Learning Contraction Policies From Offline Data

Navid Rezazadeh, *Graduate Student Member, IEEE*, Maxwell Kolarich, *Graduate Student Member, IEEE*, Solmaz S. Kia, *Senior Member, IEEE*, and Negar Mehr, *Member, IEEE*

***Abstract*—This letter proposes a data-driven method for learning convergent control policies from offline data using Contraction theory. Contraction theory enables constructing a policy that makes the closed-loop system trajectories inherently convergent towards a unique trajectory. At the technical level, identifying the contraction metric, which is the distance metric with respect to which a robot's trajectories exhibit contraction is often non-trivial. We propose to jointly learn the control policy and its corresponding contraction metric while enforcing contraction. To achieve this, we learn an implicit dynamics model of the robotic system from an offline data set consisting of the robot's state and input trajectories. We propose a data augmentation algorithm for learning contraction policies using this learned dynamics model. We randomly generate samples in the state space and propagate them forward in time through the learned dynamics model to generate auxiliary sample trajectories. We then learn both the control policy and the contraction metric such that the distance between the trajectories from the offline data set and our generated auxiliary sample trajectories decreases over time. We evaluate the performance of our proposed framework on simulated robotic goal-reaching tasks and demonstrate that enforcing contraction results in faster convergence and greater robustness of the learned policy.**

***Index Terms*—Deep learning methods, machine learning for robot control, reinforcement learning.**

I. Introduction

While learning-based controllers have achieved significant success, they still lack safety guarantees. For instance, in general, the temporal evolution of a robot's trajectories under a learned policy cannot be certified. On the other hand, when a system's dynamics are known, control-theoretic properties, such as stability and contraction, directly examine the temporal progression of a system's states to verify whether a system remains within a safe set, and whether the system's trajectories converge. In this letter, we seek to enforce the desired temporal evolution of the closed-loop system's states while learning the policy from an offline set of data, i.e. we seek to learn control policies such that under the learned policy, the convergence of a robot's trajectories is achieved.

To achieve such trajectory convergence, our design approach leverages Contraction theory [1]. Contraction theory provides a framework for identifying the class of nonlinear dynamic systems that have asymptotic convergent trajectories. Intuitively, a region of the state space is a contraction space if the distance between any two close neighboring trajectories decays over time. This notion of convergence is relevant to many robotic tasks such as tracking controllers where we want a robot to either reach a goal or track a reference trajectory. In this letter, we want to learn policies from offline data such that they achieve convergence of a robot's trajectories in closed loop. While contraction theory provides a simple and intuitive characterization of convergent trajectories, finding the distance metric with respect to which a robot's trajectories exhibit contraction – which is called the *contraction metric* – is often non-trivial. To address this challenge, we propose to *jointly* learn the robot policy and its corresponding contraction metric.

We learn the robot dynamics model from an offline data set consisting of the robot's state and input trajectories. This setting is similar to the setting of offline model-based reinforcement learning (RL) where a dynamics model and a policy are learned from a set of robot trajectories that are collected offline. Learning from offline data is appropriate for safety-critical applications where online data collection is dangerous [2]. We learn a dynamics model of the system from the data and propose a data augmentation algorithm for learning contraction policies. Randomly sampled states are propagated forward in time through the learned dynamics model to generate auxiliary sample trajectories. We then learn both our policy and our contraction metric such that the distance between the robot trajectories from the data set and the auxiliary sample trajectories decreases over time. Learning contraction policies is particularly relevant to offline RL as it allows us to regard the errors in the learned dynamics model as external disturbances and obtain a tracking error bound in regions where the learning errors of the dynamics model are bounded [3], [4].

We evaluate the performance of our proposed framework on a set of simulated robotic goal-reaching tasks. The performance of our proposed framework is compared with a number of control algorithms. We demonstrate that as a result of enforcing contraction, the robot's trajectories converge faster to the goal position with a higher degree of accuracy. It is further shown that learning contraction policies increases the robustness of the learned policy with respect to learned dynamics model mismatch, i.e. enforcing contraction increases the robustness of the learned policies. In summary, our contributions are the following:

- We propose a framework for learning convergent robot policies from an offline data set using Contraction theory.
- We develop a data augmentation algorithm for learning contraction policies from the offline data set.
- We provide a formal analysis for bounding contraction policy performance as a function of dynamics model mismatch.

Manuscript received September 9, 2021; accepted January 3, 2022. Date of publication January 25, 2022; date of current version February 2, 2022. This letter was recommended for publication by Associate Editor G. Chalvatzaki and Editor J. Kober upon evaluation of the reviewers' comments. The work of Navid Rezazadeh and Solmaz S. Kia were supported by NSF, under CAREER Award ECCS-1653838. *(Corresponding author: Navid Rezazadeh.)*

Navid Rezazadeh and Solmaz S. Kia are with the Department of Mechanical and Aerospace Engineering, University of California, Irvine 92697 USA (e-mail: nrezazad@uci.edu; solmaz@uci.edu).

Maxwell Kolarich and Negar Mehr are with the Aerospace Engineering Department, University of Illinois Urbana-Champaign, Urbana 61801 USA (e-mail: mak13@illinois.edu; negar@illinois.edu).

Digital Object Identifier 10.1109/LRA.2022.3145100

- We perform numerical evaluations of our proposed policy learning framework and demonstrate that enforcing contraction results in favorable convergence and robustness performance.

The organization of this letter is as follows. In Section II, we provide an overview of the related and prior work. We provide an overview of Contraction theory in Section III and present our problem formulation in Section IV. We then discuss our proposed framework in Section V. Section VI provides a discussion and analysis of the robustness of learned contraction policies. In Section VII, we evaluate and compare the performance of our policy learning algorithm. Finally, we will conclude the letter in Section VIII.

II. Related Work

For systems with unknown dynamics, several offline RL algorithms have been developed recently which either directly learn a policy using an offline data-set [2], [5]–[7] or learn a surrogate dynamics model from the offline data to learn an appropriate policy [8], [9]. However, the majority of such RL algorithms lack formal safety guarantees, and the convergent behavior of the learned policies is not certified [10], [11].

When the system dynamics are known, robust and certifiable control policy design can be achieved through various control-theoretic methods such as reachability analysis [12], Funnels [13], [14], and Hamilton-Jacobi analysis [15], [16]. Lyapunov stability criteria, Contraction Theory, and Control Barrier Functions have also been extensively utilized for providing strong convergence guarantees for nonlinear dynamical systems [1], [17]–[20]. However, even when the dynamics are known, finding a proper Lyapunov function or a control barrier function is itself a challenging task. To address these challenges, learning algorithms have been utilized for learning the Lyapunov and Control Barrier Functions [21]–[23]. In [10] and [24], a framework for learning contraction metrics was proposed for systems with known dynamics.

Various recent works have considered combining control-theoretic tools with learning algorithms to enable learning safe policies even when dynamics are unknown. For instance, [25], [26] consider learning stable dynamics models. In [27], Contraction theory is used to learn stabilizable dynamics models of unknown systems. In [11], [28], Lyapunov functions are used for ensuring the stability of the learned policies. In [29], it is proposed to learn the system dynamics and its corresponding Lyapunov function jointly to ensure the stability of the learned dynamics model.

In this work, we consider learning contraction policies from offline data for systems with unknown dynamics. Our work is closely related to [10] and [24], where Contraction theory has been used for certifying convergent trajectories. The current work is different in that, unlike these approaches where dynamics are explicitly known and assumed to be control-affine, we consider access to only an offline data set. We assume that we can learn an implicit model of system dynamics, in the form of a neural network function approximator, and provide robustness guarantees with respect to the errors of the learned dynamics model. Moreover, we develop a novel method for learning contraction policies which can be be applied to general nonlinear dynamical systems.

