# SINDy-PETS: A Flexible Physics-Informed and Probabilistic MBRL Architecture

Brown University

*Authors:*
Yizhong Hu
Mason Lee
Gaweł Kuś
Alan John Varghese

**Abstract**

Model-based reinforcement learning (MBRL) has shown promise in addressing the limitations of traditional model-free reinforcement learning, particularly in terms of sample efficiency and the ability to backpropagate through learned differentiable environments. However, MBRL often falls short in performance due to model bias, which arises from the inability to accurately represent the environment, exploding particle rollouts, and inaccurate state-representations. In this paper, we propose a novel approach to MBRL by combining probabilistic models with physics-based models to capture environmental uncertainty while benefiting from the sample efficiency brought about by a physical bias. We introduce two methods for incorporating physics-informed bias into Probabilistic Ensembles with Trajectory Sampling (PETS): Residual Correction (RC) and Opinion Input (OI). Our approach uses the Sparse Identification of Nonlinear Dynamics (SINDy) algorithm to learn the physics model with a robust backend to inject useful properties such as conservation of mass, energy and momuentum. SINDy-PETS is most useful in data-sparse environments, especially rigid body systems governmed by ODEs / PDEs. We evaluated this algorithm on Cartpole environment and provide preliminary results that show signs of effectiveness, though they remain inconclusive. Further research is needed to determine the full potential of this approach in mitigating model bias and improving MBRL performance in real-world, noisy environments.

## 1 Introduction

Traditional model-free reinforcement learning solely relies on the interaction between the agent and environment to optimize policy, which may not always be practical due to factors such as high interaction cost or potential structural damage [1, 2]. In contrast, model-based reinforcement learning has emerged as a promising alternative where the agent learns an internal representation of the environment to interact with. This paradigm offers two notable benefits over its model-free counterpart: sample efficiency and the ability to backpropagate through the learned differentiable environment during policy optimization [1, 2, 3]. The former is often achieved in dyna-style methods that exploit sample efficiency alone [4, 5], requiring less data than model-based RL. Meanwhile, some approaches also leverage the differentiability of the learned model in the training process [1, 2, 3].

However, model-based approaches can fall short in performance compared to model-free approaches due to model bias, which can lead to compounded error accumulation during roll-out. Model bias arises from the model's inability to accurately represent the environment, stemming from unaccounted phenomena such as the presence of aleatoric uncertainty (noise) in the environment. Moreover, in the case of epistemic uncertainty (lack of knowledge), the model may also suffer from bias due to limited information available for representing the environment.

Recent approaches to mitigating the issue of model bias involves incorporating the uncertainty of the model within the planning process. A seminal work in this domain is PILCO [1], wherein the authors establish a probabilistic model of the environment utilizing a Gaussian process (GP). Additionally, they exploit the model's differentiability, enabling the computation of policy gradients analytically in closed form. Nevertheless, PILCO encounters computational challenges in two specific instances: (1) when state variables are high-dimensional, due to the curse of dimensionality, and (2) in the presence of a vast quantity of data, as GP necessitates inverting a matrix corresponding to the number of data points.

The scalability of PILCO was addressed in Deep PILCO [6], which employs a Bayesian deep neural network to model the dynamics. Despite these advancements, model-based reinforcement learning (RL) has yet to achieve the asymptotic performance of model-free algorithms. Probabilistic ensembles with trajectory sampling (PETS) succeeded in attaining the asymptotic performance of model-free algorithms, primarily by effectively integrating uncertainty into the dynamics model. PETS employs high-capacity neural network (NN) models that incorporate uncertainty through an ensemble of bootstrapped models. Each model encodes distributions rather than point predictions, thus rivaling the performance of model-free methods on standard benchmark control tasks while significantly reducing sample complexity. A noteworthy advantage of PETS over previous probabilistic MBRL algorithms is its ability to distinguish aleatoric and epistemic uncertainty because isolating epistemic uncertainty is highly useful for exploration.

