Investigating Compounding Prediction Errors in Learned Dynamics Models

Nathan Lambert NOL@BERKELEY.EDU
Kristofer Pister KSJP@BERKELEY.EDU
University of California, Berkeley

Roberto Calandra RCALANDRA@FB.COM
Facebook AI Research

Abstract

Accurately predicting the consequences of agents' actions is a key prerequisite for planning in robotic control. Model-based reinforcement learning (MBRL) is one paradigm which relies on the iterative learning and prediction of state-action transitions to solve a task. Deep MBRL has become a popular candidate, using a neural network to learn a dynamics model that predicts with each pass from high-dimensional states to actions. These "one-step" predictions are known to become inaccurate over longer horizons of composed prediction – called the compounding error problem. Given the prevalence of the compounding error problem in MBRL and related fields of data-driven control, we set out to understand the properties of and conditions causing these long-horizon errors. In this paper, we explore the effects of subcomponents of a control problem on long-term prediction error: including choosing a system, collecting data, and training a model. These detailed quantitative studies on simulated and real-world data show that the underlying dynamics of a system are the strongest factor determining the shape and magnitude of prediction error. Given a clearer understanding of compounding prediction error, researchers can implement new types of models beyond "one-step" that are more useful for control.

Keywords: Dynamics Modeling, Robotics, Model-based Reinforcement Learning

1. Introduction

Learning accurate dynamics models is crucial to many real-world robotics applications. Central to the origins of dynamics model learning is the field of optimal control, where a system can be controlled when given access to an analytical dynamics model (Kirk, 2004). To apply these successful control and planning techniques to systems without known dynamics, system identification emerged as a set of techniques designed to compute a representative set of system parameters (e.g. mass of a robotic component) key to expressing the motion and use them with optimal control (Ljung, 1999). With the growth of data-driven systems, learning dynamics models for control has shifted to be increasingly task-centric, online, and free of prior knowledge of a system. The field of model-based reinforcement learning (MBRL) showcases this process and has been used to solve many robotic tasks by iteratively learning a black-box model (Deisenroth and Rasmussen, 2011; Chua et al., 2018; Nagabandi et al., 2019; Janner et al., 2019). Central to recent successes in model-based reinforcement learning are one-step dynamics models; where they have been used for online model predictive control (MPC) (Chua et al., 2018; Nagabandi et al., 2019; Lambert et al., 2019) or value-estimation and offline rollouts of imagined policies (Janner et al., 2019). These models are known to be subject to an issue of *compounding prediction error* (Clavera et al., 2018; Wang et al., 2019), where long-

horizon prediction often diverge over time to the point of being unusable. In this work, we look to understand and study the causes of these compounding errors in order to improve the performance of future model-predictive agents.

Regardless of the prediction errors, the simple one-step parametrization with deep neural networks has been successful over a variety of real-world and simulated applications. In order to predict far into the future, the models use composed function passes, where even small errors can grow rapidly to make the predictions difficult to rely on for ranking proposed actions. Specifically, this paper is centered around the questions of: Why is there compounding error? What are the numerical characteristics of the compounding error? What model design choices most heavily impact compounding error? In order to address this, the experiments are structured to address three major factors in deploying a learned model for control: the underlying system dynamics, system noise, and system dimensionality; the distribution of collected data; and how the model is trained. The paper contains a wide analysis on deep one-step predictive models on simulated and real-world systems showing the strongest correlation between unstable underlying system dynamics and high-error predictions. With this depth, a set of takeaways are included for the reader to be better aware of why and how compounding error could impact their systems. With this information, we believe that practices in MBRL and other data-driven predictive agents can be improved to better leverage dynamics models for control.

2. Related Works

Model-based Reinforcement Learning Much recent work in model-based reinforcement learning (MBRL) uses one-step models, without explicitly addressing compounding error, to iteratively learn the agents environment and then leverage it for control (Deisenroth and Rasmussen, 2011; Williams et al., 2017; Chua et al., 2018; Lambert et al., 2019; Nagabandi et al., 2019). Given the recent successes, the training of the forward one-step models is not well studied and a variety of numerical behaviors have been recorded. For example, the Model-based Policy Optimization (MBPO) algorithm uses short horizons to avoid the compounding error problem (Janner et al., 2019), but there-in loses out on a lot of the potential for having a learned model and anticipating the future. With another MBRL algorithm, probabilistic ensembles with trajectory sampling (PETS) (Chua et al., 2018), dynamically tuning the predictive horizon and model training dramatically improves the downstream performance (Zhang et al., 2021). The centrality of predictive horizon to performance indicates that the compounding error over time is important to future of MBRL, but its causes are not well understood. Another horizontal study of model learning in MBRL proposes and details the metrics of many different model variants used in MBRL (Kégl et al., 2021), but does not provide insight in how to limit the effects of compounding error.

Learning One-step Dynamics Models Single-step learned dynamics models are effective across many domains despite their well known compounding errors. Early examples include using one-step models for studying robot dynamics (Punjani and Abbeel, 2015) or learning a model for linear controllers (Bansal et al., 2016). Additionally, other model types such as single-step Gaussian Processes (GPs) (Deisenroth and Rasmussen, 2011) and linear models (Fu et al., 2016; Bansal et al., 2017) have been applied to multiple lower-dimensional robotic learning tasks. GPs are exciting due to their more structured handling of uncertainty, which can allow for safety in long-horizon predictions (Koller et al., 2018), but they have not scaled as well as deep learning in high-dimension and large-data tasks. There has been substantial development in deep learning and other data-driven

methods to facilitate more useful methods for learning models for control. Structured mechanical models (Gupta et al., 2020) and Lagrangian neural networks (Cranmer et al., 2020) both use a constrained learning setup to restrict their one-step dynamics models to those satisfying smooth constraints and de-valuing transitions likely to be governed by random noise. These methods have been shown to be data-efficient for smooth systems, but their effect on compounded prediction accuracy is not yet studied.

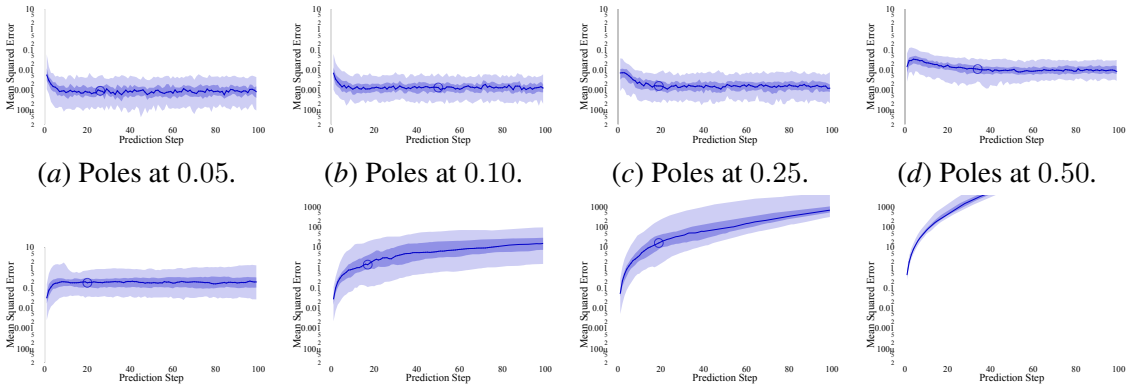
Mitigating Compounding Error Compounding error is referenced in many recent MBRL papers, including state-of-the-art algorithms (Clavera et al., 2018; Chua et al., 2018) and benchmarking efforts (Wang et al., 2019). Recently, more methods have proposed potential solutions. (Heess et al., 2015) avoids compounding error by only predicting with the model with real observations as inputs to avoid distribution drift via long predictions. Most methods avoid compounding error by tweaking the dynamics model optimization, including imitation-learning inspired predictive models (Venkatraman et al., 2015), multi-step value estimators (Asadi et al., 2019), and flexible prediction horizons (Xiao et al., 2019). Most work using model predictive control uses predictive horizons of 20-30 steps with limited cross validation of model accuracy versus prediction horizon or agent performance, as suggested in (Lambert et al., 2020a). The trajectory-based model proposes a new training paradigm to avoid the compounding error by embedding time dependence in prediction, but it is currently limited by its requirement of closed form controllers (Lambert et al., 2020b).

3. Background

Markov Decision Process We now describe the formulation we use to evaluate compounding error. At time t, the environment is represented by the state $s_t \in \mathbb{R}^{d_s}$, the action $a_t \in \mathbb{R}^{d_a}$, and the reward function $r(s_t, a_t) : \mathbb{R}^{d_s \times d_a} \mapsto \mathbb{R}$ following the Markov Decision Process (MDP) (Bellman, 1957). With this data, the task is learn a dynamics model $f_\theta(s_t, a_t)$ to represent the transition function $f : \mathbb{R}^{d_s \times d_a} \mapsto \mathbb{R}^{d_s}$. The data $\mathcal{D} = \left\{(s_i, a_i, s_{i+1})\right\}_{i=1}^{N}$ used to train the model is often subject to the behavior of an agents' control law $\pi(\cdot)$, called the policy.

One-step Dynamics Models Given the state s_t, and the action a_t, one-step dynamics models predicts the next state s_{t+1} of an MDP. A given model object can use different prediction formulations. We use the delta-state formulation that is popular for regularizing the prediction distribution, $s_{t+1} = s_t + f_\theta(s_t, a_t)$, and compare to the true-state variant, $s_{t+1} = f_\theta(s_t, a_t)$. The true-state models are denoted with a -S, such as the true-state probabilistic ensemble *PE-S*. These formulations can be used with multiple loss functions, including Mean Squared Error (MSE) for deterministic models and Negative Log Likelihood (NLL) for probabilistic models. All model types can be used with an ensemble that weights predictions across multiple trained models, denoted with *E*.

