Exploration Behavior of Untrained Policies

Jacob Adamczyk JACOB.ADAMCZYK001@UMB.EDU
University of Massachusetts Boston, United States

Abstract

Exploration remains a fundamental challenge in reinforcement learning (RL), particularly in environments with sparse or adversarial reward structures. In this work, we study how the architecture of deep neural policies implicitly shapes exploration before training. We theoretically and empirically demonstrate strategies for generating ballistic or diffusive trajectories from untrained policies in a toy model. Using the theory of infinite-width networks and a continuous-time limit, we show that untrained policies return correlated actions and result in non-trivial state-visitation distributions. We discuss the distributions of the corresponding trajectories for a standard architecture, revealing insights into inductive biases for tackling exploration. Our results establish a theoretical and experimental framework for using policy initialization as a design tool to understand exploration behavior in early training.

1. Introduction

Effective exploration is crucial for reinforcement learning agents operating in high-dimensional or sparse-reward environments. While much focus has been placed on designing explicit exploration bonuses or strategies, we shift attention to a more implicit source of exploration: the parameterization of the policy network. Importantly, studying the effect of policy architecture does not require training of additional networks [3], or introduction of exploration-dependent rewards [8, 17].Untrained policies are responsible for determining the initial data distribution from which deep RL agents begin training. As such, the properties of deep policies at initialization is important for understanding exploration in the early training regime. We hypothesize that the choice of architecture and the corresponding initialization distribution implicitly determine the entropy and geometry of exploration. As a step toward understanding this phenomenon, we provide some initial theoretical and experimental results to support this hypothesis. We leverage infinite-width and continuous-time limits to understand random policy behavior in a toy model with corresponding simulations.

Contributions: In this work, we show that (1) fixed policies lead to ballistic trajectories where correlations and smoothness in the network dominate (2) random policy initializations at each step can be used to generate diffusive trajectories with a heavy-tailed steady-state distribution and (3) combining these methods can provide new pathways for controlled exploration in deep RL.

Related Work: Exploration has been studied from many angles [6]. Our work leverages the infinite-width limit [5, 11] and analysis of the Fokker-Planck equation which have both proven to be fruitful avenues of research for various ML communities. Neural architecture search (NAS) might attempt to optimize the policy architecture directly [21], e.g. for better exploration strategies. Rather than searching in the space of possible architectures, we instead approach the problem through the lens of closed-form solutions, attempting to understand the inductive biases of a fixed architecture and its impact on exploration behavior.

© J. Adamczyk.

2. Background

2.1. Reinforcement Learning

Focusing on the effects of untrained policies, we only need the basic and usual definitions: a state space, $\mathcal{S}$, action space $\mathcal{A}$, deterministic transition dynamics (a function mapping state and action to successor-state) $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. Let $\pi_\theta : \mathcal{S} \to \mathcal{A}$ be a policy represented by a feedforward neural network with parameters $\theta \in \mathbb{R}^n$ randomly initialized according to some specified scheme (e.g. Xavier-Glorot). We will not introduce any (intrinsic or exploration-dependent) reward functions and instead focus on the reward-free Markov processes that govern the agent's data collection process.

Operating in the policy-based RL setting, a network π_θ is trained to maximize returns (e.g. REINFORCE [20], TRPO [15], PPO [16]). Value-based algorithms such as Q-learning and its offspring [4, 10, 19] typically use ϵ-greedy approaches, and the additional non-linear mapping from value to policy space can complicate the analysis, so this connection is left to future work.

