SINDy+RL: Pendulum and Nonstationary Oscillator Control

Hirsa Kia

December 14, 2025

Abstract

This project investigates a particular SINDy-RL approach in which Sparse Identification of Nonlinear Dynamics (SINDy) is used to learn compact surrogate models that drive reinforcement-learning-style control of nonlinear systems. Two benchmark environments are considered: the classic swing-up pendulum and a second-order oscillator with time-varying stiffness and damping. In both cases, SINDy is trained from limited interaction data and embedded in a model-based RL loop that performs receding-horizon planning over candidate action sequences. For the pendulum, online SINDy-RL is compared with an offline variant to illustrate the value of continuous model adaptation. For the nonstationary oscillator, online SINDy-RL is compared with a neural-network model-based agent and a model-free Deep Q-Network (DQN). Simulation results show that online SINDy-RL adapts to parameter drift and tracks sinusoidal references more reliably than the DQN agent, while achieving comparable performance to the neural-network baseline and maintaining interpretability through sparse coefficient estimates. The pendulum experiments illustrate that useful swing-up policies can be obtained from a SINDy surrogate trained on limited samples, albeit with sensitivity to modeling error and data quality.

Contents

1 Introduction

Model-based reinforcement learning (MBRL) seeks to improve sample efficiency by learning a predictive model of the environment and using it for planning or policy improvement, instead of relying solely on trial-and-error interactions. In continuous-control domains this typically involves high-capacity neural networks, which demand large datasets and yield models that are difficult to interpret and analyze. Sparse Identification of Nonlinear Dynamics (SINDy) provides an appealing alternative by discovering low-dimensional governing equations from data, selecting only a small subset of candidate basis functions.

1.1 Very important disclaimer

This work is investigating a very limited and narrow alley: 1. we are investigating one particular augmentation of SINDy to RL design. 2. the considered environments are only two benchmark systems.
The central question is whether SINDy models learned from limited interaction data can support reinforcement-learning-style control that is both data-efficient and interpretable, and how such SINDy-RL controllers compare with baselines when the dynamics are stationary versus drifting.

Bear in mind that many alternative architectures and training protocols are possible and could plausibly achieve even stronger performance.[1]

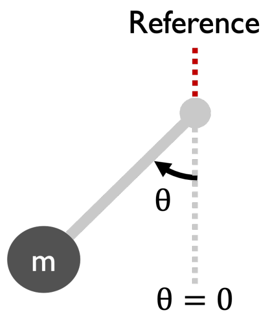

Figure 1: Pendulum

2 Methods

2.1 Environments

The considered systems are: a swing-up pendulum with fixed physical parameters but noisy observation space, and a nonstationary oscillator whose stiffness and damping drift over time. In both environments, the controller aims to track a reference signal while penalizing velocity and control effort.

[1]**Mini but important disclaimer**: In writing of the report, AI has been used.

Pendulum (stationary). The pendulum environment uses the state $x = [\cos\theta, \sin\theta, \dot{\theta}]^\top$ and a bounded torque input u. The task is to swing the pendulum from near the downward position to the upright position and keep it near $\theta = \pm\pi$, while penalizing deviations and control magnitude via a quadratic cost in angle, angular velocity, and torque. In all pendulum experiments the physical parameters are fixed; only the data-collection and learning protocols (offline vs. online, noiseless vs. noisy observations) differ.

Nonstationary oscillator. The oscillator has state $x = [p, v]^\top$ with dynamics

$$\dot{p} = v, \qquad \dot{v} = -\omega(t)^2 p - \gamma(t) v + u,$$

where $\omega(t)^2$ and $\gamma(t)$ are time-varying stiffness and damping coefficients, respectively. Both follow piecewise-smooth schedules that define three phases (start-up, relaxation, and steady), as illustrated in Figure 3. The control objective is to track a sinusoidal reference position $p_{\text{ref}}(t)$ while minimizing

$$\ell(p, u, t) = (p - p_{\text{ref}}(t))^2 + 0.01u^2.$$

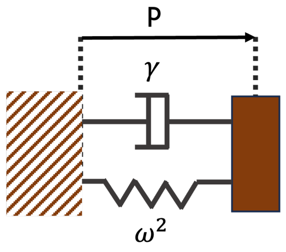

Figure 2: Oscillator

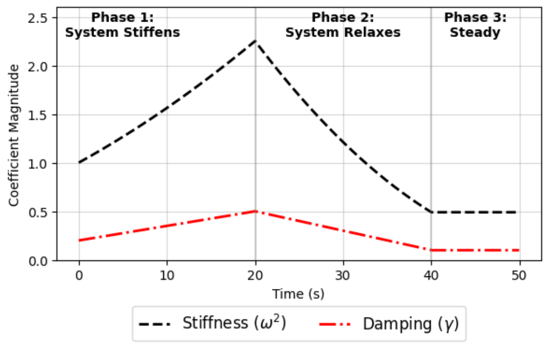

Figure 3: Time-varying stiffness $\omega(t)^2$ and damping $\gamma(t)$ for the nonstationary oscillator.