In this work we primarily compare the delta-state and true-state prediction formulations with simple deterministic models (*D*) and with rich probabilistic ensembles (*PE*). For the probabilistic models, we propagate the trajectories with expectation based propagation, more options for the *PE* models are detailed in Chua et al. (2018). We compare one-step models to *linear models* (LIN) based on least-squares learning of a linear predictor and *zero models* (ZERO) that return a predicted state of the zero vector at each step. Additional model training details are included in Sec. A.2.



(*a*) Poles at 0.05. (*b*) Poles at 0.10. (*c*) Poles at 0.25. (*d*) Poles at 0.50.

(*e*) Poles at 0.75. (*f*) Poles at 0.90. (*g*) Poles at 0.95. (*h*) Poles at 1.00.

Figure 1: Showing the compounding errors, formally the per-step MSE (median, 65^{th}, and 95^{th} percentiles), of state-space systems shown in Eq. (3) with different poles, ρ. Compounding errors vary substantially with the underlying poles of the environment and diverge when the poles approach instability. All models are trained and evaluated on separate datasets of 100 trajectories.

4. Methodology

4.1. Problem Formulation: Multi-step Compounding Error

For the prediction of the long-term future, the one-step dynamic model is recursively applied as

$$\hat{s}_{t+h} = f_\theta(\dots f_\theta(f_\theta(s_t, a_t), a_{t+1}) \dots, a_{t+h}) \,. \tag{1}$$

Any parametrization of the dynamics model f carries a prediction error $\epsilon_t = \hat{s}_t - s_t$. It is often observed that this error grows multiplicatively by the next prediction's input being subject to all past errors in the prediction, as

$$\hat{s}_{t+h} = f_\theta(\dots f_\theta(f_\theta(s_t, a_t) + \epsilon_t, a_{t+1}) + \epsilon_{t+1} \dots, a_{t+h}) + \epsilon_{t+h} \,. \tag{2}$$

The central metric we will use to quantify and visualize compounding error is the mean-squared prediction error (MSE) at each step. The action sequence used with a given trajectory, $\{a_0, a_1, \dots, a_h\}$ is provided when planning the trajectory and measuring its performance. To that end, we train a dynamics model f_θ, and use it to predict to a horizon h, steps into the future, generating a predicted trajectory $\hat{s}_i \forall i \in [1, h]$. Then, the predicted error is computed by summing across all of the state dimensions at the given time-step as $\text{MSE}_t = \sum_{d=0}^{d_s} \|\hat{s}_{t,d} - s_{t,d}\|_2^2$.

For each experiment and environment, the MSE is normalized per-state to $[0, 1]$ to make the calculated MSE represent average predictive accuracy proportional to the relative state error rather than the numerical error (e.g. to normalize across states of different state types like positions and velocities). We hope this makes the errors shown more intuitive – an mean squared error of 1 represents an error across each state averaging to 100%.

4.2. Experimental Setting

Here we provide a concise summary of the studied systems, with more details available in Sec. A.1.

State-space System To test the possible causes of compounding error, we test the ability of deep one-step models to predict variations of a clearly defined system. Consider a state-space system defined with a state $s \in \mathbb{R}^3$ and an action $a \in \mathbb{R}$ as $s_{t+1} = \boldsymbol{A}s_t + \boldsymbol{B}a_t + \omega_t$. Therein, we define $\boldsymbol{A}$ and $\boldsymbol{B}$ as follows to control the poles of the system:

$$\boldsymbol{A} = \begin{bmatrix} \rho & a_1 & a_2 \\ 0 & \rho & a_3 \\ 0 & 0 & \rho \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}. \tag{3}$$

We set the desired eigenvalues, or poles, of the system to be ρ. The other parameters of the system are sampled randomly for each trial as $a_i, b_i \sim \mathcal{U}(-1, 1)$, which act as a source of uncertainty. The default process-noise in the environment $\omega_i \sim \mathcal{U}(-0.01, 0.01)$. All actions are chosen randomly from $\mathcal{U}(-1, 1)$ and act via the randomly generated $\boldsymbol{B}$ matrices.

Other Environments We present simulated results from three other simulated environments: the Cartpole, Reacher, and Quadrotor tasks (for more details, see (Lambert et al., 2020b)). In the Cartpole ($d_s = 4$, $d_a = 1$) and Quadrotor ($d_s = 9$, $d_a = 4$) tasks the agent attempts to balance the agent around an unstable fixed point. In the Reacher manipulation task ($d_s = 15$, $d_a = 5$), the agent must control the end-effector to a randomly generated point in the state-space.

5. Experimentally Studying Compounding Error

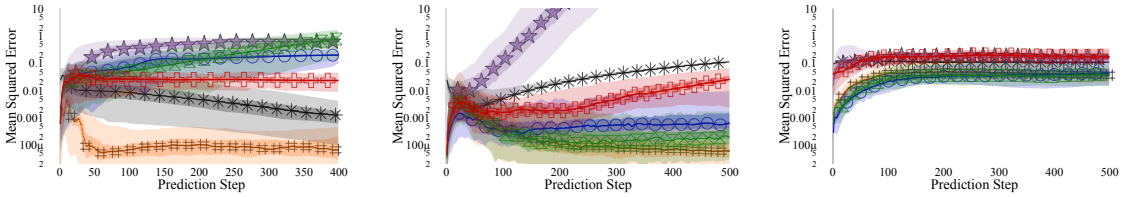
In this section we progress through different components of learning a model for control and detail their effect on compounding prediction error. We start with the central consideration – the underlying dynamics of the system, then we investigate two key important characteristics of a system: noise levels and state-space dimensionality. Next, we explore an important question for MBRL: the effect of dataset density on prediction accuracy. Finally, we conclude by showing that many model training effects such as the model parametrization and normalization play a lesser role in compounding errors. Across these experiments we enact large sweeps across key properties to illustrate macro trends, which comes at the loss of specificity on a per-environment basis – specific applications can benefit from finer analysis of these variables. Code for reproducing the experiments is available [1] and additional experiments are included in Sec. B.

5.1. System: Underlying Dynamics

The underlying dynamics of a system to be modelled controls the relative complexity of the prediction landscape. One way to characterize the behavior of a system's behavior is through the poles, often computed as the eigenvalues for linear systems. To continue the state-space example, a discrete-time system is *stable* when the eigenvalues are within the unit-circle, $||\rho|| < 1$. We vary the eigenvalues of our state-space system shown in Eq. (3) across a range of mostly stable and psuedo-stable values to compare the compounding error.

The results in Fig. 1 indicate that as eigenvalues approach instability, prediction accuracy quickly degrades. For stable systems, the error growth over time does not compound, but decreases over a short horizon before reaching steady state. Those systems with truly unstable poles, $||\rho|| > 1$, diverge so rapidly that plotting and computing the magnitude of error is computationally intractable. It

1. Code: https://github.com/natolambert/continuousprediction/tree/compound



(*a*) Cartpole. $d_s + d_a = 5$. (*b*) Quadrotor. $d_s + d_a = 13$. (*c*) Reacher. $d_s + d_a = 20$.

LIN (#) ZERO (∗) D (●) D-S (★) PE (⧖) PE-S (+)

Figure 2: Comparing the MSE of prediction error per-step (median, 65^{th}, and 95^{th} percentiles) on common model types and parametrizations on simulated robotic tasks of different dynamics, simulators, and dimension. There is a trend of error of predictions increasing with the task difficulty, but there is high variability in the performance of any one model type when comparing across platforms. All model types are trained and evaluated on the same datasets, maintaining separate datasets of 100 trajectories for test-train split.

is also interesting that when the poles are notably stable, the relative *stable-ness* of the system does not have a large baring on prediction accuracy, as shown by the similarity in error between poles of $\rho < 0.25$. Examples of the compounding error on different simulated robotic tasks is shown in Fig. 2, which show substantial variation in compounding errors. With complex robotic systems it is often difficult to directly identify the eigenvalues, further increasing the difficulty of cross-platform comparisons. The Cartpole and Quadrotor environments represent stabilization tasks, which have similar error profiles when compared to the Reacher maniuplation task. The variation between platforms in both the magnitude and shape of compounding error motivates a deeper study of the causes, which could be revealed in other properties of the system such as the dimension or noise.

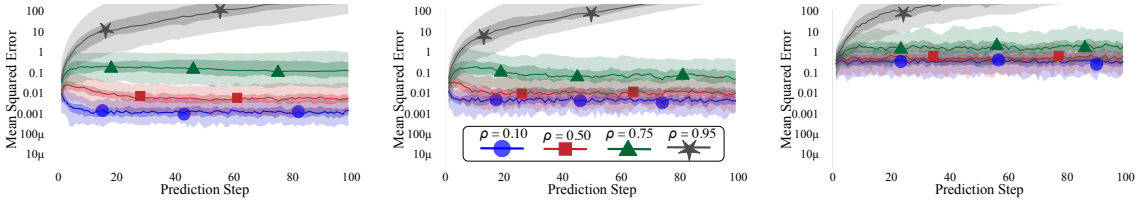
Remark 1 *The underlying dynamics of a system have a heavy impact on when the prediction accuracy will be poor. Unstable systems should deploy one-step models with extra care.*

Remark 2 *The shape of long-term prediction error for simple, stable-systems is not one of multiplicative growth. As the horizon increases the error rapidly reaches a steady-state value.*

5.2. System: Process Noise

The underlying noise within the dynamics has a substantial impact on measurement and evolution of any dynamical system. The default state-space system has only process noise, sampled uniformly $\omega_i \sim \mathcal{U}(-0.01, 0.01)$. To measure the effects of this noise, we measure the prediction accuracy with the following multiples of the original noise: $0\times, 10\times, 100\times$. The results, shown in Fig. 3 indicate that noise can control the maximum accuracy. This floor is a primary contributor to model inaccuracy for stable poles $\rho = \{0.1, 0.5, 0.75\}$, but when the poles approach instability $\rho = 0.95$, the compounding error is similar across all noise levels.