An alternative approach to tackle the issue of model bias is to include strong physics biases to improve the models accuracy and thereby reduce the roll-out error. Recently, a number of works [7, 5] have shown that including strong physics biases in model-based reinforcement learning (MBRL) can lead to state-of-the-art performance with high sample efficiency. However, these algorithms were tested only on idealized, simulated environments. Their application in real environments, which can be noisy, with uncertain dynamics, is not known. Recent work [8] has shown that using neural networks (NN) to model discrepancies caused by systemic physics-based errors can effectively enhance performance. However, deterministic NNs often exhibit significantly poorer performance than probabilistic ensemble methods when used for planning purposes [9]. Additionally, MBRL using physics dynamics models have been limited to environment's where the system of ODE/PDEs are known prior to agent deployment. This forces the creation of highly specific environment configurations, loss functions, and model architectures. A general MBRL agent should be able to enter an unknown environment quickly and learn a dynamics model that extrapolates well for planning. In this work, we aim to address this problem by combining probabilistic models and physics based models to capture the uncertainty of the environment while benefiting from the sample-efficiency brought about by a physical bias.

However, in many practical applications, the physics model is not known a priori. Arora et al. [7] and Ebers et al. [8] address this scenario and develop frameworks that infer physics model using SINDy (Sparse Identification of Nonlinear Dynamics [10]), a symbolic regression method. Arora et al. [7] used SINDy to build a model-based reinforcement learning framework, which achieved the best performance on standard control benchmarks such as Cartpole, inverted pendulum, mountain car, and pendulum swingup. Despite the outstanding performance, the benchmark tasks chosen for evaluation are relatively easy and low-dimensional problems. It remains unclear how well these model-based approaches would perform in real-world noisy environments.

Our research provides a novel approach to model-based reinforcement learning, balancing the flexibility of probabilistic modeling with biases due to the prior knowledge of the physics of the environment.

## 2 Introducing Physics-Informed bias to PETS

By traditional practice of RL, dynamical systems roughly follows a deterministic state transition model:

$$s_{t+1} = T(s_t, a_t) \tag{1}$$

where $\mathbf{s}_t \in \mathbb{R}^{d_s}$ and $\mathbf{a}_t \in \mathbb{R}^{d_a}$ are the states and actions at time $t$ respectively.

However, for physical systems, it is often easier to model a change in state, similar to Euler's Method:

$$s_{t+1} = s_t + f(s_t, a_t) \tag{2}$$

We can predict the change in state instead by defining $\delta_t = s_{t+1} - s_t = f(s_t, a_t)$.

## 2.1 Physics-Informed PETS

PETS models the system as a Markov Decision Process (MDP), and an ensemble of Gaussian neural networks produces next state predictions in a parameterized form of Gaussian distribution:

$$\tilde{\delta}_t \sim \mathcal{N}(\mu_\theta(\mathbf{s}_t, \mathbf{a}_t), \mathbf{\Sigma}_\theta(\mathbf{s}_t, \mathbf{a}_t)). \tag{3}$$

The NN, with parameters $\theta$, gives predictions for the mean and covariance. It is trained with cross-entropy loss:

$$\text{loss}(\theta) = \sum_{t=1}^{N} \left[\mu_\theta(\mathbf{s}_t, \mathbf{a}_t) - \delta_t\right]^T \mathbf{\Sigma}_\theta(\mathbf{s}_t, \mathbf{a}_t) \left[\mu_\theta(\mathbf{s}_t, \mathbf{a}_t) - \delta_t\right] + \log\det \mathbf{\Sigma}_\theta(\mathbf{s}_t, \mathbf{a}_t). \tag{4}$$

Note that we assume different state variables are independent, which gives a diagonal covariance matrix represented by a vector in $\mathbb{R}^{d_s}$.

To introduce physics-bias, we propose two methods: Residual Correction (**RC**) and Opinion Input (**OI**). Assume that we can obtain an approximate physics model $\tilde{f}(\mathbf{s_t}, \mathbf{a_t})$, in the residual correction (**RC**) case, correction from Gaussian NN is combined additively:

$$\mu_\theta(\mathbf{s_t}, \mathbf{a_t}) = \tilde{f}(\mathbf{s_t}, \mathbf{a_t}) + \epsilon_\theta(\mathbf{s_t}, \mathbf{a_t}) \tag{5}$$

where $\epsilon_\theta$ are the corrections provided by the Guassian NN. If $\tilde{f}$ is correct, no correction is needed to be made. The term $\tilde{f}(\mathbf{s}_t, \mathbf{a}_t)$ corresponds to part of the dynamics model learned by the physics-biased approach (SINDY), while $\epsilon_\theta(\mathbf{s}_t, \mathbf{a}_t)$ aims to resolve the uncertainty, and parts of the dynamics that cannot be easily captured by the physics part (e.g. due to wrong assumptions about the model form).

