Lyapunov Density Models: Constraining Distribution Shift in Learning-Based Control

Katie Kang [1] Paula Gradu [1] Jason Choi [1] Michael Janner [1] Claire Tomlin [1] Sergey Levine [1]

Abstract

Learned models and policies can generalize effectively when evaluated within the distribution of the training data, but can produce unpredictable and erroneous outputs on out-of-distribution inputs. In order to avoid distribution shift when deploying learning-based control algorithms, we seek a mechanism to constrain the agent to states and actions that resemble those that it was trained on. In control theory, Lyapunov stability and control-invariant sets allow us to make guarantees about controllers that stabilize the system around specific states, while in machine learning, density models allow us to estimate the training data distribution. Can we combine these two concepts, producing learning-based control algorithms that constrain the system to in-distribution states using only in-distribution actions? In this work, we propose to do this by combining concepts from Lyapunov stability and density estimation, introducing Lyapunov density models: a generalization of control Lyapunov functions and density models that provides guarantees on an agent's ability to stay in-distribution over its entire trajectory.

1. Introduction

Learning-based control algorithms, including model-free and model-based reinforcement learning, have shown the capability to surpass the performance of traditional model-based control on a number of complex, difficult-to-model systems (Kalashnikov et al., 2018; Akkaya et al., 2019; Li et al., 2021), by leveraging data in place of approximate mathematical models. However, the data-dependent nature of these learning-enabled controllers also leads to new challenges. One such challenge is that learned models can only provide reliable predictions when queried within the training data distribution. If the inputs to a learning-based controller deviate far from its training data, the controller can suffer from *model exploitation*, in which mispredictions from the machine learning model causes the controller to output suboptimal or even catastrophic action commands.

To mitigate the negative effects of distribution shift, can we design controllers that perform the desired task *while ensuring the system remains in-distribution*? One approach is to learn a density model of the training data, and use it to constrain or regularize the controller to prevent the agent from taking low-likelihood actions or visiting low-likelihood states (Richter & Roy, 2017; McAllister et al., 2019; Fujimoto et al., 2019; Kumar et al., 2019; Wu et al., 2019). However, because density models are not aware of system dynamics, this kind of approach "greedily" chooses actions that are in-distribution at the next timestep, but does not provide a mechanism for *staying* in-distribution over a long horizon. On the other hand, guaranteed long-horizon constraint satisfaction has been widely studied in control theory. Lyapunov functions map each state to a non-negative scalar which must decrease as the system evolves, providing a mechanism for ensuring the stability of a system over an infinite horizon. However, these notions of stability are formulated relative to fixed points of the system, and unrelated to any data or distribution.

The goal of our work is to provide a mechanism for ensuring that a system controlled by a learning-based policy will remain in-distribution throughout its trajectory. To this end, the main contribution of our work is the Lyapunov density model (LDM), a function that combines the data-aware aspect of density models with the dynamics-aware aspect of Lyapunov functions. Analogous to how control Lyapunov functions are used to guarantee that a controller stabilizes a system, we show how LDMs can be used to design controllers that are guaranteed to keep a system in-distribution, where naïvely using a density model will fail. We also present an algorithm for learning the LDM and its associated policy function directly from a training set of transitions, without assuming any knowledge of the underlying dynamics or active system interaction. Since learning the LDM is itself a data-dependent process which may suffer from distribution shift, we also provide theoretical analysis of our

[1]University of California, Berkeley. Correspondence to: Katie Kang <katiekang@eecs.berkeley.edu >.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

algorithm, showing that performing control with a LDM outperforms using a density model, even under approximation error. LDMs are a general tool for synthesizing and verifying controllers to stay in-distribution, and can be used in a variety of different learning-based control applications. We present a method of using an LDM in model-based RL as a constraint on the model optimizer, and evaluate this method empirically. The project webpage can be found at: https://sites.google.com/berkeley.edu/ldm/

2. Related works

A large body of work in control theory focuses on providing guarantees about the evolution of dynamical systems. To guarantee stability, one could use Lyapunov functions for uncontrolled systems (Sastry, 1999), and control Lyapunov functions for controlled systems (Sontag, 1989). To guarantee that an agent remains within a set of safe states, one could use control-barrier functions (Ames et al., 2017) or Hamilton-Jacobi reachability value functions (Bansal et al., 2017). All of these approaches, in addition to our work, provide conditions for infinite-horizon guarantees based on the control invariance of the level sets of the functions. However, these methods usually assume access to the ground-truth dynamics model, whereas our method only uses data from the system. More importantly, these methods are primarily concerned with satisfying manually-designed constraints based on limitations of the physical system. In contrast, our method derives constraints from the training data distribution in order to avoid visiting out-of-distribution regions where the learning-based controller is prone to be erroneous.

Some recent works take a data-driven approach to maintaining safety or stability, without assuming access to a ground-truth dynamics model (Berkenkamp et al., 2017; Cheng et al., 2019; Dai et al., 2021; Chang et al., 2019; Mehrjou et al., 2020; Mittal et al., 2020; Richards et al., 2018; Fisac et al., 2019; Robey et al., 2020; Tu et al., 2022). Although this line of work shares superficial similarities with our work, such as learning a "Lyapunov-like" function , the two serve orthogonal purposes. The primary goal of such safe learning methods is to satisfy physical safety constraints, while our goal is to maintain the reliability of learning-based controllers by staying in-distribution.

A number of prior methods, particularly in model-free and model-based offline RL (Nair et al., 2020; Laroche et al., 2019; Kumar et al., 2019; Fujimoto et al., 2019; Kumar et al., 2020; Kidambi et al., 2020; Yu et al., 2020), focus on learning a good policy from offline data by avoiding distributional shift. Although we also aim to learn from offline data and avoid distributional shift, our focus is different. Our aim is to learn a general function that can prevent distributional shift for any downstream controller. The Lyapunov density model is not associated with any task, its role is only to provide a constraint such that any downstream policy that satisfies this constraint avoids excessive distributional shift.

3. Problem Formulation

Consider a deterministic, continuous-state, discrete-time dynamical system $s_{t+1} = f(s_t, a_t)$ with state space $\mathcal{S}$ and action space $\mathcal{A}$. We start with a dataset of transitions $\{(s_t^i, a_t^i, s_{t+1}^i)\}_{i=1}^N$ generated from a distribution $P(\mathrm{s}, \mathrm{a})$, which is used to train a learning-enabled control system. We will say a state-action tuple (s_t, a_t) is *in-distribution* if $P(s_t, a_t) \geq c$, for some density level $c > 0$. We denote the set of in-distribution states and actions as $\mathcal{D}_c := \{(s_t, a_t) | P(s_t, a_t) \geq c\}$. We wish to query our learned model or policy, used for performing a downstream task, only on points in $\mathcal{D}_c$, to mitigate distribution shift at test-time. While being in-distribution or in-support does not *necessarily* guarantee that the learned models or policies will be accurate, this condition is simple and has been used widely in prior work (Fujimoto et al., 2019; Laroche et al., 2019; Kumar et al., 2019; Dean et al., 2020).

Because we do not have access to the ground truth training data distribution $P(\mathrm{s}, \mathrm{a})$, we could instead approximate it with a density model $P_\theta(s_t, a_t)$, using techniques such as energy-based models (Hinton, 2002) and flow models (Durkan et al., 2019). However, even if $P_\theta(s_t, a_t)$ is learned effectively, it is not obvious how to design a controller such that $P_\theta(s_t, a_t) \geq c$ is satisfied for an agent's entire trajectory. The main challenge lies in the sequential nature of control problems: a controller that satisfies $P_\theta(s_t, a_t) \geq c$ at the time t could still take the agent to a future state s_{t+n} for which there exists no action a_{t+n} that satisfies $P_\theta(s_{t+n}, a_{t+n}) \geq c$, inevitably driving the agent into a low density region. See Figure 1 for an illustrative example.

4. Background on CLFs

In order to perform long-horizon reasoning about the data distribution in conjunction with the system dynamics, we turn to a widely studied tool in controls: control Lyapunov functions (CLFs). CLFs are a tool for verifying that a system is stabilizable, meaning that system can be controlled to remain close to an equilibrium point over an infinite horizon. First, we introduce formal definitions for an equilibrium point, stabilizability, and a CLF.

Definition 4.1 (Equilibrium Point)**.** A state s_e is an equilibrium point if $\exists$ action $a_e \in \mathcal{A}$ such that $f(s_e, a_e) = s_e$.

Definition 4.2 (Stabilizable in the sense of Lyapunov)**.** A system is stabilizable (in the sense of Lyapunov) if $\forall \epsilon > 0$, $\exists \delta$ such that for all $s_0 \in \mathcal{S}$ such that $||s_0 - s_e|| \leq \delta$, there exists $\{a_t\}_{t=0}^{\infty}$ such that the resulting $\{s_t\}_{t=0}^{\infty}$ satisfies $||s_t - s_e|| \leq \epsilon \; \forall t \geq 0$. (Murray et al., 2017, Ch.4.4)

Definition 4.3 (Control Lyapunov Function)**.** A continuous

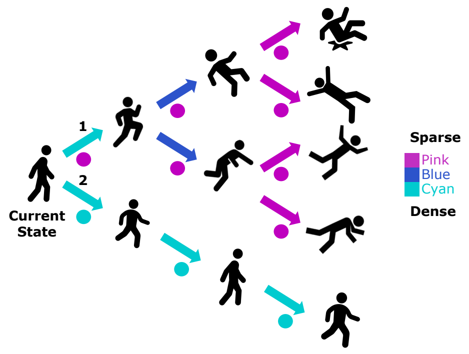

Figure 1. Illustrative example of how naively constraining the agent with a density model can lead to failure. Arrows denote possible transitions, and color denotes the likelihood of a state and action under the data distribution. Suppose our goal was for the agent to remain at a higher density than blue. Using the density model as a constraint on the actions ($P_\theta(s_t, a_t) \geq c_{blue}$) would allow the agent to take action 1 or 2 from the current state. However, taking action 1 will inevitably cause the agent to go to the pink states, leading to failure. Instead, constraining the agent's actions with the lowest density that the agent will encounter in the *future* (denoted by coloring of dots below arrows) will restrict the agent to only take action 2 at the current state, which ensures the agent stays above the blue density threshold throughout its trajectory. In order to *remain* in-distribution, the agent must not only avoid states and actions that are currently unfamiliar, but also those that may lead to unfamiliar states and actions *in the future*.

and radially unbounded function W: $\mathcal{S} \to \mathbb{R}$ is a control Lyapunov function if the following conditions hold:

1. $\forall s \in \mathcal{S}, \exists a \in \mathcal{A}$ s.t. $W(s) \geq W(f(s,a))$,
2. $\forall s \neq s_e, W(s) > 0$,
3. $W(s_e) = 0$.

A CLF maps each state to a scalar "energy level". The first condition of CLFs mandates that the system can always be controlled to remain at or below its current energy level. This means that for an energy threshold, if an agent is inside the set of states with energies below that threshold, then it can be controlled to remain within that set for all time. This property of CLFs is formalized below.

Definition 4.4 (Control-Invariant Set)**.** A set of states $\mathcal{C} \subseteq \mathcal{S}$ is called a control-invariant set if $\forall s_0 \in \mathcal{C}, \exists\{a_t\}_{t=0}^{\infty}$ such that $s_t \in \mathcal{C}\ \forall t \geq 0$. (Borrelli et al., 2017, Ch.10.9)

Theorem 4.5. *For a control Lyapunov function W, the sublevel set $\{s \mid W(s) \leq c\}$ is control-invariant for all $c \geq 0$.*

Combining Theorem 4.5 with the second and third conditions of CLFs, which mandates that the equilibrium point has the lowest energy, we can use the sublevel sets of the CLF to extract arbitrarily small control-invariant sets which contain the equilibrium point. This ensures that the agent is never able to deviate far from the equilibrium point, thereby guaranteeing stabilizability.

Theorem 4.6. *A system has a CLF if there exists an action sequence $\{a_t\}_{t=0}^{\infty}$ such that the resulting trajectory $\{s_t\}_{t=0}^{\infty}$ is stabilizable in the sense of Lyapunov with respect to x_e. (Murray et al., 2017, Theorem 4.4.4)*

5. Lyapunov Density Models

Density models provide us with an estimate of the training data distribution, while CLFs allow us to make guarantees about the long-horizon evolution of a dynamical system. In this section, we derive Lyapunov density models, a union of the two ideas which can be used to guarantee a system's ability to stay within regions of high data density.

In order for a learned model or policy to be queried only on points in $\mathcal{D}_c = \{(s_t, a_t) | P(s_t, a_t) \geq c\}$ when evaluated in closed-loop, the agent must only visit states from which it can be controlled to remain in $\mathcal{D}_c$. This desired property is exhibited by a control-invariant set with a state-dependent action space of $\mathcal{A}(s_t) = \{a_t | P(s_t, a_t) \geq c\}$. We define the notion of a control-invariant set in the joint state and action space to simplify notation.

Definition 5.1 (State-Action Control-Invariance)**.** A set $\mathcal{G} \subseteq \mathcal{S} \times \mathcal{A}$ is a state-action control-invariant set if $\forall (s_0, a_0) \in \mathcal{G}$, $\exists \{a_t\}_{t=1}^{\infty}$ such that $(s_t, a_t) \in \mathcal{G}\ \forall t \geq 0$.

Let $\mathcal{G}_c$ be a state-action control-invariant set inside $\mathcal{D}_c$. By Definition 5.1, if an agent is in state s_0 and takes action a_0 such that $(s_0, a_0) \in \mathcal{G}_c$, then there exists a trajectory $(\{s_t\}_{t=1}^{\infty}, \{a_t\}_{t=1}^{\infty})$ that can satisfies $P(s_t, a_t) \geq c$ for all $t = 0, .., \infty$, which guarantees that the agent is able to remain in-distribution *for all future time steps*.

How can we construct a state-action control-invariant set $\mathcal{G}_c$? Recall we can obtain a control-invariant set associated with *each* level of a CLF (Theorem 4.5). We adapt this property to derive a function such that 1) each sublevel-set is a state-action control-invariant set, and 2) each level of the function is associated with a density level of a probability distribution. We call this a Lyapunov density model (LDM).

Definition 5.2 (Lyapunov Density Models)**.** A function $G(s_t, a_t)$: $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ continuous in s_t and a_t is a Lyapunov Density Model for the dynamical system $s_{t+1} = f(s_t, a_t)$ and a density function $P : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ if the following conditions hold:

1. $\forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}, \exists a_{t+1} \in \mathcal{A}$ such that $G(s_t, a_t) \geq G(f(s_t, a_t), a_{t+1})$
2. $\forall (s_t, a_t) \in \mathcal{S} \times \mathcal{A}, G(s_t, a_t) \geq -\log(P(s_t, a_t))$

We formalize the key property of Lyapunov Density Models (LDMs) in Theoremm 5.3 below, proven in Appendix A.2.