2.2. Smooth Networks Yield Ballistic Trajectories

For short times, sufficiently smooth neural networks can induce trajectories with a dominating drift term: "ballistic motion". Loosely speaking, the structure of a neural net induces similar outputs (actions) for nearby states, and when compounded with smooth dynamics, this results in agents with very smooth (non-diffuse) trajectories. This result is visualized in Figures 1 and 2 and is formalized more precisely below:

Assumption 1 (Transition Dynamics Locality) *The environment dynamics are "local": there exists a $\delta > 0$ such that for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$, the distance between s and any corresponding successor state $s' = f(s, a)$ is upper-bounded $|s' - s| < \delta$.*

Lemma 1 (Short-Time Ballistic Behavior of Lipschitz Neural Policies) *Let $\pi_\theta : \mathbb{R}^d \to \mathbb{R}^d$ be a stochastic policy sampled from a neural network prior, such that the realized sample function is L_π-Lipschitz. Let the deterministic transition dynamics $s_{t+1} = f(s_t, a_t)$ satisfy Assumption 1, and also be Lipschitz continuous with respect to the action space: $|f(s, a) - f(s, a')| \leq L_f|a - a'|$ and state space, $|f(s, a) - f(s', a)| \leq L_s|s - s'|$ for all $s, s' \in \mathcal{S}$ and $a, a' \in \mathcal{A}$. Let the constant $c = \pi_\theta(s_0)$ denote the initial action and let $L_s = 1$ (the $L_s \neq 1$ regime is dominated by exponential time dependence, and is further discussed in the Appendix). Then for any initial state s_0 the agent's trajectory approximately follows the straight-line path ct, up to a bounded error:*

$$|(s_t - s_0) - ct| \leq \frac{1}{2}\delta L_a L_\pi t^2.$$

Remark 2 *Although the previous lemma holds for all times, a linear approximation holds only for short timescales, $t \ll \frac{2c}{\delta L_a L_\pi}$, beyond which, deviations caused by curvature (resulting from non-linearities in the deep neural net) become large enough to dominate the effective mean velocity, c.*

The Lipschitz constant of a neural policy can be computed [1, 7, 18] or roughly bounded *a priori* to make Lemma 1 implementable. Although the assumptions and structure of the above dynamics may be limiting in some environments, they provide a starting point to develop intuition for the relationship between architecture, dynamics, and "depth"[1] of trajectories. We will focus on generalizing these results in future work.

1. In the sense of "deep exploration" [14].

If a fixed policy network is used at the beginning of training (e.g. to fill up a replay buffer), the agent will not observe diverse trajectories when periodically reset to the initial state, (a common setup in episodic RL). As an extreme alternative, we can instead consider initializing a *new* policy (from a fixed distribution, say) at each timestep and study its effects on the agent's trajectories.

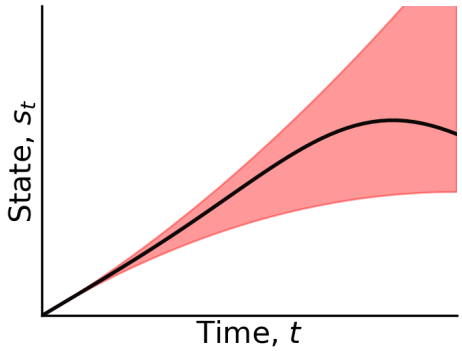

Figure 1: Though trajectories can change direction (as evidence in the plot on the right), on short timescales, the trajectories can be well-approximated with linear drift.

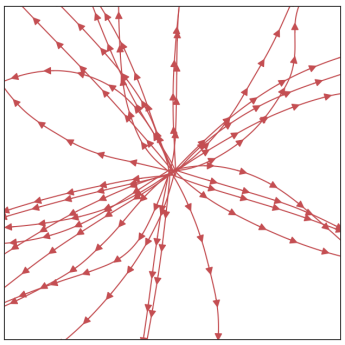

Figure 2: Standard MLPs used for deep RL (with two hidden layers, 256 hidden nodes, ReLU activation) produce ballistic trajectories.

