2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

An Iterative Online Approach to Safe Learning in Unknown Constrained Environments

Minh Vu[1] and Shen Zeng[1]

Abstract— **This paper presents an iterative learning technique to safely guide a nonlinear system with unknown dynamics through an environment with unspecified constraints. The presented approach leverages the system's local dynamics to incrementally explore the environment and learn the appropriate control, which allows us to avoid the data-intensive task of learning an accurate global system model. Due to the local nature of this approach, the system's safe operating region does not need to be pre-specified as long as local areas of the constraints can be identified when the system approaches those areas. The functionality and efficiency of the proposed approach are demonstrated through simulation of a unicycle and a high-dimensional nonlinear quadcopter, indicating the system's ability to learn dynamics from data and safely navigate unknown environments.**

I. Introduction

Deploying autonomous robots in safety-critical applications requires systems that are capable of rapid learning and adapting to unknown environments while satisfying essential constraints. One such example is the Perseverance rover in NASA's Mars 2020 mission, which needs to explore the unknown terrain of Mars to search for habitability and potential evidence of life. Changes in the Martian atmosphere like dust storms and solar flares, however, can significantly alter the environment and dynamics of the rover, requiring the system to promptly learn its new dynamics to navigate safely in the newly changed environment.

These intriguing and crucial applications have sparked a lot of research interest in machine learning and control communities [1]–[3]. A primary and shared research objective is to develop systematic approaches to safe learning and adaptation that intelligently enhance the autonomous operation of robots (as compared to the existing methods of manually designing safe fallback mechanisms [4]).

One such effort is the development of safe reinforcement learning (RL), which aims to learn a policy that maximizes the expected return while ensuring the satisfaction of safety constraints. To achieve this goal, safe RL approaches have utilized methods such as reward shaping, policy optimization with constraints, and expert demonstration [5]. Nevertheless, these model-free approaches often allow for a certain level of constraint violation and do not guarantee safety during learning. In fact, safety is learned through environmental interactions, and its assurance is only approximately achieved after a sufficient learning period [6].

To ensure safety during the learning process, a variety of model-based learning techniques have been proposed, including reachability analysis, Lyapunov-related methods, and model predictive control [1]–[3]. These approaches rely on a simplified model to keep the system within a predetermined safe region and have been demonstrated on physical platforms [7]. Despite these progresses, existing model-based learning methods still rely on the assumptions (or bounds) of model errors to ensure safety, which in some cases limit the performance of the learning algorithms. For examples, as model uncertainty increases, especially when the system dynamics is unknown, the assumed bounds must become highly conservative, thereby heavily restricting the learning performance and freedom [8].

In this paper, we present a novel approach to model-based learning that can ensure safety while promoting efficient learning and freedom in exploration. In particular, the advocated approach focuses on utilizing only the system's *local* dynamics for effective exploration and control. To achieve this, our approach first learns a local model of the system from data and then uses it to incrementally and safely guide the control process and further exploration. During the learning process, we subsequently alternate between control synthesis and exploration to maintain a good model quality (based on a mixed measure of desired metrics). This alternating mechanism (i.e., pausing the control and updating the model when it is required) allows us to gradually expand the valid region of the model and, more importantly, tailor this expansion to the purpose of control. This is the hallmark of our approach, which enables efficient and safe learning and ultimately minimizes the restrictive (global) model dependence found in existing model-based learning frameworks.

In the next section, we introduce problem formulation and some preliminary backgrounds. In Section III, we present the iterative control methodology, which enables the system to learn its own dynamics (from scratch) and safely navigate unknown constrained environments. We present the numerical results in Section IV and conclude the paper in Section V.

II. Problem Formulation

Consider the point-to-point control problem in the form

$$\begin{aligned} \underset{u_0,\ldots,u_{N-1}}{\text{minimize}} \quad & \|x_N - x_{\text{target}}\|^2 \\ \text{subject to} \quad & x_{k+1} = f(x_k, u_k), \quad x_0 = x_{\text{start}} \\ & g(x_k) \leq 0 \end{aligned} \tag{1}$$

where f and g are nonlinear functions denoting the system dynamics and constraints, $u_k \in \mathbb{R}^m$ are the control inputs, and $x_{\text{start}}, x_{\text{target}} \in \mathbb{R}^n$ are the initial state and the desired

[1]Department of Electrical and System Engineering, Washington University in St. Louis, St. Louis, MO, USA, emails: {minhvu, s.zeng}@wustl.edu.

terminal state, respectively. Note that for each time step, x_k is computed by iterating the system dynamics, i.e.,

$$\begin{aligned} x_k &= f(x_{k-1}, u_{k-1}) = f(f(x_{k-2}, u_{k-2}), u_{k-1}) \\ &= \cdots = f(\ldots, f(f(x_0, u_0), u_1), \ldots, u_{k-1}) \end{aligned}$$

which makes the above problem highly nested and nonlinear. In the context of data-driven control where f and g are unknown, a direct approach using this "global" viewpoint is known to be difficult [9]. In the following, we utilize an iterative control methodology to provide a "local" approach to this problem. In particular, we will break the above nonlinear coupled dynamics and simplify it into a linear relation (which is locally valid along a nominal controlled trajectory). Then, in each trial, we leverage this linear relation to make a small improvement toward the desired solution of (1). This approach allows us to leverage only the system local dynamics to iteratively and safely steer the system to the target without the need for a high-fidelity global model.