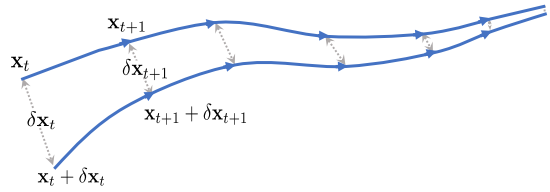

Fig. 1. The schematic of two neighboring trajectories that exhibit contraction. The distance between the trajectories decreases over time: $\|\delta\mathbf{x}_{t+1}\| < \|\delta\mathbf{x}_t\|$, i.e. trajectories converge.

III. Contraction Theory

Contraction theory assesses the stability properties of dynamical systems by studying the convergence behavior of neighboring trajectories [1]. The convergence is established by directly examining the evolution of the weighted Euclidean distance of close neighboring trajectories.

Formally, consider a differentiable autonomous discrete-time dynamical system $g(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$ defined as

$$\mathbf{x}_{t+1} = g(\mathbf{x}_t), \tag{1}$$

with Jacobian

$$\nabla g(\mathbf{x}_t) = \left.\frac{\partial g(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_t}. \tag{2}$$

Now, consider a differential displacement $\delta\mathbf{x}_t$. The differential displacement dynamics at $\mathbf{x}_t$ are governed by

$$\delta\mathbf{x}_{t+1} = \nabla g(\mathbf{x}_t)\delta\mathbf{x}_t. \tag{3}$$

The system dynamics $g(\mathbf{x}_t)$ are contractive if there exists a full rank state dependent metric $\boldsymbol{\Theta}(\mathbf{x}) \in \mathbb{R}^n \times \mathbb{R}^n$ such that the system trajectories satisfy

$$\|\boldsymbol{\Theta}(\mathbf{x}_t)\delta\mathbf{x}_t\| > \|\boldsymbol{\Theta}(\mathbf{x}_{t+1})\delta\mathbf{x}_{t+1}\|. \tag{4}$$

Equation (4) indicates that the weighted distance between any two infinitesimally close states decreases as the dynamics evolve [1]. When $\boldsymbol{\Theta}(\mathbf{x}) = \mathbf{I}$ the distances between trajectories are measured in the Euclidean norm sense. Fig. 1 illustrates the behavior of two trajectories of a contractive system when $\boldsymbol{\Theta}(\mathbf{x}) = \mathbf{I}$.

For a small finite displacement $\Delta\mathbf{x}_t$, as an approximation of infinitesimal small displacement $\delta\mathbf{x}_t$, the first-order Taylor expansion of the system dynamics allows us to locally approximate the forward evolution of the displacement

$$\nabla g(\mathbf{x}_t)\Delta\mathbf{x}_t \approx g(\mathbf{x}_t + \Delta\mathbf{x}_t) - g(\mathbf{x}_t). \tag{5}$$

Thus, we may approximate the contraction condition (4) as

$$\|\boldsymbol{\Theta}(\mathbf{x}_{t+1})\left(g(\mathbf{x}_t + \Delta\mathbf{x}_t) - g(\mathbf{x}_t)\right)\| - \|\boldsymbol{\Theta}(\mathbf{x}_t)\Delta\mathbf{x}_t\| < 0. \tag{6}$$

Establishing a system as contractive allows for several useful stability properties to be deduced. We state motivating results from [1] in the following definition and proposition.

Definition 1: Given the discrete-time system $\mathbf{x}_{t+1} = g(\mathbf{x}_t)$, a region of the state space is called a *contraction region* with respect to a uniformly positive definite metric $\mathbf{M}(\mathbf{x}_t) = \boldsymbol{\Theta}(\mathbf{x}_t)^T\boldsymbol{\Theta}(\mathbf{x}_t)$, if in that region

$$\nabla g(\mathbf{x}_t)^T\mathbf{M}(\mathbf{x}_{t+1})\nabla g(\mathbf{x}_t) - \mathbf{M}(\mathbf{x}_t) < 0, \tag{7}$$

Proposition 1: A convex contraction region contains *at most one equilibrium point*.

It is shown in [1] that (7) is equivalent to condition (4) holding for all $\mathbf{x}_t$ and $\delta\mathbf{x}_t$ in the contraction region. Thus, by Proposition 1, we may conclude that a unique equilibrium exists within a convex region if (4) holds everywhere inside the region. Therefore, (6) represents a useful numerical analog that can be enforced in order to drive a region towards being contractive. By choosing a set of $\Delta\mathbf{x}_t$, we will use condition (6) to enforce contracting behavior of the closed-loop system.

Going beyond autonomous systems, when a system is subject to control input $\mathbf{u}_t$, i.e., $\mathbf{x}_{t+1} = g(\mathbf{x}_t) = f(\mathbf{x}_t, \mathbf{u}_t)$, contraction theory can be used to design state feedback policies $\mathbf{u}_t = \mathbf{u}(\mathbf{x}_t)$ such that the closed-loop system trajectories converge to a given reference state. This may be done by determining $\mathbf{u}(\mathbf{x}_t)$ such that the convex region of interest is contractive and the unique equilibrium is the desired reference state. Such a control design process is outlined in the following sections.

IV. Problem Formulation

We consider the problem of control policy design for a robot with unknown discrete-time dynamics model $f(\mathbf{x}, \mathbf{u}) : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$, where $\mathcal{X} \in \mathbb{R}^n$ is convex, and $\mathcal{U} \in \mathbb{R}^m$. We assume that we can use an offline data set $\mathcal{D}$ consisting of tuples of state transitions and control inputs $(\mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{u}_t)$ satisfying the unknown system dynamics

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t). \tag{8}$$

Our objective is to obtain a data-driven state-feedback control policy $\mathbf{u}_t = \mathbf{u}(\mathbf{x}_t)$ to steer the system (8) towards a desired reference state $\mathbf{x}^r \in \mathbb{R}^n$, i.e. $\mathbf{x}_t \to \mathbf{x}^r$ as $t \to \infty$. To compensate for the lack of knowledge of the true system dynamics, we propose using a model of the system dynamics that we learn from the offline data $\mathcal{D}$. Note that this indicates that our learned dynamics model may still not be available explicitly and may only be available as implicit dynamics such as neural network approximators. More specifically, we aim to design a control policy $\mathbf{u}(\mathbf{x}_t)$ that leverages the learned dynamics model

$$\mathbf{x}'_{t+1} = f'(\mathbf{x}_t, \mathbf{u}_t), \tag{9}$$

to drive the system asymptotically to $\mathbf{x}^r$.

V. Learning Deep Contraction Policies

To develop a policy that results in contractive behavior, we seek to enforce the approximate condition in (6), requiring the weighted distances of close neighboring trajectories to decrease over time. To enforce this condition, we need to ensure that we have sufficiently close neighboring points for each point within our training set. However, our training data set may not include such neighboring trajectories. We augment our data set with auxiliary trajectories that enable us to enforce this condition at each data point. That is, for each $\mathbf{x}_t \in \mathcal{D}$, we augment our data set with a $\Delta\mathbf{x}_t$ sampled from

$$\Delta_t = \left\{ \Delta\mathbf{x}_t \in \mathbb{R}^n \;\middle|\; \|\Delta\mathbf{x}_t\| < \lambda \right\}, \tag{10}$$

where the parameter λ is set in the training process. We sample points from Δ_t to ensure that $\Delta\mathbf{x}_t$ is a small displacement with respect to the training data set. Then, for each data point $\mathbf{x}_t$, we create the auxiliary state $\tilde{\mathbf{x}}_t = \mathbf{x}_t + \Delta\mathbf{x}_t$. Both of these points are propagated through our learned dynamics model to calculate the states at the next time step: $\mathbf{x}'_{t+1} = f'(\mathbf{x}_t, \mathbf{u}(\mathbf{x}_t))$ and $\tilde{\mathbf{x}}'_{t+1} = f'(\tilde{\mathbf{x}}_t, \mathbf{u}(\tilde{\mathbf{x}}_t))$. The initial state, the auxiliary state, and the predicted evolution of these two states are then combined into a tuple $(\mathbf{x}_t, \tilde{\mathbf{x}}_t, \mathbf{x}'_{t+1}, \tilde{\mathbf{x}}'_{t+1})$. The collection of all such tuples over all $\mathbf{x}_t \in \mathcal{D}$ form the augmented data set $\mathcal{D}'$.