2.3. Random Policy Networks as Gaussian Processes

Rather than a single policy generating an entire trajectory (as is typically the case), we seek to obtain a more diffusive (and hence potentially more exploratory) trajectory distribution. To this end, the control policy π_θ is now re-sampled sequentially, producing a stochastic trajectory ensemble (even with deterministic dynamics and deterministic action outputs from the policies):

Definition 3 (Trajectory Under a Random Policy Ensemble) *Let $\mathcal{T}_\theta = \{s_0, s_1, \ldots, s_T\}$ denote a trajectory generated by:*

$$s_{t+1} = f\left(s_t, \pi_{\theta_t}(s_t)\right), \quad \theta_t \sim \mathcal{P}_\theta$$

where each θ_t is independently drawn from the initialization distribution $\mathcal{P}_\theta$, and π_{θ_t} is the corresponding deterministic policy network.

With this setup, we can leverage the following closed-form expression for the policy in the infinite-width limit (cf. also Eq. 1 of [9] and surrounding discussion for more details):

Theorem 4 (Neal [11], Infinite-Width Limit as Gaussian Process) *Let π_θ be a feedforward neural network with nonlinear activation $\phi(0) = 0$ (e.g., ReLU or Tanh), and i.i.d. weights and biases, with zero mean. In the limit as the width of all hidden layers tends to infinity, the distribution over functions π_θ converges to a Gaussian Process:*

$$\pi_\theta \xrightarrow{d} \mathcal{GP}(0, K(s, s')),$$

where K is an architecture-dependent kernel.

The GP then produces a structured action covariance when a new policy is sampled at each timestep. Formally, let states $s, s' \in \mathcal{S} \doteq \mathbb{R}^d$ be given. The covariance between actions (w.r.t. draws from the initial parameter distribution) at the two states is given by: $\mathrm{Cov}\left[\pi_\theta(s), \pi_\theta(s')\right] = K(s, s') \in \mathbb{R}^{d\times d}$. Thus, the kernel corresponding to the chosen architecture (and hence the parameter initialization) gives rise to action correlations across the state-space. Below, we show an example of this structure in a policy with one (infinitely wide) hidden layer and ReLU activation, for its simplicity and popularity in the literature.

Note that these properties are complementary to that of the fixed policy, whose architecture is typically Lipschitz, implying the ballistic trajectories discussed above.

2.4. Case Study: ReLU Network and the Induced State Distribution

For ReLU networks with Gaussian weight initialization, the infinite-width kernel is given by:

$$K(s, s') = \frac{\sigma_w^2}{\pi}|s||s'|\left(\sin\theta + (\pi - \theta)\cos\theta\right) + \sigma_b^2, \tag{1}$$

where $\cos\theta = \frac{\langle s, s'\rangle}{|s||s'|}$. This kernel is not stationary (i.e. it does not only depend on a radial distance between inputs), and induces a diffusion term that grows with $|s|^2$. Thus, far from the origin, exploration becomes more diffuse. Stationary kernels (e.g. Radial Basis Functions [2]) depending on $|s - s'|$ on the other hand would result in constant diffusion coefficient, which may be of interest in some environments.

In the special case of the one hidden layer, infinitely wide ReLU, we take weights and biases drawn from centered Gaussians with variance σ_w, σ_b, respectively. In this case, the policy is mean zero and has diffusion coefficient $K(s, s) = \Sigma(s) = \sigma_b^2 + \frac{\sigma_w^2}{\pi}\|s\|^2$. For the simplest linear dynamics $s_{t+1} = s_t + \pi(s_t)$, we can model the stochastic policy as a random walk in state space. In the continuous time (equivalently "small action") limit, the Fokker-Planck equation can be employed to study the dynamical distribution $p(s, t)$ over state space:

$$\frac{\partial p}{\partial t} = -\mu_0^\top \nabla p + \frac{1}{2}\nabla^2 : (\Sigma(s)p), \tag{2}$$

which in the long-time limit (assuming stationarity) can be written as:

$$\nabla^2\left[\left(\sigma_b^2 + \frac{\sigma_w^2}{\pi}\|s\|^2\right)p_\infty(s)\right] = 0.$$