III. METHODS

In this Section, we first review the iterative methodology which was introduced in [10]. In section III-B, we build upon the basic idea of iterative control to equip the system with the ability to safely navigate unknown environments. In Section III-C, we consider the case where both the environment and the system dynamics are unknown, and the system must learn its local dynamics from data and subsequently use it to safely guide the control process and further exploration.

A. Basics of Iterative Control Methodology

Given an initial state x_0, a nominal control signal $U := \left[u_0^\top, u_1^\top, \ldots, u_{N-1}^\top\right]^\top$, and the resulting nominal state trajectory $X_U := [x_1^\top, x_2^\top, \ldots, x_N^\top]^\top$ obtained from the evolution of the system, we consider the following linearization

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

where $A_k := \dfrac{\partial f}{\partial x}(x_k, u_k)$, $B_k := \dfrac{\partial f}{\partial u}(x_k, u_k)$ denote the derivatives of f with respect to the state and the input, respectively. Then, by iterating (2), we have

$$\begin{aligned} \delta x_1 &= B_0 \delta u_0 \\ \delta x_2 &= A_1 B_0 \delta u_0 + B_1 \delta u_1 \\ &\vdots \\ \delta x_N &= A_{N-1} \ldots A_1 B_0 \delta u_0 + \cdots + B_{N-1} \delta u_{N-1}. \end{aligned}$$

$$\text{Let } H := \begin{bmatrix} B_0 & 0 & \cdots & 0 \\ A_1 B_0 & B_1 & & \vdots \\ \vdots & \vdots & & 0 \\ A_{N-1} \cdots A_1 B_0 & A_{N-1} \cdots A_2 B_1 & \cdots & B_{N-1} \end{bmatrix}.$$

Given sufficiently small $\Delta U := \left[\delta u_0^\top, \delta u_1^\top, \ldots, \delta u_{N-1}^\top\right]^\top$, the main idea of iterative control is to utilize the above linearizations to quantify changes of the system trajectory due to small perturbations of the nominal control input, i.e.,

$$\Delta X := \begin{bmatrix} \delta x_1 \\ \vdots \\ \delta x_N \end{bmatrix} = H \Delta U, \text{ and } X_{U+\Delta U} = X_U + H \Delta U \tag{3}$$

where $X_{U+\Delta U}$ denotes the resulting state trajectory obtained from the perturbed input $U + \Delta U$. The advantage of this approximation is that it simplifies the nonlinear "coupled" dynamics and enables us to effectively capture the effect of small input perturbations via a simple linear relation, i.e., $\Delta X = H \Delta U$.

Using this idea, given a nominal (arbitrary) control input U and the corresponding state trajectory X, in each trial, one can consider the optimization problem

$$\underset{\Delta U}{\text{minimize}} \quad \|x_N + H_N \Delta U - x_{\text{target}}\|^2 + \lambda \|\Delta U\|^2 \tag{4}$$

to steer the system step-by-step closer to a desired target, where $H_N = \begin{bmatrix} A_{N-1} \cdots A_1 B_0 & \cdots & B_{N-1} \end{bmatrix}$ denotes the N-horizontal block of H, and $\lambda \geq 0$ is a regularization parameter that enforces a penalty on the magnitude of ΔU to ensure incremental updates and the appropriateness of (3) (the appropriate choice of λ is important and will be discussed in detail in Section III-B). From [10], the iterative scheme for steering the system to the target without state constraints is as follows.

Algorithm 1 Steering the system to a target state

Require: Desired terminal state x_{target}, an initial input U.
1: Apply the input U to the system and calculate A_k, B_k.
2: Calculate H_N.
3: Solve for ΔU^* of the optimization problem (4).
4: Update the control input via $U = U + \Delta U^*$.
5: Repeat steps 1-4 until $\|x_N - x_{\text{target}}\| \leq \epsilon_{\text{terminal}}$.

B. Iterative Control in Constrained Environments

For autonomous systems to operate safely in unknown environments, safety needs to be carefully incorporated into the control methodology. Here, it is worth noting that existing safety analysis typically requires the constraints to be readily described in the form of $g(x) \leq 0$. In the context of learning, this strikes us as odd since the learning system often has only limited information of the environment and thus is not capable of capturing the constraint in the exact form. To address this issue, our approach will relax this specification and thus will not require the safe region to be predetermined, i.e., to know the form of $g(x) \leq 0$ in advance. Instead, the approach is designed to work with local information of the constraint and only requires the (local) constraint's boundary to be identified when the system gets close to those areas.

