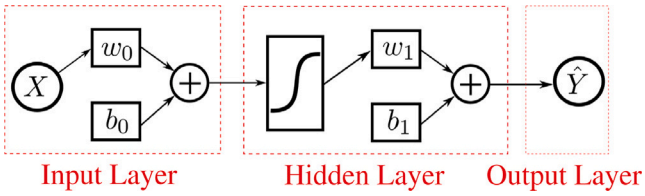


Fig. 1. Schematic diagram of a single-layer feedforward neural network where $X \in \mathbb{R}^{n_x}$ and $\hat{Y} \in \mathbb{R}^{n_y}$ represent the input and output, respectively, and $w_0 \in \mathbb{R}^{n_n \times n_x}$, $w_1 \in \mathbb{R}^{n_y \times n_n}$, $b_0 \in \mathbb{R}^{n_n}$, $b_1 \in \mathbb{R}^{n_y}$ are the weight matrices and bias vectors for the hidden and output layers, respectively.

where $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$, and $k \in \mathbb{N}_0$ denotes time index. We consider that the states of the system are fully observable and collect their measurements at each time step. We use $D_{\text{training}}^{(\text{In})} := \{[x_k^{(i)^\top}, u_k^{(i)^\top}]^\top\}$, *i.e.*, the recorded state and control at each time step, as an input to the learning model and $D_{\text{training}}^{(\text{Out})} := \{x_{k+1}^{(i)}\}$ as the corresponding output. To capture the system dynamics F (in the region of the training data), we consider the following model assumption.

Assumption 3.1. For any $\epsilon > 0$, there exists $\delta > 0$ such that if the training data is δ-sufficiently dense on a domain $\mathscr{D}$ (*i.e.*, for any $[x^\top, u^\top]^\top \in \mathscr{D}$, there is $[x_k^{(i)^\top}, u_k^{(i)^\top}]^\top \in D_{\text{training}}^{(\text{In})}$ such that $\|[x^\top, u^\top]^\top - [x_k^{(i)^\top}, u_k^{(i)^\top}]^\top\| \leq \delta$), then the F_{model} obtained from the training process has an ϵ-accuracy (*i.e.*, for any $[x^\top, u^\top]^\top \in \mathscr{D}$, $\|F(x,u) - F_{\text{model}}(x,u)\| \leq \epsilon$, $\|\partial_x F(x,u) - \partial_x F_{\text{model}}(x,u)\| \leq \epsilon$, and $\|\partial_u F(x,u) - \partial_u F_{\text{model}}(x,u)\| \leq \epsilon$, where ∂_x and ∂_u denote the derivatives with respect to x and u).

Assumption 3.1 is based on recent advances in deep learning theory, which indicates that a deep learning model can accurately learn the dynamic function F up to a desired precision if the training data is sufficiently dense (for more details, see new results on generalization bound analysis of deep neural networks [31,32]). We will use the assumption to characterize the model accuracy with respect to the density of the training data. The denser the training data, the more accurate the learning model becomes. The assumption will later be utilized (in Section 3.3) to choose the exploration threshold to ensure the desired density of the training data and the convergence of the overall control design.

Neural networks are known to be universal function approximators, for which Assumption 3.1 can be satisfied [33]. In our implementation, we use a standard structure of a fully-connected feedforward neural network provided by the MATLAB Machine Learning Toolbox [34]. By properly training the model, we capture F, the evolution of the system, in the neural network's forward propagation, which will later be used for prediction and control.

In addition to the computation of F, our control method (as introduced in Section 2) requires repetitive calculations of the system dynamics' derivatives $\partial F/\partial x$ and $\partial F/\partial u$. Instead of approximating these derivatives using finite differences, which potentially requires many interactions with the system, we compute the Jacobian of F symbolically from the learned model to minimize system interaction and the computational cost of the overall control design.

3.2. Symbolic computation of the local dynamics

To better illustrate the idea, we start with a single-layer feedforward neural network, as in Fig. 1. The equations of the forward propagation are

$$\begin{aligned} Y_0 &= w_0 X + b_0 \\ \hat{Y} &= w_1 g(Y_0) + b_1. \end{aligned} \tag{13}$$

The activation function utilized in the hidden layer is the tansig function, *i.e.*, $g(z) = \frac{2}{1+e^{-2z}} - 1$, whose derivative is $\frac{d}{dz}g(z) = 1 - g^2(z)$. Now,

Table 1
CPU time (in seconds) for one iteration of the iterative control synthesizing process.

System dimension	Finite differences	Symbolic computation	Improvement ratio
3	0.18	0.08	2.25
5	0.92	0.12	7.67
16	13.51	0.79	17.10

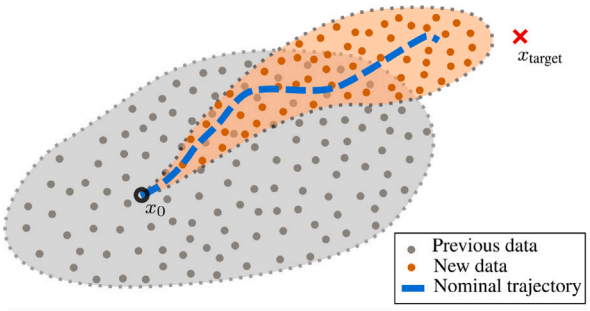


Fig. 2. Local exploration and acquisition of data. The process is conducted by collecting (sufficiently dense) data around a trajectory that extends outside the explored region, resulting in a guided expansion of the data cloud toward the target state.