Theorem 5.3. *Any sub-level set of an LDM G, $\mathcal{G}_c := \{(s_t, a_t) : G(s_t, a_t) \leq -\log(c)\}$, is a state-action control-invariant set s.t. $\forall (s_t, a_t) \in \mathcal{G}_c$, $P(s_t, a_t) \geq c$.*

An LDM $G(s_t, a_t)$ is determined by a dynamical system $s_{t+1} = f(s_t, a_t)$ and a data distribution $P(s_t, a_t)$ [1], and maps each state and action pair to a scalar value representing a negative log density level $-\log(c)$. By Theorem 5.3, we can simultaneously represent a state-action control-invariant set inside $\mathcal{D}_c$ for *all* values of c between 0 and $\max_{s_t,a_t} P(s_t, a_t)$ using the sub-level sets of $G(s_t, a_t)$, which we denote as $\mathcal{G}_c$. Since $\mathcal{G}_c$ is a state-action control-invariant set, as long as the agent only visits states and executes actions inside $\mathcal{G}_c$, its trajectory is guaranteed to stay within the c-thresholded support of the data distribution. Thus, using an LDM, we can verify that an agent can stay in-distribution for any desired density level.

We define a notion of a maximal LDM to represent the LDM with the largest state-action control-invariant sets for a given dynamical system and probability distribution. In our later discussion on deriving control policies with LDMs, we make use of maximal LDMs in order to impose the least restriction on the downstream task-solving policy.

Definition 5.4 (Maximal LDMs)**.** An LDM $G(s_t, a_t)$ is maximal if its each sublevel set $\mathcal{G}_c = \{(s_t, a_t) | G(s_t, a_t) \leq -\log(c)\}$ is the largest state-action control-invariant set contained inside the corresponding sub-level set of the data distribution, $\mathcal{D}_c = \{(s_t,a_t) | P(s_t,a_t) \geq c\}$.

The maximal LDM can be represented in closed form with the following equation:

$$G(s_0, a_0) = \min_{\{a_t\}_{t=1}^{\infty}} \max_{t\geq 0} -\log(P(s_t, a_t)). \tag{1}$$

Proposition 5.5. *If $G(s, a)$ defined by (1) is continuous, then it is a maximal LDM for $f(s, a)$ and $P(s, a)$.*

Here, we introduce an illustrative example of LDMs for a 2D linear system, given by

$$s_{t+1} = Fs_t + Ga_t, \tag{2}$$

where $F = e^{A\Delta t}$, $G = A^{-1}(e^{A\Delta t} - I)B$, with $A = \begin{bmatrix} \beta & \omega \\ -\omega & \beta \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $s := \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ and $\beta, \omega > 0$. The system spirals outwards from the origin when no action is applied, but can be stabilized to the origin with an control input.

Figure 2 shows example data distributions (left) and their associated maximal LDMs (right) for this system. In particular, case (a) represents a scenario where we sample zero-mean Gaussian action uniformly across the bounded state space. The data is dense near $a_t = 0$, and sparse at higher action magnitudes. Since the natural dynamics is unstable, the system can only maintain a state that does not escape the high-density support near the origin. This results in the level set of $G(s, a)$ for a high density level (cyan) to

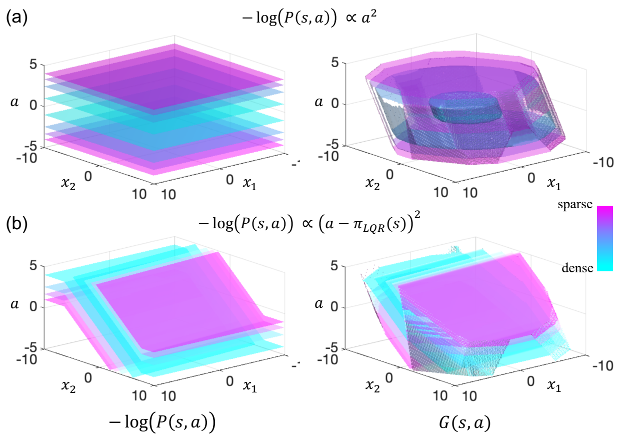

Figure 2. Level sets of various data distribution $P(s, a)$ (left) and the corresponding level sets of the maximal LDMs $G(s, a)$ (right) for the linear system example (2). The color closer to cyan indicates denser region with lower $-\log(P(s, a))$ or lower level of $G(s, a)$. (a) Gaussian policy with zero-mean and (b) Gaussian policy with mean $\pi_{LQR}(s)$ (stabilizing LQR controller).

be very small. In contrast, in case (b), the data distribution is collected by running a noisy feedback controller which stabilizes the system. In this case, the dense region of the data already inherits a stabilizing property. Therefore, it turns out that the invariant set for the high-density level maintains most volume of the original high-density region. Appendix A.1 provides more details and examples.

6. Learning LDMs from Data

In order to make use of LDMs, we need a practical way of constructing them from data. In this section, we present an algorithm for learning a maximal LDM. First, to represent the distribution of our training data, we train a density model $E(s_t, a_t) = -\log(P(s_t, a_t))$. Next, we introduce the following LDM backup operator:

$$\mathcal{T}G(s, a) = \max\{E(s, a), \min_{a' \in \mathcal{A}} \gamma G(f(s, a), a')\} \tag{3}$$

Analogous to value iteration or Q-learning (here, the LDM $G(s, a)$ plays a similar role to the Q-function), where one can solve for the value function via dynamic programming by iteratively applying the Bellman backup, we can solve for the maximal LDM by iteratively applying the LDM backup. By initializing $G(s, a)$ to $-\log(P(s, a))$ and performing the update rule $G_{k+1} \leftarrow \mathcal{T}G_k$, $\lim_{k\to\infty} G_k$ will converge to the solution of Equation (1) when $\gamma = 1$ (a formal statement and proof are provided in Appendix A.4). While this method can easily be implemented without any discount, corresponding to $\gamma = 1$, the use of a discount factor aids the theoretical analysis [2].

[1]LDMs are, in fact, a generalization of both Lyapunov functions and density models; for more details, see Appendix A.3.

[2]Discounted iteration of Equation 3 yields an LDM that satisfies $G(s_0, a_0) = \min_{\{a_t\}_{t=1}^{\infty}} \max_{t\geq 0} -\gamma^t \log(P(s_t, a_t))$, which

The above procedure makes use of the true dynamics $s_{t+1} = f(s_t, a_t)$. However, in reality we only have access to a dataset of transitions $D = \{(s_t^i, a_t^i, s_{t+1}^i)\}_{i=1}^N$, which may be used to learn a dynamics model or a policy for solving a downstream task. Thus, we would like to learn the LDM from this same dataset. We present a fitted version of LDM training, in which the learned LDM $\hat{G}_{k+1}$ is optimized using samples by iteratively applying:

$$\hat{G}_{k+1} \in \arg\min_{G \in \mathcal{G}} \sum_{(s,a) \in D} \left(G(s,a) - \mathcal{T}\hat{G}_k(s,a)\right)^2 \quad (4)$$

We analyze this procedure formally in Sec. 8.

Finally, in order to use LDMs in high-dimensional and realistic domains, we present a practical algorithm for training LDMs with deep neural networks. We represent the density model $E_\theta(s_t, a_t) = -\log(P(s_t, a_t))$ with a neural network, and train it as a flow model (Durkan et al., 2019) (implementation details in Appendix D.3). We also train a neural network model of the LDM $G_\phi(s_t, a_t)$ in conjunction with a policy $\pi_\psi(s_t)$. This algorithm is structurally similar to standard off-policy actor-critic RL methods, and is implemented in a similar way in practice by modifying standard deep RL implementations, replacing the standard Bellman backup with our LDM backup (Eq. 3). The policy is optimized to output the actions that minimize the LDM (Eq. 5), and the policy's actions are used to perform the LDM backup (Eq. 6), because exactly solving for the minimizing action at each step is difficult with continuous, high-dimensional action spaces:[3]

$$\psi = \text{argmin}_\psi \mathbb{E}_{s_t \sim p_D}[G_\phi(s_t, \pi_\psi(s_t))] \quad (5)$$

$$\phi = \text{argmin}_\phi \mathbb{E}_{s_t, a_t, s_{t+1} \sim p_D}[(G_\phi(s_t, a_t) - \max\{E_\theta(s_t, a_t), \gamma G_\phi(s_{t+1}, \pi_\psi(s_{t+1}))\})^2] \quad (6)$$

Following the implementation in soft-actor critic (SAC) (Haarnoja et al., 2018), we also make use of a number of deep RL techniques to stabilize training. These include using double value networks ϕ_1, ϕ_2, target networks $\bar{\phi}_1, \bar{\phi}_2$, and an additional entropy term in the training objective with automatic tuning, $\alpha\mathcal{H}(\pi_\psi(.|s_t))$. Furthermore, because we are learning the LDM from an offline dataset, we follow the implementation in CQL(Kumar et al., 2020) by adding a conservative regularization term, CQL($\mathcal{H}$), in the LDM update to regulate underestimation [4] in the LDM on out-of-distribution actions. A summary of our practical algorithm can be found in Alg. 1. We make use of this algorithm in our experiments in Section 9. More implementation details can be found in Appendix D.3.

essentially counts densities far into the future as being lower than they really are, which we can intuitively treat as either as a heuristic variance reduction technique or a way to account for greater uncertainty far in the future. We additionally perform ablation experiments to analyze the effects of using a discount $\gamma < 1$ empirically (Appendix D.6).

[3]In our theoretical analysis, we will still assume that the minimization of the LDM with respect to a can be done exactly, since the policy has unrestricted access to $\mathcal{T}\hat{G}_k$.

[4]Since Q-values are *maximized* while LDMs are *minimized*, our conservative term handles *underestimation*, not *overestimation*.

7. Control with LDMs

In this section, we discuss how LDMs can be used with learned models to produce control strategies that remain in-distribution while accomplishing a task. We focus on applying LDMs to model-based reinforcement learning, a problem setting that is known to be particularly sensitive to distribution shift, though in principle LDMs could be used with other learned models, including model-free or imitation learning policies, all of which are susceptible to distributional shift. Model-based RL methods can use learned models to train policies, or use learned models directly for control via planning (e.g., via model-predictive control, MPC). We focus on the latter category of methods to provide a simple evaluation of LDMs, which is amenable to both theoretical and empirical analysis.

7.1. MPC with an LDM Constraint

In our analysis, we will apply the LDM to constrain a short-horizon MPC controller that uses a learned model, such that the controller stays close to the training distribution where the model is likely to be accurate. The model $f_\xi(s_t, a_t)$ is trained on the same dataset as the LDM, which we assume is sampled from $P(s_t, a_t)$, using empirical risk minimization on a sample-based estimate of the loss $L_f(\xi)$:

$$L_f(\xi) = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim p_D}[(f_\xi(s_t, a_t) - s_{t+1})^2]. \quad (7)$$

This model can then be used with MPC to select actions $\mathbf{a}_{1:H}$. The LDM, trained via Alg. 1, modifies the standard MPC procedure by imposing an additional constraint:

$$a^*_{1:H} = \text{argmax}_{a_{1:H}} \sum_{t=1}^{H} r(s_t, a_t) \quad (8)$$

$$\text{s.t.} \quad s_{t+1} = f_\xi(s_t, a_t) \quad \forall 1 \le t \le H-1$$

$$G_\phi(s_t, a_t) \le -\log(c) \quad \forall 1 \le t \le H \quad (9)$$

As in standard MPC, the controller executes the first action, and then replans. Theorem 5.3 guarantees that, for a correct LDM, as long as the agent is in the sub-level set of the LDM at $-\log(c)$, it is able to satisfy $P(s_t, a_t) \ge c$ throughout its trajectory. Thus, under a correct LDM on Line (9), this method of constraining MPC is guaranteed to produce trajectories in the c-thresholded support of the data.

Finally, since Algorithm 1 aims to learn the maximal LDM, it is the *least restrictive* constraint for making the guarantee, giving the agent the most flexibility for solving the task. Of course, in practice the LDM itself might suffer from approximation errors; we analyze these further in Section 8.

Algorithm 1 LDM Training [practical algorithm]

```
Initialize parameter vectors $\theta$, $\phi_1$, $\phi_2$, $\bar{\phi}_1$, $\bar{\phi}_2$, $\psi$, and $\alpha$
Train flow model $E_\theta(s_t, a_t)$
for num LDM training steps do
    Take gradient step on LDM networks:
        $\phi_i \leftarrow \phi_i - \lambda_G \nabla_{\phi_i} \Big( \mathbb{E}_{s_t, a_t, s_{t+1} \sim p_D} [(G_{\phi_i}(s_t, a_t) - \max\{E_\theta(s_t, a_t), \gamma G_{t+1}\})^2 + \text{CQL}(\mathcal{H})] \Big)$ for $i \in \{1, 2\}$
        where $G_{t+1} = \max\{G_{\bar{\phi}_1}(s_{t+1}, \pi_\psi(s_{t+1})), G_{\bar{\phi}_2}(s_{t+1}, \pi_\psi(s_{t+1}))\}$
    Update LDM target networks:
        $\bar{\phi}_i \leftarrow \tau \phi_i + (1 - \tau) \bar{\phi}_i$ for $i \in \{1, 2\}$
    Take gradient step on policy network:
        $\psi \leftarrow \psi - \lambda_\pi \nabla_\psi \Big( \mathbb{E}_{s_t \sim p_D} [G_\phi(s_t, \pi_\psi(s_t))] - \alpha \mathcal{H}(\pi_\psi(.|s_t)) \Big)$
    Take gradient step on entropy coefficient:
        $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha \Big( \alpha(\mathcal{H}(\pi_\psi(.|s_t)) - H_{\text{tar}}) \Big)$
end
```

7.2. MPC with a Density Model Constraint

One may wonder if imposing the density constraint for a longer horizon by planning with the constraint

$$E_\theta(s_t, a_t) \leq -\log(c) \quad \forall 1 \leq t \leq H \tag{10}$$

in place of Equation (9) would alleviate the myopic behavior shown in Figure 1. We find that the answer is *no*. We construct an example scenario, extending the one in Figure 1, for which planning with with a density model constraint fails even under perfect knowledge of the dynamics and data distribution, but planning with an LDM can succeed.

Example 7.1. Let $\mathcal{S} = \mathcal{A} = \mathbb{Z}$, $f(s, a) = s + a$, $r(s, a) = a$ and $s_0 = 0$. For any $H \geq 1, \epsilon > 0$, $\exists$ data distribution $\mathcal{D}_{H,\epsilon}$ for which planning with constraint (10) will lead to s_{H+1} s.t. $P(s_{H+1}, a) \leq \epsilon \ \forall a \in \mathcal{A}$. However, constraining via the LDM (9) ensures we only reach states s_t for which $\exists a_t$ with $P(s_t, a_t) \geq \frac{1}{2(H+1)}$.

Appendix B contains the explicit construction, a discussion of its interpretation, and the proof of the claims in Ex. 7.1. The failure mode of (10) highlighted above depends on the horizon of the planning problem. This is because larger H enables MPC to plan further into the future, and hence satisfy the density lower bound over more time steps. However, in practice one cannot set H to be too large because exceedingly large horizons are computationally expensive, lead to accumulated model error, and potentially increase variance (Tassa et al., 2012; Deisenroth et al., 2015). Current model-based RL methods that use MPC typically utilize horizons as low as $H = 10$ or $H = 20$ (Chua et al., 2018), $150\times$ lower than the full task length.