The idea is to put a thin layer with a controllable thickness along the (local) constrained regions so that if any point x_i of the system's trajectory touches this layer, it automatically imposes a half-space constraint preventing x_i from

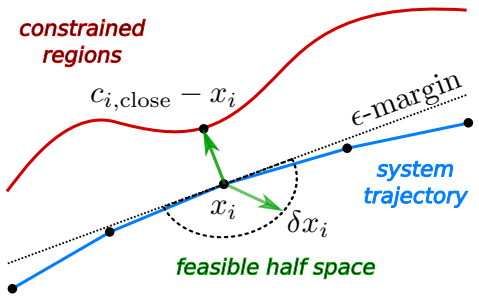

Fig. 1. The schematic diagram of the local constraint consideration where safety is ensured with a (user-defined) margin of ϵ. If a point x_i of the system's trajectory reaches the safety margin, a half-space constraint is activated to prevent x_i from getting closer to the unsafe region.

evolving forward in that direction, as illustrated in Figure 1. To implement this idea, we first denote $\mathcal{C}_i := \{c_1, \ldots, c_p\}$ as the local constraint corresponding to each x_i. Whenever x_i approaches the constraint, i.e., $\min_{j=1,\ldots,p} d(x_i, c_j) =: d(x_i, \mathcal{C}_i) \leq \epsilon$, we consider the following condition

$$(c_{i,\text{close}} - x_i)^\top \delta x_i \leq 0 \tag{5}$$

where $c_{i,\text{close}}$ denotes the point in $\mathcal{C}_i$ that is closest to x_i. This expression imposes a half-space constraint preventing x_i from getting closer to $\mathcal{C}_i$ in the next trial. The approach also gives us a systematic way to precisely control how close the system can get to the constrained regions, giving us a (user-definable) ϵ-margin of safety.

In practical applications, it is often desirable to make the system aware of the constraint early and approach it in an incremental fashion. To this end, one could consider establishing some mild safety condition at a farther distance and make the condition more restrictive as the system approaches the constraint. For example, whenever $d(x_i, \mathcal{C}_i) \leq \bar{\epsilon}$ where $\epsilon \ll \bar{\epsilon}$, the following condition could be considered

$$(c_{i,\text{close}} - x_i)^\top \delta x_i \leq d(x_i, \mathcal{C}_i) - \epsilon \tag{6}$$

This expression lets the system continue to approach the constrained region (if necessary) until it reaches the ϵ-margin of safety, i.e., $d(x_i, \mathcal{C}_i) \leq \epsilon$.

Another way of thinking about this approach is that it acts as a pre-collision warning mechanism that gets activated whenever the system's states get near the constrained regions. The approach allows us to start considering safety as soon as constraint regions are detected and gradually increase the safety requirement as the system approaches the constraints. With safety considerations, problem (4) is now extended to

$$\begin{aligned} \underset{\Delta U}{\text{minimize}} \quad & \|x_N + H_N \Delta U - x_{\text{target}}\|^2 + \lambda \|\Delta U\|^2 \\ \text{subject to} \quad & \delta x_i = H_i \Delta U, \ i \in \mathcal{I} \\ & (c_{i,\text{close}} - x_i)^\top \delta x_i \leq d(x_i, \mathcal{C}_i) - \epsilon, \ i \in \mathcal{I} \end{aligned} \tag{7}$$

where $\mathcal{I} = \{i \in \{1, 2, ..., N\} \,|\, d(x_i, \mathcal{C}_i) \leq \bar{\epsilon}\}$ is the index set denoting parts of the system trajectory that are near the constrained regions.

For the choice of λ, we accept ΔU^* and reduce λ, if the updated control steers the system closer to the target, i.e., $J(U + \Delta U^*) < J(U)$ where $J(U) := \|x_N(U) - x_{\text{target}}\|^2$. Otherwise, we increase λ and recompute the update ΔU^*. This adaptive mechanism will keep increasing λ to enforce a strict penalty on the magnitude of ΔU^* and recompute ΔU^* until $J(U+\Delta U^*) < J(U)$. Once $J(U+\Delta U^*) < J(U)$, λ is slowly decreased (in the subsequent trial), which gradually relaxes the penalty on the magnitude of ΔU^* to allow for a larger control update and more progress toward the overall solution. An analysis of this adaptive mechanism is presented in Proposition 1. The iterative scheme for the system to safely navigate unknown constrained environments is as follows.

Algorithm 2 Iterative control in constrained environments

Require: Desired terminal state x_{target}, an initial (arbitrary) input U, an adaptive rate $\alpha \in (0, 1)$, and the objective cost $J(U) = \|x_N(U) - x_{\text{target}}\|^2$.