With radial symmetry $p_\infty(s) = f(r)$, $r = \|s\|$, we obtain the solution:

$$f(r) \propto \left(\sigma_b^2 + \frac{\sigma_w^2 \cdot r^2}{\pi}\right)^{-d/2} \sim \frac{1}{r^d}. \tag{3}$$

This describes a heavy-tailed stationary distribution (resembling Cauchy or power-law distributions) and is normalizable if $\sigma_b > 0$. Note that as $\sigma_b^2 \to 0$, the tails become increasingly heavy, and exploration is more likely to reach distant regions of state space.

2.5. Experiments

Unsurprisingly, an agent's exploration behavior depends on the nature of its policy parameterization. To demonstrate our theoretical results, we show that ballistic or diffusive motion can be controlled via policy "resets". As illustrated in Figure 3, placing a barrier in state space with a narrow hallway drastically reduces the likelihood of uniformly exploring state space (i.e., passing through the hallway). Ballistic trajectories, generated by a fixed MLP (red), often do not pass through the hallway: they follow straight paths, and small errors result in failed exploration. In contrast, the trajectories from policy networks that are re-initialized at every step (green) exhibit diffusive, fat-tailed distributions that will eventually (but slowly) explore across the barrier. A hybrid switching strategy, where a fixed MLP is used for the first n steps before switching to per-step re-initialization, captures both the initial linear drift and later diffusion, enabling more effective exploration in this scenario. Within the linear regime, choosing a value of $n \ll L_\pi^{-1}$ (cf. Lemma 1) ensures that trajectories will escape the initial state, while $n > 2\Delta/\delta$ can ensure that agents reach the states of interest (e.g. those a distance Δ from the origin) before diffusing. Studying this tradeoff with a more quantitative definition of "exploration" in more complex environments is the focus of future work. The policy reset can also be done stochastically with increasing probability, based on agent observations, which may be especially relevant in the deep RL setting, perhaps providing additional insight to the [13].

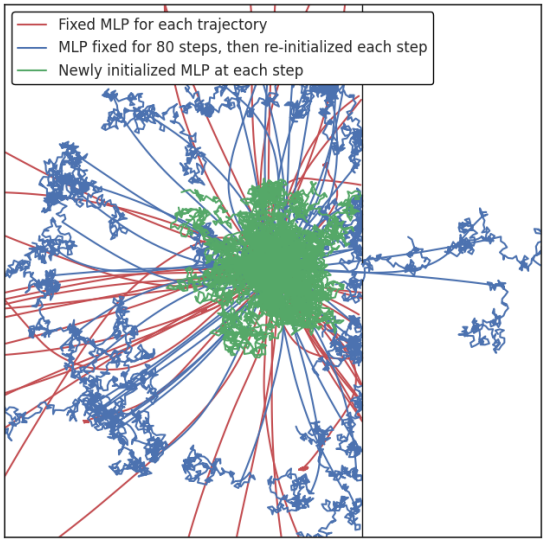

Figure 3: **Exploration through a narrow hallway.** Ballistic trajectories from a fixed MLP struggle to pass through the barrier; stepwise re-initialization produces overly-diffusive motion, while a hybrid strategy leverages both behaviors for efficient exploration.

3. Discussion

In this work, we explored how the architecture and initialization of policy networks can shape the exploration dynamics of reinforcement learning agents. By modeling untrained policies as draws from Gaussian Processes (GPs) in the infinite-width limit, we demonstrated that distinct neural architectures and parameterizations implicitly induce nontrivial priors over agent behavior. For simple dynamics models, these behaviors are analytically tractable, giving insight into the exploration distribution. These priors manifest themselves as specific patterns in trajectory geometry, ranging from ballistic drift to heavy-tailed diffusive exploration.