The main takeaway is that density thresholding fails even with the true dynamics f and distribution P. Using an LDM constraint, we can plan with any horizon and ensure that we will stay within some reasonably well-visited region (and hence we can support arbitrarily long task-horizon).

8. Theoretical Analysis

So far our discussion has assumed that the LDM is learned perfectly, but this is not the case in practical settings due to approximation error. We analyze the effect of such errors on the LDM next. We aim to answer the following two questions: (1) Can we bound the error in the LDM when it is trained from data? (2) How does this error influence the ability of the LDM to keep MPC procedures (Eq. 8) in-distribution? Our analysis will show that even with approximation error, the LDM retains provable guarantees for keeping MPC within high-density regions, in contrast to the naïve constraint based on the density model. We conclude the section by returning to Example 7.1 and showing that the learned LDM (via Eq. 4) is still able to remedy the myopic tendencies of naive density thresholding.

8.1. Error Bound of LDM Learning Procedure

The goal of this section is to give a formal guarantee for repeatedly performing the update in Eq. 4. One cause for error in our updates is the lack of access to the true data distribution. What we will ultimately converge to is the fixed point of Eq. 3, the γ-discounted LDM using the density model $E(s, a)$: $\hat{G}^\star \doteq \min_{\{a_t\}_{t=1}^\infty} \max_{t \geq 0} \gamma^t E(s_t, a_t)$. Hence, our main result will analyze convergence to this quantity. To do so, we first introduce a quantity that captures the maximal level of 'recoverability' of a system, i.e., how many times larger the maximal reachable density from any given initial conditions can be.

Definition 8.1 (Recoverability)**.** Define the recoverability of the system to be

$$R \doteq \sup_{\substack{s_0, T \geq 0, \{a_t\}_{t=0}^T \\ \text{s.t. } P(s_0, a_0) > 0}} \frac{P(s_T, a_T)}{P(s_0, a_0)} \ \text{ s.t. } s_{t+1} = f(s_t, a_t).$$

Intuitively, a small R would mean that once we reach a low density region, we cannot hope to go back to a high density region, while a large R means that we can considerably multiply the density of any initial condition by acting appropriately. We define the P-norm of a function $\|g\|_P \doteq \mathbb{E}_{(s,a)\sim P}|g(s,a)|$ and its ∞-norm $\|g\|_\infty \doteq \sup_{s,a}|g(s,a)|$. The main result (proved in Appendix B) is stated below:

Proposition 8.2. *Let $\hat{G}_0 = E$. The result of applying the update in Eq. 4 K times satisfies:*

$$\|\hat{G}_K - \hat{G}^\star\|_P \leq \frac{R \cdot \epsilon_{\text{ls}}}{1-\gamma} + \gamma^K \cdot \|E - \hat{G}^\star\|_\infty, \qquad (11)$$

where $\epsilon_{\text{ls}} \doteq \max_{t\in[K-1]} \|\hat{G}_{t+1} - \mathcal{T}\hat{G}_t\|_P$.

Proposition 8.2 quantifies the approximation error from executing the update (Eq. 4) and its dependence on the recoverability factor (R), discount (γ), # of iterations (K), $\|E-\hat{G}^\star\|_\infty$, and the generalization error of the least squares fit (ϵ_{ls}). The quantity ϵ_{ls} is simply the accuracy of fit under a given distribution, and it is standard to assume that this decreases as the dataset becomes larger. Bounds on this quantity fall under the study of supervised learning, and we state a standard bound in Appendix B, along with a corollary regarding its implication for Proposition 8.2.

Below, we give two additional remarks which may capture the convergence behavior of the LDM learning procedure more tightly under certain situations. Firstly, in cases where R is large, we can bound error in terms of the *one-step* recoverability parameter $r \doteq \sup_{s_t,a_t,a_{t+1}}\{\frac{P(f(s_t,a_t),a_{t+1})}{P(s_t,a_t)} \text{ s.t. } P(s_t,a_t) > 0\}$. This captures how much the density can decrease in a single step, which may be orders of magnitude smaller than R.

Remark 8.3. *$\hat{G}_K$ also satisfies $\|\hat{G}_K - \hat{G}^\star\|_P \leq \frac{\epsilon_{\text{ls}}}{1-r\gamma} + (r\gamma)^K \cdot \|E - \hat{G}^\star\|_P$ for $\gamma < r^{-1}$.*

Secondly, there are scenarios (such as when the system is stable) under which we may set $\gamma = 1$.

Remark 8.4. *If $\exists K_{\text{fin}}$ such that the system evolving more than K_{fin} times leads to states where the difference between the density model and the LDM value becomes negligible ($\leq \epsilon_{\text{fin}}$), we can set $\gamma = 1$ and satisfy $\|\hat{G}_K - \hat{G}^\star\|_P \leqslant R \cdot K_{\text{fin}} \cdot \epsilon_{\text{ls}} + \epsilon_{\text{fin}}$ where $\hat{G}^\star$ is the undiscounted LDM.*

As a final note, the LDM as formulated above addresses fully observed settings, but in Appendix E we provide the natural extension of the LDM to non-Markovian/partial state observations. In fact, the *same* procedure as before still provides a distributional shift guarantee, but with an additional error term that accounts for the variability in the observation for the same underlying state (Prop. E.1).

8.2. Error Sensitivity of the LDM Barrier for MPC

In this section we show that, even in the presence of approximation and discount error, a learned LDM constraint (9) retains provable guarantees on staying in high density regions, which means that MPC with an LDM constraint will not query the model $\hat{f}$ on inputs where it was not trained. There are three main sources of approximation error: error in the density model, error from training the LDM, and error from imposing a discount. For cleaner presentation, we assume here the density model satisfies a pointwise error bound: $\exists\epsilon_p > 0$ s.t. $|\log P(s,a) + E(s,a)| \leq \epsilon_p$. In Appendix C.3, all results are restated assuming a probably approximately correct error instead. Our result follows:

Proposition 8.5. *For any probability mass function P and initial state s_0, taking action a_0 with $\hat{G}(s_0,a_0) \leq -\log c$ guarantees that $\forall t$, there is a sequence $a_{1:t}$ for which*

$$\log P(s_t,a_t) \geq \gamma^{-t}\log c - \frac{\gamma^{-t}R\,\epsilon_{\text{ls}}\exp\epsilon_p}{c(1-\gamma)} - \epsilon_P$$

as $K \to \infty$. Further, if the assumption in Remark 8.4 holds, using $\gamma = 1$ ensures

$$\log P(s_t,a_t) \geq \log c - \frac{(RK_{\text{fin}}\epsilon_{\text{ls}} + \epsilon_{\text{fin}})\exp\epsilon_p}{c} - \epsilon_P$$

The γ^{-t} factor in the bound above suggests that distributional shift is harder to avoid further in the future. However, if we are instead interested in the difference between the sum of discounted rewards the agent planned for and the one the agent actually experience, then we get a guarantee which does not decay exponentially in time. Let us denote the 'planned-for' discounted cumulative reward for a state-action sequence $\mathbf{s} = s_{0:T}, \mathbf{a} = a_{0:T}$ by $\hat{R}_T(\mathbf{s},\mathbf{a}) \doteq \sum_{t=0}^{T-1}\gamma^t r(\hat{f}(s_t,a_t),a_{t+1})$ and the 'true' reward $R_T(\mathbf{s},\mathbf{a}) \doteq \sum_{t=0}^{T-1}\gamma^t r(f(s_t,a_t),a_{t+1})$. Assuming that the composition of the reward with the model has error which satisfies $\sup_{a'}|r(f(s,a),a') - r(\hat{f}(s,a),a')| \leq \frac{\epsilon_r}{\sqrt{P(s,a)}}$ for some $\epsilon_r \geq 0$ (which encompasses a $\sim P(s,a)^{-0.5}$ model error together with the Lipschitz property of the reward function), we derive the following corollary from Proposition 8.5:

Corollary 8.6. *For any starting state s_0, taking action a_0 with $\hat{G}(s_0,a_0) \leq -\log c$ with $c \geq \frac{(1-\gamma)+R\epsilon_{\text{ls}}\exp\epsilon_P}{(1-\gamma^{T-1}(\epsilon_P+2\log\epsilon_r))(1-\gamma)}$ guarantees that there is a sequence $a_{1:T}$ that yields $s_{1:T}$ satisfying*

$$|\hat{R}_T(\mathbf{s},\mathbf{a}) - R_T(\mathbf{s},\mathbf{a})| \leq (1+\epsilon_P+2\log\epsilon_r)\cdot\frac{1-\gamma^T}{1-\gamma} + T\cdot\left(\log\frac{1}{c} + \frac{R\epsilon_{\text{ls}}\exp\epsilon_P}{c(1-\gamma)}\right)$$

We see that the first term captures the accumulation of density error over the trajectory and the second term captures how close we can get to this upper bound as a function of the horizon T, the threshold c, the discount γ, and the model errors. For example, if we have $\epsilon_{\text{ls}} \approx 0$ and $c \to 1$, the difference would be dominated by $\approx (1+\epsilon_p)\frac{1-\gamma^T}{1-\gamma}$. By bounding the deviation of the real world discounted reward from the discounted reward expected by the controller, Corollary 8.6 translates the LDM guarantee into a bound on how "over optimistic" the model-based trajectory optimizer can possibly be. Note also that the guarantee applies every time we commit to an action (for $T \gg H$ potentially), and is refreshed/extended under iterative replanning.

8.3. LDM vs. Density Model for MPC

In Example 7.1, we constructed a situation where constraining MPC with an oracle density model of the data distribution can lead to myopic behavior, whereas the true LDM is able to prevent it. A natural question is whether the learnt (hence imperfect) LDM is still capable of eliciting the desired behavior? The answer is *yes*, made precise below:

Example 8.7 (Continuation of Example 7.1)**.** In the setting of Prop 7.1, if MPC with LDM is feasible for $\hat{c} \geq (R(H+2)\epsilon_{\text{ls}} + 2\epsilon_p) \exp \epsilon_p + \epsilon$, we are guaranteed to only reach states with $P(s_t, a_t) \geq \frac{1}{2(H+1)}$. For small ϵ_p, we only need $\hat{c} \gtrapprox R(H+2)\epsilon_{\text{ls}} + \epsilon$.

Intuitively, if $\epsilon_p, \epsilon_{\text{ls}}$ are sufficiently small, we can expect the approximate LDM to succeed in filtering out the path which leads to dangerously low density regions, and, in this example, stay in regions with probability even higher than the formal guarantee. Example 8.7 is an exemplification of how Prop. 8.5 shows that the LDM always ensures bounded distributional shift, solving the underlying horizon-dependent vulnerability of the naïve density constraint to situations where violation after H is inevitable if the wrong action is taken at the current step.

For a more general comparison of the approximate LDM and the density model constraints, we derive an analogue to Props. 8.5 & 8.6 for the density threshold within the horizon H. In contrast to the purely statistical LDM analysis, providing guarantees for density thresholding requires additional smoothness assumptions on E and f. In Appendix C.3, we instantiate these assumptions and give the corresponding guarantees for density model constrained MPC.

To conclude, our results bound the error from using an *imperfect* LDM as an MPC constraint, showing that it will more effectively constrain short-horizon MPC (with small values of H) than a naïve density constraint.

9. Experiments

In this section, we present an experimental evaluation of our method. Our experiments aim to compare LDMs with other techniques for avoiding distribution shift. Prior works in model-based RL that consider this problem generally utilize density models or other error estimation schemes, such as ensembles (Chua et al., 2018; Kidambi et al., 2020; Yu et al., 2020). To provide an apples-to-apples comparison, we will compare LDMs to a baseline that uses a density model as a constraint (10), as well as one that uses the variance of an ensemble of dynamics models as a constraint. These baselines are broadly representative of "greedy" constraints that forbid actions that violate some "local" metric, as opposed to the LDM, which takes into account *future* state probabilities. We also compare to a naïve baseline that does not employ any constraint at all. In our experiments, a static dataset is used to train the density model, LDM, ensemble models, and the dynamics model used for downstream control. We perform MPC with the learned dynamics model, analogous to prior model-based RL techniques (Chua et al., 2018), and constrain the planning procedure with the LDM or the baseline constraint functions. Further details about our implementation are given in Appendix D.

We conduct our evaluation on two RL benchmark environments, hopper and lunar lander (Brockman et al., 2016), and a medical application, SimGlucose (Xie, 2018). The objectives for the hopper and lunar lander tasks are to control the agent to reach different target locations (x position for hopper, and landing pad for lunar lander), and the objective for SimGlucose is to maintain the patient's glucose level close to a target setpoint. To effectively solve these tasks, the learning-based policy must not only accurately model complex dynamics (e.g. to perform a delicate landing procedure without crashing for lunar lander), but also be flexible to different target goals (e.g. different target x positions for the hopper require different behavior, such as hopping forwards or backwards, so directly copying the actions in the data would not be effective). For more details on the datasets and tasks, see Appendix D.1.

We aim to answer the following questions: 1) how does the value of the threshold used for the MPC constraint influence the behavior of the policy? 2) how does the performance of a MPC controller with an LDM constraint compare to that of using a density model constraint, an ensemble constraint, and no constraint?

To answer our first question on how the choice of the MPC constraint threshold influences the resulting policy, we aim to measure how the policy return and the degree to which the policy can keep the agent in-distribution changes as the threshold value changes. However, it is not completely clear how to objectively measure whether a generated trajectory is "in-distribution." Our tasks involve a termination conditions that ends a trajectory when the agent enters a "failure" state (e.g., falling over for the hopper; becoming dangerously hypo/hyperglycemic for SimGlucose). Because the points

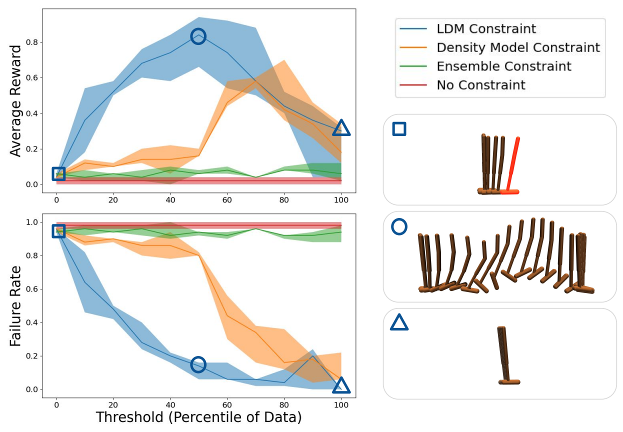

Figure 3. Evaluation of average reward and failure rate over different threshold values for the hopper task (left), and example trajectories from an LDM-constrained policy with thresholds at 0, 50, and 100 (right). The x axis of the plots on the left represents the percent of the dataset whose value under the constraint function falls below the associated threshold c. More specifically, $x = \text{percentile}(\{W(s_t^i, a_t^i)\}_{i=1}^N, c)$, where W is the constraint function. We use this representation of constraint values in order to make them comparable across different constraint functions. A hopper figure colored red represents termination due to failure.

in our dataset are concentrated in non-failing states, we use the rate of failure as a proxy for the fraction of trajectories that go out of distribution. Figure 3 shows an example of the average return and failure rate across different thresholds for the hopper task (For the other tasks, see Appendix D.7). We see that for low threshold values, the policies have low reward and high failure rate, as a consequence of excessive model exploitation (similar to the "no constraint" case). For high threshold values, the policies have low reward and low failure rate, because the over-conservative LDM constraint keeps the agent in-distribution but doesn't leave enough flexibility for the agent to perform the desired task. For a concrete example, consider the figures of the hopper trajectories from the LDM constrained policy at different threshold values: the low threshold trajectory consists of the hopper falling over, the medium threshold trajectory has the hopper successfully hopping to the goal, and the high threshold trajectory consists of the hopper standing still, which effectively stays in-distribution but doesn't accomplish the task. Thus, the threshold can be thought of as the user's knob for controlling the tradeoff between protecting against model error vs. flexibility for performing the desired task.