1: Apply the input U to the system and calculate A_k, B_k.
2: Calculate H.
3: Solve for ΔU^* of the optimization problem (7).
4: If $J(U + \Delta U^*) < J(U)$, set $\lambda = (1 - \alpha)\lambda$.
   Else set $\lambda = (1 + \alpha)\lambda$ and repeat step 3.
5: Update the control input via $U = U + \Delta U^*$.
6: Repeat step $1 - 5$ until $\|x_N - x_{\text{target}}\|^2 \leq \epsilon_{\text{terminal}}$.

Proposition 1: For each U, there exists a $\lambda > 0$ such that $J(U + \Delta U^*) \leq J(U)$ where $J(U) = \|x_N(U) - x_{\text{target}}\|^2$.

Proof: First, note that the H_N is indeed the derivative of x_N with respect to the control input U, i.e.,

$$\begin{aligned} \left.\frac{dx_N}{dU}\right|_U &= \begin{bmatrix} \left.\frac{\partial x_N}{\partial u_0}\right|_U & \left.\frac{\partial x_N}{\partial u_1}\right|_U & \cdots & \left.\frac{\partial x_N}{\partial u_{N-1}}\right|_U \end{bmatrix} \\ &= \begin{bmatrix} A_{N-1} \ldots A_1 B_0 & A_{N-1} \ldots B_1 & \cdots & B_{N-1} \end{bmatrix} = H_N. \end{aligned}$$

From [11], we have that $\Delta U^* \to 0$ as $\lambda \to \infty$. Then, by applying the first-order Taylor expansion to $x_N(U)$, we have $J(U+\Delta U^*) = \|x_N(U+\Delta U^*) - x_{\text{target}}\|^2 = \|x_N(U) + H_N \Delta U^* + R(\Delta U^*) - x_{\text{target}}\|^2 \leq \|x_N(U) + H_N \Delta U^* - x_{\text{target}}\|^2 + \|R(\Delta U^*)\|^2 + 2\|x_N(U) + H_N \Delta U^* - x_{\text{target}}\| \|R(\Delta U^*)\|$, where $R(\Delta U^*) := [r_1(\Delta U^*), \ldots, r_n(\Delta U^*)]^\top$, r_i denotes the coordinate-wise residual of the Taylor expansion.

From multivariate Taylor's theorem [12], the coordinate-wise residuals can be upper bounded $r_i(\Delta U^*) \leq M_i \|\Delta U^*\|^2$ for some constants M_i and thus

$$\|R(\Delta U^*)\|^2 = \sum_{i=1}^{n} r_i^2(\Delta U^*) \leq n \left(M \|\Delta U^*\|^2 \right)^2$$

where $M := \max_i M_i$. By incorporating the upper bound of the residuals into the above Taylor expansion, we have $J(U + \Delta U^*) \leq \|x_N(U) + H_N \Delta U^* - x_{\text{target}}\|^2 + nM^2 \|\Delta U^*\|^4 + 2\|x_N(U) + H_N \Delta U^* - x_{\text{target}}\| \sqrt{n} M \|\Delta U^*\|^2$.

Now, since $\Delta U^* \to 0$ as $\lambda \to \infty$, there exists λ such that $\lambda \geq nM^2 \|\Delta U^*\|^2 + 2\|x_N(U) + H_N \Delta U^* - x_{\text{target}}\| \sqrt{n} M$, and as a result $J(U+\Delta U^*) \leq \|x_N(U) + H_N \Delta U^* - x_{\text{target}}\|^2 + \lambda \|\Delta U^*\|^2 \leq \|x_N(U) - x_{\text{target}}\|^2 = J(U)$. The last inequality is due to the fact that ΔU^* is a unique solution of (7), which yields a no larger objective value compared to $\Delta U = 0$. ■

Due to the generality of $g(x)$, the constraint can possibly impede the convergence of our algorithm (see Appendix for more details). However, if the constraint does not violate such conditions, one can ensure convergence of Algorithm 2.

Theorem 1: Assume that $d(x_i, \mathcal{C}_i) - \epsilon > 0, i \in \mathcal{I}$, which indicates that the system can strictly maintain the margin of safety. If the local linearization (2) of the original control system is N-step controllable, the iterative control synthesis (Algorithm 2) converges, and the system reaches the target.