2.2 SINDy Dynamics Modeling

The overall SINDy-RL pipeline is shown in Figure 4. A replay buffer stores recent transitions (x_t, u_t, x_{t+1}). A feature library $\Phi(x, u)$ maps states and actions into candidate nonlinear terms. For the pendulum, Φ includes constants, $\cos\theta$, $\sin\theta$, $\dot{\theta}$, the control u, and low-order products such as $\sin\theta\,\dot{\theta}$ and $u\,\dot{\theta}$. For the oscillator, Φ uses constants, p, v, u, and interactions such as p^2, pv, and u^2. Approximate derivatives are obtained via

$$\dot{x}_t \approx \frac{x_{t+1} - x_t}{\Delta t},$$

and SINDy solves the sparse regression problem

$$\dot{X} \approx \Phi(X, U)\Xi$$

using a sequentially thresholded least-squares procedure that alternates ridge regression with coefficient pruning. The resulting sparse matrix Ξ defines the dynamics model $f_{\text{SINDy}}(x, u)$.

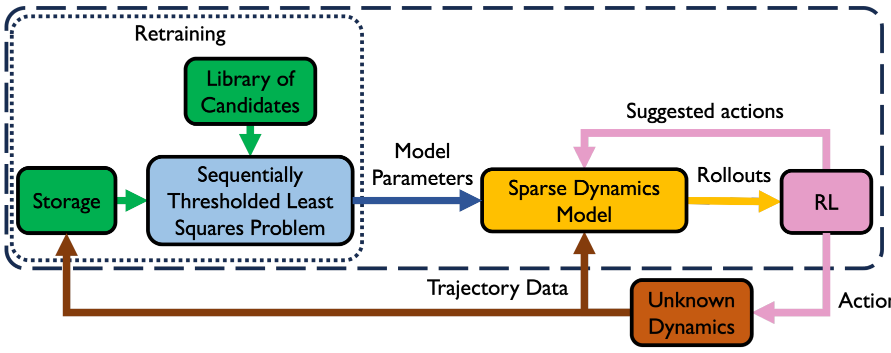

Figure 4: SINDy-RL pipeline: transitions are stored, lifted by a feature library, and used in sparse regression to obtain a dynamics model that feeds into the RL/planning module.

2.3 SINDy-RL Control and Baselines

Receding-horizon SINDy-RL. For both environments, the SINDy model is embedded in a receding-horizon planner. At each step, the current state initializes a rollout, and a population of candidate action sequences over horizon H is sampled. For each candidate, the SINDy dynamics are integrated forward with Euler updates

$$x_{k+1} = x_k + \Delta t\, f_{\text{SINDy}}(x_k, u_k),$$

and the cumulative cost $\sum_{k=0}^{H-1} \ell(x_k, u_k, t + k\Delta t)$ is estimated. The first action of the lowest-cost sequence is applied to the real environment, and the resulting transition is stored for future SINDy updates.

Online vs. offline SINDy. This distinction is used in both the pendulum and oscillator experiments. In the *offline* SINDy-RL variant, an initial dataset is collected under a simple exploration policy, SINDy is fit once on this dataset, and the resulting model is used for the entire episode without further updates. In the *online* variant, the same planner is used but SINDy is periodically refit on the latest buffer contents, enabling its coefficients to adapt either to noisy observations (pendulum) or to drifting stiffness and damping (oscillator).

Additional comparison models (oscillator only). For the nonstationary oscillator, several additional baselines are considered:

- **Least-squares (LS) dynamics model:** Linear regression on the same feature library as SINDy, using ridge regularization with coefficient $\alpha = 10^{-4}$, no sparsity thresholding, and refitting every 100 time steps on a sliding window of the most recent 2000 transitions.
- **Neural network (NN) dynamics model:** A feedforward network with input dimension equal to the state–action vector, two hidden layers of 64 units each with ReLU activation, and a 2-dimensional linear output predicting $(\dot{p}, \dot{v})$.
- **Model-free Deep Q-Network (DQN):** A two-hidden-layer network with 64 ReLU units per layer, mapping the oscillator state $x = [p, v]^\top$ to Q-values over a discrete action set of 11 torques uniformly spaced in $[-4, 4]$. The agent uses ϵ-greedy exploration (ϵ decayed multiplicatively by 0.995 per step from 1.0 to 0.05), a replay buffer of size 2000, discount factor $\gamma = 0.95$, and Adam with learning rate 10^{-3}.