To answer our second question, we compare the performance of performing MPC with an LDM constraint, a density model constraint, an ensemble constraint, and no constraint for each task in Figure 4. The performance of the constrained MPC methods depends on the user-chosen threshold value for the constraints. For a fair comparison, we report the performance for the best threshold for each

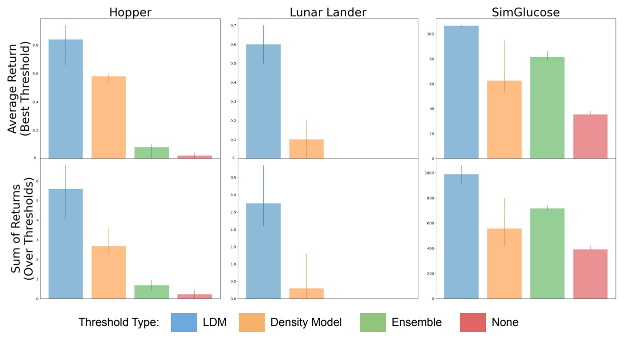

Figure 4. Evaluation of the average reward attained by an MPC controller constrained with an LDM, a density model, an ensemble, and no constraint. The top row shows the best performance over thresholds for each method, and the bottom row shows the performance summed in aggregate over all thresholds. The bars show the median performance over random seeds, and the top and bottom of the error bars represent the 25/75 percentiles. We evaluated 5 random seeds for each task.

method in the top row. Furthermore, to assess each method's sensitivity to the choice of threshold, we also report the sum of the performances over a range of thresholds for each method (area underneath each curve in the average rewards plot of Fig. 3) in the bottom row. See Appendix D.7 for a full sweep of threshold values vs. performance for each task. We see that the unconstrained MPC policy was rarely successful in performing the task (red). In comparison, the density model (orange) and ensemble (green) constrained MPC methods yielded better performance by considering the data distribution while planning with a learned model. Finally, the LDM constrained MPC policy (blue) is able to most effectively perform the task for each tested environment, because it is able to reason about the data distribution in a dynamics aware fashion.

10. Discussion and Future Work

We presented Lyapunov density models (LDMs), a tool that can ensure that an agent remains within the distribution of the training data. We provide a definition of the LDM, discuss its properties, and present a practical algorithm that learns LDMs from data. Furthermore, we provide a method of using an LDM in control, and present theoretical and empirical results showing its benefits to using density models. By making a formal connection between data distribution and control-invariance, we believe that our framework is a step towards developing learning-based control algorithms that resolve the issue of model unreliability due to distribution shifts. In this work, we focused on applying our framework in model-based RL; an exciting direction for future work would be to combine an LDM with other types of algorithms, such as model-free RL or imitation learning.

References

Agarwal, A., Jiang, N., Kakade, S. M., and Sun, W. Reinforcement learning: Theory and algorithms, 2021. URL https://rltheorybook.github.io/rltheorybook_AJKS.pdf.

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Ames, A., Xu, X., Grizzle, J., and Tabuada, P. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62: 3861–3876, 2017.

Antos, A., Szepesvari, C., and Munos, R. Value-iteration based fitted policy iteration: Learning with a single trajectory. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pp. 330–337, 2007. doi: 10.1109/ADPRL.2007.368207.

Bansal, S., Chen, M., Herbert, S. L., and Tomlin, C. Hamilton-jacobi reachability: A brief overview and recent advances. *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2242–2253, 2017.

Berkenkamp, F., Turchetta, M., Schoellig, A. P., and Krause, A. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems*, 2017.

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

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Chang, Y.-C., Roohi, N., and Gao, S. Neural lyapunov control. *ArXiv*, abs/2005.00611, 2019.

Cheng, R., Orosz, G., Murray, R. M., and Burdick, J. W. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models, 2018.

Dai, H., Landry, B., Yang, L., Pavone, M., and Tedrake, R. Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152*, 2021.

Dean, S., Taylor, A. J., Cosner, R. K., Recht, B., and Ames, A. D. Guaranteeing safety of learned perception modules via measurement-robust control barrier functions, 2020.

Deisenroth, M. P., Fox, D., and Rasmussen, C. E. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, Feb 2015. ISSN 2160-9292. doi: 10.1109/tpami.2013.218. URL http://dx.doi.org/10.1109/TPAMI.2013.218.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Neural spline flows. *Advances in Neural Information Processing Systems*, 2019.

Fisac, J. F., Lugovoy, N. F., Rubies-Royo, V., Ghosh, S., and Tomlin, C. J. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8550–8556, 2019.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration, 2019.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Hsu, D., Kakade, S. M., and Zhang, T. Random design analysis of ridge regression, 2014.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation, 2018.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

Kumar, A., Fu, J., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Laroche, R., Trichelair, P., and Combes, R. T. D. Safe policy improvement with baseline bootstrapping. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3652–3661. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/laroche19a.html`.

Li, Z., Cheng, X., Peng, X. B., Abbeel, P., Levine, S., Berseth, G., and Sreenath, K. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, June 2021.

McAllister, R., Kahn, G., Clune, J., and Levine, S. Robustness to out-of-distribution inputs via task-aware generative uncertainty. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2083–2089. IEEE, 2019.

Mehrjou, A., Iannelli, A., and Schölkopf, B. Learning dynamical systems using local stability priors. *arXiv preprint arXiv:2008.10053*, 2020.

Mittal, M., Gallieri, M., Quaglino, A., Salehian, S. S. M., and Koutník, J. Neural lyapunov model predictive control. *arXiv preprint arXiv:2002.10451*, 2020.

Murray, R. M., Li, Z., and Sastry, S. S. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

Nair, A., Dalal, M., Gupta, A., and Levine, S. Accelerating online reinforcement learning with offline datasets. *ArXiv*, abs/2006.09359, 2020.

Richards, S. M., Berkenkamp, F., and Krause, A. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Conference on Robot Learning*, pp. 466–476. PMLR, 2018.

Richter, C. and Roy, N. Safe visual navigation via deep learning and novelty detection. *RSS 2017*, 2017.

Robey, A., Hu, H., Lindemann, L., Zhang, H., Dimarogonas, D. V., Tu, S., and Matni, N. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 3717–3724, 2020.

Sastry, S. *Nonlinear systems : analysis, stability, and control / Shankar Sastry.* Interdisciplinary applied mathematics ; v. 10. Springer, New York, 1999. ISBN 0387985131.

Sontag, E. D. A 'universal' construction of artstein's theorem on nonlinear stabilization. *Systems & Control Letters*, 13(2):117–123, 1989. ISSN 0167-6911.

Tassa, Y., Erez, T., and Todorov, E. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913, 2012. doi: 10.1109/IROS.2012.6386025.

Tu, S., Robey, A., Zhang, T., and Matni, N. On the sample complexity of stability constrained imitation learning. In *Learning for Dynamics and Control Conference*. PMLR, 2022.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Xie, J. Simglucose v0.2.1, 2018. URL `https://github.com/jxx123/simglucose`.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J., Levine, S., Finn, C., and Ma, T. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

A. Appendix to Section 5

A.1. Additional details of 2D Linear System Example

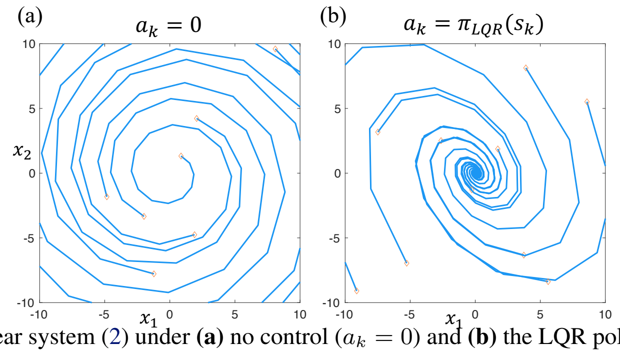

Figure 5. Phase portraits of the linear system (2) under **(a)** no control ($a_k = 0$) and **(b)** the LQR policy ($a_k = \pi_{LQR}(s_k)$), respectively. The uncontrolled dynamics of this system is unstable, therefore, there is no invariant region except for the origin. On the other hand, the LQR policy can stabilize the system to the origin, resulting in the trajectories to stay in a bounded region. Thus, the data distribution centered at this policy (case (b) in Fig. 2) will by its nature, have large state-action control invariant region.

In Figure 2 and 6, level sets of the maximal LDMs $G(s, a)$ (on the right) that can be obtained for various distributions of the data density $P(s, a)$ (on the left) are visualized. The maximal LDMs (1) are computed on the grid of (s, a) by applying the exact LDM backup (3) described in Section 6 with $\gamma = 1$. The computation is done on the state-action grid of $[-10, 10] \times [-10, 10] \times [-5, 5]$ with the size $(201, 201, 101)$, and it takes about a minute to obtain the converged $G(s, a)$ on a standard laptop. After obtaining the LDMs, from Theorem 5.3, the maximal state-control invariant sets which can maintain density levels $P(s_t, a_t) \geq c$ are obtained by taking $\mathcal{G}_c := \{(s, a) : G(s, a) \leq -\log(c)\}$.

In addition to the two cases described in the main text, in Figure 6, a toric data distribution is considered ((a) left). Note that the natural dynamics of (2) induces symmetric rotations. The invariant sets verified from the computed LDM ((a) right) reveal that indeed it is possible to maintain the system to stay in a torus with an appropriate density level. Also, we can obtain an policy for staying in this set by taking $\pi_{opt}(s) := \arg\min_{a \in \mathcal{A}} G(s, a)$. The resulting trajectory under this policy is visualized in (b).

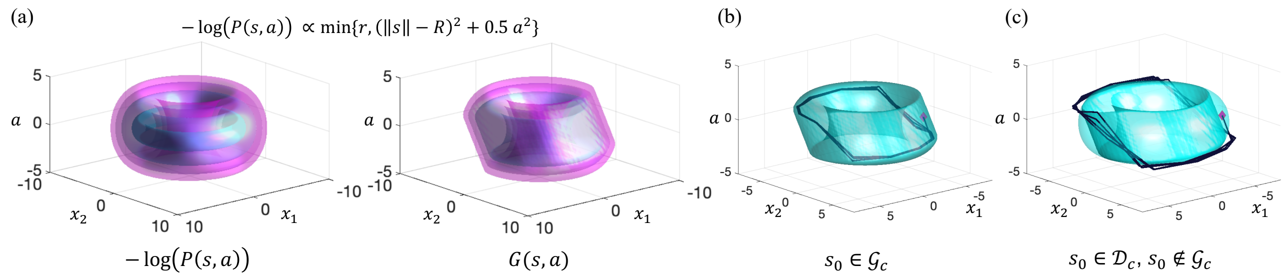
(a)
$-\log(P(s,a)) \propto \min\{r, (\|s\| - R)^2 + 0.5\,a^2\}$
(b)
(c)
a
x_1
x_2
$-\log(P(s,a))$
$G(s,a)$
$s_0 \in \mathcal{G}_c$
$s_0 \in \mathcal{D}_c,\ s_0 \notin \mathcal{G}_c$

Figure 6. The level sets of a toric data distribution for the linear system (2) are visualized in **(a)** left, and the level sets of the resulting maximal LDM are visualized in **(a)** right. In **(b)**, the smallest level set of $G(s, a)$ among the three sets in (a) are visualized in cyan again. The trajectory under the policy $\pi_{opt}(s) = \arg\min_{a \in \mathcal{A}} G(s, a)$, starting at an initial state s_0 (indicated by the diamond marker) that is contained in $\mathcal{G}_c = \{(s, a) : G(s, a) \leq -\log(c)\}$ is visualized in black. This shows that the obtained set $\mathcal{G}_c$ is indeed state-action control invariant which can be achieved by the policy $\pi_{opt}(s)$. In contrast, in **(c)**, an initial state that is not contained in $\mathcal{G}_c$ but still contained in $\mathcal{D}_c = \{(s, a) | P(s, a) \geq c\}$ (the bigger level set in cyan) is selected and the same optimal policy is tested out. Although the initial state is in-distribution, it is inevitable that the trajectory escapes $\mathcal{D}_c$.

A.2. Proofs of Theorems 5.3, Proposition 5.5

Proof of Theorem 5.3. Suppose $(s_t, a_t) \in \mathcal{G}_p$. By Condition 1 in Definition 5.2, we know that $\exists a_{t+1} \in \mathcal{A}$ such that $G(s_t, a_t) \geq G(f(s_t, a_t), a_{t+1})$. Furthermore, since $\mathcal{G}_p$ is the sublevel-set of $G(s_t, a_t)$ at $-\log(p)$, we have $-\log(p) \geq G(s_t, a_t) \geq G(f(s_t, a_t), a_{t+1}) = G(s_{t+1}, a_{t+1})$, so $(s_{t+1}, a_{t+1}) \in \mathcal{G}_p$. Now suppose $(\mathbf{s}_0, \mathbf{a}_0) \in \mathcal{G}_p$. By proof by induction, we know that $\exists \{a_t\}_{t=1}^{\infty}$ such that $(s_t, a_t) \in \mathcal{G}_p \quad \forall t \geq 0$. Thus, $\mathcal{G}_p$ is a generalized control-invariant set. Finally, by Condition 2 in Definition 5.2, we know that $G(s_t, a_t) \geq -\log(P(s_t, a_t))$, so $P(s_t, a_t) \geq p \quad \forall (s_t, a_t) \in \mathcal{G}_p$. $\square$

Proof of Proposition 5.5. First, we show that $G(s,a)$, as defined by 1, is a Lyapunov density model. We prove each condition of an LDM individually:

- Condition 1: $\forall(s_t,a_t)$, $G(s_t,a_t) = \max\{-\log(P(s_t,a_t)), \min_{a_{t+1}} G(f(s_t,a_t),a_{t+1})\} \geq \min_{a_{t+1}} G(f(s_t,a_t),a_{t+1})$. Thus, $\exists a_{t+1}$ such that $G(s_t,a_t) \geq G(f(s_t,a_t),a_{t+1})$.
- Condition 2: $\forall(s_t,a_t)$, $G(s_t,a_t) = \max\{-\log(P(s_t,a_t)), \min_{a_{t+1}} G(f(s_t,a_t),a_{t+1})\} \geq -\log(P(s_t,a_t))$

Thus, by Theorem 5.3, each sub-level set of $G(s,a)$, $\mathcal{G}_p = \{(s_t,a_t)|G(s_t,a_t) \leq -\log(p)\}$, is a generalized control-invariant set such that $\forall(s_t,a_t) \in \mathcal{G}_p$, $P(s_t,a_t) \geq p$. To show that $G(s_t,a_t)$ is the *maximal* LDM, we will next show that each $\mathcal{G}_p$ is the *largest* generalized control invariant set contained inside $\mathcal{D}_p = \{(s_t,a_t)|P(s_t,a_t) \geq p\}$.