Proof: From Proposition 1, we that $J(U + \Delta U^*) \leq J(U)$. Then, since $J(U)$ is bounded bellow, the sequence of $\{J(U)\}$ generated by Algorithm 2 is a convergent sequence. Proposition 1 also shows that $J(U + \Delta U^*) \leq V_{\Delta U^*,\lambda} \leq J(U)$ where $V_{\Delta U^*,\lambda}$ denotes the optimal objective value of (7). Thus, $J(U) - V_{\Delta U^*,\lambda} \leq J(U) - J(U + \Delta U^*)$, which leads to $V_{\Delta U^*,\lambda} \to J(U)$ as $J(U) - J(U + \Delta U^*) \to 0$. This implies $\Delta U^* \to 0$ since ΔU^* is the unique solution of (7). Given $d(x_i, \mathcal{C}_i) - \epsilon > 0, i \in \mathcal{I}$, $\Delta U^* = 0$ solves (7) if and only if $H_N^\top (x_N - x_{\text{target}}) = 0$, as shown in the Appendix. On the other hand, we have H_N is full rank since (2) is N-step controllable [13]. This implies $x_N \to x_{\text{target}}$. ∎

C. Learning Safe Control in Unknown Environments

In the previous subsection, we presented a technique for enabling systems with known dynamics to safely navigate unknown environments. We now let the system learn its own dynamics from data then subsequently use it to safely guide the control process. To this end, our idea is to have the system explore the operating region in an incremental fashion and only let it collect data in the regions that are safe and necessary for the control purpose. To achieve this, we integrate data collection and model learning into the control design through a sequential process. Specifically, we first excite the system around the starting point and use a neural network to learn the corresponding local dynamics. Then, given the estimated dynamics, we employ the iterative control (as introduced in Section III-B) to steer the system closer to the target. Whenever the system trajectory starts departing from the explored region, we execute a new round of excitation to locally inspect the unexplored region and better capture the overall dynamics, as illustrated in Figure 2. This sequential process is then repeated until the system reaches the target, which is implemented as follows.

1) (**S1**) **Safe local exploration:** Given a nominal input U, we consider applying to the system a slightly perturbed test input, i.e., $\tilde{U} := U + \Delta\tilde{U}$, where the small random perturbation $\Delta\tilde{U}$ is designed by solving the following optimization problem

$$\begin{aligned} \underset{\Delta\tilde{U}}{\text{minimize}} \quad & \sum_{i\in\mathcal{I}} (c_{i,\text{close}} - x_i)^\top H_i \Delta\tilde{U} \\ \text{subject to} \quad & \|\Delta\tilde{U}\|_\infty \leq \frac{\beta\, d_{\text{safe}}}{\|H\|_\infty} \end{aligned} \tag{8}$$

for $\beta \in (0, 1)$ and $d_{\text{safe}} := \min_{i=1,\ldots,N} \; d(x_i, \mathcal{C}_i) - \epsilon$. The solution of (8) leverages the (local) dynamics to ensure safe exploration by making $\|\Delta\tilde{X}\|_\infty < d_{\text{safe}}$, and encourages the system to explore away from the constraints, as shown in Figure 2. At each time step, we store the corresponding input and the state values obtained from the perturbed trajectory to create a new data set $D_{\text{new}} = \{D_{\text{new}}^{(\text{In})}; D_{\text{new}}^{(\text{Out})}\}$, where $D_{\text{new}}^{(\text{In})} = [\tilde{x}_k^\top, \tilde{u}_k^\top]^\top$ and $D_{\text{new}}^{(\text{Out})} = \tilde{x}_{k+1}$, and then repeat this process K times with different values of β.

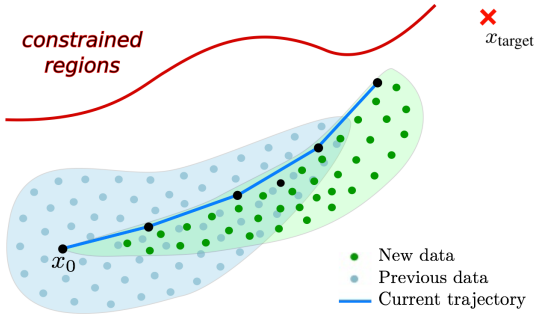

Fig. 2. Local exploration and further acquisition of data. The process is conducted by collecting data around a trajectory that extends outside the previously explored region. Based on the information of the system's local dynamics, the expansion of the data cloud (toward the desired target) is safely directed away from the constraint.

2) (**S2**) **Learning local dynamics:** We combine the newly collected data D_{new} together with the previous data D_{prev} to form a training set $D_{\text{training}} = D_{\text{new}} \cup D_{\text{prev}}$. We then (re)train a neural network to better capture the dynamics of the system in the combined explored region. In addition to the computation of f, our control method (introduced in Section III-B) requires repetitive calculations of $\partial f/\partial x$ and $\partial f/\partial u$. Instead of approximating these derivatives using finite differences, which may lead to many trials, we compute the Jacobian of f symbolically to improve data efficiency [14].
3) (**S3**) **Optimizing control inputs:** With the newly estimated dynamics and the nominal input U, we apply Algorithm 2 to iteratively steer the system closer to x_{target}. In each iteration, to measure the distance between a current trajectory and the explored region, we consider the maximum distance from each point $[x_k^\top, u_k^\top]^\top$ of the trajectory to the training dataset, i.e.,