An interesting observation of the learning process is the relation between the random actions, which the networked is informed of, and that of the process noise. By default, the input matrices $\boldsymbol{B}$ are randomized along with the control policy in each trajectory, so they computationally impossible for the general dynamics model to learn. In Fig. 4, the actions are zeroed so they no longer contribute



(*a*) 0× noise. (*b*) 10× noise. (*c*) 100× noise.

Figure 3: Comparing prediction accuracy when increasing the levels of process noise in the system above and below the default of $\omega_t \sim \mathcal{U}(-0.01, 0.01)$ on all dimensions (median, 65^{th}, and 95^{th} percentiles). The error between a system with default (shown in Fig. 1) and zero process noise shows that the default noise from the random action matrices determines the resulting prediction accuracy. An interesting feature is that when increasing the process noise from 10× to 15×, the modelling accuracy degrades by a factor of 15.

as a disturbance on system dynamics, and the prediction accuracy shape is similar, but improves performance by about 100× when compared to Fig. 1.

For the robotic tasks shown in Fig. 2, the default noise varies dramatically. The Reacher has an unreported and low noise level (hidden within the Mujoco simulator), the Cartpole has uniform noise $\mathcal{U}(-0.1, 0.1)$ on all states, and the Quadrotor has a noise sampled from $\mathcal{N}(0, 0.0001)$. None of these simulators vary the level of noise relative to the type of the state variables (for example, a position in meters can take much lower magnitudes than an angle in degrees). The learned models on the quadrotor system converged to substantially lower levels, potentially indicating that deep models continue to improve with substantial noise reduction.

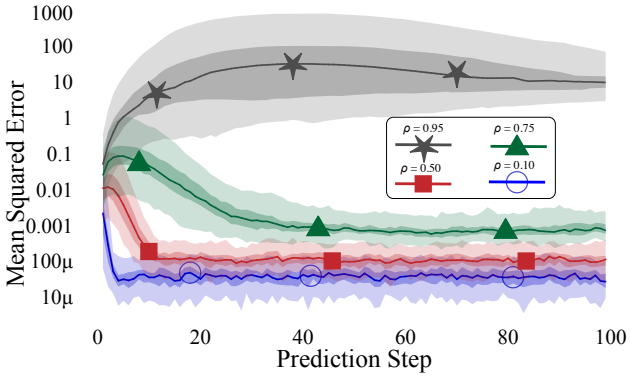
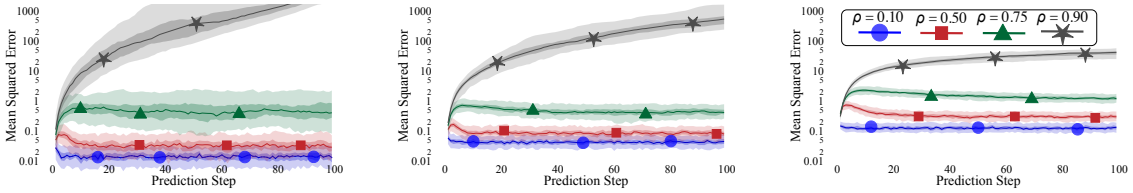


Figure 4: Showing how the randomly sampled input matrices, $\boldsymbol{B}$, and actions affect the per-step MSE with different eigenvalues (median, 65^{th}, and 95^{th} percentiles) by collecting new data and evaluate newly trained models with $\boldsymbol{B} = 0$. The random actions are the second leading cause of prediction error behind the unstable eigenvalues.

Remark 3 *Process noise impacts the peak prediction accuracy of a one-step dynamics model, but does not cause composed predictions to diverge more rapidly.*

5.3. System: State-space Dimension



(a) $d_s = 9$. (b) $d_s = 27$. (c) $d_s = 81$.

Figure 5: Comparing compounding error with different state dimensions to see if input-output prediction size challenges the models when the underlying dynamics are regularized (shown is the MSE with median, 65^{th}, and 95^{th} percentiles). State size does not have a substantial effect on the modeling error (the decreased variance of the error at each step in the state-sizes could be due to averaging over more state dimensions).

A large motivation to using deep models for predicting dynamics is the ability to extend to higher dimensional tasks. While early work has shown that deep networks are useful for high dimensional tasks (such as (Nagabandi et al., 2019; Lambert et al., 2019) with state-action spaces over 40 dimensional), given the difficulty of comparing across different systems more controlled studies of prediction dimensionality are needed. Learning one-step models scale the input and output dimensions with respect to the environment state. To evaluate this, we scale the dimensionality of the state-space system from 3 to 9, 27, and 81, which is shown in Fig. 6. The effect of increasing the state without normalizing the underlying dynamics, $\boldsymbol{A}$, is a rapid increase in the compounding error because the matrix norm grows with state-dimension.

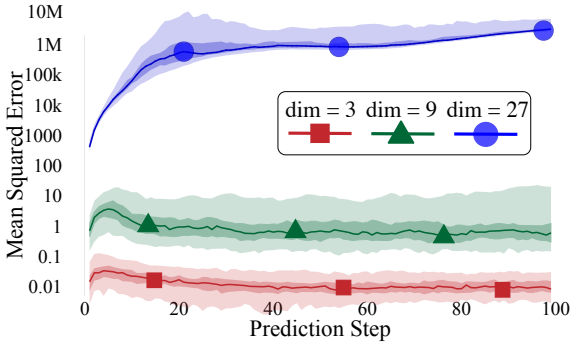
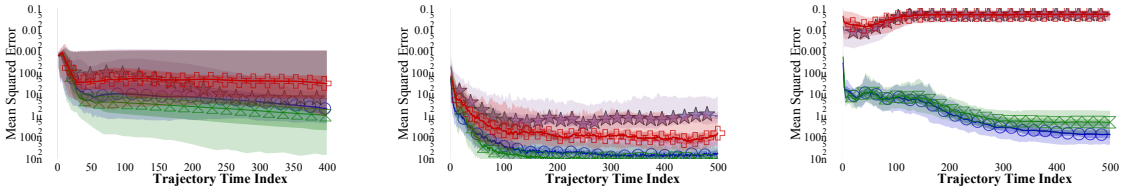


Figure 6: The prediction accuracy versus *unregularized* state-dimension growth given a set pole at $\rho = 0.5$ for all systems predictions. Specifically, in this figure the matrix norm, $||\boldsymbol{A}||_\infty$, of the state-dynamics grows with dimension.

The underlying trajectories act more unstable when increasing the dimension of the state-space system because each state is a weighted sum of the current states. Without normalization the summation representing a linear transition continues to grow with state dimension. As a second experiment, the maximum row norm of the state-dynamics matrix, $\boldsymbol{A}_\infty$, is bounded to isolate the effects of prediction-dimension from the strong effects of system stability. As the dimension increases in this subsection, the dynamics are regularized so that $||\boldsymbol{A}||_\infty \leq 3$, as in the default system. With such normalization, the standard model types suffer from an increase in baseline prediction error, but the rate of error compounding does not grow, shown in Fig. 5. Increasing the dimension of the state also reduces the variance of the compounding MSE, which is likely due to averaging over more states rather than a change in prediction dynamics.

Remark 4 *Increasing the dimension of the system state while maintaining a similar underlying dynamics does not contribute a notable increase to compounding prediction error.*



(*a*) Cartpole. (*b*) Crazyflie. (*c*) Reacher.

D (●) D-S (★) PE (⧖) PE-S (+)

Figure 8: The per-step model prediction error, rather than accumulated prediction error, (median, 65^{th}, and 95^{th} percentiles) across trajectories in simulated robotic experiments. This highlights the relative error of a true state-action pair at a given time index in a trajectory. In these examples, the per-step error decreases as the controllers stabilize the robots towards the stationary target points.

5.4. System: Data Distribution & Density

Understanding how model accuracy relates to an underlying training distribution is crucial to advancements in deep learning. Scaling the state-dimension of a system effectively reduces the density of data. Another axis for comparing the density of training data for a learned model is to compare model accuracy along trajectories understanding the relative density with respect to time. For stable systems, the data will likely be more dense at higher time indices (as is the case for the Cartpole and the Crazyflie), but for other systems the data-distribution over time in trajectories can take complex forms. As a proxy to density, we observe the per-step prediction error when predicting from the true state and action to the next state along each trajectory from the initial state s_0 and at each intermediate state s_t (rather than the composed predictions as in most of this work). The results of the across-time one-step errors are shown in Fig. 8 for the simulated robots and Fig. 7 for the state-space system.