we consider the neural network as a generic map $L : X \mapsto \hat{Y}$, and denote the derivative of L with respect to the input as $\frac{dL}{dX}$. By applying the chain rule to (13), we have

$$\begin{aligned}\left.\frac{dL}{dX}\right|_X &= \left.\frac{d\hat{Y}}{dg}\right|_{g(Y_0)} \cdot \left.\frac{dg}{dY_0}\right|_{Y_0} \cdot \left.\frac{dY_0}{dX}\right|_X \\ &= w_1 \cdot \operatorname{diag}\left(1 - g^2(Y_0)\right) \cdot w_0\end{aligned} \tag{14}$$

where $\operatorname{diag}(v)$ is a diagonal matrix whose entries are elements of vector v. This idea can be readily extended to a general multi-layer perceptron (MLP). More specifically, for a neural network with h numbers of hidden layers, the forward propagation is

$$\begin{aligned}Y_0 &= w_0 X + b_0 \\ Y_1 &= w_1 g(Y_0) + b_1 \\ &\vdots \\ \hat{Y} &= w_h g(Y_{h-1}) + b_h\end{aligned}$$

where w_i and b_i for $i \in \{1, \dots, h\}$ are the weight matrix and the bias vector for the ith hidden layer, respectively. Similar to the process in (14), the derivative of L with respect to the input of a MLP is computed as follows.

$$\begin{aligned}\left.\frac{\partial L}{\partial X}\right|_X &= \left.\frac{\partial \hat{Y}}{\partial g}\right|_{g(Y_{h-1})} \cdot \left.\frac{\partial g}{\partial Y_{h-1}}\right|_{Y_{h-1}} \cdot \\ &\quad \cdots \left.\frac{\partial Y_1}{\partial g}\right|_{g(Y_0)} \cdot \left.\frac{\partial g}{\partial Y_0}\right|_{Y_0} \cdot \left.\frac{\partial Y_0}{\partial X}\right|_X \\ &= w_h \cdot \operatorname{diag}\left(1 - g(Y_{h-1})^2\right) \cdot \\ &\quad \cdots w_1 \cdot \operatorname{diag}\left(1 - g(Y_0)^2\right) \cdot w_0 \\ &= \left(\prod_{i=h}^{1} w_i \cdot \operatorname{diag}\left(1 - g^2(Y_{i-1})\right)\right) \cdot w_0.\end{aligned} \tag{15}$$

The presented approach allows us to significantly reduce the number of system interactions and the computational cost as compared to computing the derivative using finite differences. Table 1 illustrates the computational gains across three test examples (*i.e.*, a simple pendulum, a pendulum on a cart, and a 3-dimensional quadcopter), highlighting the added benefits when applying to more complex systems.

3.3. Integrating model learning to control design

In this subsection, we describe the sequential integration of model learning and control design. To explore the environment and learn the model in a controlled manner, our approach is to integrate data collection and model learning into the control design and let the control design actually guide the data collection and model learning. More specifically, we first excite the system around an initial state and collect sufficiently dense data. Under Assumption 3.1, we can capture the system dynamics in the local region. Then, using the estimated dynamics, we employ the iterative control (as introduced in Section 2) to steer the system closer to the target. Whenever the system trajectory starts departing from the explored region, we execute a new round of explorations to locally inspect the unexplored region and better capture the overall dynamics, as illustrated in Fig. 2. This sequential process is then repeated until the system reaches the target, which is summarized in Algorithm 2 and is implemented through the following three steps.

1. (S1) **Local exploration:** Given a nominal input $U^{(i)}$, we consider applying to the system a slightly perturbed test input, *i.e.*, $\tilde{U}^{(i)} := U^{(i)} + R_N^{(i)}$ where $R_N^{(i)}$ denotes a small random input of N time steps. Then, for each time step, we store the states and the corresponding input values obtained from the perturbed trajectory and form a new data set $D_{\text{new}} = \{D_{\text{new}}^{(\text{In})}; D_{\text{new}}^{(\text{Out})}\}$ where $D_{\text{new}}^{(\text{In})} = [\tilde{x}_k^{(i)\top}, \tilde{u}_k^{(i)\top}]^\top$ and $D_{\text{new}}^{(\text{Out})} = \tilde{x}_{k+1}^{(i)}$. To sufficiently explore the local area, we repeat this process (for K times) until D_{new} is sufficiently dense. Note that the exploratory trajectories all begin at x_0 and remain close to the nominal trajectory.
2. (S2) **Learning local dynamics:** We combine the newly collected data D_{new} together with the previous data D_{prev} to form a training dataset $D_{\text{training}} = D_{\text{new}} \cup D_{\text{prev}}$. Then, we train a learning model (as described in Section 3.1) to better capture the system's dynamics on the combined explored region.
3. (S3) **Optimizing control inputs:** Given the nominal input $U^{(i)}$ and the newly estimated dynamics, we apply Algorithm 1 to steer the system step-be-step closer to x_{target}. In each iteration, to measure the distance between a current trajectory and the explored region, we consider the maximum (Hausdorff) distance from each point $[x_k^{(i)\top}, u_k^{(i)\top}]^\top$ of the trajectory to the training dataset, *i.e.*,

$$\mathbb{d}_{\text{extend}}^{(i)} := \max_{k=0,1,\dots,N-1} d([x_k^{(i)\top}, u_k^{(i)\top}]^\top, D_{\text{training}}^{(\text{In})}).$$

If the extended distance $\mathbb{d}_{\text{extend}}^{(i)}$ is greater than a certain preset threshold $\mathbb{d}_{\text{threshold}}$, it indicates that the current trajectory is about to depart from the explored region, and as a result, further data acquisition is required. Note that for the purpose of theoretical analysis, $\mathbb{d}_{\text{threshold}} = \delta$ can be chosen to ensure D_{training} to be sufficiently dense. Additionally, to ensure that the current estimate of the system's dynamics is accurate for the control design even in the explored region (due to potential overfittings), we also monitor the prediction error of the model, *i.e.*,

$$\mathbb{e}_{\text{model}}^{(i)} := \max_{k=0,1,\dots,N-1} \|F(x_k^{(i)}, u_k^{(i)}) - F_{\text{model}}(x_k^{(i)}, u_k^{(i)})\|^2$$

where $F_{\text{model}}(x_k^{(i)}, u_k^{(i)})$ are the model predictions and $F(x_k^{(i)}, u_k^{(i)})$ are the real data collected online. If $\mathbb{d}_{\text{extend}}^{(i)} > \mathbb{d}_{\text{threshold}}$ or $\mathbb{e}_{\text{model}}^{(i)} > \mathbb{e}_{\text{threshold}}$, we stop the control synthesizing process and start the next exploration **S1**.