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

If the extended distance d_{extend} is greater than a certain preset threshold $d_{\text{threshold}}$, it indicates that the current trajectory is about to depart from the explored region, and as a result, further data acquisition is required. To ensure that the current estimation of the system's dynamics is accurate for control design even in the explored region (due to potential overfittings), we also monitor the prediction error of the model, i.e.,

$$e_{\text{predict}} := \|f_{\text{model}}(x_k, u_k) - f(x_k, u_k)\|.$$

Whenever $d_{\text{extend}} \geq d_{\text{threshold}}$ or $e_{\text{predict}} \geq e_{\text{threshold}}$, we stop the current control synthesizing iteration and start the next exploration **S1**.

To summarize, the above approach learns a local model of the system's dynamics from data and immediately utilizes it to design safe control signals (with the use of Algorithm 2). Whenever the system starts to venture into the unexplored region or experiences inaccurate model prediction (due to

potential overfittings), the program pauses the current control, executes a new round of exploration, and updates the model. This sequential process is repeated until the system reaches the desired target, as summarized in Algorithm 3.

Algorithm 3 Learning safe control in unknown environments

Require: x_0, x_{target}, and an initial (arbitrary) input U.
1: Apply **S1** to explore local areas around a current trajectory.
2: Apply **S2** to update the model on the explored areas.
3: Apply **S3** using Algorithm 2 to steer the system closer to the target until $d_{\text{extend}} > d_{\text{threshold}}$ or $e_{\text{predict}} < e_{\text{threshold}}$.
4: Repeat step $1-3$ until $\|x_N - x_{\text{target}}\|^2 \leq \epsilon_{\text{terminal}}$.

IV. Numerical Implementation

We demonstrate the effectiveness of our approach with a classical unicycle example and a quadcopter example. The systems' dynamics and environments are assumed to be unknown in both examples. The results demonstrate the effectiveness of our approach in learning safe controls for systems in unknown environments.

A. Unicycle

Consider the unicycle model described by the dynamics

$$\begin{aligned}\dot{x} &= v\cos(\theta)\\ \dot{y} &= v\sin(\theta)\\ \dot{\theta} &= \omega\end{aligned}$$

where x and y indicate the position of the unicycle, and θ is the heading angle of the unicycle. Here, v and ω are the control inputs denoting the moving and steering velocities.

We set $x_0 = (-8,-8,0)^\top, x_{\text{target}} = (8, 6.5, 0)^\top$ and use Matlab ode45 to simulate the system for $N = 150$ steps. To capture the system's dynamics, we use a fully connected feedforward network with two hidden layers (each layer consists of 10 tansig activation units) from Matlab's Neural Network Toolbox. We apply Algorithm 3 with $d_{\text{threshold}} = 5$, $e_{\text{threshold}} = 0.1$, $K = 5$, and learn the desired maneuver of the unicycle in 150 trials. Figure 3 illustrates the sequential and gradual learning process. Note that due to the consideration of (8), explorations are directed away from the constraints while remaining close to the main controlled trajectories. This mechanism results in a safe, efficient, and highly targeted exploration of the state-action space, which is mainly tailored to the purpose of control.

B. Quadcopter

We now consider a high-dimensional quadcopter model with complex nonlinear dynamics to highlight the efficacy of the proposed method for learning safe controls. The adopted model is described by a system of nonlinear ODEs with 12 states and 4 control inputs and takes the form of

$$\dot{x} = f_d(x) + \sum_{i=1}^{4} f_i(x) u_i^2$$

where 4 control u_i denote the rotating speeds of four motors, f_d and f_i are the nonlinear drift dynamics and the control vector fields, respectively [15].