All oscillator controllers use the same instantaneous cost $\ell(p, v, u, t)$, the same control bounds, and the same nonstationary parameter schedule, allowing a direct comparison of tracking performance and robustness to drift.

3 Simulation Results

3.1 Pendulum: Stationary Dynamics, Online vs. Offline SINDy-RL

The pendulum experiments evaluate SINDy-RL in a stationary setting under two learning regimes. In the *offline* configuration, a short dataset of noise-free trajectories is first collected using a simple exploration policy. A SINDy model is fit once to this dataset and then held fixed while the receding-horizon planner optimizes a swing-up policy on the learned dynamics. In the *online* configuration, the same planner is used but the SINDy model is updated periodically from noisy streaming observations: after every fixed number of steps, recent transitions are appended to the replay buffer and the sparse regression is refit on a sliding window of the most recent data.

Figure 5 shows the online case. Despite measurement noise, the adaptive SINDy model and controller eventually drive the pendulum angle to the upright target and keep it there with small oscillations. Figure 6 shows the offline case: the controller converges to the opposite stable equilibrium (downward) instead of the desired upright configuration, illustrating how a fixed model identified from limited data can mislead the planner even when the underlying dynamics are time-invariant.

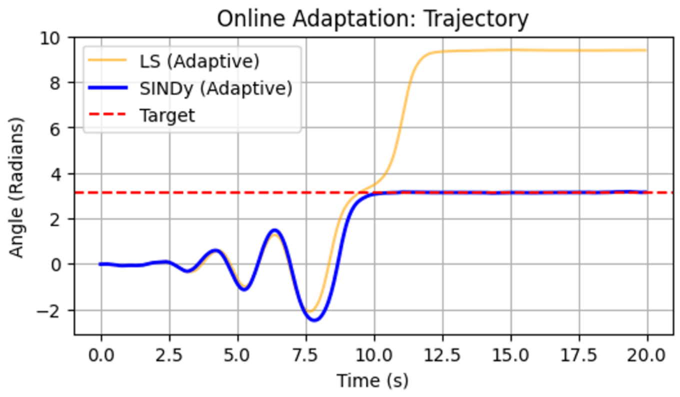

Figure 5: Pendulum, online SINDy-RL with noisy observations: the adaptive SINDy model drives the angle toward and maintains it near the upright target.

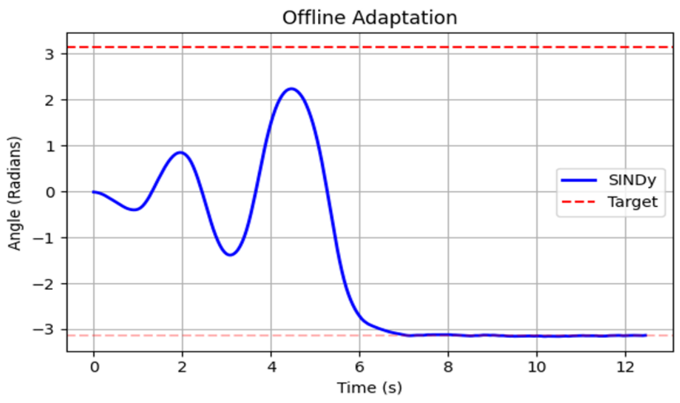

Figure 6: Pendulum, offline SINDy-RL with a fixed model identified from limited data: the controller converges to the wrong (downward) equilibrium.

3.2 Nonstationary Oscillator: SINDy-RL, NN Model, and DQN

The nonstationary oscillator experiments assess robustness to drifting dynamics. Here the stiffness $\omega(t)^2$ and damping $\gamma(t)$ follow the schedule in Figure 3, producing three phases with distinct effective dynamics. Three controllers are compared on this environment:

- **SINDy-RL:** model-based RL with an online SINDy dynamics model refit periodically on recent transitions.
- **NN model-based RL:** the same receding-horizon planner, but using an online neural-network dynamics model trained on the same replay buffer.
- **DQN:** a model-free Deep Q-Network acting directly on the environment without an explicit dynamics model.