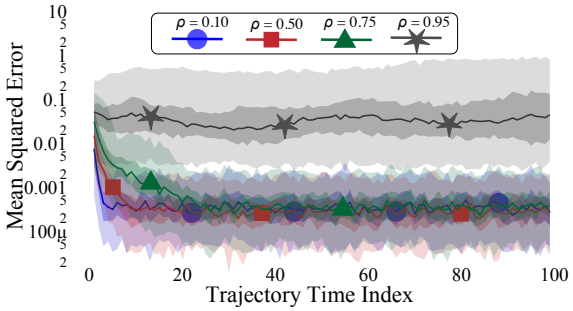
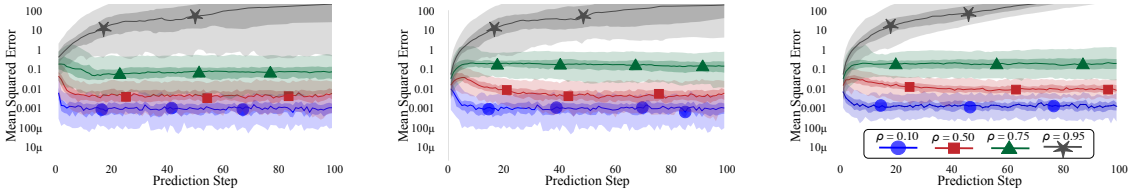


Figure 7: The per time index error for varying state-space systems. Again, unexpectedly, the error does not worsen further into the trajectories where state-space coverage is less dense.

Remark 5 *Dynamics model accuracy mirrors the distribution of training data, which is not necessarily correlated with the task of interest.*

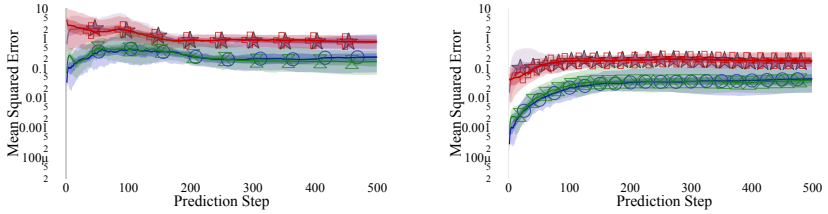
5.5. Model: Prediction Formulation & Training

Different model parametrizations, particularly ensembles and probabilistic loss functions, have been shown to improve the peak performance of MBRL algorithms (Chua et al., 2018; Janner et al., 2019). It is important to identify if these models are uniformly more accurate in predictions, or if their integration with controllers is important to the performance gains. As additional model comparisons, we include two *simple model* baselines often omitted in recent MBRL work: a linear



(*a*) Zero prediction model. (*b*) Linear model. (*c*) Default Neural Network.

Figure 9: Comparing the compounding error of the state-space system with simple linear and zero prediction models (shown is the MSE, 65^{th}, and 95^{th} percentiles per pole). On the simple state-space system, the simple models perform comparably to the neural network, but this does not indicate they would be as useful for control.



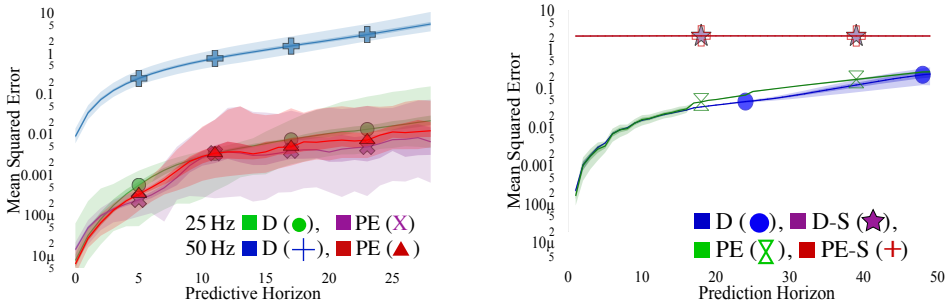
(*a*) **Without** $\cos\theta, \sin\theta$ transform. (*b*) **With** $\cos\theta, \sin\theta$ transform.

D (●) D-S (★) PE (⧖) PE-S (+)

Figure 10: Comparing the prediction accuracy (MSE median, 65^{th}, and 95^{th} percentiles) on the Reacher environment with (*right*) and without (*left*) transforming the joint angles from radians θ_i to an expanded state for each joint $(\cos\theta_i, \sin\theta_i)$ to account for angle wrap-around the interval $[0, 2\pi]$. The joint angle transformation, while increasing the state dimension from 10 to 15 improves the prediction accuracy substantially on short horizons and at convergence.

model (LIN) and a model predicting 0 (ZERO) to provide context for the prediction errors presented. These simple models performances are highlighted in Fig. 9 for the state-space system and in Fig. 2 for the other simulated environments. These simple models are extremely strong baselines in terms of compounding error, but our work does not study their usefulness for control (e.g. the zero prediction model would be useless for control).

Another popular modelling tool is shift from a true one-step prediction to that of a delta-state formulation. For the simulated environments in Fig. 2, there *can* be an improvement by using the delta-state parametrization or an probabilistic ensemble, but it is not constant across all systems. Importantly, especially when deploying on real systems, is that the ranking of prediction accuracy per-model is not consistent across environment. Another implementation trick used in MBRL and other applications of model-learning for control is to map angles and other state-variables that may have discontinuities to smooth representations. For example, with angles, the state-space can be expanded to be the sine and cosine of each angle, such as done by default in the Reacher environment. The increase in state dimension for smooth state-space improves the prediction accuracy notably at short horizons ($h < 50$) and at convergence in Fig. 10 – confirming results presented in Sec. 5.3 that it is not a crucial factor for prediction accuracy when the underlying dynamics are constant.



(*a*) Crazyflie Quadrotor. (*b*) Unitree A1 Quadruped.

Figure 11: Looking at the real-world prediction divergence on a flying (a) and walking robot (b) two training sets for a Euler angles of a flying robot (median, 65^{th}, and 95^{th} percentiles).

We tested numerous other model types and neural network parametrizations (e.g. depth, layer size, parameter tuning, normalization, training set size, etc.) on the state-space system, but they had minimal effect on the prediction accuracy. The additional results can be found in the Appendices.

Remark 6 *Different model parametrizations can have substantial impact on prediction accuracy, but more importantly the ranking of models is not maintained across environments, so multiple models should be validated on every new application.*

6. Case Studies of Compounding Error with Real World Data

In this section we showcase the prediction accuracy of models trained on real-world dynamics data from a quadrotor and a quadruped. The quadrotor is a high-speed, high-noise system with datasets from two different control frequencies (showcasing the potential prediction challenge with low state-signal to noise ratio, which is studied further in the Appendix). The quadruped shows how high-dimensional state-action spaces can reduce the prediction accuracy of true-state models. [2]

Quadrotor: The Crazyflie is a micro-aerial vehicle that masses only $27\,\mathrm{g}$, is $9\,\mathrm{cm}^2$ and performs on-board sensor fusion with an MPU-9250 inertial measurement unit ($s \in \mathbb{R}^3$, $a \in \mathbb{R}^4$). The dataset used corresponds primarily to episodes of flights from 1 to $5\,\mathrm{s}$ attempting to stabilize the Euler angles of the robot (data is from Lambert et al. (2019)). Most of these sequences are unstable and end with failed control where the Euler angles diverge or the robot collided with a wall due to drift of unmeasured states. The prediction accuracy is shown for a training set in Fig. 11(a). Note that a prediction of the same horizon in steps translates to a longer prediction in *time* when the model is trained on data with a lower frequency. The results show that there is a clear gain in prediction accuracy with the lower frequency.

Quadruped: As another evaluation of compounding data, we have randomly created episodes of length 50 from a batch of 3800 points of state-action data from the Unitree A1 Quadruped ($a_t \in \mathbb{R}^{60}$, $s_t \in \mathbb{R}^{52}$). This data comes from non-episodic data of the quadruped walking with a trot gait. The action space is a multi-modal controller representing a unstudied problem in MBRL for control.When a leg corresponding to one of the 12 motors (3 per leg) is in contact with the ground,

2. To contribute additional data to study compounding error on another platform, contact nol@berkeley.edu.

all actions are set to 0 except torque indicator, and vice-versa when the leg is in the air. The error shown when predicting a high-dimensional input-output relation is shown in Fig. 11(b).

7. Takeaways

Here, we outline a few key observations that should be considered when understanding the compounding prediction error on a new system. These should be the points of focus when model-learning for control is applied on new systems:

- **"*No Free Lunch*" Applies to Model Accuracy**: Given a fixed dataset, changing between different models will shift where error is present in the state-action space and over different predictive horizons. There will be no model that is perfect for one task, so designers should match their model to the desired controller.
- **Dynamics Dominates the Model Accuracy**: The properties of the dynamic system being modeled often has substantially greater effect on the prediction accuracy compared to model parametrization or training parameters. This point covers the accepted optimization procedures in the literature, but more complex optimizations, such as Automatic Machine Learning of dynamics models for MBRL (Zhang et al., 2021), is an exception that shifts modeling from an accuracy problem to one of maximization of reward.
- **Long-horizon Errors Can Level-off**: The results show that for many simulated applications, the error only compounds over an initial horizon h after which the error levels off or grows slowly. In most cases, this levelling happens when predictions are already useless for control purposes. However, we can postulate that might exist cases when the levelling off of error is sufficiently low that long-horizon predictions could be leveraged to solve sparse tasks.
- **Simple Models for Simple Systems**: In our low-dimensional experiments, simple linear models and deterministic neural networks provide a strong baseline that should be considered in real-world applications.
- **Low-to-Zero Noise is not a Accuracy Guarantee**: Transitioning from moderate to low to zero noise has diminishing returns on prediction accuracy, indicating that even simulated environments with no noise can still be difficult modelling tasks.