An alternative is Opinion Input (**OI**), where the output of the physics model is not directly used in the next-state prediction, but is used at the discretion of the Gaussian NN:

$$\mu_\theta(\mathbf{s}_t, \mathbf{a}_t, \tilde{f}(\mathbf{s}_t, \mathbf{a}_t) + \mathbf{s}_t) \tag{6}$$

In this case, the Gaussian NN is not required to completely take the "opinion" of the physics model. It can decide how much the model can be trusted based on data collected.

## 2.2 Learning the physics model with SINDy

The physics model $\tilde{f}$ is learned using the SINDy algorithm, which we elaborate in this subsection. SINDy models the system continuously, which is approximated with Euler's Method. We consider $g(\mathbf{s}, \mathbf{a}) = \frac{1}{\Delta t} f(\mathbf{s}, \mathbf{a})$. It governs the evolution of the state vector $\mathbf{s}(t) \in \mathbb{R}^n$, which is affected by a control input $\mathbf{a}(t) \in \mathbb{R}^q$:

$$\frac{d\mathbf{s}(t)}{dt} = g(\mathbf{s}(t), \mathbf{a}(t); \mu). \tag{7}$$

Our objective is to extract the dynamics model $g$, from a sequence of observations of the state vector $\mathbf{S} = [\mathbf{s}(1), \mathbf{s}(2), \cdots, \mathbf{s}(N)]^T \in \mathbb{R}^{N \times d_s}$ and control input $\mathbf{A} = [\mathbf{a}(1), \mathbf{a}(2), \cdots, \mathbf{a}(N)]^T \in \mathbb{R}^{N \times d_a}$ at different time instances. To achieve this goal, we use SINDy which relies on sparse regression over a dictionary of $k$ features which we construct:

$$\Theta(S, A) = \left[\theta_1(S, A), \theta_2(S, A), \ldots, \theta_k(S, A)\right], \tag{8}$$

where $\theta_i(S, A) \in \mathbb{R}^{N \times (d_s + d_a)} \to \mathbb{R}^{N \times m}$ is a candidate function, and the functions $\theta_i(S, A)$ can be any combination of $S$ and $A$. Here the dictionary of candidate functions is constructed by the user and generally includes polynomials and Fourier terms.Using a finite difference method or other means we can compute the time derivative of the state vector $\dot{S} = [\dot{s}(1), \dot{s}(2), \cdots, \dot{s}(N)]^T \in \mathbb{R}^{N \times n}$. SINDy solves the sparse linear regression problem:

$$\dot{S} = \Theta(S, A) \cdot \Xi \tag{9}$$

The physics model, $\tilde{f} = g(s(t), a(t))\Delta t$, is recovered from $\Xi$, which solves the above sparse regression problem. States and actions can be discretized to make predictions for the discrete-time model: $\mathbf{s}_t = \mathbf{s}(t \cdot \Delta t)$, $\mathbf{a}_t = \mathbf{a}(t \cdot \Delta t)$.

## 2.3 Agent and Trajectory Roll-out

Given the dynamics model, the agent makes decisions with Model Predictive Control (MPC). MPC formulates control as an optimization problem, deciding a sequence of actions over a finite time-horizon that maximizes the rewards. Observing $s_t$ at time $t$, to decide action $a_t$, we roll out the trajectory for each Gaussian NN in the ensemble, and maximize the average expected value reward over each trajectory of length $T$:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n}\sum_{p=1}^{P} \mathbb{E}_{\theta_k}\left[\sum_{\tau=t}^{t+T} r(s_\tau^{(p)}, a_\tau)\right] \\ \text{where} \quad & s_{\tau+1}^{(p)} = s_\tau^{(p)} + \tilde{\delta}_\tau^{(p)} \qquad \forall \tau \in t+1 : t+T, p \in 1 : P \\ & \tilde{\delta}_t^{(p)} \sim \mathcal{N}(\mu_{\theta_p}(\mathbf{s}_t, \mathbf{a}_t), \mathbf{\Sigma}_{\theta_p}(\mathbf{s}_t, \mathbf{a}_t)), \end{aligned}$$

where we have an $n$-ensemble of Gaussian NNs, each with parameters $\theta_k$ and a predicted trajectory $s_\tau^{(k)}$ MPC be accomplished with any trajectory optimizer, such as CEM, ICEM, or MPPI.