Suppose there exists a generalized control-invariant set $\mathcal{S}_p$ such that $\mathcal{G}_p \subset \mathcal{S}_p \subseteq \mathcal{D}_p$. Let (s_0,a_0) be a point in $\mathcal{S}_p$ but not in $\mathcal{G}_p$. Because $\mathcal{S}_p$ is a generalized control-invariant set inside $\mathcal{D}_p$, for initial condition (s_0,a_0), $\exists\{a_t\}_{t=1}^{\infty}$ such that $\max_{t\geq 0} -\log(P(s_t,a_t)) \leq -\log(p)$. Because (s_0,a_0) is not in $\mathcal{G}_p$, $G(s_0,a_0) > -\log(p)$. This is a contradiction, because $G(s_t,a_t)$ is by definition $\min_{\{a_t\}_{t=1}^{\infty}} \max_{t\geq 0} -\log(P(s_t,a_t))$, so there cannot exist $\{a_t\}_{t=1}^{\infty}$ such that $\max_{t\geq 0} -\log(P(s_t,a_t)) \leq -\log(p) < G(s_0,a_0)$. Thus, $\mathcal{G}_d$ must be the largest generalized control invariant set. □

A.3. LDM Connections to Density models and CLFs

There are two intuitive ways to view an LDM: 1) a Lyapunov function that "stabilizes" towards a distribution rather than a single equilibrium point, or 2) a density model of the lowest data density over an agent's trajectory.LDMs are in fact a generalization of both Lyapunov functions and density models. For the special case of a static system $s_{t+1} = f(s_t,a_t) = s_t$, $-\log(P(s_t,a_t))$ is a valid LDM for the data distribution $P(s_t,a_t)$. Note that this is just an example, not a necessary condition for the LDM to coincide with the density model.

On the other hand, an LDM can be viewed as a somewhat more expressive version of a Lyapunov function, because it is a function of states and actions, rather than just states, and is tied to a distribution rather than an equilibrium point. However, if the data distribution is peaked at an equilibrium point, then we can use the LDM to recover a CLF that stabilizes the system around that equilibrium point.

Lemma A.1. *Let $\mathbf{s}_e$ be an equilibrium point of the system, and $\mathbf{a}_e$ be the associated steady state action. If an LDM $G(s_t,a_t)$ is radially unbounded and has a unique minimizer at $(\mathbf{s}_e,\mathbf{a}_e)$, then $W(s_t) = \min_{a_t\in\mathcal{A}} G(s_t,a_t) - G(\mathbf{s}_e,\mathbf{a}_e)$ is a control Lyapunov function of the system.*

Proof of Lemma A.1. We prove each condition of a control Lyapunov function individually:

- Continuity: Since $G(s_t,a_t)$ is continuous, $W(s_t) = \min_{a_t\in\mathcal{A}} G(s_t,a_t) - G(s_e,a_e)$ is also continuous by Berge's Maximum theorem.
- Radial Unboundedness: Let $a_t^* = \text{argmin}_{a_t\in\mathcal{A}} G(s_t,a_t)$. $||s_t|| \to \infty \implies ||[s_t,a_t^*]|| \to \infty \implies G(s_t,a_t^*) \to \infty \implies W(s_t) = \min_{a_t\in\mathcal{A}} G(s_t,a_t) - G(s_e,a_e) \to \infty$
- Condition 1: From Condition 1 of Definition 5.2, we know that $\forall(s_t,a_t)$, $\exists a_{t+1}$ such that $G(s_t,a_t) \geq G(f(s_t,a_t),a_{t+1})$. Let $a_t^* = \text{argmin}_{a\in\mathcal{A}} G(s_t,a_t)$, $\exists a_{t+1}$ such that $G(s_t,a_t^*) - G(s_e,a_e) \geq G(f(s_t,a_t^*),a_{t+1}) - G(s_e,a_e)$. Thus, $W(s_t) = \min_{a_t\in\mathcal{A}} G(s_t,a_t) \geq \min_{a_{t+1}\in\mathcal{A}} G(f(s_t,a_t^*),a_{t+1}) - G(s_e,a_e) = W(f(s_t,a_t^*))$. Thus, $\forall s \in \mathcal{S}$, $\exists a \in \mathcal{A}$ such that $W(s) \geq W(f(s,a))$.
- Condition 2: Since (s_e,a_e) is the unique minimizer of $G(s,a)$, $\forall(s,a) \neq (s_e,a_e)$, $G(s,a) > G(s_e,a_e)$. Thus, $\forall s \neq s_e$, $\min_{a\in\mathcal{A}} G(s,a) > G(s_e,a_e)$, so $W(s) = \min_{a\in\mathcal{A}} G(s,a) - G(s_e,a_e) > 0$.
- Condition 3: $W(s_e) = \min_{a_t\in\mathcal{A}} G(s_e,a_t) - G(s_e,a_e) = G(s_e,a_e) - G(s_e,a_e) = 0$.

A.4. Convergence of the value iteration algorithm for the maximal LDM.

Lemma A.2. *Define $G'_T(s_0,a_0) := \min_{\{a_t\}_{t=1}^{T}} \max_{t\in[0,T]} E(s_t,a_t)$ with $E(s,a) := -\log P(s,a)$. Then*

$$G'_{T+1}(s_0,a_0) = \max\left\{E(s_0,a_0), \min_{a_1\in\mathcal{A}} G'_T(s_1,a_1)\right\}, \tag{12}$$

where $s_1 = f(s_0, a_0)$.

Proof. To simplify the notation, we use $a_{m:n}$ to indicate the sequence $\{a_t\}_{t=m}^n$.

$$G'_{T+1}(s_0,a_0) = \min_{a_{1:T+1}} \max_{t\in[0,T+1]} E(s_t,a_t) = \min_{a_{1:T+1}} \max\left\{E(s_0,a_0), \max_{t\in[1,T+1]} E(s_t,a_t)\right\}$$
$$= \max\left\{E(s_0,a_0), \min_{a_1\in\mathcal{A}}\left\{\min_{a_{2:T+1}} \max_{t\in[1,T+1]} E(s_t,a_t)\right\}\right\} = \max\left\{E(s_0,a_0), \min_{a_1\in\mathcal{A}} G'_T(s_1,a_1)\right\}.$$

□

Note that Lemma A.2 implies that the finite-horizon version of the maximal LDM satisfies the Bellman principle associated with the LDM backup operator $\mathcal{T}$ (3) with $\gamma = 1$.

Theorem A.3. *The value iteration algorithm with* $G_{k+1} \leftarrow \mathcal{T}G_k$*, with* $\mathcal{T}$ *defined in* (3) *under* $\gamma = 1$ *and with initialization* $G_0(s,a) = -\log(P(s,a))$*, results in 1)* $G_k = G'_k$ *for* $\forall\, k \geq 0$*, and 2) for any* $(s,a) \in \mathcal{S}\times\mathcal{A}$ *such that the maximal LDM* $G(s,a)$ *defined in* (1) *is finite,* $\lim_{k\to\infty} G_k(s,a) = G(s,a)$.

Proof. 1) is a direct corollary of Lemma A.2. Then, by the definition of G'_k,

$$G_{k+1}(s,a) = \min_{a_{1:k+1}} \max_{t\in[0,k+1]} E(s_t,a_t) = \min_{a_{1:k+1}} \max\left\{\max_{t\in[0,k]} E(s_t,a_t), E(s_{k+1},a_{k+1})\right\}$$
$$\geq \min_{a_{1:k+1}} \max_{t\in[0,k]} E(s_t,a_t) = G_k(s,a).$$

In a similar way, one can easily proof that $G(s,a) \geq \min_{a_{1:\infty}} \max_{t\in[0,k]} E(s_t,a_t) = G_k(s,a)$ for $\forall\, k \geq 0$. Therefore, for all $(s,a) \in \mathcal{S}\times\mathcal{A}$ such that $G(s,a)$ is finite, $\{G_k(s,a)\}_{k=0}^\infty$ is a non-decreasing sequence bounded above. By monotone convergence theorem, this sequence converges to its supremum which is by definition, the maximal LDM $G(s,a)$. □

B. Appendix to Section 7

Proof of Proposition 7.1. Let K be such that $1/K \leq 2(H+1)\epsilon$. We construct $\mathcal{D}$ by its generating law as follows:

$$P_\mathcal{D}(s,-1) = \begin{cases} \frac{1}{2(H+1)} & \text{if } s \in \{-(H-1),\dots,0\} \\ 0 & \text{o.w.} \end{cases} \qquad P_\mathcal{D}(s,1) = \begin{cases} \frac{1}{2(H+1)} & \text{if } \in \{0,\dots,H-1\} \\ 0 & \text{o.w.} \end{cases}$$
$$P_\mathcal{D}(-H,0) = \frac{1}{2(H+1)} \qquad P_\mathcal{D}(H,k) = \begin{cases} \frac{1}{2(H+1)\mathbf{K}} & \text{for } k \in \{0,\dots,(K-1)\} \\ 0 & \text{o.w.} \end{cases}$$

and $P_\mathcal{D}(s,a) = 0$ wherever not defined above. Note that the above probabilities sum to 1 and that the maximal p for which the MPC problem with a naïve distribution constraint is feasible is $p^\star = \frac{1}{2(H+1)}$. Because our reward is a and because taking $a_{1:H} = 1$ satisfies the constraint at the very first timestep $t = 1$, we will take $a_1 = 1$. From hereon out, the dataset contains only '1' actions on states $\overline{1, H-1}$ so we are doomed to always take $a_t = 1$ and hence reach $s_{H+1} = H$ where by construction $P_\mathcal{D}(s_{H+1},a) < \epsilon$ for any action a.

This construction corresponds to data collected from a game where the agent can go right or left or stay still with $-H$ being the only terminal state and that the player deterministically stops the collection after its $H + 1^{th}$ action. This time, if it took H actions and didn't finish, it takes one random action to see what happens then ends the data collection. The uncertainty is therefore caused by the unstable behavior of the player at state $-H$.

Finally, it remains to show that the LDM has the promised guarantee on this construction. First note that taking

$$a_t = \begin{cases} -1 \text{ if } s < H \\ 0 \text{ o.w.} \end{cases}$$

guarantees we always have $P_\mathcal{D}(s_t,a_t) \geq \frac{1}{2(H+1)}$ arbitrarily far in the future. Using any non-trivial LDM, and in particular the maximal LDM, as a constraint will guarantee that we satisfy $P(s_t,a_t)$ due to Theorem 5.3. Hence, via the LDM constraint, even if the data was collected from short trajectories, we can give infinite-horizon guarantees. □

C. Appendix to Section 8

We first prove Proposition 8.2 as stated in the main text in C.1, then derive bounds on ϵ_{ls} with an associated corollary that contains all the dependencies in C.2.

C.1. Proof of Proposition 8.2

Proof of Proposition 8.2. Consider arbitrary G_1, G_2 and their optimal safety policies. If $G_1(f(s,a), \pi_1(f(s,a))) \leq G_2(f(s,a), \pi_2(f(s,a)))$, we have that:

$$\begin{aligned}
\mathcal{T}G_2(s,a) - \mathcal{T}G_1(s,a) &= \max\{E(s,a), \gamma G_2(f(s,a), \pi_2(f(s,a)))\} - \max\{E(s,a), \gamma G_1(f(s,a), \pi_1(f(s,a)))\} \\
&\leq \max\{E(s,a), \gamma G_2(f(s,a), \pi_1(f(s,a)))\} - \max\{E(s,a), \gamma G_1(f(s,a), \pi_1(f(s,a)))\} \\
&= \begin{cases} 0 \text{ if } \gamma G_2(f(s,a), \pi_1(f(s,a))) \leq E(s,a) \\ \gamma G_2(f(s,a), \pi_1(f(s,a))) - \max\{E(s,a), \gamma G_1(f(s,a), \pi_1(f(s,a)))\} \text{ o.w.} \end{cases} \\
&\leq \gamma(G_2(f(s,a), \pi_1(f(s,a))) - G_1(f(s,a), \pi_1(f(s,a))))
\end{aligned}$$

otherwise (if $G_1(f(s,a), \pi_1(f(s,a))) > G_2(f(s,a), \pi_2(f(s,a)))$, analogously (by symmetry) we have

$$\mathcal{T}G_1(s,a) - \mathcal{T}G_2(s,a) \leq \gamma(G_1(f(s,a), \pi_2(f(s,a))) - G_2(f(s,a), \pi_2(f(s,a))))$$

so letting $\pi_{12}(s) \doteq \begin{cases} \pi_1(s) \text{ if } G_1(s, \pi_1(s)) \leq G_2(s, \pi_2(s)) \\ \pi_2(s) \text{ o.w.} \end{cases}$ we have

$$|\mathcal{T}G_1(s,a) - \mathcal{T}G_2(s,a)| \leq \gamma |G_1(f(s,a), \pi_{12}(f(s,a))) - G_2(f(s,a), \pi_{12}(f(s,a)))|$$

We can apply this recursively to obtain that for any s, a and $t \geq 1$, there exists a sequence of actions $a_0 = a$, $a_{1:t}$ and corresponding states $s_0 = s$, $s_{k+1} = f(s_k, a_k)$ for $k = \overline{0, t-1}$ such that

$$|\mathcal{T}^{(t)}G_1(s,a) - \mathcal{T}^{(t)}G_2(s,a)| \leq \gamma^t |G_1(s_t, a_t) - G_2(s_t, a_t)|$$

and using the recoverability factor from Definition 8.1, we can bound (note that $(s,a) = (s_0, a_0)$):

$$\begin{aligned}
\|\mathcal{T}^{(t)}G_1(s,a) - \mathcal{T}^{(t)}G_2(s,a)\|_P &\leq \gamma^t \sum_{s_0, a_0} |G_1(s_t, a_t) - G_2(s_t, a_t)| \cdot P(s_0, a_0) \\
&\leq R\gamma^t \sum_{s_t, a_t} |G_1(s_t, a_t) - G_2(s_t, a_t)| \cdot P(s_t, a_t) \\
&= R\gamma^t \|G_1 - G_2\|_P
\end{aligned}$$

or, for continuous state-action spaces,

$$\begin{aligned}
\|\mathcal{T}^{(t)}G_1(s,a) - \mathcal{T}^{(t)}G_2(s,a)\|_P &\leq \gamma^t \int_{s_0, a_0} |G_1(s_t, a_t) - G_2(s_t, a_t)| \cdot P(s_0, a_0) \\
&\leq R\gamma^t \int_{s_t, a_t} |G_1(s_t, a_t) - G_2(s_t, a_t)| \cdot P(s_t, a_t) \\
&= R\gamma^t \|G_1 - G_2\|_P
\end{aligned}$$

Now since $\mathcal{T}\hat{G}^\star_\gamma = \hat{G}^\star$, by triangle inequality we have:

$$\begin{aligned}
\|\hat{G}_K - \hat{G}^\star\|_P &\leq \|\mathcal{T}\hat{G}_{K-1} - \mathcal{T}\hat{G}^\star\|_P + \|\hat{G}_K - \mathcal{T}\hat{G}_{K-1}\|_P \\
&\leq \|\mathcal{T}^{(2)}\hat{G}_{K-2} - \mathcal{T}^{(2)}\hat{G}^\star\|_P + \|\mathcal{T}\hat{G}_{K-1} - \mathcal{T}^{(2)}\hat{G}_{K-2}\|_P + \epsilon_{\text{ls}} \\
&\leq \|\mathcal{T}^{(2)}\hat{G}_{K-2} - \mathcal{T}^{(2)}\hat{G}^\star\|_P + R\gamma\epsilon_{\text{ls}} + \epsilon_{\text{ls}} \\
&\leq \ldots \\
&\leq \|\mathcal{T}^{(K)}\hat{G}_0 - \hat{G}^\star\|_P + R(\gamma^{K-1} + \ldots + 1) \cdot \epsilon_{\text{ls}} \\
&\leq \frac{R(1-\gamma^K)}{1-\gamma} \cdot \epsilon_{\text{ls}} + \min\{R\gamma^K \|E - \hat{G}^\star\|_P, \gamma^K \|E - \hat{G}^\star\|_\infty\}
\end{aligned}$$

which is clearly less than the bound in the Proposition statement. Note that there may be cases where $R\gamma^K \|E - \hat{G}^\star\|_P < \gamma^K \|E - \hat{G}^\star\|_\infty$, i.e. if the maximal deviation is very large but the P-weighted deviation is not. We chose the stated formulation to disentangle the dependency on R for presentability purposes.