Assumption 3.2. (i) Since $F \in C^2(\mathbb{R}^{n+m}, \mathbb{R}^n)$ has a bounded domain, the derivatives of F is bounded, which is denoted by $\|A_k^{(i)}\| \le \bar{D}$ and $\|B_k^{(i)}\| \le \bar{D}$. Suppose $\bar{D}$ is known. (ii) Suppose $0 < \underline{\hat{\sigma}} \le \sigma_{\min}(\hat{H}^{(i)}) \le \sigma_{\max}(\hat{H}^{(i)}) = \|\hat{H}^{(i)}\| \le \bar{r}$ for all iterations.

Algorithm 2 Integration of model learning and iterative control design

Require: x_0, x_{target}, and an (arbitrary) initial $U^{(i=0)}$.

1. Apply **S1** to explore the local region around the current nominal trajectory.
2. Apply **S2** to update the estimated dynamics on the explored area.
3. Apply **S3** with the use of Algorithm 1 to steer the system closer to the target until $\mathbb{d}^{(i)}_{\text{extend}} > \mathbb{d}_{\text{threshold}}$ or $\mathbb{e}^{(i)}_{\text{model}} > \mathbb{e}_{\text{threshold}}$.
4. Repeat step 1 – 3 until $\|x_N - x_{\text{target}}\|^2$ falls below a desired tolerance.

Theorem 3.3. *Suppose Assumptions 3.1 and 3.2 hold. Then, the sequential learning and control design converges, and the system is steered to the desired target.*

Proof. We will show that $\hat{H}^{(i)}$ computed from the learning model F_{model} satisfies $\|H^{(i)} - \hat{H}^{(i)}\| \le \frac{1}{4}\sigma_{\min}(\hat{H}^{(i)})$. This ensures Assumption 2.5 is satisfied for the use of Theorem 2.7.

First, choose ϵ such that $N\epsilon(\epsilon + \bar{D})^{N-1} + \cdots + 2\epsilon(\epsilon + \bar{D}) + \epsilon \le \frac{1}{4}\underline{\hat{\sigma}}$ and choose δ accordingly as in Assumption 3.1. Set $\mathbb{d}_{\text{threshold}} = \delta$ to ensure D_{training} is sufficiently dense. Thus, from Assumption 3.1, we have $\|\hat{A}^{(i)}_k - A^{(i)}_k\| \le \epsilon$ and $\|\hat{B}^{(i)}_k - B^{(i)}_k\| \le \epsilon$, where $\hat{A}^{(i)}_k := \partial_x F_{\text{model}}(x^{(i)}_k, u^{(i)}_k)$, $\hat{B}_k := \partial_u F_{\text{model}}(x^{(i)}_k, u^{(i)}_k)$. By applying the reverse triangle inequality, we have

$$\|\hat{A}^{(i)}_k\| - \|A^{(i)}_k\| \le \|\hat{A}^{(i)}_k - A^{(i)}_k\| \le \epsilon \Longrightarrow \|\hat{A}^{(i)}_k\| \le \epsilon + \bar{D}.$$

Now, since $\hat{H}^{(i)} = \begin{bmatrix} \hat{A}^{(i)}_{N-1} \cdots \hat{A}^{(i)}_1 \hat{B}^{(i)}_0 & \cdots & \hat{B}^{(i)}_{N-1} \end{bmatrix}$, $\|H^{(i)} - \hat{H}^{(i)}\|$ can be expressed and upper bounded by

$$\begin{aligned} &\left\| \begin{bmatrix} A^{(i)}_{N-1} \cdots A^{(i)}_1 B^{(i)}_0 - \hat{A}^{(i)}_{N-1} \cdots \hat{A}^{(i)}_1 \hat{B}^{(i)}_0 & \cdots & B^{(i)}_{N-1} - \hat{B}^{(i)}_{N-1} \end{bmatrix} \right\| \\ &\le \|A^{(i)}_{N-1} \cdots A^{(i)}_1 B^{(i)}_0 - \hat{A}^{(i)}_{N-1} \cdots \hat{A}^{(i)}_1 \hat{B}^{(i)}_0\| + \cdots + \|B^{(i)}_{N-1} - \hat{B}^{(i)}_{N-1}\|. \end{aligned} \tag{16}$$

Since each norm of the right hand side of (16) can be upper bounded, which depends on the number of terms and not specific $A^{(i)}_k, B^{(i)}_k, \hat{A}^{(i)}_k, \hat{B}^{(i)}_k$, to simplify the notation, we use $G^{(i)}_k$ and $\hat{G}^{(i)}_k$ to generically denote $A^{(i)}_k, B^{(i)}_k$ and $\hat{A}^{(i)}_k, \hat{B}^{(i)}_k$. For a norm with k terms, we have

$$\begin{aligned} &\| G^{(i)}_1 G^{(i)}_2 \cdots G^{(i)}_k - \hat{G}^{(i)}_1 \hat{G}^{(i)}_2 \cdots \hat{G}^{(i)}_k \| \le \| (G^{(i)}_1 - \hat{G}^{(i)}_1) G^{(i)}_2 \cdots G^{(i)}_k \\ &+ \hat{G}^{(i)}_1 (G^{(i)}_2 - \hat{G}^{(i)}_2) G^{(i)}_3 \cdots G^{(i)}_k + \cdots + \hat{G}^{(i)}_1 \cdots \hat{G}^{(i)}_{k-1} (G^{(i)}_k - \hat{G}^{(i)}_k) \| \\ &\le \epsilon \bar{D}^{k-1} + \epsilon(\epsilon + \bar{D})\bar{D}^{k-2} + \cdots + \epsilon(\epsilon + \bar{D})^{k-1} \le k\epsilon(\epsilon + \bar{D})^{k-1}. \end{aligned} \tag{17}$$