## 2.4 Algorithm Summary

Physics-Informed model and PETS can be trained concurrently with the following algorithm

**Algorithm 1** Physics-Informed PETS

```
1: Initialize data $\mathbb{D}$ with a random controller for one trial.
2: Pre-train physics-informed model $\tilde{f}$ using synthetic data.
3: if synthetic train then
4:     Create imaginary roll-outs using $\tilde{f}$ and pre-train probabilistic ensemble (PE).
5: end if
6: for Trial k = 1 to K do
7:     Train physics model $\tilde{f}$ given data $\mathbb{D}$.
8:     Train PE $\delta(s_t, a_t) = f(s_t, a_t) + \epsilon_\theta(s_t, a_t)$ dynamics model given $\mathbb{D}$, freeze parameters of $\tilde{f}$.
9:     for Time t = 0 to TaskHorizon do
10:        for Actions sampled $a_{t:t+T} \sim$ Policy Optimizer, 1 to NSamples do
11:            Propagate state particles $s_\tau^{(p)}$ using TS and $\tilde{f}|\mathbb{D}, a_{t:t+T}$
12:            Clip the exploding particle rollouts to 5 STD of state normalization.
13:            Evaluate actions as $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^{P} r(s_\tau^P, a_\tau)$.
14:            Update CEM distribution.
15:        end for
16:        Execute first action $a^t$ (only) from optimal actions $a^{t:t+T}$.
17:        Record outcome $\mathbb{D} \leftarrow \mathbb{D} \cup \{s_t, a_t^*, st+1\}$.
18:    end for
```

Similar to PETS, we can also use Model Predictive Control (MPC) as our policy [9] to maximize the utilization of our learned environment model (see Algorithm 2). Compared to deep RL algorithms that utilize some type of value estimation and implicitly modeling the dynamics of the environment, MPC skips this step and use the environment model for such a value estimation, to directly optimize for the policy.

MPC is a method where the sequence of controls $u_{t:t+N}$ with a finite task horizon $N$ under the rewards and the trajectory $x_{t:t+N}$ constrained by the dynamics model $x_{\tau+1} = f(x_\tau, a_\tau)$ is optimized with respect to the total rewards.

For state and action constraints $\{g_i\}_{i=1:k}$. Only one step is taken with this control, and at the next time step, the same optimization is performed again.

We note that in this schema, our optimization constraints $s_{\tau+1} = f(s_\tau, a_\tau)$ are stochastic constraints, as there is a stochastic element on the transition function. To perform such an optimization, we follow the PETS paper and employ the Cross-Entropy Method (CEM).

By using CEM, we can fully utilize the stochastic transition model. Since the model of the environment does not change when we switch to a different task, it can also be easily adapted to other tasks on the same environment. Additionally, CEM does not require much training to work, and it has reliability from being investigated much outside of the ML context.

## 3 Experiment

We perform our experiments on the Cartpole environment with and without friction [11]. with $m_c = 1.0$, $m_p = 0.1$, $g = 9.8$, $l = 0.5$. The force on the cart is between $-10.0$ and $10.0$. For experiments with friction, the coefficients are $\mu_c = 0.05$. $\mu_p = 0.1$. The cart is given a reward of 1.0 for every time step it reaches the outside of the stabilized state space, which is $\pm 2.4$ for $x$, and $\pm 12^o$ for $\theta$.

We tested with a friction-less cart-pole model and SINDy model as the physics prior. both SINDy and PETS are trained on the data of 1 trial with random actions before it starts acting with MPC controller.

For different ways of combining physics model and PETS, separate trials are performed on PETS only and physics model only for control, and proposed physics-informed methods Residual Correction (RC) and Opinion Input (OI) as well.