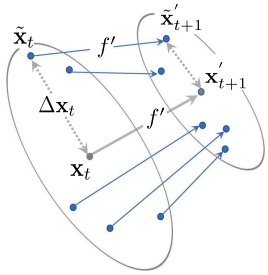

Fig. 2. We sample a small displacement $\Delta\mathbf{x}_t$ around the data point $\mathbf{x}_t$ to augment an auxiliary point $\tilde{\mathbf{x}}_t = \mathbf{x}_t + \Delta\mathbf{x}_t$ to our data set. Then, we propagate the auxiliary state $\tilde{\mathbf{x}}_t$ and the actual state $\mathbf{x}_t$ through our learned dynamics model f' under feedback control law $\mathbf{u}(\mathbf{x}_t)$ to calculate the next states: $\tilde{\mathbf{x}}'_{t+1}$ and $\mathbf{x}'_{t+1}$, respectively. Finally, we require that the weighted distance between the two states decreases over time as stated in condition (11).

Now, we want to evaluate the contracting behavior of the controller $\mathbf{u}(\mathbf{x}_t)$ through the learned model on the augmented data set $\mathcal{D}'$. Thus, we seek to enforce condition (6) for the elements of $\mathcal{D}'$

$$\|\boldsymbol{\Theta}(\mathbf{x}'_{t+1})(\tilde{\mathbf{x}}'_{t+1} - \mathbf{x}'_{t+1})\| - \|\boldsymbol{\Theta}(\mathbf{x}_t)\,\Delta\mathbf{x}_t\| < 0, \tag{11}$$

with respect to a contraction metric $\boldsymbol{\Theta}(\mathbf{x}_t)$. Contractive behavior is illustrated in Fig. 2, showing the weighted distance between $\tilde{\mathbf{x}}_t$ and $\mathbf{x}_t$ decays as the system evolves to $\tilde{\mathbf{x}}'_{t+1}$ and $\mathbf{x}'_{t+1}$. We evaluate the approximate contraction condition only at the states $\mathbf{x}_t$ that exist in the data set $\mathcal{D}$. This is due to the fact that the dynamics model is learned from $\mathcal{D}$ and hence f' is expected to behave the most accurately at these points, which in turn will increase the quality of the learned policy. This will enforce contractive behavior with respect to the learned dynamics model f'. Later we will discuss how we can ensure contractive behavior of the closed-loop behavior of the true dynamics model f.

Since in general, the contraction metric $\boldsymbol{\Theta}(\mathbf{x})$ is not known, and it is directly coupled to the control policy, we propose a learning-based approach to jointly learn both the control policy and the metric with respect to which the policy exhibits contraction. We refer to such a policy as a deep contraction policy. To this end, let us start by assuming that we know a control policy $\mathbf{u}(\mathbf{x})$ that makes $f'(\mathbf{x}, \mathbf{u}(\mathbf{x}))$ contractive. Consider now that we want to learn a corresponding contraction metric. Let this contraction metric be represented by a model $\hat{\boldsymbol{\Theta}}(\mathbf{x}; \mathbf{w}_\Theta)$ which is parameterized by weights $\mathbf{w}_\Theta$. We then obtain the best parameters of this contraction metric, denoted by $\mathbf{w}^\star_\Theta$, from

$$\mathbf{w}^\star_\Theta = \underset{\mathbf{w}_\Theta}{\operatorname{argmin}}\; L_{\boldsymbol{\Theta}}(\mathcal{D}'; \mathbf{w}_\Theta), \tag{12}$$

where

$$L_{\boldsymbol{\Theta}}(\mathcal{D}'; \mathbf{w}_\Theta) = \mathbb{E}_{\mathcal{D}'}\Big(\|\hat{\boldsymbol{\Theta}}(\mathbf{x}'_{t+1}; \mathbf{w}_\Theta)(\tilde{\mathbf{x}}'_{t+1} - \mathbf{x}'_{t+1})\| - \|\hat{\boldsymbol{\Theta}}(\mathbf{x}_t; \mathbf{w}_\Theta)\Delta\mathbf{x}_t\| \Big). \tag{13}$$

The term $\|\hat{\boldsymbol{\Theta}}(\mathbf{x}'_{t+1}; \mathbf{w}_\Theta)(\tilde{\mathbf{x}}'_{t+1} - \mathbf{x}'_{t+1})\| - \|\hat{\boldsymbol{\Theta}}(\mathbf{x}_t; \mathbf{w}_\Theta)\,\Delta\mathbf{x}_t\|$ is an approximate measure of the contraction condition (11) which ideally should be negative for all elements of $\mathcal{D}'$. Since enforcing (11) directly results in a non-differentiable optimization, we minimize (13) as a proxy

for (11). Note that $L_{\mathbf{\Theta}}$ is computed over all data points in $\mathcal{D}'$. When paired with differentiable contraction metric $\hat{\mathbf{\Theta}}(\mathbf{x};\mathbf{w}_\Theta)$, the choice of loss function (13) is differentiable and is amenable to gradient decent optimization.

Now, let's consider the more general case where both the policy and its contraction metric are unknown. We want to learn both the state-feedback policy and the contraction metric together. We want to learn a control policy represented by a function approximator $\hat{\mathbf{u}}(\mathbf{x};\mathbf{w_u})$, parameterized by weights $\mathbf{w_u}$, such that the closed-loop system is contractive with respect to the metric model $\hat{\mathbf{\Theta}}$. To achieve this, we propagate the initial data points in $\mathcal{D}'$ with the control policy model $\hat{\mathbf{u}}(\mathbf{x};\mathbf{w_u})$ as $\mathbf{x}'_{t+1} = f'(\mathbf{x}_t, \hat{\mathbf{u}}(\mathbf{x}_t;\mathbf{w_u}))$ and $\tilde{\mathbf{x}}'_{t+1} = f'(\tilde{\mathbf{x}}_t, \hat{\mathbf{u}}(\tilde{\mathbf{x}}_t;\mathbf{w_u}))$.

We obtain the parameters of the contraction metric $\mathbf{w}_\Theta$, denoted by $\mathbf{w}^\star_\Theta$, and the parameters of the control policy $\mathbf{w_u}$, denoted by $\mathbf{w}^\star_{\mathbf{u}}$, by minimizing a loss function L_u over the data set $\mathcal{D}'$

$$(\mathbf{w}^\star_{\mathbf{u}}, \mathbf{w}^\star_\Theta) = \underset{\mathbf{w_u},\mathbf{w}_\Theta}{\operatorname{argmin}}\ L_u(\mathcal{D}';\mathbf{w_u},\mathbf{w}_\Theta), \tag{14}$$

where

$$L_u(\mathcal{D}';\mathbf{w_u},\mathbf{w}_\Theta) = \mathbb{E}_{\mathcal{D}'}\Big(\|\hat{\mathbf{\Theta}}(\mathbf{x}'_{t+1};\mathbf{w}_\Theta)(\tilde{\mathbf{x}}_{t+1} - \mathbf{x}'_{t+1}))\| - \|\hat{\mathbf{\Theta}}(\mathbf{x}_t;\mathbf{w}_\Theta)\Delta\mathbf{x}_t\|\Big). \tag{15}$$