8. Future Work

Other Prediction Modalities Many other model types and trajectory propagation techniques exist that are well suited to a more narrow spectrum of problems than deep neural networks. Linear models are suited to linear systems (Fu et al., 2016; Bansal et al., 2017), Gaussian processes are useful for lower dimensionalities and dataset sizes (Wang et al., 2006; Deisenroth and Rasmussen, 2011; McKinnon and Schoellig, 2017), trajectory-based neural networks are useful with closed form control laws (Lambert et al., 2020b), and new physics-based neural networks are yet to be deployed for control (Krishnapriyan et al., 2021; Jiahao et al., 2021). When dynamics models learn distributions instead of specific transitions, the method for propagation of the imagined trajectory can heavily impact both compounding error and downstream control. In this work, only expectation-based propagation is used for probabilistic models and probabilistic ensembles. Crucially, careful

understanding of the various model types' strengths and weaknesses with respect to compounding error will yield improved performance when paired with a suitable controller.

Dynamics Modeling with Distribution Shift In model-based reinforcement learning, the data used to train the model changes with each step. This can take two forms: the amount of data in the replay buffer and the relative shape of the data distribution. There are complicated relationships in model-based reinforcement learning between model accuracy, task-performance, and data-distribution that are not studied in this paper. Recent work suggests that optimizing solely for prediction accuracy does not result in maximum task-performance (Lambert et al., 2020a; Zhang et al., 2021). These data-properties are very difficult to quantify but crucial for performance – for example, the relative density of labeled data points and the underlying difficulty of a transition to model both effect prediction accuracy and are not well understood.

9. Conclusion

Accurately understanding the dynamics of a robotic system with finite data can enable many forms of control. In this work we characterize the compounding error problem that emerges in long-horizon prediction with learned one-step, neural network models. In hopes of the improvement of tools used to model these systems, we show that underlying system dynamics have a dramatic effect on the short- and long-term prediction accuracy for any system. With this understanding, model-learning for control can design advanced controllers with a more precise relationship between the prediction and action-decision error.

Acknowledgements

The authors would like to thank Tianyu Li for providing data from the Unitree A1 Quadruped and Howard Zhang for participating in useful discussions.

References

Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michel L Littman. Combating the compounding-error problem with a multi-step model. *arXiv preprint arXiv:1905.13320*, 2019.

Somil Bansal, Anayo K Akametalu, Frank J Jiang, Forrest Laine, and Claire J Tomlin. Learning quadrotor dynamics using neural network for flight control. In *IEEE Conference on Decision and Control*, pages 4653–4660, 2016.

Somil Bansal, Roberto Calandra, Ted Xiao, Sergey Levine, and Claire J Tomiin. Goal-driven dynamics learning via bayesian optimization. In *IEEE Conference on Decision and Control (CDC)*, pages 5168–5173, 2017.

Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

Frank M Callier and Charles A Desoer. *Linear system theory*. Springer Science & Business Media, 2012.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Neural Information Processing Systems*, pages 4754–4765, 2018.

Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization. In *Conference on Robot Learning*, pages 617–629. PMLR, 2018.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

Marc P. Deisenroth and Carl E. Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *International Conference on Machine Learning*, pages 465–472, 2011.

Justin Fu, Sergey Levine, and Pieter Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4019–4026. IEEE, 2016.

Wojciech Giernacki, Mateusz Skwierczyński, Wojciech Witwicki, Paweł Wroński, and Piotr Kozierski. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. In *IEEE International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 37–42, 2017.

Jayesh K Gupta, Kunal Menda, Zachary Manchester, and Mykel Kochenderfer. Structured mechanical models for robot learning and control. In *Learning for Dynamics and Control*, pages 328–337. PMLR, 2020.

Nicolas Heess, Greg Wayne, David Silver, Timothy Lillicrap, Yuval Tassa, and Tom Erez. Learning continuous control policies by stochastic value gradients. *arXiv preprint arXiv:1510.09142*, 2015.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, pages 12498–12509, 2019.

Tom Z Jiahao, M Ani Hsieh, and Eric Forgoston. Knowledge-based learning of nonlinear dynamics and chaos. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(11):111101, 2021.

Balázs Kégl, Gabriel Hurtado, and Albert Thomas. Model-based micro-data reinforcement learning: What are the crucial model properties and which to choose? In *International Conference on Learning Representations*, 2021.

Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2004.

Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. Learning-based model predictive control for safe exploration. In *2018 IEEE conference on decision and control (CDC)*, pages 6059–6066. IEEE, 2018.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Nathan Lambert, Daniel S Drew, Joseph Yaconelli, Sergey Levine, Roberto Calandra, and Kristofer SJ Pister. Low-level control of a quadrotor with deep model-based reinforcement learning. *IEEE Robotics and Automation Letters*, 4(4):4224–4230, 2019.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. In *Learning for Dynamics and Control*, pages 761–770. PMLR, 2020a.

Nathan Lambert, Albert Wilcox, Howard Zhang, Kristofer SJ Pister, and Roberto Calandra. Learning accurate long-term dynamics for model-based reinforcement learning. *International Conference on Decision and Control (CDC)*, 2020b.

Lennart Ljung. System identification. *Wiley encyclopedia of electrical and electronics engineering*, pages 1–19, 1999.

Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics and Automation magazine*, 19(3):20–32, 2012.

Christopher D McKinnon and Angela P Schoellig. Learning multimodal models for robot dynamics online with a mixture of gaussian process experts. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 322–328. IEEE, 2017.

Anusha Nagabandi, Kurt Konoglie, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. *arXiv preprint arXiv:1909.11652*, 2019.

Ali Punjani and Pieter Abbeel. Deep learning helicopter dynamics models. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3223–3230. IEEE, 2015.

William D Stanley, Gary R Dougherty, Ray Dougherty, and H Saunders. Digital signal processing. 1988.

Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. Improving multi-step prediction of learned time series models. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Jack Wang, Aaron Hertzmann, and David J Fleet. Gaussian process dynamical models. In *Advances in neural information processing systems*, pages 1441–1448, 2006.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721, 2017.

Chenjun Xiao, Yifan Wu, Chen Ma, Dale Schuurmans, and Martin Müller. Learning to combat compounding-error in model-based reinforcement learning. *arXiv preprint arXiv:1912.11206*, 2019.

Baohe Zhang, Raghu Rajan, Luis Pineda, Nathan Lambert, André Biedenkapp, Kurtland Chua, Frank Hutter, and Roberto Calandra. On the importance of hyperparameter optimization for model-based reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 4015–4023. PMLR, 2021.

Appendices

The Appendices contain the following content:

- Sec. A contains additional environment (see Sec. A.1) details and the model training parameters (see Sec. A.2).
- Sec. B contains additional experiments investigating compounding error. The additional experiments are broken up into two sections: 1) additional effects of model properties are discussed in Sec. B.1 and 2) other impacts on and lenses for studying compounding error are illustrated in Sec. B.2.

Appendix A. Extra Experimental Details

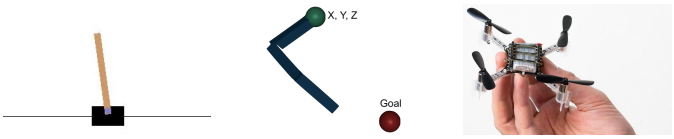
A.1. Additional Environment Details

State-space System For a discrete-time state-space system, the time evolution of the state over time can be solved for explicitly. This time evolution, shown in Eq. (4), is the goal of what these composed one-step systems attempt to model, yet result in compounding errors. The solution to the discrete transition dynamics takes the form an transient response from the initial state and a forced response corresponding to the applied action sequence:

$$s_t = \underbrace{\boldsymbol{A}^t \boldsymbol{s}_0}_{\text{Transient}} + \underbrace{\sum_{l=0}^{k-1} A^{k-l-1} B a(l)}_{\text{Forced Response}} \tag{4}$$

For the state-space system, the time by which the prediction error converges to its minimum for a given pole is proportional to the number of steps by which the transient of the initial state will decay. The transient is proportional to powers of the dynamics matrix, $||A||^k$, which is proportional to the powers of the eigenvalues. For the poles in $\{0.01, 0.05, 0.1, 0.25, 0.5\}$ the number of time-steps until the steady state error is reached is proportional to the mean transient decay in Tab. 1. Example trajectories and one-step predictions are shown in Fig. 13. In this parametrization, the input at any time is randomly sampled, so the forced response becomes a source of noise.

Cartpole We evaluate predictions of state and reward of cartpole agents conditioned on a varied Linear Quadratic Regulator (LQR) (Callier and Desoer, 2012) (Fig. 2).



(*a*) Cartpole. (*b*) Reacher. (*c*) Quadrotor.

Figure 12: Experimental platforms.

Quadrotor - Simulated The quadrotor model is based off the Crazyflie (Giernacki et al., 2017) – an 27 g, open-source micro-aerial vehicle. The 12 state Euler-step simulation follows (Mahony et al., 2012). The simulated controller is a linear, pitch and roll Proportional-Derivative controller with randomly sampled parameters.

Reacher The task associated with the environment is to maneuver the end-effector of the arm from an initial position state to an end position state. Our experiments control the agent using a Proportional-Integral-Derivative (PID) controller with randomly generated parameter vectors $\boldsymbol{K} \in \mathbb{R}^{15}$.