Our analysis showed that Lipschitz continuity in deterministic policy networks leads to smooth, directional (ballistic) trajectories which may limit early exploration. In contrast, policies sampled at each timestep from an architectural prior induce highly stochastic (but structured) trajectory distributions. We studied these trajectories in the continuous-time limit, employing a Fokker–Planck equation, revealing a connection between a network's kernel and steady-state distribution. We found that ReLU-based policies generate quasi-Cauchy state distributions in free space, encouraging broad

exploration. Notably, the magnitude and shape of the induced kernel dictate the correlation between actions at different states, determining whether trajectories are locally consistent or rapidly de-correlating, when using the per-step policy sampling scheme.

These findings suggest a new lens for understanding exploration. Rather than a process to be tuned through learning or incentivized via external bonuses, one can envisage exploration as an architectural and initialization design problem. By tailoring policy architectures to match the topology or reward sparsity of a target environment, one can influence early-stage exploration "zero-shot". Future work will study the relationship to resetting parameters during training [12], which may have the additional benefit of improving exploration ability. This discussion aligns with a broader incentive to align network architectures with environments, an area in deep reinforcement learning that has not yet been explored.

References

[1] Aritra Bhowmick, Meenakshi D'Souza, and G Srinivasa Raghavan. Lipbab: Computing exact lipschitz constant of relu networks. In *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part IV 30*, pages 151–162. Springer, 2021.

[2] Martin Dietrich Buhmann. Radial basis functions. *Acta numerica*, 9:1–38, 2000.

[3] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[4] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[5] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[6] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.

[7] Fabian Latorre, Paul Rolland, and Volkan Cevher. Lipschitz constant estimation of neural networks via sparse polynomial optimization. *arXiv preprint arXiv:2004.08688*, 2020.

[8] Sam Lobel, Akhil Bagaria, and George Konidaris. Flipping coins to estimate pseudocounts for exploration in reinforcement learning. In *International Conference on Machine Learning*, pages 22594–22613. PMLR, 2023.

[9] Lassi Meronen, Martin Trapp, and Arno Solin. Periodic activation functions induce stationarity. *Advances in Neural Information Processing Systems*, 34:1673–1685, 2021.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[11] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[12] Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2022.

[13] Evgenii Nikishin, Junhyuk Oh, Georg Ostrovski, Clare Lyle, Razvan Pascanu, Will Dabney, and Andre Barreto. Deep reinforcement learning with plasticity injection. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 37142–37159. Curran Associates, Inc., 2023.

[14] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[17] Adrien Ali Taiga, William Fedus, Marlos C Machado, Aaron Courville, and Marc G Bellemare. On bonus-based exploration methods in the arcade learning environment. *arXiv preprint arXiv:2109.11052*, 2021.

[18] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31, 2018.

[19] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[20] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[21] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

Appendix

We first provide a useful inequality for obtaining our main result, Lemma 1, first focusing on the case of $L_s = 1$:

Lemma 5 *The sequence of inequalities*

$$\epsilon_{t+1} \leq \epsilon_t + kt \tag{4}$$

has iterates that are upper bounded by $\epsilon_t \leq kt^2/2 + \epsilon_0$.

Proof The proof is straightforward by induction. For the base case, note that $\epsilon_0 \leq k(0)^2/2 + \epsilon_0$. From the inductive hypothesis,

$$\begin{aligned}
\epsilon_{t+1} &\leq \epsilon_t + kt \\
&\leq \frac{kt^2}{2} + \epsilon_0 + kt \\
&= \frac{k(t^2 + 2t)}{2} + \epsilon_0 \\
&= \frac{k(t+1)^2 - k}{2} + \epsilon_0 \\
&= \frac{k(t+1)^2}{2} + \epsilon_0 - \frac{k}{2} \\
&\leq \frac{k(t+1)^2}{2} + \epsilon_0
\end{aligned}$$

which is the form of iterate $(t+1)$, thus completing the proof. ■

When $L_s \neq 1$, the recursive form is instead controlled by an exponential behavior, L_s^t:

Lemma 6 *The sequence of inequalities*

$$\epsilon_{t+1} \leq A\epsilon_t + kt \tag{5}$$

has iterates that are upper bounded by $\epsilon_t \leq kt\frac{A^t - 1}{A-1} + \epsilon_0$.