Loss function (15) ensures that the region of interest $\mathcal{X}$ is contractive with respect to $\hat{\mathbf{\Theta}}$ and the learned dynamics model f'. However, so far, there has been no mechanism to ensure that the unique equilibrium of the contractive system is indeed the desired reference value $\mathbf{x}^r$. To alleviate this, we need the learning process to be aware of the desired reference value, which we would like to be the equilibrium of the contraction region. The measure of awareness that we introduce is based on the ability of the controller $\hat{\mathbf{u}}(\mathbf{x};\mathbf{w_u})$ to steer the system from an initial state $\mathbf{x}_0 \in \mathcal{X}$ to the desired state value $\mathbf{x}^r$ in k time steps, i.e. how close $\mathbf{x}'_k$ gets to $\mathbf{x}^r$ where $\mathbf{x}'_k$ is the k^{th} state value of the process f'. Therefore, to enforce the system's states to contract to $\mathbf{x}^r$, we add another penalty term to our loss function to obtain the final loss function utilized for learning the policy and contraction metric:

$$L(\mathcal{D},'\mathcal{Y};\mathbf{w}_\Theta,\mathbf{w_u}) = L_u(\mathcal{D}';\mathbf{w}_\Theta,\mathbf{w_u}) + \alpha L_{\text{tr}}(\mathcal{Y};\mathbf{w_u}), \tag{16}$$

where

$$L_{\text{tr}}(\mathcal{Y};\mathbf{w_u}) = \sum_{\mathbf{x}_0\in\mathcal{Y}} \|\mathbf{x}'_k(\mathbf{x}_0) - \mathbf{x}^r\|, \tag{17}$$

is the tracking loss with $\alpha \in \mathbb{R}_{>0}$ as the penalty factor. Here, $\mathbf{x}'_k(\mathbf{x}_0)$ is the k^{th} state value of the process $\mathbf{x}'_{t+1} = f'(\mathbf{x}'_t, \hat{\mathbf{u}}(\mathbf{x}'_t;\mathbf{w_u}))$, initialized at $\mathbf{x}'_0 = \mathbf{x}_0$ where $\mathbf{x}_0$ is drawn from a countable set $\mathcal{Y} \in \mathcal{X}$. The number of time steps k is set by the designer and, as the reader may infer, affects the transient behavior of the closed-loop system.

The algorithm describing Learning Contraction Policies from Offline Data is outlined in Algorithm 1

Algorithm 1: Learning Deep Contraction Policies.

Input :
  Data set: $(\mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{u}_t) \in \mathcal{D}$
  Set of sampled initial states : $\mathbf{x}_0 \in \mathcal{Y}$
  Reference state: $\mathbf{x}^r$
Init :
  $f'(\mathbf{x}_t, \mathbf{u}_t) \leftarrow$ learned dynamics using $\mathcal{D}$
  $\mathbf{w}_\Theta, \mathbf{w_u} \leftarrow$ randomly sampled
for N_{epochs} **do**
  Calculate $\mathbf{x}'_k$'s using $\mathcal{Y}$ and $f'(\mathbf{x}_t, \mathbf{u}(\mathbf{x}_t;\mathbf{w}_f))$
  Calculate $L_{\text{tr}}(\mathbf{w_u})$ using $\mathbf{x}_0 \in \mathcal{Y}$ and $\mathbf{x}_k$'s
  $\Delta\mathbf{x}_t \leftarrow$ uniform random sample from Δ_t
  Create $\mathcal{D}'$ using sampled $\Delta\mathbf{x}_t$
  Calculate $L_u(\mathbf{w}_\Theta, \mathbf{w_u})$ using $\Delta\mathbf{x}_t$'s and $\mathcal{D}'$
  $L(\mathbf{w}_\Theta, \mathbf{w_u}) \leftarrow L_u(\mathbf{w}_\Theta, \mathbf{w_u}) + \alpha L_{\text{tr}}(\mathbf{w_u})$
  Calculate gradients $\nabla_{\mathbf{w}_\Theta} L$ and $\nabla_{\mathbf{w_u}} L$
  Update $\mathbf{w}_\Theta$ and $\mathbf{w_u}$
end for

VI. Contraction of True Dynamics Under the Learned Policy

A major concern regarding control policy design using a learned model from offline data is that of model mismatch. In this section, we focus on verifying the convergent behavior of the true system dynamics under the learned policy. In order to bound the controller performance degradation, we assume a known upper bound on the Lipschitz constant of the model error $f(\mathbf{x}_t, \mathbf{u}_t) - f'(\mathbf{x}_t, \mathbf{u}_t)$, which we denote as $\mathsf{L}_{f-f'}$. In practice, such an upper bound may be estimated by fitting a Reverse Weibull distribution over the data set $\mathcal{D}$ [30], [31].

Lemma 1: Consider an unknown system $f(\mathbf{x},\mathbf{u})$ and its learned model $f'(\mathbf{x},\mathbf{u})$ with an upper-bound estimation on the Lipschitz constant of $f(\mathbf{x},\mathbf{u}) - f'(\mathbf{x},\mathbf{u})$ as $\mathsf{L}_{f-f'}$. The error between the learned model and the unknown system is bounded by ε, i.e. $\|f(\mathbf{x},\mathbf{u}) - f'(\mathbf{x},\mathbf{u})\| < \varepsilon$, for all $(\mathbf{x},\mathbf{u}) \in \mathcal{X}\times\mathcal{U}$ where

$$\varepsilon = \max_{(\mathbf{x}_t,\mathbf{x}_{t+1},\mathbf{u}_t)\in\mathcal{D}} \|\mathbf{x}_{t+1} - f'(\mathbf{x}_t,\mathbf{u}_t)\| + \mathsf{L}_{f-f'}\mathsf{D} \tag{18}$$

with $\mathsf{D} = \max_{(\mathbf{x},\mathbf{u})\in\mathcal{X}\times\mathcal{U}} \min_{(\mathbf{x}_t,\mathbf{u}_t)\in\mathcal{D}} \| \begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} - \begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix} \|$.

Proof: See Appendix.

The constant D in Lemma 1 is the maximum distance that a point $(\mathbf{x},\mathbf{u}) \in \mathcal{X}\times\mathcal{U}$ can have from its nearest data point $(\mathbf{x}_t,\mathbf{u}_t) \in \mathcal{D}$.

Deep contraction policy learning proposed in Algorithm 1 ideally ensures contractive behavior of the controlled learned system $f'(\mathbf{x}_t, \mathbf{u}(\mathbf{x}_t))$ at the states $\mathbf{x}_t \in \mathcal{D}$. More specifically, by defining an approximate measure of contraction condition (4) as $C_{g(\mathbf{x}_t)} : \mathcal{X}\times\Delta_t \to \mathbb{R}$

$$C_{g(\mathbf{x}_t)}(\mathbf{x}_t, \Delta\mathbf{x}_t) = \|\hat{\mathbf{\Theta}}(g(\mathbf{x}_t))(g(\tilde{\mathbf{x}}_t) - g(\mathbf{x}))\| - \|\hat{\mathbf{\Theta}}(\mathbf{x}_t)\Delta\mathbf{x}_t\|,$$

the controlled learned model being contractive is equivalent to $\mathbb{E}_{\Delta\mathbf{x}_t}(C_{f'(\mathbf{x}_t,\hat{\mathbf{u}}(\mathbf{x}_t))}(\mathbf{x}_t,\Delta\mathbf{x}_t)) < 0$ for all $\mathbf{x}_t \in \mathcal{D}$ and $\Delta\mathbf{x}_t \in \Delta_t$. Hence, it remains for us to verify whether the learned policy exhibits contraction with the true unknown system dynamics in the sense of contraction condition (4), i.e. $\mathbb{E}_{\Delta\mathbf{x}_t}(C_{f(\mathbf{x}_t,\hat{\mathbf{u}}(\mathbf{x}_t))}(\mathbf{x}_t,\Delta\mathbf{x}_t)) < 0$ for all $\mathbf{x}_t \in \mathcal{X}$ and $\Delta\mathbf{x}_t \in \Delta_t$. We seek to derive a condition under which we are guaranteed that the controlled true dynamics are also contractive with the learned policy. To arrive to such quantification, we begin with contraction of the learned model $C_{f'(\mathbf{x}_t,\hat{\mathbf{u}}(\mathbf{x}_t))}(\mathbf{x}_t,\Delta\mathbf{x}_t)$ at the training points, $\mathbf{x}_t \in \mathcal{D}$ and end with an upper bound estimation

of the contraction of the true dynamics $C_{f(\mathbf{x}_t,\hat{\mathbf{u}}(\mathbf{x}_t))}(\mathbf{x}_t,\Delta\mathbf{x}_t)$ at any points $\mathbf{x}\in\mathcal{X}$. The following Proposition establishes the condition under which the approximate contraction measure holds for the true robot dynamics under the trained $\mathbf{u}(\mathbf{x}_t)$.