As remarked in footnote, if we have $\gamma = 1$, we would instead get:

$$\|\hat{G}_K - \hat{G}^\star\|_P \leq R \cdot K \cdot \epsilon_{\text{ls}} + \sup_{P^{(K)}} \|E - \hat{G}^\star\|_{P^{(K)}}$$

where the supremum is taken over all probability distributions that could be reached by acting according to any arbitrarily complicated, possibly non-stationary policy for K steps (applied to each state-action pair). This bound could be useful if regardless of how we act $P^{(K)}$ concentrates to a region where E and $\hat{G}^\star$ coincide. It is envisonable that this could happen for stable systems for example.

Finally, Remark 8.3 follows by considring only $t = 1$, using the definition of one-step recoverability and recursively obtaining:

$$\begin{aligned}\|\hat{G}_K - \hat{G}^\star\|_P &\leq \|\mathcal{T}\hat{G}_{K-1} - \mathcal{T}\hat{G}^\star\|_P + \|\hat{G}_K - \mathcal{T}\hat{G}_{K-1}\|_P \\ &\leq r\gamma\|\hat{G}_{K-1} - \hat{G}^\star\| + \epsilon_{\text{ls}} \\ &\leq \dots \\ &\leq \frac{(1-(r\gamma)^K)\cdot\epsilon_{\text{ls}}}{1-r\gamma} + (r\gamma)^K\|E - \hat{G}^\star\|_P\end{aligned}$$

as stated. □

Depending on the tradeoff between r and R, different approaches to choosing γ may be better. For example, if $r \ll R$ or if r is very close to 1, then it would be worthwhile to select $\gamma = r^{-1}$ so that we can rely on the guarantees of Remark 8.3 instead. On the other hand, if r is not much smaller than R it could be better to just try to optimize for best γ closer to 1 and rely on the main guarantee.

We also make a final comment the converge behavior we expect in scenarios where Remark 8.4 holds.

Remark C.1 (Convergence to undiscounted LDM). *In this case if the dataset has an expected trajectory length K_{avg}, then by Markov's inequality we can get that $K_{\text{fin}} \leq K_{\text{avg}} \cdot (\#\ traj)^{1/4}$ with probability at least $1 - (\#\ traj)^{-1/4}$. As such, since $|D| \approx K_{\text{avg}} \cdot (\#\ traj)$, we could expect that with high probability, $K \times \epsilon_{\text{ls}}$ scales as $(\#\ traj)^{-1/4}$, hence ensuring converge to the undiscounted LDM.*

C.2. Bounding the Generalization Error ϵ_{ls}

We can state the following standard results based on some complexity measure of the function class $\mathcal{G}$. For example we can use the following standard result:

Lemma C.2 (Least Squares Generalization Bound). *Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$ where $x_i \in \mathcal{X}$ and $x_i, y_i \sim \mu$, $y_i = f^\star(x_i) + \epsilon_i$, where $|y_i| \leq Y$ and $|\epsilon_i| \leq \sigma$ for all i, and $\{\epsilon_i\}$ are independent from each other. Given a function class $\mathcal{F} : \mathcal{X} \to [0, Y]$ such that $\min_{f\in\mathcal{F}} \mathbb{E}_{x\sim\mu}(f^\star(x) - f(x))^2 \leq \epsilon_{\text{aprx}}$. If we denote $\hat{f} = \arg\min_{f\in\mathcal{F}} \sum_{i=1}^n (f(x_i) - y_i)^2$. With probability at least $1 - \delta$, we have:*

$$\mathbb{E}_\mu\left[(\hat{f}(x) - f^\star(x))^2\right] \leq \frac{144Y^2 \ln(|\mathcal{F}|/\delta)}{n} + 8\epsilon_{\text{aprx}}$$

For a proof see Lemma 6 from Hsu et al. (2014) or Lemma A.11. from Agarwal et al. (2021).

Corollary C.3. *With probability at least $1 - \delta$ we have*

$$\epsilon_{\text{ls}} \leq \frac{12E_{\max}\sqrt{\ln(|\mathcal{G}|^2 K/\delta)}}{\sqrt{|D|}} + 3\sqrt{\epsilon_{\text{aprx}}}$$

where $\epsilon_{\text{aprx}} \doteq \max_{G\in\mathcal{G}} \min_{G'\in\mathcal{G}} \mathbb{E}_P\left[(G'(s,a) - \mathcal{T}G(s,a))^2\right]$ is the inherent Bellman error and $E_{\max} \doteq \max_{(s,a)\ s.t.\ P(s,a)>0} E(s,a)$ is the maximal value our density model takes on regions that have $P(s,a) > 0$.

C.3. Proofs for Section 8.2

We will prove the guarantee under the 'with high probability' version of the density model error assumption from the main text:

Assumption C.4. There exist $\epsilon_p \geq 0, \delta_p \in [0,1]$ such that

$$\mathbb{P}_{(s,a)\sim_{\text{unif.}}\mathcal{S}\times\mathcal{A}}\left[|E(s,a) + \log P(s,a)| > \epsilon_p\right] \leq \delta_p$$

i.e. at most a fraction δ_p of the state-action space has error bigger than ϵ_p.

We have the following guarantee:

Proposition C.5. *For any probability mass function P, starting from any state s_0, taking action a_0 s.t. $\hat{G}(s_0, a_0) \leq -\log c$ guarantees that $\forall t$, there is a sequence $a_{1:t}$ for which with probability at least $1 - 2\delta_p$*

$$\log P(s_t, a_t) \geq \gamma^{-t} \log c - \frac{\gamma^{-t} R\epsilon_{\text{ls}} \exp \epsilon_p}{(1-\gamma)c} - \frac{\gamma^{K-t} \exp \epsilon_p \|E - \hat{G}^\star\|_\infty}{c} - \epsilon_p$$

if we use a discount $\gamma < 1$. Alternatively, if we use $\gamma = 1$, with probability at least $1 - 2\delta_p$,

$$\log P(s_t, a_t) \geq \log c - \frac{R \cdot K \cdot \epsilon_{\text{ls}}\ \exp \epsilon_p}{c} - \frac{\exp \epsilon_p \cdot \sup_{P^{(K)}} \|E - \hat{G}^\star\|_{P^{(K)}}}{c} - \epsilon_p$$

where $P^{(K)}$ is any probability distributions that can be reached by some sequence of actions (see Proof C.1 for more).

Proposition C.5 clearly encompasses the first clause of Proposition 8.5 by taking $\delta_p = 0$ and $K \to \infty$. The second clause from the main text also follows directly in the case that the scenario in Remark 8.4 occurs. As a minor side note, we can bound $\exp \epsilon_p \leq 1 + \epsilon_p + \epsilon_p^2$ for $\epsilon_p < 1.79$ in order to avoid having both logarithms and exponentials.

Proof of Proposition C.5. First note that we can always replace the LDM with $\hat{G}(s,a) = \max\{\hat{G}_K(s,a), E(s,a)\}$, which never increases the errors in 8.2/8.4/8.3 and can be seen as a refinement of avoidable fitting error.[5] This refinement ensures we have $\hat{G}(s_0,a_0) \leq -\log c$ implies that $E(s_0,a_0) \leq -\log c$. Due to Assumption C.4, with probability at least $1-\delta_p$, $E(s_0,a_0) \geq -\log P(s_0,a_0) - \epsilon_p$ and hence by rearranging $P(s_0,a_0) \geq ce^{-\epsilon_p}$. Since for now we assume P is a probability mass function, Proposition 8.2 implies that

$$\begin{aligned} |\hat{G}(s_0,a_0) - \hat{G}^\star(s_0,a_0)| &\leq \frac{R\epsilon_{\text{ls}} e^{\epsilon_p}}{(1-\gamma)c} + \frac{\gamma^K e^{\epsilon_p} \|E - \hat{G}^\star\|_\infty}{c} \\ \Rightarrow \quad \hat{G}^\star(s_0,a_0) &\leq \hat{G}(s_0,a_0) + \frac{R\epsilon_{\text{ls}} e^{\epsilon_p}}{(1-\gamma)c} + \frac{\gamma^K e^{\epsilon_p} \|E - \hat{G}^\star\|_\infty}{c} \end{aligned}$$

by the definition of $\hat{G}^\star$ this implies that for any t there is a sequence of actions $\{a_\tau\}_{\tau=1}^t$ for which starting from s_0, a_0 we can guarantee

$$\gamma^t E(s_t,a_t) \leq \hat{G}^\star(s_0,a_0) \leq -\log c + \frac{R\epsilon_{\text{ls}} e^{\epsilon_p}}{(1-\gamma)c} + \frac{\gamma^K e^{\epsilon_p} \|E - \hat{G}^\star\|_\infty}{c}$$

Since the probability that Assumption C.4 breaks twice is bounded by $2\delta_p$ due to union bound, we have that with probability at least $1-2\delta_p$,

$$\begin{aligned} \gamma^t(-\log P(s_t,a_t) - \epsilon_p) &\leq -\log c + \frac{R\epsilon_{\text{ls}} e^{\epsilon_p}}{(1-\gamma)c} + \frac{\gamma^K e^{\epsilon_p} \|E - \hat{G}^\star\|_\infty}{c} \\ \Rightarrow \quad \gamma^t(\log P(s_t,a_t) + \epsilon_p) &\geq \log c - \frac{R\epsilon_{\text{ls}} e^{\epsilon_p}}{(1-\gamma)c} - \frac{\gamma^K e^{\epsilon_p} \|E - \hat{G}^\star\|_\infty}{c} \\ \Rightarrow \quad \log P(s_t,a_t) &\geq \gamma^{-t} \log c - \frac{\gamma^{-t} R\epsilon_{\text{ls}} e^{\epsilon_p}}{(1-\gamma)c} - \frac{\gamma^{K-t} e^{\epsilon_p} \|E - \hat{G}^\star\|_\infty}{c} - \epsilon_p \end{aligned}$$

[5]This is a natural slight modification that we have deferred simply for presentation purposes.

The second statement follows analogously from the guarantee from Proof C.1 which implies

$$\hat{G}^\star(s_0,a_0) \le \hat{G}(s_0,a_0) + \frac{R\cdot K\cdot \epsilon_{\text{ls}}\, e^{\epsilon_p}}{c} + \frac{e^{\epsilon_p}\cdot \sup_{P(K)}\|E-\hat{G}^\star\|_{P(K)}}{c}$$

where $\hat{G}^\star$ is now undiscounted so we directly obtain

$$\log P(s_t,a_t) \ge \log c - \frac{R\cdot K\cdot \epsilon_{\text{ls}}\, e^{\epsilon_p}}{c} - \frac{e^{\epsilon_p}\cdot \sup_{P(K)}\|E-\hat{G}^\star\|_{P(K)}}{c} - \epsilon_p$$

concluding the full statement of the bound. $\square$

Now we plug the above in multiple times and do some manipulation to get the form presented in Corollary 8.6:

Proof of Corollary 8.6. Using our assumption on the error scaling we have that:

$$\begin{aligned}
|\hat{R}_T(\mathbf{s},\mathbf{a}) - R_T(\mathbf{s},\mathbf{a})| &= \sum_{t=0}^{T-1}\gamma^t\left(r(f(s_t,a_t),a_{t+1}) - r(\hat{f}(s_t,a_t),a_{t+1})\right)\\
&\le \sum_{t=0}^{T-1}\frac{\gamma^t\cdot\epsilon_r}{\sqrt{P(s_t,a_t)}}\\
&= \sum_{t=0}^{T-1}\frac{\gamma^t}{\sqrt{\epsilon_r^{-2}P(s_t,a_t)}}
\end{aligned}$$

For $\epsilon_r^{-2}P(s_t,a_t) \ge 0.08104$, we have that $\frac{1}{\sqrt{\epsilon_r^{-2}P(s_t,a_t)}} \le 1 - \log\left(\epsilon_r^{-2}P(s_t,a_t)\right) = 1 + 2\log\epsilon_r - \log P(s_t,a_t)$ which enables us to plug in the guarantee of Proposition 8.5 multiplied by γ^t iteratively. For now let's assume we picked c such that $P(s_t,a_t) \ge \epsilon_r^2\cdot 0.08104$ (note that for good enough model this is <u>very</u> small) for $t = \overline{0,T-1}$ (we will derive the setting required at the end). In that case, we have:

$$\begin{aligned}
|\hat{R}_T(\mathbf{s},\mathbf{a}) - R_T(\mathbf{s},\mathbf{a})| &= \sum_{t=0}^{T-1}\gamma^t\cdot(1+2\log\epsilon_r - \log P(s_t,a_t))\\
&\le \sum_{t=0}^{T-1}\left[\gamma^t(1+2\log\epsilon_r)) - \gamma^t\log P(s_t,a_t)\right]\\
&\le \sum_{t=0}^{T-1}\left[\gamma^t(1+2\log\epsilon_r) + \gamma^t\epsilon_P + \cdot\left(-\log c + \frac{R\epsilon_{\text{ls}}\exp\epsilon_P}{c(1-\gamma)}\right)\right]\\
&= (1+\epsilon_P+2\log\epsilon_r)\cdot\frac{1-\gamma^T}{1-\gamma} + T\cdot\left(\log\frac{1}{c} + \frac{R\epsilon_{\text{ls}}\exp\epsilon_P}{c(1-\gamma)}\right)
\end{aligned}$$

as desired. Finally we need to see what setting of c ensures we can apply the $\frac{1}{\sqrt{x}} \le 1 - \log x$ inequality throughout the trajectory. Using again our main guarantee it suffices to ensure

$$\log c - \frac{R\epsilon_{\text{ls}}\exp\epsilon_P}{c(1-\gamma)} \ge \gamma^{T-1}(\epsilon_P + 2\log\epsilon_r + \log 0.08104)$$