Applying (17) to (16), we have

$$\|H^{(i)} - \hat{H}^{(i)}\| \le N\epsilon(\epsilon + \bar{D})^{N-1} + \cdots + 2\epsilon(\epsilon + \bar{D}) + \epsilon.$$

Thus, $\|H^{(i)} - \hat{H}^{(i)}\| \le \frac{1}{4}\underline{\hat{\sigma}} \le \frac{1}{4}\sigma_{\min}(\hat{H}^{(i)})$. Together with Assumption 3.2.ii, it implies that Assumption 2.5 holds. Then, it follows from Theorem 2.7 that the sequential learning and control converges, and the system is steered to the target. □

Note that the above analysis requires the upper bound of the derivative of F, *i.e.*, $\bar{D}$, to be known. If the system dynamics is not available, this quantity needs to be estimated from data. To this end, through measurements, one could estimate how fast the system's state changes (the derivatives of F) and thus estimate the upper bound $\bar{D}$. In experimental settings, additional techniques may be needed to minimize the effect of noise on the estimates [35,36].

In addition, we note that most model-based learning controls, different from our approach, are often formulated to obtain a "global" control solution, *e.g.*, see [37,38] for the use of value functions where the solution is represented as a function of the state. If there exists an incorrect estimate of the system dynamics (which often occurs at the beginning of the learning process), these global approaches will quickly extrapolate from the incorrect model and result in severe drifts and degraded control performance [3,7,8].

Our approach, on the other hand, does not suffer from such model exploitation. The panacea lies in the local nature of our control design, which utilizes the system's *local* dynamics around a current trajectory to improve the control and thus effectively avoids the exploitation associated with learning the *global* dynamics of the system. To ensure that the locally estimated dynamics are appropriate, the dynamics are continuously evaluated and immediately updated when the system ventures into an unexplored region or experiences overfitting (as indicated by the conditions in **S3**).

The iterative and local nature of this integration between model learning and control also gives us the ability to gradually expand and learn a data model in concurrence with the continuing progress of control. This naturally results in an efficient and targeted exploration of the state–action space that is mainly tailored to the purpose of control.

4. Implementation and evaluation

In this section, we illustrate the effectiveness of the proposed framework with simulation of a benchmark control example and physical implementation. In both cases, we do not assume any prior knowledge of the systems and let them learn their suitable controls from scratch by following the steps outlined in Algorithm 2.

4.1. Simulation of a cart-pendulum system

The dynamics of an inverted pendulum on a cart is

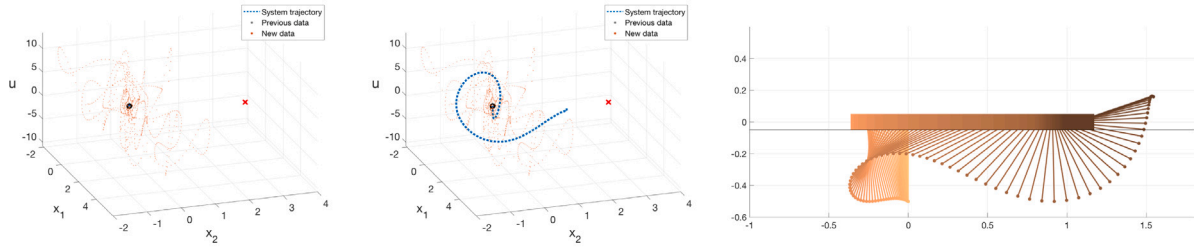
$$\begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \dot{x}_3 &= \frac{lmx_4^2 \sin x_2 + u + mg \cos x_2 \sin x_2}{M + m(1 - \cos^2 x_2)} \\ \dot{x}_4 &= -\frac{lmx_4^2 \cos x_2 \sin x_2 + u \cos x_2 + (M + m)g \sin x_2}{lM + lm(1 - \cos^2 x_2)} \end{aligned}$$

where x_1, x_2, x_3 and x_4 are the position of the cart, the angle of the pendulum, their corresponding velocities, and $M = 0.8$, $m = 0.5$, $l = 0.5$, $g = 9.81$.

In this problem, we learn the control applied to the cart so that the pendulum is swung from a downward position at $x = [0, 0, 0, 0]^\top$ to an upward position at $x_{\text{target}} = [1, \pi, 0, 0]^\top$. To capture the system's dynamics, we employ a neural network with two hidden layers (each layer consists of 10 hyperbolic tangent activation units) provided by MATLAB Toolbox. The Levenberg–Marquardt algorithm is used for the training [39]. To achieve a swing-up control, we apply Algorithm 2 with $N = 100$, $\Delta T = 0.02$, $\mathbb{d}_{\text{threshold}} = 5$, and $\mathbb{e}_{\text{threshold}} = 0.01$ to gradually steer the system from x_0 to x_{target}.

During the process, if the system starts departing from the explored region, a new round of exploration is executed to inspect the locally unexplored region to better capture the overall dynamics. This mechanism, as observed in Fig. 3, results in a rather efficient exploration of the state–action space in which the sample data are mostly located around the controlled trajectories. As a result, the explored region is closely connected and gradually expanded with the progress of control, allowing us to reduce excessive (and potentially undesirable) explorations.