Proposition 2: Let $\mathbb{E}_{\Delta\mathbf{x}_t}(C_{f'(\mathbf{x}_t,\hat{\mathbf{u}}(\mathbf{x}_t))}(\mathbf{x}_t,\Delta\mathbf{x}_t))<0$ for all $\mathbf{x}_t\in\mathcal{D}$. Let the Lipschitz constant of $\hat{\mathbf{\Theta}}_{ij}(\mathbf{x}_t)$, $f(\mathbf{x}_t,\mathbf{u}_t)-f'(\mathbf{x}_t,\mathbf{u}_t)$, $f'(\mathbf{x}_t,\mathbf{u}_t)$, $\hat{\mathbf{u}}(\mathbf{x}_t)$, $f'(\mathbf{x}_t,\hat{\mathbf{u}}(\mathbf{x}_t))$ and $C_{f'(\mathbf{x}_t,\hat{\mathbf{u}}(\mathbf{x}_t))}(\mathbf{x}_t,\Delta\mathbf{x}_t)$ be given as $\mathsf{L}_{\mathbf{\Theta}_{ij}},\mathsf{L}_{f-f'},\mathsf{L}_{f'},\mathsf{L}_{\mathbf{u}},\mathsf{L}_{f'_u}$, and L_C, respectively. Additionally, let $|\hat{\mathbf{\Theta}}_{ij}(\mathbf{x}_t)|<\gamma$, λ be given by (10), and ε be given as in (18). Then the true dynamics (8) are contractive under the trained controlled policy $\hat{\mathbf{u}}(\mathbf{x}_t)$, i.e. $\mathbb{E}_{\Delta\mathbf{x}_t}(C_{f(\mathbf{x}_t,\hat{\mathbf{u}}(\mathbf{x}_t))}(\mathbf{x}_t,\Delta\mathbf{x}_t))<0$ for all $\mathbf{x}_t\in\mathcal{X}$ and $\Delta\mathbf{x}_t\in\Delta_t$, if

$$\zeta+\lambda(\varepsilon\tau\mathsf{L}_{f'_u}+(\varepsilon\tau+n\gamma)\mathsf{L}_{f-f'}(1+\mathsf{L}_{\mathbf{u}}))<0, \tag{19}$$

where $\tau=\sqrt{\sum_{ij}\mathsf{L}^2_{\mathbf{\Theta}_{ij}}}$ and $\zeta=\max_{\mathbf{x}\in\mathcal{X}}\min_{\mathbf{x}_t\in\mathcal{D}}C(\mathbf{x}_t)+\mathsf{L}_C\|\mathbf{x}_t-\mathbf{x}\|$.

Proof: See Appendix.

VII. Implementation & Evaluation

We evaluate the performance of the contraction policies in a set of goal-reaching robotic tasks by comparing our method against a number of offline control methods suitable for systems with learned dynamics models. In particular, we compare our framework with the following algorithms:

1) *MPC:* An iterative Linear Quadratic Controller (iLQR) as described in [32] ran in a receding horizon fashion.
2) *Learning Without Contraction:* To evaluate the effectiveness of the contraction penalty, we further evaluate the robot's performance in the absence of any contraction terms in the loss function.
3) *Reinforcement Learning:* We also use the state-of-the-art offline RL method Conservative Q-Learning (CQL) [33] for further comparisons.

We evaluate the performance of our approach on two different robotic settings involving nonlinear dynamical systems of varying complexity represented by neural networks. The dynamics of these systems have closed-form expressions, but it is assumed that we do not have access to such expressions. We assume that we only have access to a set of system trajectories and learn a dynamics model from the state-action trajectories. The learned dynamics are represented as neural networks to the model-based control methods: deep contraction policy, MPC controller, and contraction-free learning. The RL implementation develops the policy directly from the same offline data set that is used to train the dynamics model in a model-free fashion. This allows us to implement our algorithm on the learned systems while having an analytical baseline to compare against to quantify robustness. Additionally, we consider state and control sets $\mathcal{X},\mathcal{U}$ defined by box constraints in order to constrain the size of the training data. Clearly for such constraints, $\mathcal{X}$ is convex. The dynamical systems we have chosen for our performance evaluation are as follows:

1) *2D Planar Car:* A planar vehicle that is capable of controlling its acceleration, α, and angular velocity, ω. Here $\mathbf{x}:=[p_x,p_y,\theta,v]$ and $\mathbf{u}:=[\alpha,\omega]$ where p_x,p_y are the planar positions, v is the velocity, and θ is the heading angle. The system dynamics are governed by: $\dot{\mathbf{x}}=[v\cos(\theta),v\sin(\theta),\omega,\alpha]^\top$.
2) *3D Drone:* An adaptation of a drone model that is given by [10] and [34]. This model describes an aerial vehicle capable of directly controlling the rate of change of its normalized thrust $\dot{F}$, and Euler Angles, $\dot{\phi},\dot{\theta},\dot{\psi}$. Here $\mathbf{x}:=[p_x,p_y,p_z,v_x,v_y,v_z,F,\phi,\theta,\psi]$ and $\mathbf{u}:=[\dot{\phi},\dot{\theta},\dot{\psi}]$ where p_i,v_i are the translational positions and velocities along the i_{th} axis, respectively. Omitting the first order integrators in p_i,F,ϕ,θ,ψ for brevity, the dynamics can then be expressed as $[\dot{v}_x,\dot{v}_y,\dot{v}_z]=[-F\sin(\theta),F\cos(\theta)\sin(\phi),g-\cos(\theta)\cos(\phi)]$, where g is the acceleration due to gravity.

For both systems we assume a timestep of $\Delta t=0.1$s and a final time of $T=10$s.

A. Learning System Dynamics

All of the continuous dynamical systems described above are represented to our controllers as fully connected neural networks which capture the discretization of the model integration: $\mathbf{x}_{t+1}-\mathbf{x}_t\approx f'(\mathbf{x}_t,\mathbf{u}_t;\mathbf{w}_f)$. The training dataset $\mathcal{D}$ is generated by aggregating reference trajectories through the state space generated from an iLQR controller applied directly to the true dynamics $f(\mathbf{x}_t,\mathbf{u}_t)$. The reference trajectories $\Phi(\mathbf{x}_t,\mathbf{u}_t)$ were chosen such that $\mathbf{x}_t\in\mathcal{X}$ and $\mathbf{u}_t\in\mathcal{U}$ for all t. Trajectory data was used in order to implement a discounted multistep prediction error as in [35] until sufficient integration accuracy was achieved.

B. Controller Implementation

The contraction metric and control policy neural networks, $\hat{\mathbf{\Theta}}(\mathbf{x}_t;\mathbf{w}_\Theta)$ and $\hat{\mathbf{u}}(\mathbf{x}_t;\mathbf{w}_\mathbf{u})$, are trained according to Algorithm 1. For our ablation study, we remove the contraction penalty term and simply find a policy for minimizing the tracking error norm. Without a contraction penalty, the impact of contraction conditions during the learning process vanishes. In order to create a controller for this case, each $\mathbf{x}_t\in\mathcal{D}$ is forward evolved a number of time steps under the learned control policy and trained with a discounted cumulative loss of the tracking error norms over each timestep.

For the MPC controller, the iLQR planner utilizes the learned dynamics model in order to calculate the linearization relative to the state and control inputs. This linearization is used along with weighting matrices $\mathbf{Q}=100\mathbf{I},\mathbf{R}=1000\mathbf{I}$ in order to calculate an iLQR control law.