Poles (ρ)	Transient decay $k : \|\|\boldsymbol{A}^k \boldsymbol{x}_0\|\| < 1 \times 10^{-4}$	Delta-state labels mean($\|\|s_{t+1} - s_t\|\|$) $\pm \sigma(\cdot)$	True-state labels mean($\|\|s_{t+1}\|\|$) $\pm \sigma(\cdot)$
0.01	4.7	0.019 ± 0.063	0.011 ± 0.021
0.05	6.1	0.020 ± 0.066	0.012 ± 0.020
0.10	7.4	0.018 ± 0.052	0.012 ± 0.026
0.25	11.2	0.017 ± 0.044	0.016 ± 0.048
0.50	21.8	0.020 ± 0.042	0.033 ± 0.087
0.75	55.5	0.029 ± 0.051	0.134 ± 0.299
0.90	168.3	0.086 ± 0.156	1.284 ± 2.610
0.95	N.A.	0.225 ± 0.351	8.511 ± 14.30
1.00	N.A.	7.442 ± 11.164	201.2 ± 450.5
1.10	N.A.	$6.5 \times 10^4 \pm 1.9 \times 10^5$	$7.0 \times 10^5 \pm 1.9 \times 10^6$

Table 1: Dataset properties for state-space systems with different eigenvalues. The transient decay is the number of discrete transitions on average by which the transient term in Eq. (4) decays below 1×10^{-4} (chosen based on the steady state prediction error that the most stable poles converge to on average). The mean and standard deviations of the data labels represent a relative challenge for the models – as the input and output normalizers need to compress a wider range of data to $\mathcal{N}(0, 1)$, the more sensitive the learning process becomes (for more on normalization, see Sec. B.1.2).

Quadrotor - Real World Due to the high noise on the accelerations measured by on-board sensors, we evaluate predicting a restricted state of Euler angles from direct motor voltages as:

$$s_t = \begin{bmatrix} \text{Yaw:}\psi & \text{Pitch:}\theta & \text{Roll:}\phi \end{bmatrix}, \quad a_t = \begin{bmatrix} V_1 & V_2 & V_3 & V_4 \end{bmatrix} \tag{5}$$

When discretizing dynamics, lower sample rates yield more unstable eigenvalues, but the system can also gain by having relatively lower signal-to-noise ratio on the measured states.

Quadruped - Real World The state, $s_t \in \mathbb{R}^{52}$, corresponds to the following:

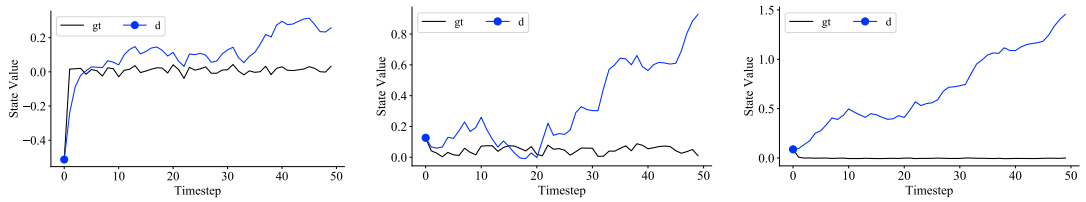
$$s_t = \begin{bmatrix} \boldsymbol{x} & \dot{\boldsymbol{x}} & \boldsymbol{\omega} & \boldsymbol{x}_k^f & \psi & \theta & \phi & \alpha_i & \dot{\alpha}_i & c_k^f \end{bmatrix} \tag{6}$$

Here, $\boldsymbol{x} \in \mathbb{R}^3$ is the position of the robot base, $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular rates of the robot base, $\boldsymbol{x}_k^f \in \mathbb{R}^3$ is the position of the kth foot, ψ, θ, ϕ are the Euler angles, α_i is the joint angle for each of the 12 motors, and c^f is an indicator if each foot is in contact with the ground. The action, $a_t \in \mathbb{R}^{60}$, is a bimodal input, where for each of the 12 motors on the robot has 5 dimensional action space of desired joint position and velocities ($\alpha_i^*, \dot{\alpha}_i^*$), low-level PID control coefficients (K_i^p, K_i^d), and set torque (τ_i):

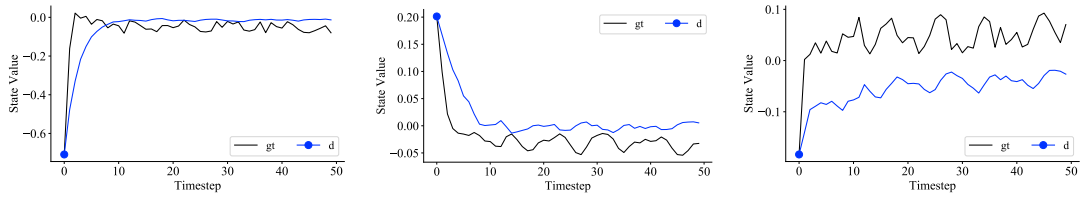
$$a_t = \begin{bmatrix} \alpha_i^* & \dot{\alpha}_i^* & K_i^p & K_i^d & \tau_i \end{bmatrix} \tag{7}$$

A.2. Model Training

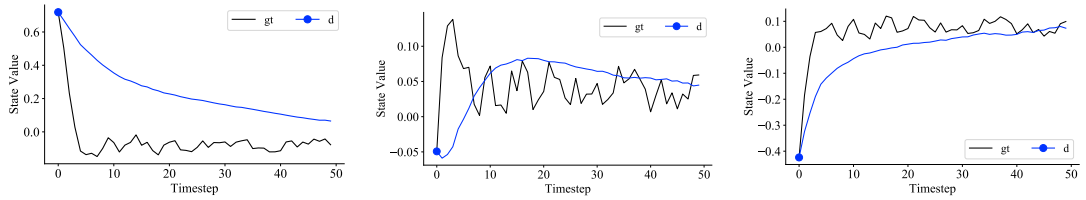
To learn a model of the dynamics, we use a feedforward neural network with two hidden layers of width 256. Ensemble models use $E = 5$ members. The models are trained on 100 trajectories



(*a*) Poles at 0.1, state index 0. (*b*) Poles at 0.1, state index 1. (*c*) Poles at 0.1, state index 2.



(*d*) Poles at 0.25, state index 0. (*e*) Poles at 0.25, state index 1. (*f*) Poles at 0.25, state index 2.



(*g*) Poles at 0.5, state index 0. (*h*) Poles at 0.5, state index 1. (*i*) Poles at 0.5, state index 2.

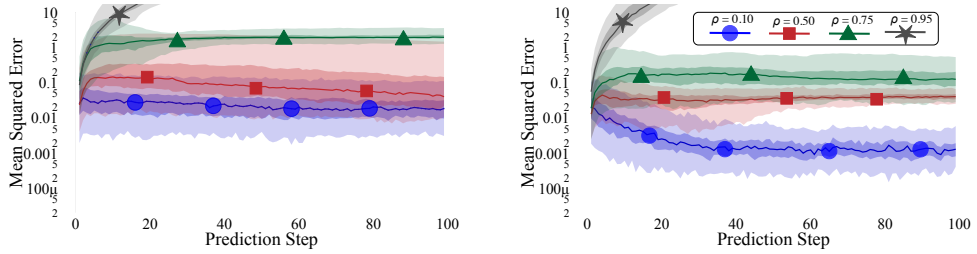
Figure 13: Example trajectories for the state-space system in three dimensions showcasing the predictions of deterministic models for $\rho = 0.1, 0.25, 0.5$. The three states for each pole correspond to one example trajectory, and the matrices and initial states are different for each of the representative poles shown. *gt* is the true state dynamics and *d* is the one-step model.

– all with different matrices $\boldsymbol{A}, \boldsymbol{B}$ with the same poles for the state-space systems. For the other environments, control policies are randomly sampled to create diverse training and testing data. The models are trained for 20 epochs with a learning rate of 0.0003 for deterministic and 0.000025 for probabilistic models with the Adam optimizer. Deterministic models use a batch size of 32 and probabilistic models use a batch size of 64. All state data is normalized to a unit Gaussian and action data is normalized to $[-1, 1]$ for training. The equations for computing the loss during training are shown for MSE and NLL:

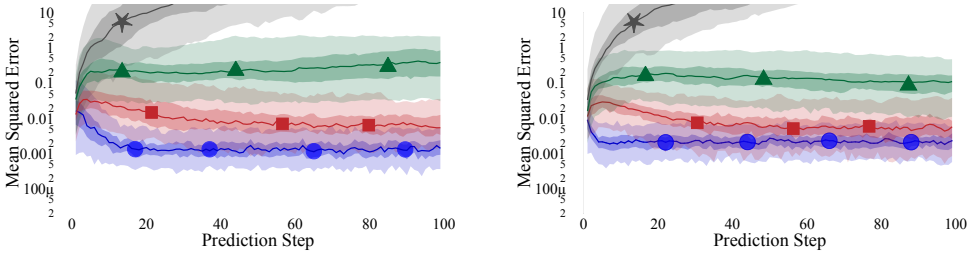
$$l_{\text{MSE}} = \sum_{n=1}^{N} \| f_\theta(s_n, a_n) - s_{n+1} \|_2^2 \,, \tag{8}$$

$$l_{\text{NLL}} = \sum_{n=1}^{N} [\mu_\theta(s_n, a_n) - s_{n+1}]^T \Sigma_\theta^{-1}(s_n, a_n) [\mu_\theta(s_n, a_n) - s_{n+1}] + \log \det \Sigma_\theta(s_n, a_n) \,. \tag{9}$$

Important to the convergence of supervised learning is the shape and magnitude of the training data. In Tab. 1, we compare the training data shape for the different state-space system eigenvalues when using true- and delta-state labels for the one-step dynamics model. As the eigenvalues become



(*a*) Training set: 1 Trajectory. (*b*) Training set: 5 Trajectories.