The Probabilistic Ensemble has a ensemble size of 5. The optimizer is Adam with a Learning Rate of $10^{-3}$ and weight decay $5 \times 10^{-5}$. The replay buffer has a size of 5000. Every time a trial is collected, the ensemble is trained on the data collected in the replay buffer for 150 epochs. We train for 10 trials

In each experiment, we tested different optimizers with MPC: CEM, iCEM, and MPPI. For each of them, we run 10 iterations on a population of 200. The planning horizon $N = 15$ For CEM and iCEM, $\alpha = 0.1$, elite ratio is 0.1. For iCEM, keep elite fraction is 0.1, population decay factor is 2.0, and colored noise exponent is 2. For MPPI, $\gamma = 0.0$, $\sigma = 1.0$, $\beta = 0.0$.

We also tested different pre-train trial lengths with SINDy: 10, 50, and 100. They are evaluated on a 2000 time-step Cartpole environment, tested on PETS only, SINDy only, RC, and OI. They are also tested on both incorporating (synthesize-train) data generated by SINDy and not incorporating.

## 4 Results

### 4.1 Enhanced Dynamics via SINDy-PETS

SINDy's ability to learn an adequate physics-based dynamics model for the Cartpole environment in fewer than 50 trials outperforms neural network approaches in data-sparse settings. As demonstrated in **??**, SINDy's rollouts exhibit lower error accumulation compared to neural networks. Figure 3 illustrates the superior performance of SINDy in this regard.

We observed that SINDy configuration 1 was particularly susceptible to particle rollouts. To address this issue, we hypothesized that pretraining the neural network with imagined physics-based rollouts under noise would enhance model performance by reducing the likelihood of significant changes resulting from unseen states. We pretrained SINDy on the original replay buffer and extracted a physics model. White noise, within three standard deviations of the current state space norm, was added to the input and output states. The neural network was then trained on 7,000 imagined rollouts for 50 epochs before fine-tuning on the original replay buffer and resuming training.

Our group made an intriguing observation regarding the suboptimal performance of a perfect dynamics model in the Cartpole environment. When using an exactly correct physics equation, we noticed fluctuating rewards that peaked and subsequently plummeted significantly. We attributed this phenomenon to inefficient exploration stemming from our Improved Cross-Entropy Method (ICEM) optimizer, a genetic algorithm that simulates thousands of roll outs across various action sequences, becoming trapped in local optima. Across our results, you will see instances of where the SINDy-PETS model reaches maximums of reward and then experiences random and severe dropoffs in rewards.

Stochastic dynamics, which introduce randomness into the system, can help CEM escape local optima by occasionally taking suboptimal actions. This approach enables exploration of other regions within the state and action space, potentially leading to better solutions. Stochastic dynamics are essential for learning a robust policy that can generalize well to unseen scenarios. Moreover, as CEM is a sample-based optimization method reliant on generating multiple rollouts to approximate the optimal policy, stochastic dynamics contribute to improved sample efficiency by generating diverse trajectories. This diversity enhances the quality of approximations, resulting in more accurate policy updates and faster convergence.

### 4.2 Hyper-Parameter Tuning and Ablation Studies

The performance of the model plays a significant role in the effectiveness of the trajectory optimizer. The trajectory optimizer relies on the model's predictions to generate action sequences, any inaccuracies