In order to train an offline reinforcement learning algorithm like CQL, the algorithm needs access to state, action, and reward pairs. We reutilize the offline iLQR trajectories created for dynamics learning as training episodes for the offline CQL RL algorithm. The reward at each time step is taken to be the negative norm of the tracking error at the next time step given the currently taken action.

C. Performance Results

In order to compare the performance of our method with the alternative implementations outlined above, we propose a number of metrics to compare the controllers:

- The time evolution of the tracking error, to quantify the controllers' ability to converge to the desired reference $\mathbf{x}^r$.
- The converged tracking error versus the initial tracking error, to quantify the controllers' ability to operate over the working space $\mathcal{X}\times\mathcal{U}$.

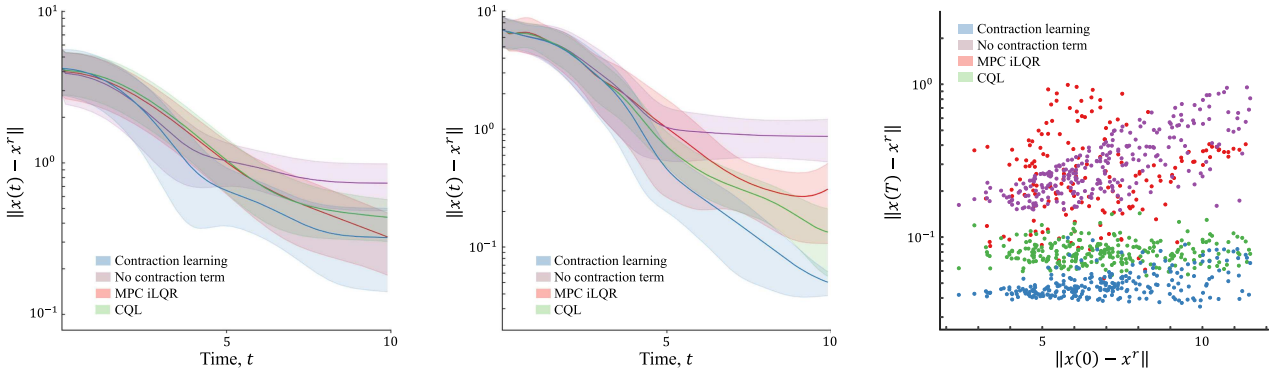

Fig. 3. Norm of the tracking error over a collection of 256 initial states for the 2D car problem (left) and the 3D drone problem (middle): The y axis is shown on a logarithmic scale and results capture the mean plus and minus one standard deviation. We see that the additional complexity of the 3D drone over the 2D car model allows for greater variation in algorithm performance. Norm of the average final tracking error versus the norm of the initial tracking error (right): As the initial states approach the boundary of the region of interest, controller performance tends to degrade.

- Root Mean Square Error (RMSE) of the tracking error versus learned model loss, to quantify the controllers' ability to deal with model mismatch.

For all analyses, the controllers were each presented with an identical set of 256 initial conditions within $\mathcal{X}$. The control methods were implemented as described above in an attempt to drive these initial states to the desired reference $\mathbf{x}^r$. The results were aggregated over the 256 initial conditions for the 2D car and 3D drone.

In the time evolution analysis, desirable controllers have trajectories that quickly converge, have minimal tracking error norm, and have high convergence precision. Results directly comparing all of the controllers relative to this performance measure for the two dynamical systems are given in Fig. 3 (left and middle). The results show that over the two different systems and a multitude of initial conditions, the contraction learning policy performs well relative to the proposed comparison controllers. For the simpler dynamic system of the two, the 2D planar car, the results are comparable among all controllers but favor the contraction controller, while the more complex drone environment shows the clear benefits of our approach. The enforcement of contraction conditions forces nearby trajectories to converge to one another, and when near the reference point, this has the effect of reducing the norm of the tracking error further than the systems designed without contraction in mind. The contraction controller consistently has the lowest mean norm of the tracking error over all the sampled initial states.

Comparing the converged tacking error, in this case, the average of the final 10 timestamps of each trajectory, versus the initial tracking error gives insight into the performance of the controllers' over the entire state and control space $\mathcal{X}$ and $\mathcal{U}$. Cases with a higher initial tracking error represent trajectories that start closer towards the boundary of our working space $\mathcal{X} \times \mathcal{U}$. Favorable controllers are ones in which the converged tracking error remains constant or grows slowly as the initial tracking error increases. Fig. 3 (right) directly shows this comparison. The results here clearly show that the MPC controller and the learned policy without the contraction terms have difficulty as the initial state norm gets further from the desired reference. For the MPC controller, the poor performance is likely caused by not having expressive enough dynamics due to the repeated linearization process. The contraction-free policy shows good performance for small initial tracking errors but quickly degrades as this value grows. Such a control method acts extremely locally. Training a collection of states to converge to the reference without the additional contraction structure does not yield favorable stability properties. The CQL policy and deep contraction learning generate trajectories with minimal degradation as the tracking error increases, with the contraction learning method consistently having the highest degree of performance.

Finally, for the 3D drone scenario, the impact of the learned dynamics model quality on the model-based controllers' performance is studied. To this end, multiple models of different quality were learned from the same offline data set. Since the CQL policy is directly learned from the offline data and does not utilize the learned dynamics model, this method is omitted from this analysis. In this case, favorable controllers are ones in which the error grows slowly with increasing model inaccuracy. Comparison of the RMSE values of the tracking error norm over the length of the trajectories for the varying quality dynamics models are shown in Table I. The contraction learning model shows favorable performance as dynamics model mismatch increases due to the robustness properties discussed in Section VI. For particularly low-quality learned dynamics models, we even see that the deep contraction policy is able to generate stabilizing controllers where the contraction-free policy and MPC controller fail to do so.

D. *Non-Control Affine Systems*

In order to quantify the ability of our deep contraction policy learning to generalize to more complex systems, we perform an illustrative analysis of our controller on the double pendulum model given in [36]. Such a system is chaotic with a non-affine control input. Fig. 4 shows the comparison of two scenarios where the designed controller was implemented on both the learned model and the true dynamics. While the controller is able to stabilize both the learned model and the true dynamics, it also governs both systems towards the reference values. The controller is able to drive the system states to exactly the reference values when applied to the learned model. However, when applied to the true dynamics, the controller positions the arms with a slight positional error while keeping the angular velocity at zero.

TABLE I
TRACKING ERROR NORM RMSE, 3D DRONE

Dynamics Model	Test Loss	Contraction learning	No contraction term	MPC iLQR
1	5.67e-05	$\mathbf{1.905 \pm 0.651}$	2.391 ± 0.968	2.161 ± 0.913
2	8.19e-05	$\mathbf{2.026 \pm 0.676}$	2.528 ± 0.880	2.966 ± 1.593
3	1.14e-04	$\mathbf{2.214 \pm 0.889}$	3.315 ± 0.917	6.458 ± 2.284
4	1.58e-04	$\mathbf{2.891 \pm 1.061}$	5.392 ± 1.290	N/A*
5	2.64e-04	$\mathbf{3.571 \pm 1.252}$	N/A*	N/A*

*Values of N/A represent cases where sufficiently stabilizing controllers were not generated.

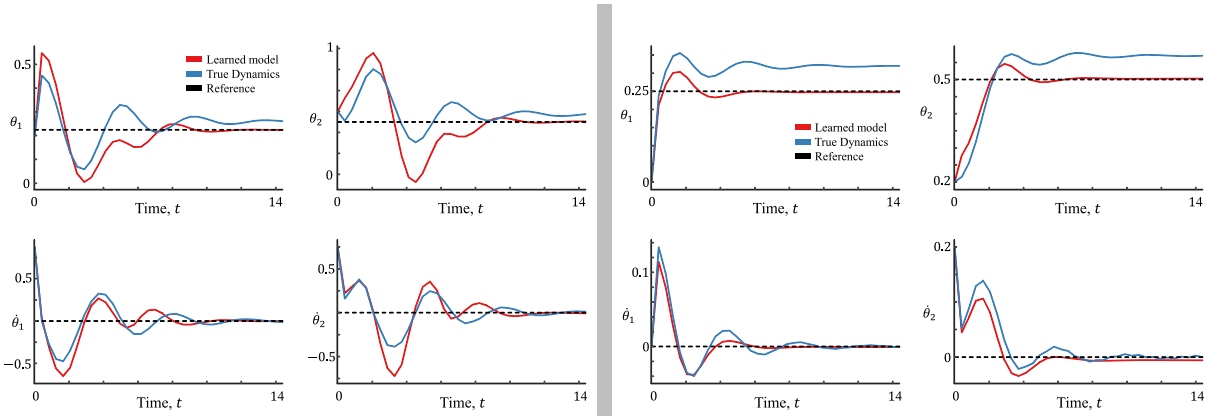

Fig. 4. Angular position and angular velocity of the double pendulum system. The controlled learned model and controlled true dynamics are shown. Results are shown for two different sets of initial conditions.