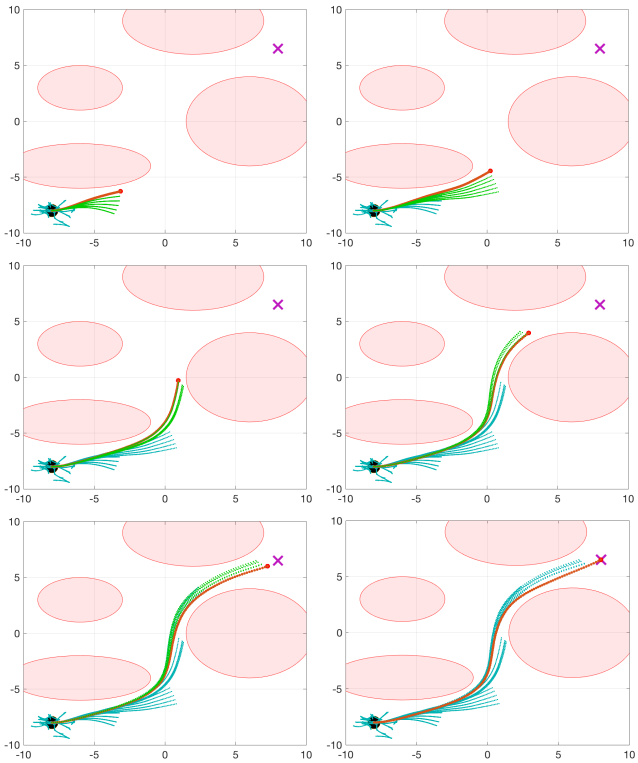

Fig. 3. A unicycle learns to navigate an unknown constrained environment through sequential exploration and control periods. As the system trajectory departs from a previously explored region or experiences inaccurate prediction, a new round of exploration is executed to inspect the new local area and better capture the overall system's dynamics. Color legend: main controlled trajectories, new rounds of explorations, previous explorations.

Given the above model, we consider learning suitable controls for the quadcopter to maneuver through a highly constrained environment and then come to a standstill at a target location with

$$\begin{aligned}x_0 &= [0.2, 0.2, 0.1, 0, 0, 0, 0, 0, 0, 0, 0, 0]^\top\\ x_{\text{target}} &= [0.8, 0.9, 0.4, 0, 0, 0, 0, 0, 0, 0, 0, 0]^\top\end{aligned}$$

where the first 6 states denote the linear and angular positions of the quadcopter, respectively, in the earth frame, and the last 6 states are the corresponding velocities. In this example, we use a neural network with two slightly larger hidden layers (each layer consists of 20 tansig activation units) and apply Algorithm 3 with $d_{\text{threshold}} = 0.25$, $e_{\text{threshold}} = 0.1$, $K = 5$ to safely learn the desired maneuver after 250 trials. Figure 3 illustrates the sequential and gradual learning process, which demonstrates that the proposed approach can effectively learn a safe maneuver in a new environment for a high-dimensional complex quadcopter system.

V. Conclusion

In this paper, we developed an incremental yet data-efficient approach to safe learning in unknown constrained environments. The presented approach learns the desired controlled maneuvers through a sequential process of control and exploration. By leveraging the system's local dynamics, the approach safely guides exploration and effectively tailors

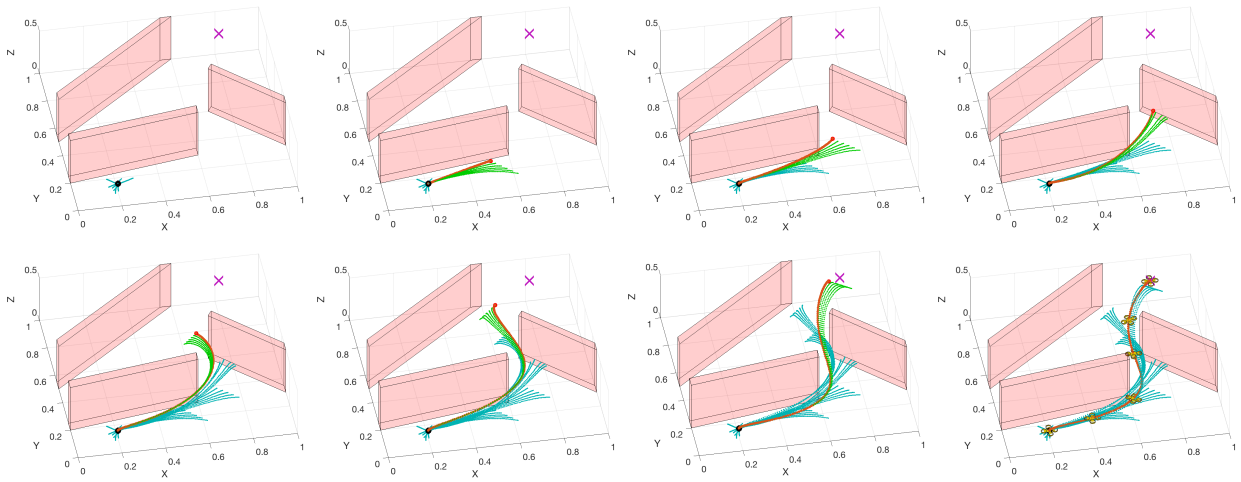

Fig. 4. Quadcopter learns to flight through an unfamiliar constrained environment directly from data. The desired maneuver is learned through sequential exploration and control periods with 250 trials. Color legend: main controlled trajectories, new rounds of explorations, previous explorations.

it to the purpose of control, which is illustrated by numerical examples including a complex high-dimensional one.