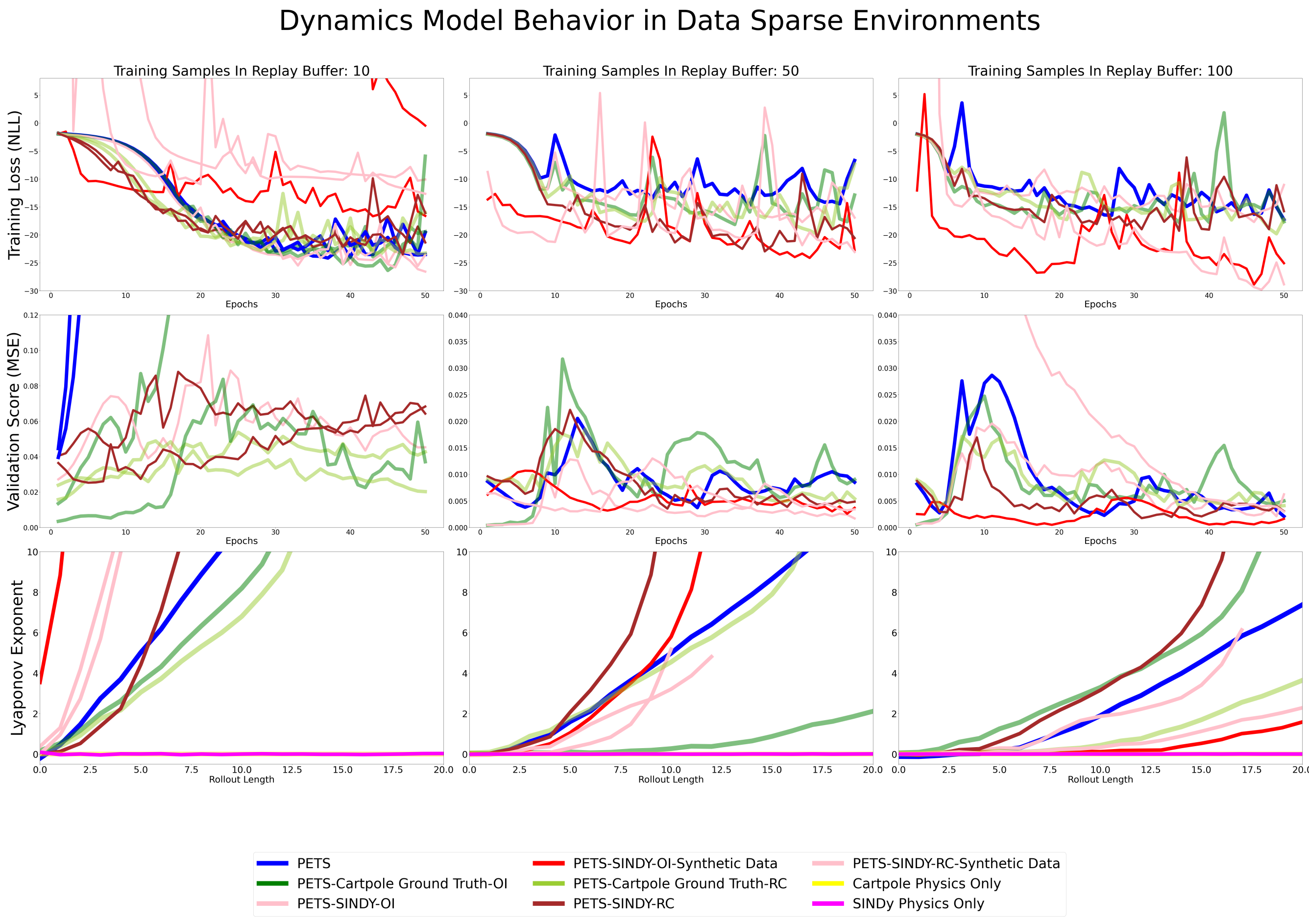

Figure 1: This figure displays the behavior of dynamics models relative to their initial samples, as evaluated by training and validation scores and the Lyapunov exponent. The figure compares the performance of multiple models across different datasets, using different metrics for model evaluation. The subplots are organized into three rows and show the training and validation scores, the Lyapunov exponent, and a legend detailing the models and datasets used.