VIII. CONCLUSION AND FUTURE WORK

In this letter, we established a framework for learning a converging control policy for an unknown system from offline data. We leveraged Contraction theory and proposed a data augmentation method for encoding the contraction conditions directly into the loss function. We jointly learned the control policy and its corresponding contraction metric. We compared our method with several state-of-the-art control algorithms and showed that our method provides faster convergence, a smaller tracking error, and lower variance of trajectories. For our future work, we would like to extend the current work to develop the stochastic confidence bounds for our proposed control design approach.

APPENDIX

Proof of Lemma 1: We ground our error analysis on the training error of the tuples $(\mathbf{x}_t, \mathbf{u}_t) \in \mathcal{D}$ and propagate the error to the general state and control tuples $(\mathbf{x}, \mathbf{u}) \in \mathcal{X} \times \mathcal{U}$.

$$\begin{aligned}\|f(\mathbf{x},\mathbf{u}) - f'(\mathbf{x},\mathbf{u})\| &\le \|f(\mathbf{x}_t,\mathbf{u}_t) - f'(\mathbf{x}_t,\mathbf{u}_t)\| \\ &+ \|(f(\mathbf{x},\mathbf{u}) - f'(\mathbf{x},\mathbf{u})) - (f(\mathbf{x}_t,\mathbf{u}_t) - f'(\mathbf{x}_t,\mathbf{u}_t))\| \\ &\le \|f(\mathbf{x}_t,\mathbf{u}_t) - f'(\mathbf{x}_t,\mathbf{u}_t)\| + \mathsf{L}_{f-f'} \left\| \begin{bmatrix}\mathbf{x}\\ \mathbf{u}\end{bmatrix} - \begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix} \right\|.\end{aligned}$$

The first and the second inequalities are obtained by adding and subtracting the terms $f(\mathbf{x}_t, \mathbf{u}_t)$ and $f'(\mathbf{x}_t, \mathbf{u}_t)$, and also using the norm and Lipschitz constant properties. If we define $E(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}, \mathbf{u})$ as the right-hand side of the second inequality, then $\max_{(\mathbf{x},\mathbf{u})\in\mathcal{X}\times\mathcal{U}} \min_{(\mathbf{x}_t,\mathbf{u}_t)\in\mathcal{D}} E(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}, \mathbf{u}) \le \varepsilon$ where

$$\varepsilon = \max_{(\mathbf{x}_t,\mathbf{x}_{t+1},\mathbf{u}_t)\in\mathcal{D}} \|\mathbf{x}_{t+1} - f'(\mathbf{x}_t, \mathbf{u}_t)\| + \mathsf{L}_{f-f'}\mathsf{D},$$

which concludes the proof.

Proof of Proposition 2: We want to derive a sufficient condition which ensures that contraction condition (11) holds for the true dynamics model. Using the learned dynamics model, the left-hand side of (11) for $\mathbf{x}_t \in \mathcal{X}$ can be bounded for the true dynamics as

$$\begin{aligned}&\|\hat{\mathbf{\Theta}}(\mathbf{x}_{t+1})(\tilde{\mathbf{x}}_{t+1} - \mathbf{x}_{t+1})\| - \|\hat{\mathbf{\Theta}}(\mathbf{x}_t)\Delta\mathbf{x}_t\| \\ &\le \|\hat{\mathbf{\Theta}}(\mathbf{x}'_{t+1})(\tilde{\mathbf{x}}'_{t+1} - \mathbf{x}'_{t+1})\| - \|\hat{\mathbf{\Theta}}(\mathbf{x}_t)\Delta\mathbf{x}_t\| \\ &+ \|(\hat{\mathbf{\Theta}}(\mathbf{x}_{t+1}) - \hat{\mathbf{\Theta}}(\mathbf{x}'_{t+1}))(\tilde{\mathbf{x}}'_{t+1} - \mathbf{x}'_{t+1})\| \\ &+ \|\hat{\mathbf{\Theta}}(\mathbf{x}'_{t+1})((\tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}'_{t+1}) - (\mathbf{x}_{t+1} - \mathbf{x}'_{t+1}))\| \\ &+ \|(\hat{\mathbf{\Theta}}(\mathbf{x}_{t+1}) - \hat{\mathbf{\Theta}}(\mathbf{x}'_{t+1}))((\tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}'_{t+1}) - (\mathbf{x}_{t+1} - \mathbf{x}'_{t+1}))\|,\end{aligned}$$

where $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \hat{\mathbf{u}}(\mathbf{x}_t))$, $\tilde{\mathbf{x}}_{t+1} = f(\tilde{\mathbf{x}}_t, \hat{\mathbf{u}}(\tilde{\mathbf{x}}_t))$, $\mathbf{x}'_{t+1} = f'(\mathbf{x}_t, \hat{\mathbf{u}}(\mathbf{x}_t))$, and $\tilde{\mathbf{x}}'_{t+1} = f'(\tilde{\mathbf{x}}_t, \hat{\mathbf{u}}(\tilde{\mathbf{x}}_t))$. The inequality holds due to addition and subtraction of proper terms and norm properties. The inequality can be further simplified using the Frobenius norm of the contraction metric $\hat{\mathbf{\Theta}}$. Since, by assumption, the entries of the contraction metric are bounded by γ, we have $\|\hat{\mathbf{\Theta}}(\mathbf{x})\|_F \le n\gamma$. Having an upper bound estimate of the Lipschitz constant of entries of the contraction metric $\mathsf{L}_{\mathbf{\Theta}_{ij}}$ and recalling that $\|(\tilde{\mathbf{x}}'_{t+1} - \mathbf{x}'_{t+1})\| \le \varepsilon$ from Lemma 1, leads to the result $\|(\hat{\mathbf{\Theta}}(\mathbf{x}_{t+1}) - \hat{\mathbf{\Theta}}(\mathbf{x}'_{t+1}))\|_F \le \varepsilon\sqrt{\sum_{ij}\mathsf{L}^2_{\mathbf{\Theta}_{ij}}}$. In addition, using the estimated Lipschitz constant