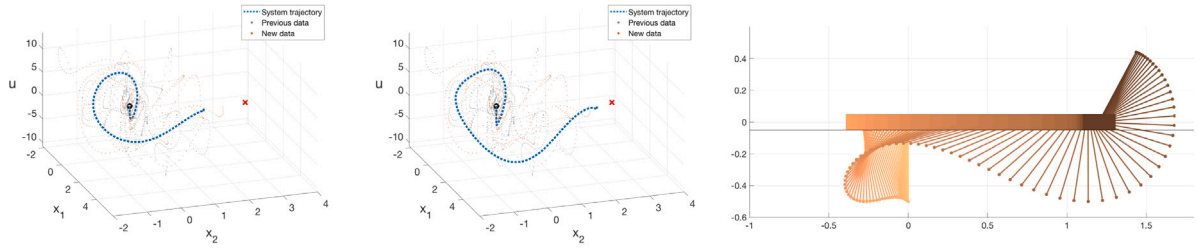
We compare the proposed framework with PILCO [14] and adaptive control with model identification (MIAC) [40], which serve as representatives of probabilistic and deterministic model-based learning, respectively. Regarding PILCO, we used an open-source implementation provided by the authors and compare the results across different runs. The presented values are the results averaged over 10 random seeds. Compared to PILCO, our approach requires a similar level of interaction with the system yet exhibits much more regulated exploration and consistent system behaviors, both during and after learning.



(a) Control synthesis begins after the 1st round of explorations

(b) Control synthesis is paused before the 2nd round of explorations

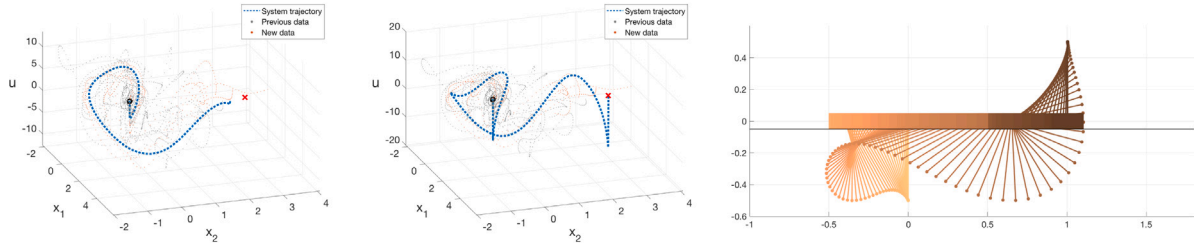
(c) Simulation of the system before the 2nd round of explorations



(d) Control synthesis resumes after the 2nd round of explorations

(e) Control synthesis is paused before the 3rd round of explorations

(f) Simulation of the system before the 3rd round of explorations

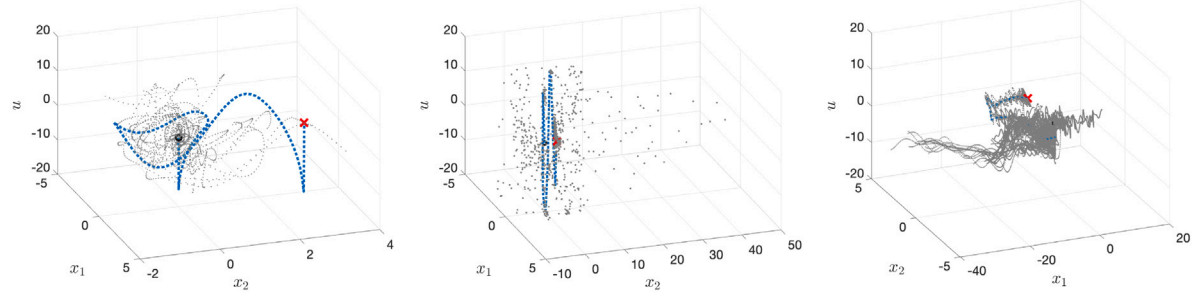


(g) Control synthesis resumes after the 3nd round of explorations

(h) Control synthesis finishes with the final controlled trajectory

(i) Simulation of the system when the control synthesis finishes

Fig. 3. Gradual expansion of the data cloud (representing the explored regions) along the evolution of the controlled trajectories. Simulations of the cart-pendulum at different stages of the learning process are also presented in which the frames are uniformly sampled in time, plotted from (light) orange to (dark) brown, as the trajectories progress. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Sequential learning and control design (b) Probabilistic model learning (PILCO) (c) Model-adaptive control (MIAC)

Fig. 4. Training data collected by three model-based learning methods and their corresponding final controlled trajectory.

To mitigate model exploitation, PILCO employs extensive random exploration, particularly during the initial stages of the learning phase. This leads to an increased exploration ratio, denoted by the proportion of the (same) state–action space encompassing the training data, in comparison to our method, as detailed in Table 2. The data collected from both approaches are visually presented in Fig. 4, affirming this observation. Note that our approach also requires less computation compared to PILCO, which is known to scale exponentially with the number of system states and hinders the implementation of PILCO on high-dimensional control systems (see, *e.g.*, [16,17,41]).

Regarding the implementation of MIAC, we use a two-hidden-layers (with 20 activation units) neural network for model identification and energy shaping method for control [42]. At each time step, the model is used to compute $\Delta E = E - E_{\text{target}}$, where $E := \frac{1}{2}\dot{\theta}^2 - \cos\theta$, and the input u is chosen such that $\Delta\dot{E} = -c\Delta E, c \geq 0$. Once the pendulum's angle is controlled to the upward position, the model's linearization is utilized to regulate the cart position. The model is updated recursively for every 100 steps. In our implementation, this rather conventional recursive application of MIAC fails to converge due to model exploitation and the lack of exploration. However, when executed sequentially in batches

Table 2
Comparison of the proposed approach with a probabilistic model-based learning (PILCO) and adaptive control with model identification (MIAC) on the cart-pendulum system.