Proof The proof is similar to that of Lemma 5, except that a factor of A is accumulated at each step in a geometric series. ■

We now prove Lemma 1:

Proof First, denote $\widetilde{s}_t = ct + s_0$, the linear approximation of the trajectory. We examine the iterates of the error, $\epsilon_t \doteq |s_t - \widetilde{s}_t|$ to verify the bound presented in the lemma.

$$\begin{aligned}
\epsilon_{t+1} &= |s_{t+1} - \widetilde{s}_{t+1}| && (6)\\
&= |f(s_t, \pi_\theta(s_t)) - f(\widetilde{s}_t, c)| && (7)\\
&= |f(s_t, \pi_\theta(s_t)) - f(s_t, c) + f(s_t, c) - f(\widetilde{s}_t, c)| && (8)\\
&\leq |f(s_t, \pi_\theta(s_t)) - f(s_t, c)| + |f(s_t, c) - f(\widetilde{s}_t, c)| && (9)\\
&\leq L_a|\pi_\theta(s_t) - \pi_\theta(s_0)| + L_s|s_t - \widetilde{s}_t| && (10)\\
&\leq L_a L_\pi |s_t - s_0| + L_s\epsilon_t && (11)\\
&\leq L_a L_\pi \delta t + L_s\epsilon_t && (12)
\end{aligned}$$

Several notes are in order: In the last three lines we (a) made the substitution of $c = \pi_\theta(s_0)$ as an estimate of the mean velocity, (b) used the Lipschitz property of the neural network π_θ, and (c) used the locality of dynamics iteratively to bound the distance in state-space. For (c), we have iterated the bound in Assumption 1 for multiple timesteps, making use of the triangle inequality.

Note that for (a), this choice leads naturally to the desired result and is easy to compute (once the first action is drawn, the estimate of the drift is complete). However, it may not lead to the tightest bounds in practice. Empirically, we have found that using a mean of actions along the trajectory can smooth out the estimate for velocity, somewhat improving the bound. Similar steps can be taken to arrive at a linear upper bound, using that e.g. $|\pi_\theta(s_t) - \langle \pi_\theta(s_t) \rangle| < g\varepsilon$. where g is a geometric factor and ε is the diameter of a ball over which the average $\langle \pi_\theta(s_t) \rangle$ is computed. In any case, the resulting bound is linear.

Using Lemma 5, we see that $\epsilon_t \leq \delta L_a L_\pi t^2/2$, neglecting the initial error term $\epsilon_0 = 0$ (since at $t = 0$ we have the correct state $\widetilde{s}_0 = s_0$ and there is no approximation). Ensuring the error does not accumulate past the linear prediction ($\delta L_a L_\pi t^2/2 < ct$) results in the upper bound on timesteps presented in the main text.

When considering $L_s \neq 1$, Lemma 6 applies and there are two distinct cases: (1) If $L_s < 1$ the dynamics are contractive, and trajectories remain roughly linear ($\propto t(1 - e^{-t/\tau})$) beyond a timescale $\tau = \ln L_s$. Intuitively, this should be expected since a Lipschitz constant less than unity is subsumed by the weaker case of $L_s = 1$. (2) If $L_s > 1$ the dynamics are explosive and trajectories exponentially separate ($\propto t(e^{t/\tau} - 1)$). Although a large global Lipschitz constant $L_s > 1$ may be required from a theoretical standpoint, in practical settings, much of state-space may be governed by relatively small local Lipschitz constants $L_s \lesssim 1$, making (sub)trajectories more well-behaved than naïvely expected by this worst-case analysis. ■