(*c*) Training set: 10 Trajectories. (*d*) Training set: 100 Trajectories.

Figure 14: Prediction error across representative poles for different training set sizes given a constant training set of 100 trajectories for each pole. The models quickly converge with only 10 training trajectories.

more unstable, the variation in the training labels grows exponentially. This effect can be counteracted with normalization if it is uniform, but the model loses the ability to differentiate between elements at fine scales, which could render the usefulness of the model low.

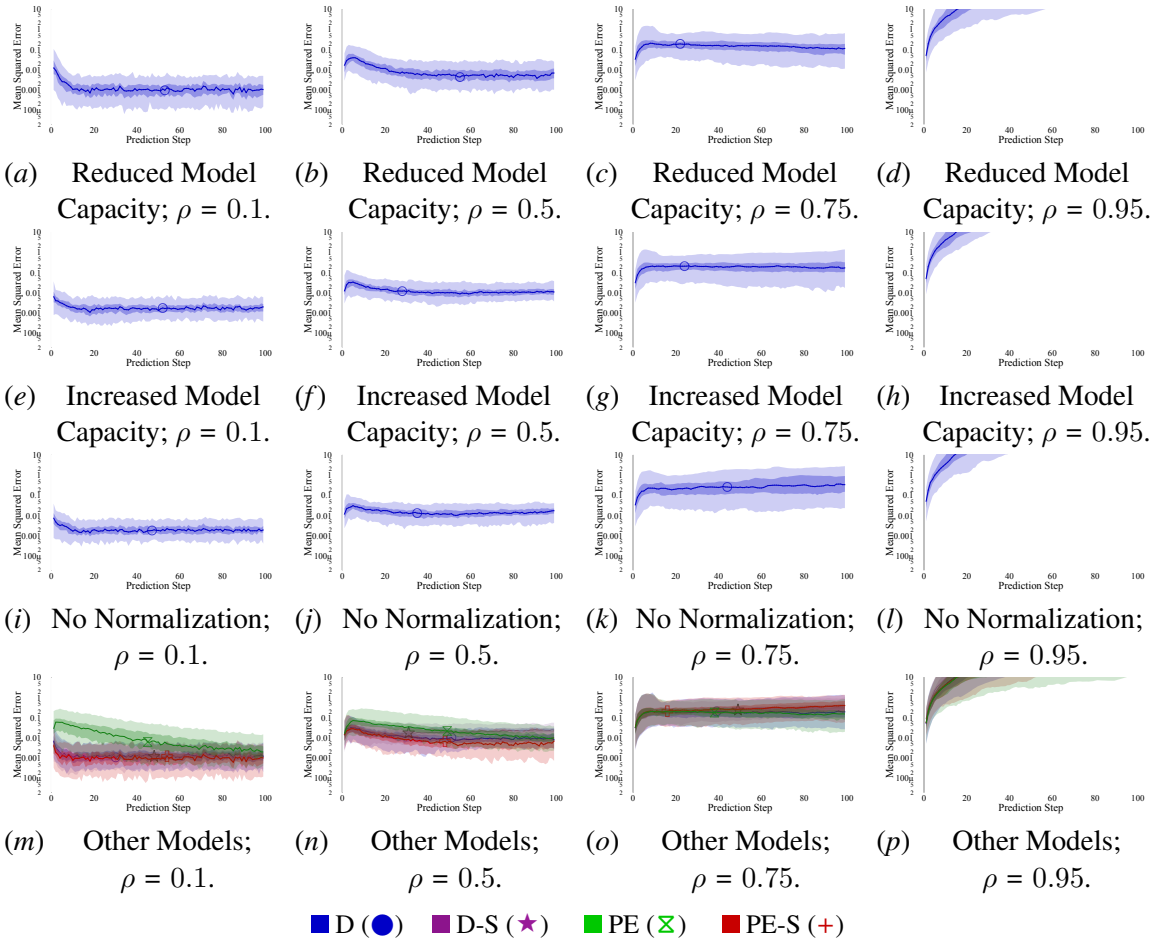
The linear model is the result of solving the least squared problem, $\arg\min_\omega ||X\omega - b||^2$, where $b = (s_{t+1} - s_t), \omega = \left[\hat{\boldsymbol{A}} \;\; \hat{\boldsymbol{B}}\right], X = \left[S \;\; U\right]$ (S and U are stacked state and action vectors). The next state is then predicted with $\hat{s}_{t+1} = \hat{\boldsymbol{A}} s_t + \hat{\boldsymbol{B}} a_t$.

Appendix B. Additional Experiments

B.1. Further Investigation of the Effect of Model Properties on Compounding Error

B.1.1. MODEL: CAPACITY

Given recent advancements in deep learning driven by large datasets with evolving model architectures, one-step dynamics models operate with simpler and smaller models and datasets. The results shown in Fig. 1 show the model prediction accuracy with a field-standard model capacity of 2 hidden layers and 250 neurons. To further examine the effects of model capacity, we also test the rate of divergence for deep predictive models on state-space systems with models with substantially fewer or greater parameters. The results of model predictive error with models of a hidden layer of size 32, shown in Fig. 15(a-d), and of a model with 3 hidden layers of width 512, shown in Fig. 15(e-h) show that for a simple task, changing the model size has little impact on prediction accuracy. The smaller model has slightly higher prediction error and variance among errors and the larger model has slightly improved prediction accuracy, though the effect is substantially less than the effects of



(*a*) Reduced Model Capacity; $\rho = 0.1$. (*b*) Reduced Model Capacity; $\rho = 0.5$. (*c*) Reduced Model Capacity; $\rho = 0.75$. (*d*) Reduced Model Capacity; $\rho = 0.95$.

(*e*) Increased Model Capacity; $\rho = 0.1$. (*f*) Increased Model Capacity; $\rho = 0.5$. (*g*) Increased Model Capacity; $\rho = 0.75$. (*h*) Increased Model Capacity; $\rho = 0.95$.

(*i*) No Normalization; $\rho = 0.1$. (*j*) No Normalization; $\rho = 0.5$. (*k*) No Normalization; $\rho = 0.75$. (*l*) No Normalization; $\rho = 0.95$.

(*m*) Other Models; $\rho = 0.1$. (*n*) Other Models; $\rho = 0.5$. (*o*) Other Models; $\rho = 0.75$. (*p*) Other Models; $\rho = 0.95$.

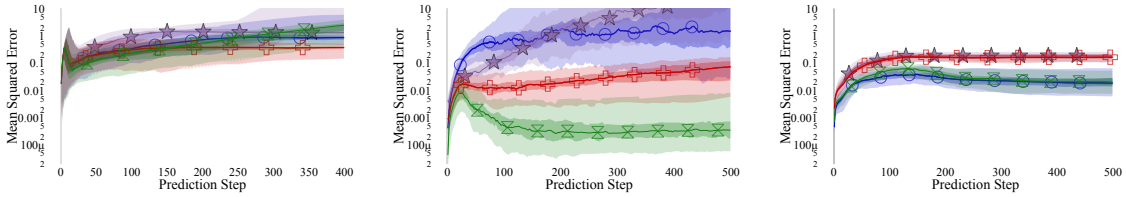
Figure 15: Comparing the effects of common modeling tools on MSE (median, 65^{th}, and 95^{th} percentiles) – changing models from top to bottom with increasing poles from right to left. Reducing the model capacity by lowering layer width from 256 to 32 units or by removing data normalization does not substantially affect prediction accuracy on the simple state-space system. The probabilistic ensemble is able to improve on prediction accuracy, though contrary to common practices, only when using the state-based predictions.

system properties studied in Sec. 5.1. Additionally, the model accuracy with different training set sizes is shown in Fig. 14, where there is not a substantial effect beyond the first few trajectories.

B.1.2. MODEL: NORMALIZATION

Tools for designing and optimizing neural networks are designed to work on data centered around unit normal distributions – *i.e.* identical and independently distributed data closely centered around 0. In robotics data where one-step models are deployed, this is often not the case, which leaves it up the user to maintain data cleaning practices for dynamics model training.

Normalization techniques map state and action variables over different ranges (bounds) and shapes (relative density) to well-behaved distributions to aid model training. The model normalizes



(*a*) Cartpole. $d_s + d_a = 5$. (*b*) Quadrotor. $d_s + d_a = 13$. (*c*) Reacher. $d_s + d_a = 20$.

D (●) D-S (★) PE (⧖) PE-S (+)

Figure 16: Showing how re-computing actions has a dramatically different effect on prediction MSE depending on the policy type (median, 65^{th}, and 95^{th} percentiles). We hypothesize that these diverging predictions could be worsened when coupled with feed-forward control policies. The data used is the same as in Fig. 2 where the policy is computed based on the predicted state, rather than mimicking the original action. Note, the predictions for the different environments are across different horizons.

the inputs and targets at training, and at prediction time utilizes these distributions to map new inputs to the latent space of the model and then back into the true distribution. Such mappings can also contribute to compounding error by pushing both inputs and targets of some validation data further outside of a training distribution. In this work, we map continuous variables to a normal distribution $\mathcal{N}(0, 1)$ and bounded variables (such as actions) to a uniform distribution $\mathcal{U}(-1, 1)$. The effects of turning this normalization off is a small increase in prediction error, shown in Fig. 15(i-l). Normalization is heavily sensitive to outliers because if some training points are substantially outside the distribution, it will further concentrate the data of interest onto a small region of the input space, resulting in a harder learning problem and one that is more sensitive to model bias.