	System interaction	Exploration ratio	Computation (in CPU) time
proposed	≈ 37 trials	≈ 8%	≈ 5 min
PILCO	≈ 35 trials	≈ 31%	≈ 30 min
MIAC	≈ 145 trials	≈ 27%	≈ 35 min

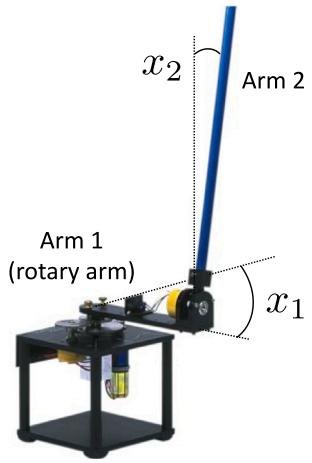


Fig. 5. The Furuta pendulum platform [43]. To control the platform, *e.g.*, to swing arm 2 to the upward position, one needs to apply suitable voltages to the DC motor to control arm 1.

of exploration and control (as in the advocated sequential design framework), MIAC can mitigate model exploitation and accomplish the control task. As illustrated in Table 2 and Fig. 4, MIAC nevertheless requires more system interaction and exploration of the state–action space, as compared to the proposed approach.

4.2. Physical implementation on a rotatory system

We present the implementation of the proposed framework on an under-actuated robot arm named the Furuta pendulum. The control objective of this two-link robot is to swing arm 2 up from the downward position by only rotating arm 1, which is controlled by a DC motor.

Our experiment uses the *"Rotary Inverted Pendulum"* from *Quanser* [43], as shown in Fig. 5, connected to a standard desktop (4 GB of RAM, 2.5 GHz) using Matlab and Simulink 2012b. The system has four states, *i.e.*, the angles of the two arms (x_1, x_2) and their corresponding angular velocities (x_3, x_4), and one input u, *i.e.*, the voltage applied to a DC motor to control arm 1. Two encoders are attached at the ends of the two arms to measure the angles of the arms and send readings to Simulink in real time, and the angular velocities are computed via an in-built time-derivative block. The input signal is fed into the platform directly via Simulink. The objective is to learn a control that steers the system from x_0 to x_{target}, which are defined as

$$x_0 = \begin{bmatrix} 0 & \pi & 0 & 0 \end{bmatrix}^\top, \quad x_{\text{target}} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^\top.$$

To this end, we use the same neural network as in the previous example and learn a swing-up control by following the process outlined in Algorithm 2. The program first explores the local area around the downward position of arm 2 and learns the corresponding dynamics. Then, it iteratively swings arm 2 step-by-step closer to the upward position. When the system starts departing from the data cloud of the explored region to enter uncharted territory or experiences slow steering progress due to overfitting, the program pauses the current control synthesis and executes a new round of explorations. This process is continued until the system reaches x_{target}. The swing-up task is successfully learned after 4 rounds of explorations with a total of 53 trials and in about 20 min. Fig. 6 illustrates this tightly integrated steering process where the explored region is gradually expanded in accordance with the iterative progress of the control design. The swing-up of the physical pendulum system is shown in Fig. 7, and the experimental details are presented in Table 3.

5. Conclusions

In this paper, we develop a framework for efficient integration of data into system control design, learning point-to-point controlled behaviors of systems in particular. We discussed in detail the importance and benefits of dissecting model learning and control design (which are both often treated from a high-level perspective) into explicit and step-by-step processes. We then recombined these basic processes into a highly integrated and data-efficient control framework in which system modeling and the acquisition of data are both tailored to the purpose of control design. Due to these distinct features, the proposed framework can avoid model exploitation and quickly learn a desirable control from scratch with just a small amount of data. We illustrated the effectiveness and robustness of the proposed framework in both simulation and experiment.

CRediT authorship contribution statement

Minh Vu: Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Yunshen Huang:** Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Conceptualization. **Shen Zeng:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Minh Vu, Yunshen Huang, Shen Zeng reports financial support was provided by National Science Foundation.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was supported by the National Science Foundation, USA under the awards CMMI-1933976.

Appendix A

We provide additional analysis for Section 2.

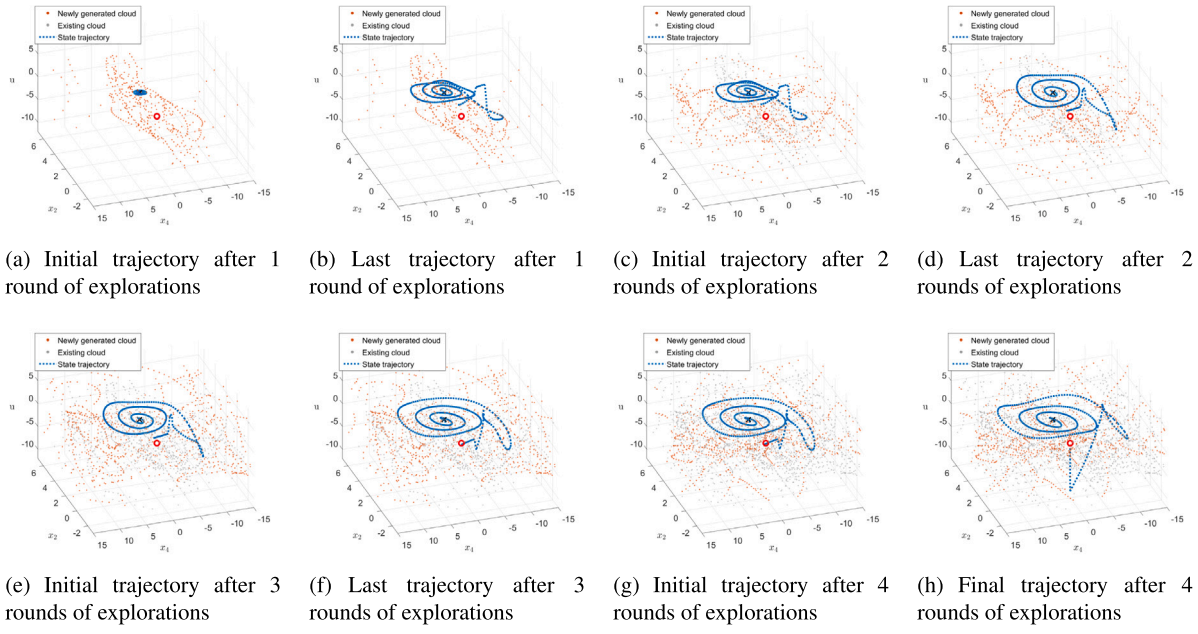
Lemma A.1. *If A is a positive definite matrix, then* $\|(\lambda I + A)^{-1}x\| \leq \|A^{-1}x\|, \forall \lambda > 0, \forall x \in \mathbb{R}^n$.