$\mathsf{L}_{f-f'}$, we have that $\|((\tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}'_{t+1}) - (\mathbf{x}_{t+1} - \mathbf{x}'_{t+1}))\| \leq \mathsf{L}_{f-f'} \| \begin{bmatrix} \tilde{\mathbf{x}} \\ \mathbf{u}(\tilde{\mathbf{x}}) \end{bmatrix} - \begin{bmatrix} \mathbf{x} \\ \mathbf{u}(\mathbf{x}) \end{bmatrix} \|$. Now, using the Lipschitz constant of $\mathbf{u}(\mathbf{x})$ as $\mathsf{L}_{\mathbf{u}}$, we have that $\|((\tilde{\mathbf{x}}_{t+1} - \tilde{\mathbf{x}}'_{t+1}) - (\mathbf{x}_{t+1} - \mathbf{x}'_{t+1}))\| \leq \mathsf{L}_{f-f'}\lambda(1 + \mathsf{L}_{\mathbf{u}})$. Finally, we can write the following inequality:

$$\begin{aligned} &\|\hat{\boldsymbol{\Theta}}(\mathbf{x}_{t+1})(\tilde{\mathbf{x}}_{t+1} - \mathbf{x}_{t+1})\| - \|\hat{\boldsymbol{\Theta}}(\mathbf{x}_t)\Delta\mathbf{x}_t\| \\ &\leq \|\hat{\boldsymbol{\Theta}}(\mathbf{x}'_{t+1})(\tilde{\mathbf{x}}'_{t+1} - \mathbf{x}'_{t+1})\| - \|\hat{\boldsymbol{\Theta}}(\mathbf{x}_t)\Delta\mathbf{x}_t\| \\ &+ \lambda(\varepsilon\tau\mathsf{L}_{f'_u} + (\varepsilon\tau + n\gamma)\mathsf{L}_{f-f'}(1 + \mathsf{L}_{\mathbf{u}})), \end{aligned} \tag{20}$$

where $\tau = \sqrt{\sum_{ij} \mathsf{L}^2_{\boldsymbol{\Theta}_{ij}}}$. With Lipschitz constant L_C, we can derive an upper bound for $C_{f'(\mathbf{x}_t, \hat{\mathbf{u}}(\mathbf{x}_t))}(\mathbf{x}, \Delta\mathbf{x}_t)$, $\mathbf{x}_t \in \mathcal{X}$ and $\Delta\mathbf{x}_t \in \Delta_t$, such that $C_{f'(\mathbf{x}, \hat{\mathbf{u}}(\mathbf{x}))}(\mathbf{x}_t, \Delta\mathbf{x}_t) < \zeta$ where

$$\zeta = \max_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{x}_t \in \mathcal{D}} C_{f'(\mathbf{x}_t, \hat{\mathbf{u}}(\mathbf{x}_t))}(\mathbf{x}_t, \Delta\mathbf{x}_t) + \mathsf{L}_C\|\mathbf{x}_t - \mathbf{x}\|.$$

Finally, by taking the expectation on Equation (20), we get

$$\begin{aligned} &\mathbb{E}_{\Delta\mathbf{x}_t}\left(\|\hat{\boldsymbol{\Theta}}(\mathbf{x}_{t+1})(\tilde{\mathbf{x}}_{t+1} - \mathbf{x}_{t+1})\| - \|\hat{\boldsymbol{\Theta}}(\mathbf{x}_t)\Delta\mathbf{x}_t\|\right) \\ &\leq \zeta + \lambda(\varepsilon\tau\mathsf{L}_{f'_u} + (\varepsilon\tau + n\gamma)\mathsf{L}_{f-f'}(1 + \mathsf{L}_{\mathbf{u}})) \end{aligned}$$

which concludes the proof.

REFERENCES

[1] W. Lohmiller and J.-J. E. Slotine, "On contraction analysis for non-linear systems," *Automatica*, vol. 34, no. 6, pp. 683–696, 1998.

[2] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," 2020, *arXiv:2005.01643*.

[3] N. Boffi, S. Tu, N. Matni, J. Slotine, and V. Sindhwani, "Learning stability certificates from data," 2020. *arXiv:2008.05952*.

[4] H. Tsukamoto, S. Chung, and J. Slotine, "Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview," *Annu. Rev. Control*, vol. 52, pp. 135–169, 2021.

[5] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[6] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," 2020, *arXiv:2006.04779*.

[7] T. Yu *et al.*, "Mopo: Model-based offline policy optimization," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 14129–14142, 2020.

[8] L. Kaiser *et al.*, "Model-based reinforcement learning for Atari," 2019, *arXiv:1903.00374*.

[9] T. Moerland, J. Broekens, and M. Jonker, "Model-based reinforcement learning: A survey," 2020, *arXiv:2006.16712*.

[10] D. Sun, S. Jha, and C. Fan, "Learning certified control using contraction metric," 2020, *arXiv:2011.12569*.

[11] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 908–919.

[12] S. Vaskov *et al.*, "Towards provably not-at-fault control of autonomous robots in arbitrary dynamic environments," in *Robotics: Sci. Syst. XV*, Univ. Freiburg, Freiburg im Breisgau, Germany, Jun. 2019.

[13] R. Tedrake, "LQR-trees: Feedback motion planning on sparse randomized trees," in *Proc. Robotics: Sci. Syst.*, Seattle, WA, Jun. 2009.

[14] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *Int. J. Robot. Res.*, vol. 36, no. 8, pp. 947–982, 2017.

[15] S. Herbert, M. Chen, S. Han, S. Bansal, J. Fisac, and C. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, 2017, pp. 1517–1522.

[16] S. Bansal, M. Chen, F. J. Fisac, and C. Tomlin, "Safe sequential path planning of multi-vehicle systems under presence of disturbances and imperfect information," in *Amer. Control Conf.*, 2017, pp. 5550–5555.

[17] H. K. Khalil and J. W. Grizzle, *Nonlinear Systems*, vol. 3. Upper Saddle River, NJ, USA: Prentice Hall, 2002.

[18] J. Choi, F. Castañeda, C. Tomlin, and K. Sreenath, "Reinforcement learning for safety-critical control under model uncertainty, using control Lyapunov functions and control barrier functions," in *Robot.: Sci. Syst.*, Corvalis, Oregon, USA, Jul. 2020, doi: 10.15607/RSS.2020.XVI.088.

[19] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Proc. Mach. Learn. Res.*, 2020, pp. 708–717.

[20] A. Zaki, A. El-Nagar, M. El-Bardini, and F. Soliman, "Deep learning controller for nonlinear system based on Lyapunov stability criterion," *Neural Comput. Appl.*, vol. 33, no. 5, pp. 1515–1531, 2021.

[21] S. Richards, F. Berkenkamp, and A. Krause, "The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Proc. Conf. Robot Learn.*, 2018, pp. 466–476.

[22] A. Robey *et al.*, "Learning control barrier functions from expert demonstrations," in *Proc. IEEE Conf. Decis. Control*, 2020, pp. 3717–3724.

[23] S. Chen, M. Fazlyab, M. Morari, G. Pappas, and V. Preciado, "Learning Lyapunov functions for hybrid systems," in *Proc. 24th Int. Conf. Hybrid Syst.: Comput. Control*, 2021, pp. 1–11.

[24] H. Tsukamoto and S.-J. Chung, "Learning-based robust motion planning with guaranteed stability: A contraction theory approach," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 6164–6171, Oct. 2021.

[25] S. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian mixture models," *Trans. Robot.*, vol. 27, no. 5, pp. 943–957, 2011.

[26] J. Umlauft and S. Hirche, "Learning stable stochastic nonlinear dynamical systems," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3502–3510.

[27] S. Singh, V. Sindhwani, J. Slotine, and M. Pavone, "Learning stabilizable dynamical systems via control contraction metrics," 2018, *arXiv:1808.00113*.

[28] A. Taylor, V. Dorobantu, H. Le, Y. Yue, and A. Ames, "Episodic learning with control Lyapunov functions for uncertain robotic systems," in *Proc. Int. Conf. Intell. Robots Syst.*, 2019, pp. 6878–6884.

[29] J. Kolter and G. Manek, "Learning stable deep dynamics models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 11128–11136.

[30] G. Wood and B. Zhang, "Estimation of the lipschitz constant of a function," *J. Glob. Optim.*, vol. 8, no. 1, pp. 91–103, 1996.

[31] C. Knuth, G. Chou, N. Ozay, and D. Berenson, "Planning with learned dynamics: Probabilistic guarantees on safety and reachability via lipschitz constants," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 5129–5136, Jul. 2021.

[32] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 4906–4913.

[33] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning," 2020, *arXiv:2006.04779*.

[34] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5883–5890.

[35] A. Venkatraman, M. Hebert, and J. Bagnell, "Improving multi-step prediction of learned time series models," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 3024–3030.

[36] T. Stachowiak and T. Okada, "A numerical analysis of chaos in the double pendulum," *Chaos, Solitons Fractals*, vol. 29, no. 2, pp. 417–422, 2006.