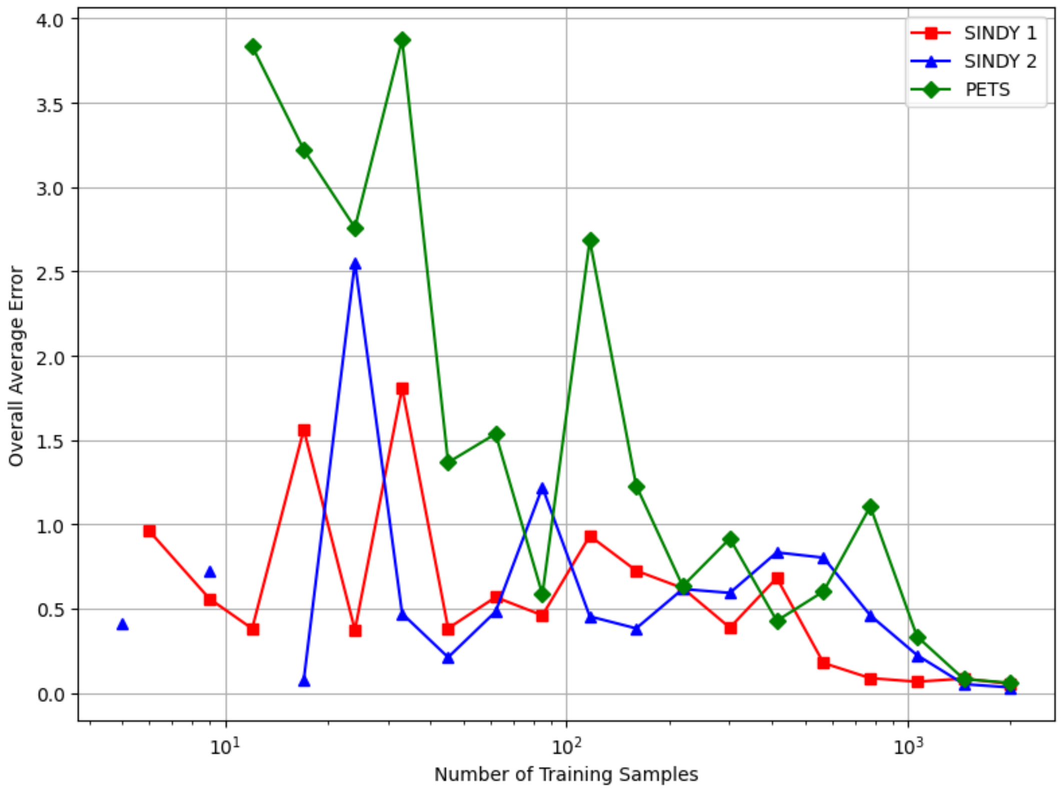

Figure 2: SINDy 1 (Residual Correction), SINDy 2 (Opinion Input), and vanilla PETS overall average error evaluated on a test data set plotted against the against the number of initial sample tests. In data-sparse environments the physics model is capable of learning with lower error. All algorithms converge as the training set exponentially increases

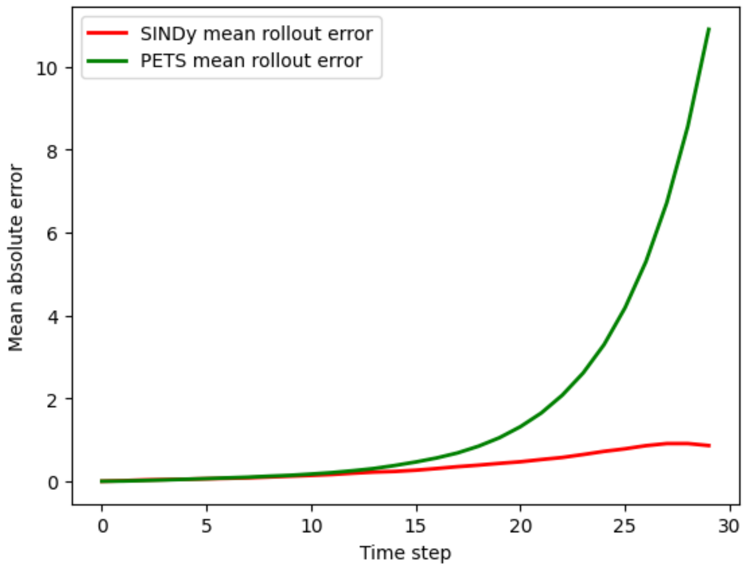

Figure 3: The mean absolute error of SINDy configuration 2 plot against PETS over a complete learning processing of maximum reward in Cart Pole. On average, non-exploding particles simulated with the physics dynamics model are more consistent that the neural network

or uncertainties in the model can have a direct impact on the quality of the planned trajectories. The optimizer, however selects the action sequences that will be applied by the agent.

Balanced performance can be achieved by tuning the hyperparameters. For this purpose we test our models with different optimizers (MPPI, CEM, ICEM), and test different optimizer hyperparameters, namely: population size (50, 100, 500), number of optimizer iterations (5, 10, 20) and agent planning horizon. Additionally, for each configuration the experiments were repeated with 3 random seeds.

Overall, it was concluded that the default hyperparameters provide the most reliable performance. No major differences were noted between the optimizers.

It was observed that SINDy marginally outperforms PETS using smaller planning horizon and smaller population sizes, which indicates on potential savings in terms of the computational costs. Furthermore, it was noted that for longer planning horizons the performance of SINDy deteriorates.