Proof. Take $\lambda > 0$ and $x \in \mathbb{R}^n$. Set $P = (\lambda I + A)^{-1}x$. Then, $\lambda P + AP = x$. Rearranging the equation, we have $P = A^{-1}(x - \lambda P)$.

On the other hand, since A is positive definite, A^{-1} is also positive definite. Thus, we have

$$\begin{aligned} 0 &\leq (x - x + \lambda P)^\top A^{-1}(x - x + \lambda P) \\ &= \lambda P^\top A^{-1}x - \lambda P^\top A^{-1}(x - \lambda P) \\ &= \lambda P^\top A^{-1}x - \lambda P^\top P. \end{aligned}$$

Thus, $\|P\|^2 \leq P^\top A^{-1}x \leq \|P\|\,\|A^{-1}x\| \Longrightarrow \|P\| \leq \|A^{-1}x\|$. □



(a) Initial trajectory after 1 round of explorations
(b) Last trajectory after 1 round of explorations
(c) Initial trajectory after 2 rounds of explorations
(d) Last trajectory after 2 rounds of explorations
(e) Initial trajectory after 3 rounds of explorations
(f) Last trajectory after 3 rounds of explorations
(g) Initial trajectory after 4 rounds of explorations
(h) Final trajectory after 4 rounds of explorations

Fig. 6. Expansion of a data cloud representing the explored regions in concurrence with the evolution of the controlled trajectories of the physical Furuta pendulum system.

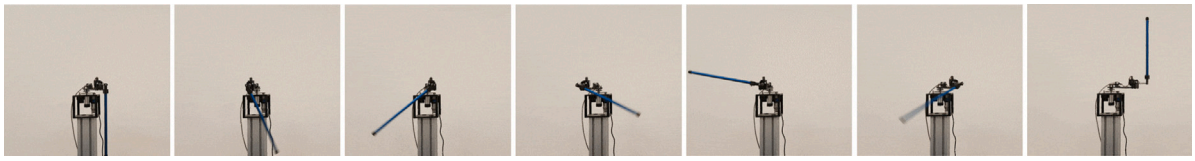


Fig. 7. Physical platform of the Furuta pendulum system with snapshots of the swing-up control carried out in 4 s. The controlled behavior is achieved after about 20 min of learning the dynamic model and the appropriate control input.

Table 3
Experimental recordings for the physical implementation of the proposed framework on the Furuta pendulum.

Round of explorations	Number of test inputs	Number of added data	$\mathbb{d}_{\text{threshold}}$	$\mathbb{e}_{\text{threshold}}$	$\|x_N - x_{\text{taget}}\|$ at the beginning	$\|x_N - x_{\text{taget}}\|$ at the end	Number of control iterations
1	8	1600	2	0.001	π	1.0731	4
2	8	1600	2	0.001	1.0731	0.6520	11
3	8	1600	2	0.001	0.6520	0.2850	3
4	8	1600	2	0.001	0.2850	0.0004	3

Lemma A.2. *If the $H^{(i)}$ matrix is full rank, $\|\Delta U_*^{(i)}\| \leq \sigma_{min}^2(H^{(i)})\,\sigma_{max}(H^{(i)})\|x_{target} - x_N^{(i)}\|$.*

Proof. From (4) and Lemma A.1, we have

$$\begin{aligned}\|\Delta U_*^{(i)}\| &\leq \|(H^{(i)^\top} H^{(i)})^{-1} H^{(i)^\top}(x_{\text{target}} - x_N^{(i)})\| \\ &\leq \sigma_{\min}(H^{(i)^\top} H^{(i)})\,\sigma_{\max}(H^{(i)})\|x_{\text{target}} - x_N^{(i)}\| \\ &\leq \sigma_{\min}^2(H^{(i)})\,\sigma_{\max}(H^{(i)})\|x_{\text{target}} - x_N^{(i)}\|\,. \quad \square \quad \square\end{aligned}$$