Hence since $\log x \ge 1 - \frac{1}{x}$, it suffices to ensure

$$\begin{aligned}
&1 - \frac{1}{c} - \frac{R\epsilon_{\text{ls}}\exp\epsilon_P}{c(1-\gamma)} \ge \gamma^{T-1}(\epsilon_P + 2\log\epsilon_r + \log 0.08104)\\
\iff &1 - \gamma^{T-1}(\epsilon_P + 2\log\epsilon_r + \log 0.08104) \ge \frac{(1-\gamma) + R\epsilon_{\text{ls}}\exp\epsilon_P}{c(1-\gamma)}\\
\iff &c \ge \frac{(1-\gamma) + R\epsilon_{\text{ls}}\exp\epsilon_P}{(1-\gamma^{T-1}(\epsilon_P + 2\log\epsilon_r + \log 0.08104))(1-\gamma)}
\end{aligned}$$

giving the tighter c lower bound. For presentation we can drop the $\log 0.08104$ (yielding a looser bound):

$$c \geq \frac{(1-\gamma) + R\epsilon_{\text{ls}} \exp \epsilon_P}{(1-\gamma^{T-1}(\epsilon_P + 2\log \epsilon_r))(1-\gamma)}$$

□

The proof of Corollary 8.7 is a similarly fast consequence of Proposition 7.1:

Proof of Corollary 8.7. It is easy to check that the construction $D_{H,\epsilon}$ from Proposition 7.1 satisfies $\sup_{P(H+2)} \|E - \hat{G}^\star\|_{P(H+2)} \leq 2\epsilon_p$ so by Proposition C.5, with probability at least $1 - 2\delta_p$,

$$\log P(s_t, a_t) \geq \log \hat{c} - \frac{(R \cdot (H+2) \cdot \epsilon_{\text{ls}} + 2\epsilon_p) \exp \epsilon_p}{\hat{c}}$$

for all t. Hence as long as $\log \frac{\hat{c}}{\epsilon} > \frac{(R\cdot(H+2)\cdot\epsilon_{\text{ls}} + 2\epsilon_p)\exp\epsilon_p}{\hat{c}}$ we are guaranteed to not exhibit the naive myopic behavior and instead ensure $P(s_t, a_t) \geq \frac{1}{2(H+1)}$ which is the best we could hope for. Since $\log x \geq 1 - x^{-1}$, it is enough to have $\hat{c}$ for which

$$\hat{c} \geq (R(H+2)\epsilon_{\text{ls}} + 2\epsilon_p) \exp \epsilon_p + \epsilon$$

as desired. □

We can also make the following remark about generalization to probability density functions.

Remark C.6 (Extension to Probability Density Functions)**.** *For P being a p.d.f. (probability density function), we can extend the above simple approach via Markov's inequality which guarantees that*

$$P(|\hat{G}^\star(s,a) - \hat{G}(s,a)| \geq a) \leq \frac{R\epsilon_{\text{ls}}}{(1-\gamma)a} + \frac{\gamma^K \|E - \hat{G}^\star\|_\infty}{a}$$

This implies that the $\mathbb{R}^{d_s \times d_a}$ hypervolume of the set of unruly state-action pairs $\mathcal{U}_{a,c} = \{(s,a) \in \mathcal{S} \times \mathcal{A}$ s.t. $|\hat{G}^\star(s,a) - \hat{G}(s,a)| \geq a$ and $P(s,a) \geq c\}$ cannot be larger than $\frac{R\epsilon_{\text{ls}}}{(1-\gamma)a} + \frac{\gamma^K\|E-\hat{G}^\star\|_\infty}{ca}$, i.e.

$$\textit{Vol}(\mathcal{U}_{a,c}) \leq \frac{R\epsilon_{\text{ls}}}{(1-\gamma)a} + \frac{\gamma^K \|E - \hat{G}^\star\|_\infty}{ca}$$

and hence bound the size of the region where the previous approach does not work (of course additionally one needs to choose a to get the best guarantee).

Finally, let us attempt to derive an analogue to Prop. 8.5 for the density threshold within the horizon H to better understand how the LDM compares to a standard density model. As discussed in the main text, we need to impose some regularity assumptions first.

Assumption C.7. Assume that E is L_p-Lipschitz, that the true dynamics f are L_f-Lipschitz in the state and that he model has error $|f(s,a) - \hat{f}(s,a)| \leq \frac{\epsilon_f}{P(s,a)}$.

Proposition C.8. *If Assumption C.7 holds, from any state s_0, taking a_0 according to (10) guarantees that $\forall t$, there is a sequence $a_{1:t}$ for which*

$$\log P(s_t, a_t) \geq \begin{cases} \log c - \frac{L_P \epsilon_f (1 + \ldots + L_f^{t-1}) \exp \epsilon_p}{c} - \epsilon_p \text{ if } t \leq H, \\ -\infty \text{ otherwise.} \end{cases}$$

Proof. Let $a'_{1:t}$ be the solution to the following problem to the $t \leq H$ horizon MPC with E and $\hat{f}$ and s_t be the sequence of states reached by playing $a'_{1:t}$ on the real model f. Under Assumption C.7, we have:

$$
\begin{aligned}
-\log P(s_t, a'_t) &\le E_\theta(s_t, a'_t) + \epsilon_p \\
&\le E_\theta(s'_t, a'_t) + L_p \|s'_t - s_t\| + \epsilon_p \\
&\le -\log c + \epsilon_p + \frac{L_p \cdot \epsilon_f \cdot (1 + \ldots + L_f^{t-1})}{c \cdot e^{-\epsilon_p}} \\
&\le -\log c + \epsilon_p + \frac{L_p \cdot \epsilon_f \cdot (1 + \ldots + L_f^{t-1}) \cdot \exp \epsilon_p}{c}
\end{aligned}
$$

where we used line follows by: where the last part follows because

$$
\begin{aligned}
||s_t - s'_t|| &= ||f(s_{t-1}, a'_{t-1}) - \hat{f}(s'_{t-1}, a'_{t-1})|| \\
&\le ||f(s_{t-1}, a'_{t-1}) - f(s'_{t-1}, a'_{t-1})|| + ||f(s'_{t-1}, a'_{t-1}) - \hat{f}(s'_{t-1}, a'_{t-1})|| \\
&\le L_f ||s_{t-1} - s'_{t-1}|| + \frac{\epsilon_f}{ce^{-\epsilon_p}} \\
&\le L_f^2 ||s_{t-2} - s'_{t-2}|| + \frac{\epsilon_f}{ce^{-\epsilon_p}}(1 + L_f) \\
&\le \ldots \\
&\le 0 + \frac{\epsilon_f (1 + \ldots + L_f^{t-1})}{ce^{-\epsilon_p}}
\end{aligned}
$$

□

Proposition C.8 shows a similar behavior to the LDM guarantee in the short-horizon. As suggested by Antos et al. (2007), the constants in analyses on fitted Q iteration are interwined with Lipschitzness of the system and of the probability function. In our case this relation springs forth more naturally since we are considering a dynamical system rather than an MDP.

D. Appendix to Section 9

D.1. Data and Tasks

Our experiments consist of a goal-directed hopper task, a goal-directed lunar lander task, and SimGlucose (Xie, 2018) , a blood glucose setpoint task. Here we present details about the task setup and training data for each domain.

D.1.1. Hopper

The goal-directed hopper task involves controlling an agent with 12 state dimensions and 3 action dimensions. The agent is randomly initialized in a x position in $[-3, 3]$, and the goal of the task is to control the agent to go to a target x location in $\{-2, -1, 0, 1, 2\}$. In our experiments, we tested each method 10 times (each from a randomly initialized starting position) for each goal location. The specific reward function we used for planning is $r(s, a) = -|s_x - s_g|$, where s_x is the x position of the agent, and s_g is the goal position.

The dataset we used to train our agents consists of stochastic expert behavior, where the experts were trained to go towards a random goal location within $[-4, 4]$ of the agent's initial position. The expert behaviors included in the dataset include hopping forwards, backwards, and stopping. In order to increase the coverage of the dataset, we added random noise to the expert's actions at each timestep. More specifically, for each trajectory, we randomly sample $Y \sim \text{Unif}(-1.5, 1.5)$, and at each timestep we add $Z \sim \text{Normal}(0, |Y|)$ to the expert's chosen actions. The size of the dataset is 8499511 transitions.

D.1.2. Lunar Lander

The goal-directed lunar lander task involves controlling an agent with 8 state dimensions and 2 action dimensions. The agent is always initialized in the middle-top of the screen, and the goal of the task is to control the aircraft to land on a target landing pad without crashing. The location of the landing pad is centered at either 4 or 6 on the horizontal axis. In our experiments, we tested each method 10 times for each landing pad location. The specific reward we used for planning is $r(s, a) = -\sqrt{(s_x - s_g)^2 + s_y^2}$, where s_x is the horizontal position of the agent, and s_g is the landing pad position, and s_y is the height of the agent.

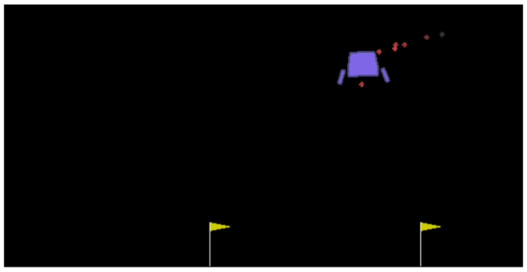

Figure 7. Visualization of the lunar lander task.

The dataset we used to train our agent consists of stochastic expert behavior, where the expert was trained to land in a random location between 1 and 10. Thus, the expert behavior in the dataset include going towards and landing in different landing pad locations. In order to increase the coverage of the dataset, we added random noise sampled from $Z \sim \text{Normal}(0, 0.9)$ to the expert's actions at each timestep. The size of the dataset is 1650022 transitions. For a visualization of the task, see Fig 7.

D.1.3. SimGlucose

The simglucose task dataset consists of a patient having their blood glucose level controller by a reference PID controller (P=0.001, I=0.00001, D=0.001) set with target 140. The state dimension is 2 (representing current blood glucose and current caloric input) and the action dimension is 1 (representing insulin administered). In our experiments, we tested each method 100 times. In order to be consistent with the PID approach as well as with our other experiment, we use the reward $r(\text{bg}) = -|\text{bg} - 140|$ during planning. This reward serves to enforce staying close to the midpoint of the safety region $(100, 180)$, matching the reference PID's target setpoint, as well as allowing for a formulation that ties setpoint maintanence and goal reaching tasks together. Note that the controller must be able to neither under nor over react to being suddenly in a more dangerous range. This is a particularly well-suited task for the LDM approach since the goal is to leverage past data and learn to filter what actions to take to ensure not getting into a hypo/hyperglycemia alert, while also trying to get to the target blood glucose level as fast as possible. The size of the dataset is 1mil transitions.

D.2. Flow Model Training

For all density models, we made use of a neural spline flow (Durkan et al., 2019). The training hyperparameters are presented below.

task	flow layers	knots K	bounds B	coupling transform	MLP	learning rate	weight decay	optimizer	train steps
Hopper	4	64	20	rational-quadratic	256-256-256	1e-4	1e-5	Adam	150k
Lunar Lander	4	64	20	rational-quadratic	256-256-256	1e-4	1e-5	Adam	150k
Glucose	2	64	20	rational-quadratic	256-256-256	1e-4	1e-5	Adam	500k

We used the output of the flow models to label our datasets for LDM training. In order to make training more robust, we manually labeled the density of early termination points to have low density for the Hopper and Lunar Lander experiments. More specifically, we labeled them to be $\min(\{E_\theta(s_i, a_i)\}_{i=1}^N) - 3 * \text{std}(\{E_\theta(s_i, a_i)\}_{i=1}^N)$, where $\{E_\theta(s_i, a_i)\}_{i=1}^N$ denotes the dataset's labels from the flow model.

D.3. LDM Training

In this section, we present the hyperparameters we used in our experiments to train an LDM, presented in Alg. 1.

First, present the training parameters for the LDM function $G_\phi(s, a)$ and its associated policy $\pi_\psi(s)$. $G_\phi(s, a)$ outputs a scalar value, and $\pi_\psi(s)$ outputs a distribution over actions, parameterized by the mean and variance of a gaussian.

$G_\phi(s,a)$ training parameters				
task	batch size	architecture	learning rate	optimizer
Hopper	256	256-256	3e-4	Adam
Lunar Lander	256	256-256	3e-4	Adam
Glucose	256	256-256	3e-4	Adam

$\pi_\psi(s)$ training parameters				
task	batch size	architecture	learning rate	optimizer
Hopper	256	256-256	1e-4	Adam
Lunar Lander	256	256-256	1e-4	Adam
Glucose	256	256-256	1e-4	Adam

Next, we discuss the details of the conservative regularization term CQL($\mathcal{H}$) (Kumar et al., 2020) we use to enable more stable offline training. Our implementation of the regularizer involves importance sampling from $\pi_\psi(s_t)$, $\pi_\psi(s_{t+1})$, and a uniform distribution over the action space, more specifically:

$$\text{CQL}(\mathcal{H}) \tag{13}$$

$$= -\beta \log \sum_{a \in \mathcal{A}} \exp G(s,a) \tag{14}$$

$$\approx -\beta \left(\frac{1}{3N_{CQL}} \sum_{a_i \sim \text{Unif}(a)}^{N_{CQL}} \left[\frac{\exp G(s,a_i)}{\text{Unif}(a)} \right] + \frac{1}{3N_{CQL}} \sum_{a_i \sim \pi_\psi(.|s_t)}^{N_{CQL}} \left[\frac{\exp G(s,a_i)}{\pi_\psi(.|s_t)} \right] \right. \tag{15}$$

$$\left. + \frac{1}{3N_{CQL}} \sum_{a_i \sim \pi_\psi(.|s_{t+1})}^{N_{CQL}} \left[\frac{\exp G(s,a_i)}{\pi_\psi(.|s_{t+1})} \right] \right) \tag{16}$$

Note that this is negative of the regularizer presented in the original work on CQL, because the authors there were concerned with regulating Q functions in the reinforcement learning setting, where higher values correspond to higher rewards. In our setting, however, lower values in the LDM correspond to higher densities (lower negative log probabilities), so this regularizer is inverted in our setting. The specific hyperparameters we use for this regularizer are:

task	β	N_{CQL}
Hopper	10	10
Lunar Lander	1	10
Glucose	1	10

Finally, we present the algorithm hyperparameters we used:

task	γ	τ	target entropy	α learning rate	α optimizer	train steps
Hopper	1	5e-3	20	1e-4	Adam	3000000
Lunar Lander	1	5e-3	2	1e-4	Adam	500000
Glucose	0.9	5e-3	0	1e-4	Adam	200000

D.4. Dynamics Model Training

In this section, we present the hyperparameters we used to train the dynamics model:

task	architecture	batch size	learning rate	weight decay	optimizer	train steps
Hopper	256-256	256	3e-4	1e-5	Adam	50000
Lunar Lander	256-256	256	3e-4	1e-5	Adam	50000
Glucose	256-256	256	3e-4	1e-5	Adam	200000

Additionally, in order to effectively learn a dynamics model for hopper, the highest dimensional system of the 4 domains, we preprocessed the training data (subtracting mean and dividing by standard deviation), and trained the neural network to output the change in state rather than the next state itself (predicting $s_{t+1} - s_t$ where s_t is the current state). For all other tasks, we directly trained on the (unnormalized) training data and predicted the next state, because this procedure was sufficient for learning a good dynamics model within the data distribution.