Figure 7 shows the resulting position trajectories. Both SINDy-RL and the NN-based controller track the sinusoidal reference closely across all three phases of the parameter schedule, with SINDy-RL additionally yielding interpretable stiffness and damping coefficients. The DQN agent, trained under the assumption of fixed dynamics, exhibits increasing tracking error and large excursions once the parameters drift, indicating substantially lower robustness to nonstationarity.

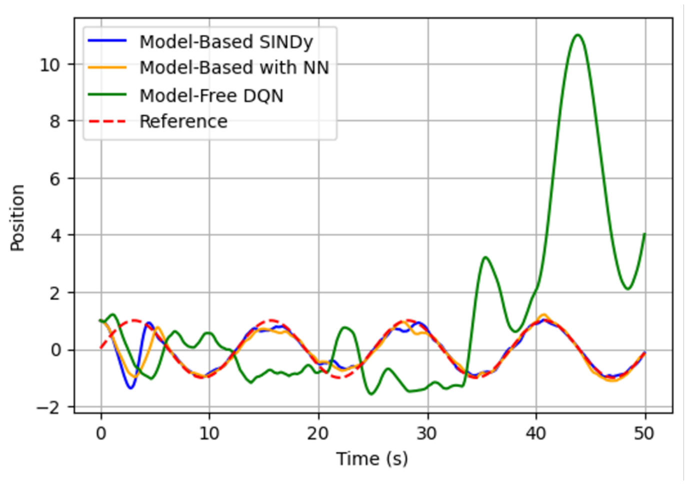

Figure 7: Nonstationary oscillator: tracking performance of SINDy-RL (sparse model-based), NN-based model-based RL, and model-free DQN.

To quantify these observations, Table 1 reports the mean tracking mean-squared error (MSE) over evaluation runs for each controller. SINDy-RL and the NN model-based controller achieve similarly low errors, whereas DQN is roughly two orders of magnitude worse.

Table 1: Nonstationary oscillator: average tracking error for each controller. MSE computed as $\frac{1}{T}\sum_t (p_t - p_{\mathrm{ref},t})^2$ over the full episode.

Method	Mean tracking MSE
SINDy-RL	0.18
NN model-based RL	0.17
DQN	11.15

Note that DQN was trained for the same number of environment steps as the model-based methods (1000 steps); with substantially more training, DQN performance would likely improve, though adapting to nonstationarity would remain challenging for any method that assumes fixed dynamics.

4 Conclusions

SINDy-RL is able to learn compact, interpretable dynamics models from modest interaction data and use them effectively for control in both the stationary pendulum and the nonstationary oscillator. On the pendulum, online adaptation enables successful swing-up even under noisy observations, whereas an offline SINDy model trained on limited data can converge to the wrong equilibrium. On the oscillator, the online SINDy-RL variant tracks drifting stiffness and damping, achieving tracking error comparable to a neural-network model-based controller (MSE of 0.18 vs. 0.17) while providing interpretable coefficient estimates that directly correspond to physical parameters. In contrast, the model-free DQN agent exhibits roughly two orders of magnitude higher tracking error (MSE of 11.15), highlighting the sample-efficiency advantage of model-based approaches in nonstationary settings.

Overall, the results support SINDy as a useful modeling layer for reinforcement learning in dynamical systems when data efficiency, robustness to changing dynamics, and interpretability all matter. Natural extensions include integrating explicit model uncertainty into the planner, applying the approach to higher-dimensional control problems, and combining SINDy with more advanced policy optimization algorithms to leverage the learned models more fully.

References

[1] Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *PNAS*, 113(15), 3932–3937.

[2] Kaiser, E., Kutz, J. N., & Brunton, S. L. (2018). Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proc. Royal Society A*, 474(2219).

[3] Zolman, N., Fasel, U., Kutz, J. N., & Brunton, S. L. (2024). SINDy-RL: Interpretable and efficient model-based reinforcement learning. *arXiv preprint arXiv:2403.09110.*

A Environments

A.1 Pendulum

With unit length and mass, the dynamics is usually written in the following form:

$$\ddot{\theta}(t) = -b\dot{\theta} - g\sin\theta(t) + T$$

Where θ is the position, $\dot{\theta}$ would be the velocity and $\ddot{\theta}$ is the acceleration of the mass.

A.2 Oscillator

The dynamics is usually written in the following form:

$$\ddot{x} = -\frac{k}{m}x - \frac{c}{m}\dot{x} + u$$

Where x is the position (same as p), $\dot{x}$ would be the velocity (same as v) and $\ddot{x}$ is the acceleration (same as $\dot{v}$) of the mass.