References

[1] Z.-S. Hou, Z. Wang, From model-based control to data-driven control: Survey, classification and perspective, Inform. Sci. 235 (2013) 3–35.
[2] C. De Persis, P. Tesi, Formulas for data-driven control: Stabilization, optimality, and robustness, IEEE Trans. Autom. Control 65 (3) (2019) 909–924.
[3] D. Bruder, B. Gillespie, C.D. Remy, R. Vasudevan, Modeling and control of soft robots using the koopman operator and model predictive control, 2019, arXiv preprint arXiv:1902.02827.
[4] C. Summers, K. Lowrey, A. Rajeswaran, S. Srinivasa, E. Todorov, Lyceum: An efficient and scalable ecosystem for robot learning, in: Learning for Dynamics and Control, PMLR, 2020, pp. 793–803.
[5] Y.S. Shao, C. Chen, S. Kousik, R. Vasudevan, Reachability-based trajectory safeguard (RTS): A safe and fast reinforcement learning safety layer for continuous control, IEEE Robot. Autom. Lett. 6 (2) (2021) 3663–3670.
[6] M. Haseli, J. Cortés, Learning koopman eigenfunctions and invariant subspaces from data: Symmetric subspace decomposition, IEEE Trans. Autom. Control 67 (7) (2021) 3442–3457.
[7] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, J. Ba, Benchmarking model-based reinforcement learning, 2019, arXiv preprint arXiv:1907.02057.
[8] T.M. Moerland, J. Broekens, C.M. Jonker, Model-based reinforcement learning: A survey, 2020, arXiv preprint arXiv:2006.16712.
[9] M. Haseli, J. Cortés, Generalizing dynamic mode decomposition: Balancing accuracy and expressiveness in koopman approximations, Automatica 153 (2023) 111001.
[10] J.G. Schneider, Exploiting model uncertainty estimates for safe dynamic control learning, in: Advances in Neural Information Processing Systems, 1997, pp. 1047–1053.
[11] J. Kober, J.A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey, Int. J. Robot. Res. 32 (11) (2013) 1238–1274.
[12] M. Janner, J. Fu, M. Zhang, S. Levine, When to trust your model: Model-based policy optimization, 2019, arXiv preprint arXiv:1906.08253.
[13] J. Ko, D.J. Klein, D. Fox, D. Haehnel, Gaussian processes and reinforcement learning for identification and control of an autonomous blimp, in: 2007 IEEE International Conference on Robotics and Automation, IEEE, 2007, pp. 742–747.
[14] M. Deisenroth, C.E. Rasmussen, PILCO: A model-based and data-efficient approach to policy search, in: International Conference on Machine Learning, PMLR, 2011, pp. 465–472.
[15] S. Levine, V. Koltun, Guided policy search, in: International Conference on Machine Learning, PMLR, 2013, pp. 1–9.

[16] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, S. Levine, Solar: Deep structured representations for model-based reinforcement learning, in: International Conference on Machine Learning, PMLR, 2019, pp. 7444–7453.
[17] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R.H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, et al., Model-based reinforcement learning for atari, 2019, arXiv preprint arXiv:1903.00374.
[18] A. Nagabandi, C. Finn, S. Levine, Deep online learning via meta-learning: Continual adaptation for model-based rl, 2018, arXiv preprint arXiv:1812.07671.
[19] A. Nagabandi, I. Clavera, S. Liu, R.S. Fearing, P. Abbeel, S. Levine, C. Finn, Learning to adapt in dynamic, real-world environments through meta-reinforcement learning, 2018, arXiv preprint arXiv:1803.11347.
[20] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, V. Sindhwani, Data efficient reinforcement learning for legged robots, in: Conference on Robot Learning, PMLR, 2020, pp. 1–10.
[21] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, S. Levine, How to train your robot with deep reinforcement learning: lessons we have learned, Int. J. Robot. Res. 40 (4–5) (2021) 698–721.
[22] S. Zeng, Iterative optimal control syntheses illustrated on the brockett integrator, IFAC-PapersOnLine 52 (16) (2019) 138–143.
[23] M. Vu, S. Zeng, Iterative optimal control syntheses for nonlinear systems in constrained environments, in: American Control Conference, ACC, IEEE, 2020, pp. 1731–1736.
[24] M. Vu, S. Zeng, An iterative online approach to safe learning in unknown constrained environments, in: IEEE Conference on Decision and Control, CDC, IEEE, 2023, pp. 7330–7335.
[25] M. Vu, S. Zeng, H. Fang, Health-aware battery charging via iterative nonlinear optimal control syntheses, IFAC-PapersOnLine 53 (2) (2020) 12485–12490.
[26] Y. Huang, W. He, S. Zeng, A differential dynamic programming-based approach for balancing energy and time optimality in motion planning, in: 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton), IEEE, 2023, pp. 1–7.
[27] E. Kamen, Fundamentals of linear time-varying systems, in: In the Control Handbook (Second Edition): Control System Advanced Methods, 2010, pp. 3.1–3.33.
[28] T. Hastie, R. Tibshirani, J.H. Friedman, J.H. Friedman, The elements of statistical learning: data mining, inference, and prediction, vol. 2, Springer, 2009.
[29] K. Königsberger, Analysis 2, Springer-Verlag, 2013.
[30] H. Bauschke, P. Combettes, Convex analysis and monotone operator theory in Hilbert spaces, in: CMS books in mathematics, 2011, DOI 10 978–1.
[31] Y. Cao, Q. Gu, Generalization bounds of stochastic gradient descent for wide and deep neural networks, Adv. Neural Inf. Process. Syst. 32 (2019).
[32] N. Golowich, A. Rakhlin, O. Shamir, Size-independent sample complexity of neural networks, in: Conference on Learning Theory, PMLR, 2018, pp. 297–299.
[33] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[34] T.M. Inc., Statistics and Machine Learning Toolbox version: 12.5 (R2022b), The MathWorks Inc., Natick, Massachusetts, United States, 2022, URL https://www.mathworks.com.
[35] H. Lanshammar, On precision limits for derivatives numerically calculated from noisy data, J. Biomech. 15 (6) (1982) 459–470.
[36] J. Pezzack, R. Norman, D. Winter, An assessment of derivative determining techniques used for motion analysis, J. Biomech. 10 (5–6) (1977) 377–382.
[37] D. Bertsekas, Dynamic programming and optimal control: Volume I, vol. 1, Athena scientific, 2012.
[38] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT Press, 2018.
[39] D.W. Marquardt, An algorithm for least-squares estimation of nonlinear parameters, J. Soc. Ind. Appl. Math. 11 (2) (1963) 431–441.
[40] I.D. Landau, R. Lozano, M. M'Saad, A. Karimi, Adaptive control: algorithms, analysis and applications, Springer Science & Business Media, 2011.
[41] N. Wahlström, T.B. Schön, M.P. Deisenroth, From pixels to torques: Policy learning with deep dynamical models, 2015, arXiv preprint arXiv:1502.02251.
[42] M.W. Spong, Energy based control of a class of underactuated mechanical systems, IFAC Proc. Vol. 29 (1) (1996) 2828–2832.
[43] Quanser, Rotary inverted pendulum, 2021, https://www.quanser.com/products/rotary-inverted-pendulum/.