D.5. Planning

In the model-based reinforcement learning algorithm presented in Section 7, we use the learned dynamics model to perform planning. In this section, we present details about the planning procedure used in our experiments.

The action trajectory that maximizes the optimization problem in (8) can be approximated with stochastic zeroth-order optimization. More specifically, we generate random action trajectories uniformly sampled from a sampling prior, and select the trajectory which maximizes the objective while satisfying a constraint that is dependent on the method. More details about the sampling prior, optimization objective and constraint are described below. If no randomly generated action trajectory satisfies the constraint, we execute the action from a learned policy which was trained to optimize the constraint function.

D.5.1. MPC parameters

In this section, we present the parameters we used to perform planning via MPC.

task	sampling prior	objective	num random actions	horizon
Hopper	normal distribution fitted around action marginal of the data	$-\lvert s_x - s_g\rvert$	1024	1
Lunar Lander	uniform distribution over action space	$-\sqrt{(s_x - s_g)^2 + s_y^2}$	8000	1
Glucose	uniform distribution over action space	$-\lvert \text{bg} - \text{bg}^\star\rvert$	1024	1

D.5.2. Constraints used in MPC

In addition to our method, with performs MPC with an LDM constraint, we evaluate our experiment on 2 baseline methods, one with a density model constraint, and one with a ensemble constraint. For the ensemble constraint, we train n_{ensemble} independent dynamics models $\{f_\xi(s,a)_i\}_{i=0}^{n_{\text{ensemble}}}$ by minimizing the loss (7), and use the negative variance of the outputs of the ensemble models as the constraint function. In our experiments, we use $n_{\text{ensemble}} = 5$. We summarize the exact equations we use to constrain MPC for each type of constraint in the table below.

constraint type	constraint function
LDM	$Q_\phi(s,a)$
density model	$E_\theta(s,a)$
ensemble	$-\text{variance}(\{f_\xi(s,a)_i\}_{i=0}^{n_{\text{ensemble}}})$

D.5.3. Backup policy

When performing constrained optimization with zeroth order optimization, it is possible for none of the sampled actions to pass the constraint. In this case, we execute the action outputted by a learned policy which is optimized to maximize the constraint used for each method.

For the LDM constraint, the backup policy is $\pi_\psi(s_t)$, the policy associated with the LDM, as described in Sec. 6.

For the backup policies for the density model and ensembles constraint, we train the backup policy in a similar fashion, by optimizing to output the action which will maximize its value under the constraint. More specifically, the loss function is $L_\pi(\psi) = \mathbb{E}_{s_t \sim p_D}[W_\phi(s_t, \pi_\psi(s_t))] - \alpha\mathcal{H}(\pi_\psi(.|s_t))$, where $W(s,a)$ represents the constraint function being used. The Algorithm for training and the hyperparameters we used for training are presented below. The main difference between this procedure and Alg. 1 is that the constraint functions here are held *fixed*, whereas in Alg. 1, the LDM function and policy were updated concurrently.

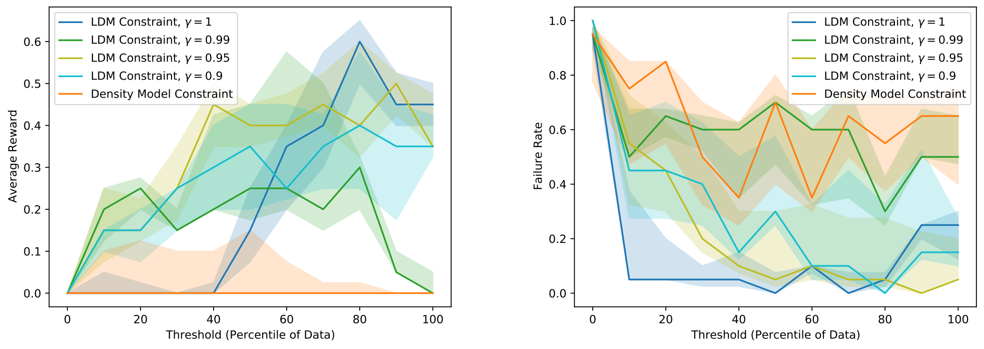

Figure 8. Effect of discount factor on LDM performance for the lunar lander task. On the left, we show the average reward for each method, and on the right, we show the failure rate for each method. Each method was evaluated with 3 random seeds.

Algorithm 2 Backup Policy Training

```
initialize parameter vectors $\psi$, and $\alpha$
for num backup policy training steps do
    $\psi \leftarrow \psi - \lambda_\pi \nabla_\psi L_\pi(\psi)$
    $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha L(\alpha)$
end
```

task	batch size	architecture	ψ learning rate	target entropy	α learning rate	optimizer	train steps
Hopper	256	256-256	3e-4	3	3e-4	Adam	500000
Lunar Lander	256	256-256	3e-4	2	3e-4	Adam	500000
Glucose	256	256-256	3e-4	1	3e-4	Adam	250000

D.6. Ablation Experiments on the Effect of Discount Factor γ

We performed ablation experiments to investigate the effect of using a discount factor on LDM training for the lunar lander task, described in Section D.1.2. We tried discount factors of 1, 0.99, 0.95, and 0.9, and compare against using a density model constraint. In Fig. 8, we show the average reward (left), and failure rate (right) of each method.

D.7. Full Experimental Results Across Threshold Sweep

Due to space constraints, we were unable to include the full sweep of threshold values vs. average reward and failure rate for all the domains in the main paper. In Fig. 9, we include the full sweeps for the lunar lander, SimGlucose, and ventilator tasks for a more comprehensive presentation of the performance of the different methods.

E. Guarantees under Partial Observability

The LDM as formulated above addresses fully observed tasks with Markovian state spaces. In many cases, the state observations may not be perfectly Markovian, and the system may be partially observed. We can extend the LDM to non-Markovian state observations, and retain appealing guarantees. Let $\mathcal{O}$ be the observation space and $h : \mathcal{S} \to \mathcal{O}$ the function that maps a state to an observation. For the remainder of the partially observed setting, we will slightly overload the notation P to refer to the density on $\mathcal{O} \times \mathcal{A}$. We analyze the behavior of the same algorithm as before when deployed under partial observability.

Intuitively, in the partially observed case, the LDMs can be seen as operating on a dynamical system that captures the *expected* transitions in observation space. Formally, we define $f^{\text{obs}} : \mathcal{O} \times \mathcal{A} \to \mathcal{O}$ as $f^{\text{obs}}(o, a) = \mathbb{E}_{s|o}[h(f(s, a))]$, which we can see as a *reduced* system capturing the expected trajectory progression, where the expectation is over all possible underlying states. More concretely, we are taking the entire distribution over possible states (which will only get narrower if we include history) and use the expected value under that distribution to define an associated dynamical system. Note that

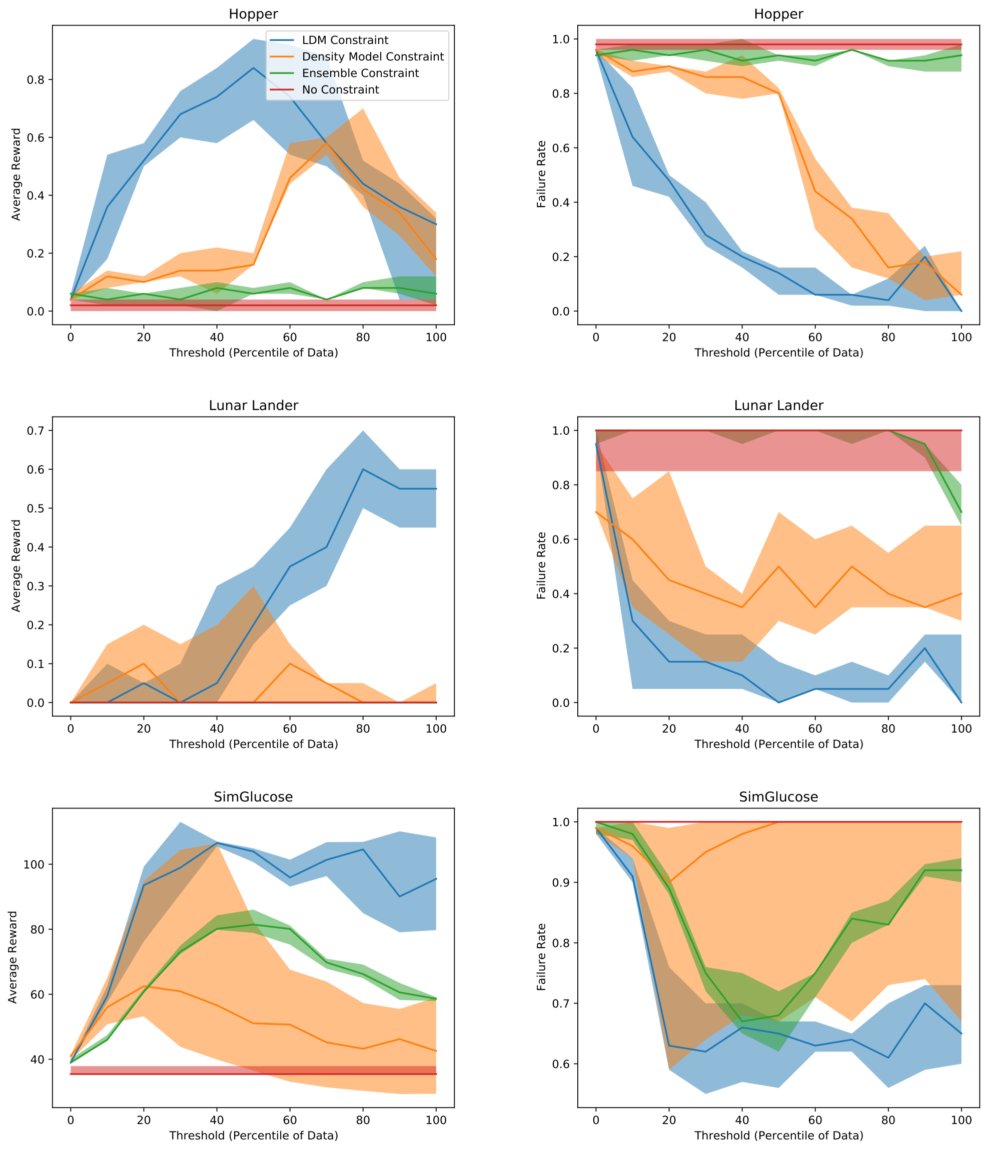

Figure 9. Evaluation of average reward and failure rate over different threshold values for the hopper, lunar lander, and SimGlucose tasks. The x axis of the plots on the left represents the percent of the dataset whose value under the constraint function falls below the associated threshold c. More specifically, $x = \text{percentile}(\{W(s_t^i, a_t^i)\}_{i=1}^N, c)$, where W is the constraint function. We use this representation of constraint values in order to make them comparable across different constraint functions. The solid line represents the median performance over random seeds, and the top and bottom of the semi-transparent region indicates the 25/75th percentiles. We ran 5 random seeds for each tasks.

the maximal LDM for the reduced system constructed in this manner can be obtained with essentially the same algorithm as before. In a sense, we are able to maintain the exact same approach, and focus only what this existing algorithm guarantees under this setting. Formally, we will update the LDM using the empirical estimate of the full information backup operator $\mathcal{T}G(o,a) = \max\{E(o,a), \min_{a' \in \mathcal{A}} \gamma G(f^{\text{obs}}(o,a), a')\}$. The only difference is that, by sampling (o, a, o'), we may have $o' \neq f^{\text{obs}}(o,a)$. As a result we cannot directly fit the above and instead go over each transition in the dataset and fit the following partial information back-up estimate $\widehat{\mathbb{E}\mathcal{T}}$, the *sampled* counterpart to Eq. 3 (averaging over possible next observations):

$$\widehat{\mathbb{E}\mathcal{T}}G(o,a) = \frac{\sum_{\mathcal{O}_D(o,a)} \max\{E(o,a), \min_{a' \in \mathcal{A}} \gamma G(o', a')\}}{|\mathcal{O}_D(o,a)|}$$

where $\mathcal{O}_D(o,a) = \{o' \in \mathcal{O} \text{ s.t. } (o,a,o') \in D\}$. Plugging this into update (4) yields:

$$\hat{G}_{k+1} \in \arg\min_{G \in \mathcal{G}} \sum_{(o,a) \in D} \Big(G(o,a) - \widehat{\mathbb{E}\mathcal{T}} G_k(o,a) \Big)^2$$

Given that the LDM has a natural extension to this setting, the main insight of this section is that the same procedure as before still provides a distributional shift guarantee, but with an additional error term that accounts for the variability in the observation for the same underlying state. This is captured in the following proposition:

Proposition E.1. *Let $\hat{G}_0 = E$. The result of applying the sample bellman update K times satisfies:*

$$\|\hat{G}_K - \hat{G}^\star\|_P \leq \frac{R \cdot \epsilon_{\text{ls,obs}}}{1-\gamma} + \gamma^K \cdot \|E - \hat{G}^\star\|_\infty,$$

for $\epsilon_{\text{ls,obs}} \doteq \max_{t \in [K-1]} \|\hat{G}_{t+1} - \widehat{\mathbb{E}\mathcal{T}}\hat{G}_t\|_P + \|\mathcal{T}\hat{G}_t - \widehat{\mathbb{E}\mathcal{T}}\hat{G}_t\|_P.$

Observe that the difference from the guarantee in Prop. 8.2 is the additional $\|\mathcal{T}\hat{G}_t - \widehat{\mathbb{E}\mathcal{T}}\hat{G}_t\|_P$ term which captures the errors due to fitting $\widehat{\mathbb{E}\mathcal{T}}$ rather than $\mathcal{T}$, since the latter cannot be computed due to lack of access to f^{obs}. As a result, we are able to formalize and motivate the behavior of our algorithm on partially observed systems, showing that: (i) the LDM is useful for bounding the probability of staying in-distribution under the expected transitions, and (ii) the convergence rate depends on how well the sampled Bellman operator estimate matches the true Bellman operator under the f^{obs}. Note that if there is no information loss, then the bounds match the previous derivations exactly.

Further, if we assume there is some some constant $M > 0$ such that the magintude of the second derivative of $\mathcal{T} \cdot G$ for any $G \in \mathcal{G}$ is bounded by M, we can guarantee that $\epsilon_{\text{ls,obs}} = \mathcal{O}\left(\sqrt{\frac{M^2 \sigma_{\text{obs}}}{|D|}}\right)$, where $\sigma_{\text{obs}} = \sup_{o,a} \text{Var}_{s|o,a}[h(f(s,a))]$ captures the maximum variance of observation transitions. Note that $\sigma_{\text{obs}} \to 0$ is equivalent to the observation capturing more and more of the true state, and hence results in the partially observed guarantee converging to its fully observed counterpart as we would expect.