APPENDIX

We characterize the constraint condition that may prevent convergence of the iterative control. In particular, we are interested in the case where $z := x_N - x_{\text{target}} \neq 0$ yet problem (7) results in a trivial solution. To this end, we first rewrite problem (7) in the form

$$\begin{aligned} \underset{\Delta U}{\text{minimize}} \quad & \frac{1}{2}\|z + H_N \Delta U\|^2 + \frac{\lambda}{2}\|\Delta U\|^2 \\ \text{subject to} \quad & L\Delta U \leq b \end{aligned} \tag{9}$$

where $L = \mathcal{N}H$, $\mathcal{N}$ and $b \geq 0$ are the matrix and vector corresponding to the set of normal vectors $n_i = c_{i,\text{close}} - x_i$ and margins $d(x_i, \mathcal{C}_i) - \epsilon$, which are activated by the constraints. Let $p = |\mathcal{I}|$ be the size (or cardinal number) of $\mathcal{I}$, $\mathbb{R}^p_- = (-\infty, 0]^p$, and $\mathbb{R}^p_+ = [0, \infty)^p$. From the optimality condition, ΔU^* is the solution of (9) if and only if

$$0 \in H_N^\top (H_N \Delta U^* + z) + \lambda \Delta U^* + L^\top (N_{\mathbb{R}^p_-}(L\Delta U^* - b))$$

where $N_\Omega(x)$ denotes the normal cone of Ω at x [16]. Thus, $\Delta U^* = 0$ is the solution of (9) if and only if $0 \in H_N^\top z + L^\top (N_{\mathbb{R}^p_-}(-b))$ which is equivalent to

$$\begin{aligned} & 0 = H_N^\top z + L^\top u, u \in N_{\mathbb{R}^p_-}(-b), -b \in \mathbb{R}^p_- \\ \iff \quad & 0 = H_N^\top z + L^\top u, u \in \mathbb{R}^p_+, u^\top b = 0, b \in \mathbb{R}^p_+ \\ \iff \quad & H_N^\top z = L^\top (-u), -u \in \mathbb{R}^p_-, -u^\top b = 0, b \in \mathbb{R}^p_+ \\ \iff \quad & H_N^\top z \in L^\top (\mathbb{R}^p_- \cap \{b\}^\perp), b \in \mathbb{R}^p_+. \end{aligned}$$

$S = L^\top (\mathbb{R}^p_- \cap \{b\}^\perp)$ characterizes the unfavorable set that could lead to a trivial solution of (9). Note that $S = \{0\}$ if $b > 0$, and it becomes the largest, i.e., $S = L^\top(\mathbb{R}^p_-)$ when $b = 0$, which denotes that all points of the current trajectory reach their (critical) safety margins. In this case, if $H_N^\top (x_N - x_{\text{target}}) \in L^\top (\mathbb{R}^p_-)$, the constraint can theoretically prevent the convergence of the presented iterative control. For example, in a particular case where the local constraint of x_N is activated, this condition indicates that the direction of $x_{\text{target}} - x_N$ aligns with the normal vector $n_N = c_{N,\text{close}} - x_N$ imposed by the constraint.

REFERENCES

[1] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.

[2] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 3387–3395.

[3] T. Lew, A. Sharma, J. Harrison, A. Bylard, and M. Pavone, "Safe active dynamics learning and control: A sequential exploration–exploitation framework," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 2888–2907, 2022.

[4] J. Guiochet, M. Machin, and H. Waeselynck, "Safety-critical advanced robots: A survey," *Robotics and Autonomous Systems*, vol. 94, pp. 43–52, 2017.

[5] N. C. Wagener, B. Boots, and C.-A. Cheng, "Safe reinforcement learning using advantage-based intervention," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 630–10 640.

[6] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[7] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[8] K.-C. Hsu, V. Rubies-Royo, C. J. Tomlin, and J. F. Fisac, "Safety and liveness guarantees through reach-avoid reinforcement learning," *arXiv preprint arXiv:2112.12288*, 2021.

[9] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 1.

[10] S. Zeng, "Iterative optimal control syntheses illustrated on the brockett integrator," *IFAC-PapersOnLine*, vol. 52, no. 16, pp. 138–143, 2019.

[11] A. E. Hoerl and R. W. Kennard, "Ridge regression—1980: Advances, algorithms, and applications," *American Journal of Mathematical and Management Sciences*, vol. 1, no. 1, pp. 5–83, 1981.

[12] T. M. Apostol, *Calculus, volume 2 (second addition*. John Wiley & Sons, 1996.

[13] J. C. Engwerda, "Control aspects of linear discrete time-varying systems," *International Journal of Control*, vol. 48, no. 4, pp. 1631–1658, 1988.

[14] S. Rodini, "Analytical derivatives of neural networks," *Computer Physics Communications*, vol. 270, p. 108169, 2022.

[15] F. Sabatino, "Quadrotor control: modeling, nonlinear control design, and simulation," 2015.

[16] H. H. Bauschke, P. L. Combettes *et al.*, *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2011, vol. 408.