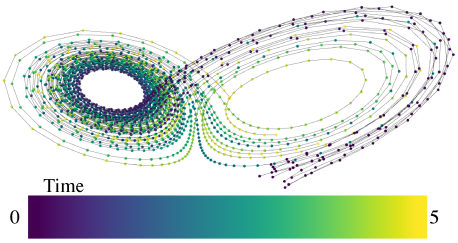
B.2. Other Factors Impacting Compounding Error

In this section, we build upon our study of system and model properties to show how some of these variables can interweave in complex manners, resulting in difficultly to forecast model performance.

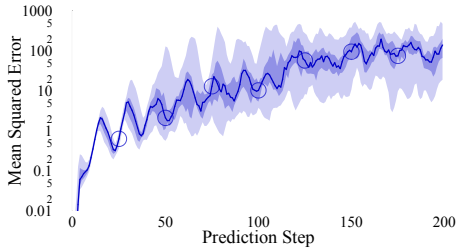
B.2.1. SYSTEM: RE-COMPUTING ACTIONS

When planning into the future there are two potential actions sequences: a logged action sequence to compare the model accuracy of a learned model to measured data and a generated action sequence to evaluate the potential usefulness of a simulated trajectory. The effect of re-computing the actions passed into the predictive model as $a_t = \pi(\hat{s}_t)$ instead of the original action sequence being provide by an oracle can be crucial to if a model will be useful under a certain controller. The action on a predicted state will take the form of $a = \pi(\hat{s}) = \pi(s + \epsilon_t)$, so the action returned varies in most the model accuracy and policy robustness to perturbation. Depending on the problem formulation, long horizon prediction is often done with an action sequence passed into the model (representing the ground truth), but model accuracy can be dramatically different if the actions are re-computed from the predicted state as computed action sequences will also exhibit compounding error.

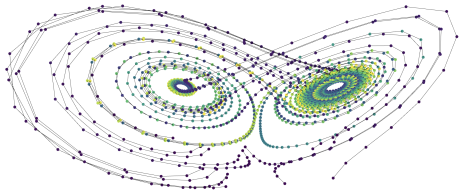
The original results of the models on the simulated robotic tasks are shown in Fig. 2 and the results where the oracle no longer provides the original action sequence are shown in Fig. 16. In this case, all environments show approximately a $10\times$ increase in error when not given the action



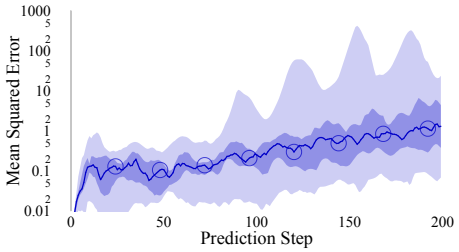
(a) Predictions - clustered s_0.



(b) Errors - clustered s_0.



(c) Predictions - broad s_0.



(d) Errors - broad s_0.

Figure 17: Highlighting the dynamics of the of the Lorenz system (*a,c*) as a challenging dynamics problem due to its chaotic dynamics that result in multi-modal behavior. The chaotic system also happens to be stable, which is reflected by its bounded prediction error per-step (*b,d*: median, 65^{th}, and 95^{th} percentiles) and a testing set of new trajectories. (*a,b*) are trajectories sampled from a more restricted initial condition, where the initial x, y, z coordinates fall in $[5, 10]$. (*c,d*) is a more diverse training and testing set, where the initial states x, y, z are sampled from $[-10, 10]$, though the dynamics model generalizes better to new previously unseen data.

sequence from an oracle. Crucially, this type of planning without a real action sequence is how most MPC algorithms compute the action with a learned model. One may expect that reflexive policies recomputing actions would diverge faster because they compute the control off only the current state, while controllers with built in damping or slew limiting can predict more accurately (such as the integral or derivative terms in a PID controller), but this trend is not clear in our simulated results.

Generated action sequences are closely linked to using the models for control, but generally model training is only evaluated on accuracy with logged data. To date, there are no methods to evaluate the potential accuracy of randomly generated actions leaving future work to understand this relationship – for example, by evaluating the bootstrapped uncertainty estimate of a probabilistic ensemble across a planned trajectory.

B.2.2. EXAMPLE: PREDICTING CHAOTIC DYNAMICS

A fundamental limit of prediction can be posed as how to predict chaotic systems. A chaotic system is defined by the idea that a small perturbation in state can grow to an exponential difference over time. As a case study, we include prediction errors for the Lorenz system Lorenz (1963), shown in Eq. (12). The canonical parameter η is often ρ, but we have replaced it to avoid overloading our

symbol for pole.

$$\dot{x} = \sigma(y - x) \tag{10}$$
$$\dot{y} = x(\eta - z) - y \tag{11}$$
$$\dot{z} = xy - \beta z \tag{12}$$

In this work, the initial states for the Lorenz system are constrained to two different distributions: $x_0, y_0, z_0 \in [5, 10]$ or $x_0, y_0, z_0 \in [-10, 10]$, resulting in a comparison between how training and test data distribution can affect dynamics model performance. Both training sets include 100 trajectories of length 500 that are evaluated on 100 previously unseen trajectories of length 200. While in practice learning to identify the three governing parameters of the dynamics could result in more accurate predictions, learning models for systems by which the analytical equations are unknown poses a problem of great interest for the field. As the simulated version of system has no noise and stable dynamics, the prediction error do not growth to infinity, but rather proportional to the separation of the two stable points, shown by the oscillations in Fig. 17.

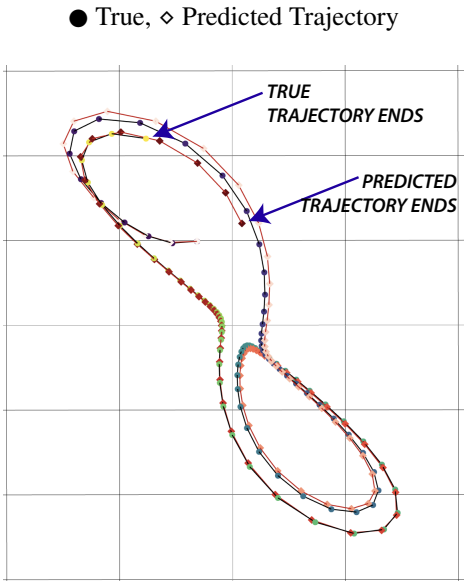


Figure 18: Example showing the prediction on the Lorenz system, where the predictions tend to advance in time and do not diverge rapidly.

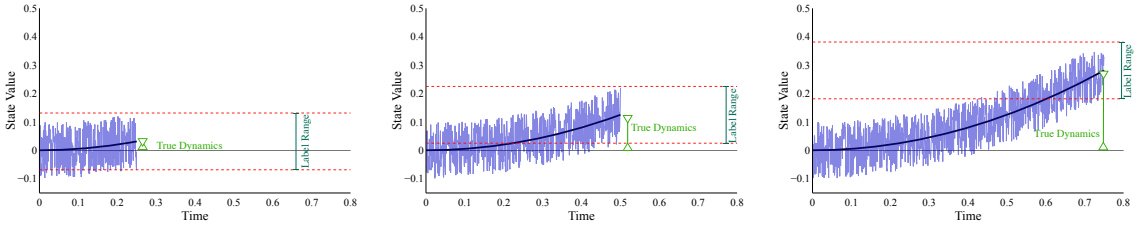
B.2.3. EXAMPLE: CONTROL FREQUENCY & SIGNAL TO NOISE RATIO

The signal to noise ratio (SNR) Stanley et al. (1988) is a metric for evaluating the relative strength of a signal that one wishes to measure to the noise that will be present in measurements, commonly deploying in digital signal processing. A related topic emerges with any dynamical system, where changing the sampling frequency of a system with uniform observation noise can implicitly change the SNR of the transition labels for supervised learning – a shorter sample time leads to higher impact of noise.

Consider a canonical control system, the double integrator, shown in Eq. (13), that is the underlying dynamics of Newtownian systems:

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \quad \boldsymbol{o}(t) = \boldsymbol{x}(t) + \omega^m(t) \tag{13}$$

With a constant input – corresponding to a constant force – solutions to this equation take the form of a quadratic function. With a set measurement noise level the dynamics of a given system when sampled at different frequencies results in modeling problems of varying difficult, which we propose understanding as a signal to (measurement) noise ratio. The true change in state, $s_{t+1} - s_t$ is corrupted by some noise from the current and previous measurements, ω_t^m and ω_{t+1}^m, acting on the current and past observations $\boldsymbol{o}$, where the relative size of the true dynamics can be described as



(*a*) 250 ms step size. (*b*) 500 ms step size. (*c*) 750 ms step size.

Figure 19: Showing how signal to noise ratio and step size are important for robotic tasks. The difference between the potential measurement regions is the magnitude of signal present in the labelled data. Crucially, a slower sampling frequency can increase the resolution of the labelled data, improving downstream prediction accuracy.

a signal-to-noise ratio (SNR):

$$\text{SNR} \approx \frac{||s_t - s_{t-1}||}{||s_t - s_{t-1} + \omega_t^m + \omega_{t-1}^m||} \tag{14}$$

Crucial to accurately modelling dynamics is for the sampling rate to be slow enough by which the noise is a minor contribution to the targets, SNR >> 1. An illustration of this example is shown in Fig. 19, where a constant noise interval illustrates the possible data labels with different sample rates. All three simulators in this work do not have measurement error, though every real system's measurement error is determined by the quality of on-board or external state measurement. In the real quadrotor system that follows in Sec. 6, we evaluate model accuracy for two sampling rates. Finally, in real systems noise distributions often take on asymmetric and complex distributions, which can be measured and understood as specific detriments to learned model accuracy.