#### 4.2.1 Comparison of Different Configurations

The evidence for physics-informed configuration against PETS only configuration in our experiments are inconclusive. The Opinion Input (OI) model is a superset of PETS only. Even when physics-informed models provide no information, given the state and action, PETS alone has been proven to be able to learn the dynamics efficiently. The few experiments that we have performed can neither prove OI is better than PETS nor similar in performance to PETS.

On the other hand, in Residual Correction (RC), we should see the PETS correction doing something completely different. When only the friction-less model is given, the correction should be zero in the friction-less case, and correct for the friction in the friction case. In this case, we can see that RC is doing much worse than PETS only. It is comparable if not worse than only given the physics model, even when the physics model gives the wrong, friction-less predictions.

This is likely due to the different hyper-parameters needed to train the residual model. The target for the Probabilistic Ensemble is very different in RC, and the original hyper-parameters may have destabilized the training. More experiments is needed to determine the correct hyper-parameters. Due to the nature of this construction, RC should perform at least as well as PETS only.

### 4.3 Learning the correction to wrong model

We conducted a series of experiments to examine the ability of the Residual Correction and Opinion Input configurations to acquire optimal actions, despite employing an inaccurate physics model. Specifically, we used analytic equations to represent the physics model for the cartpole environment. Initially, we executed a test utilizing a flawless physics model that faithfully represented the actual environment, serving as a baseline. In this scenario, both the physics model and the actual environment lacked friction. The reward versus trial plots for this particular test are depicted in Figure 4. Subsequently, we conducted a test wherein we introduced friction solely within the actual environment, while the physics model lacked terms to account for such friction. The outcomes of this test are illustrated in Figure 5. Our analysis of Figure 5 reveals that, solely relying on the physics model, the rewards fail to reach the expected value of 200. Nevertheless, employing the Opinion Input configuration enables the system to grasp the underlying dynamics and rapidly maximize the reward, similar to the performance of the PETS model. Based on these experiments, we can deduce that the Opinion Input model is capable of achieving high performance even in the presence of a slightly inaccurate physics model.

Furthermore, we conducted additional experiments utilizing the SINDy physics model, both with and without the presence of friction. The reward versus trial plots for these two scenarios are presented in Figure 6 and Figure 7, respectively. In these experiments, we observed no discernible distinction between the two cases, as the SINDy model was able to access the data with friction at the conclusion of each trial. Across all these tests, it is apparent that the Opinion Input model exhibits a slight advantage over the PETS model in terms of performance.

### 4.4 Extended rollout lengths

Finally, the models were tested on an extended the trial length. The aim of these experiments was to verify the quality of the model. In the previous experiments, which use trial length of 200 steps, most of the tested configurations yield maximum rewards within the first 2-3 trials. In longer trials, however, the agent is likely to experience more variety of states, which becomes more challenging for the model, and thus requires more versatility.

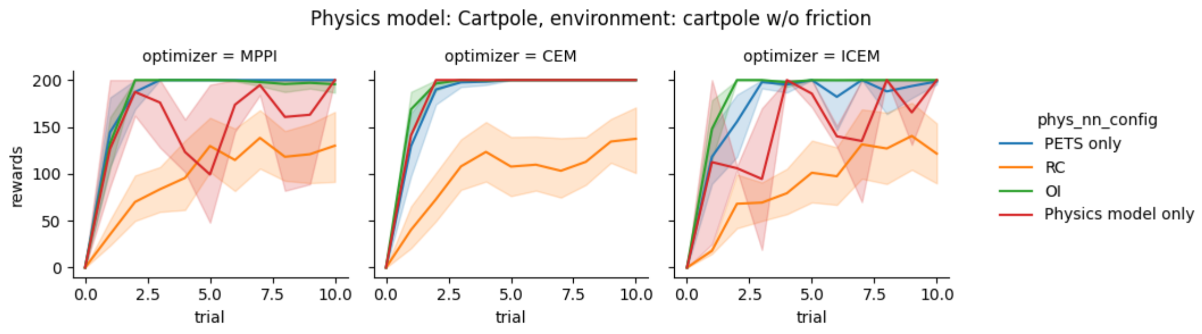

Figure 4: Reward vs trial plot for the different configurations. Here the physics model is an accurate representation of the actual environment. With CEM optimizer, the 'physics model only' configuration performs as good as PETS and Opinion Input. Across the optimizers, we see that Opinion Input marginally outperforms PETS.

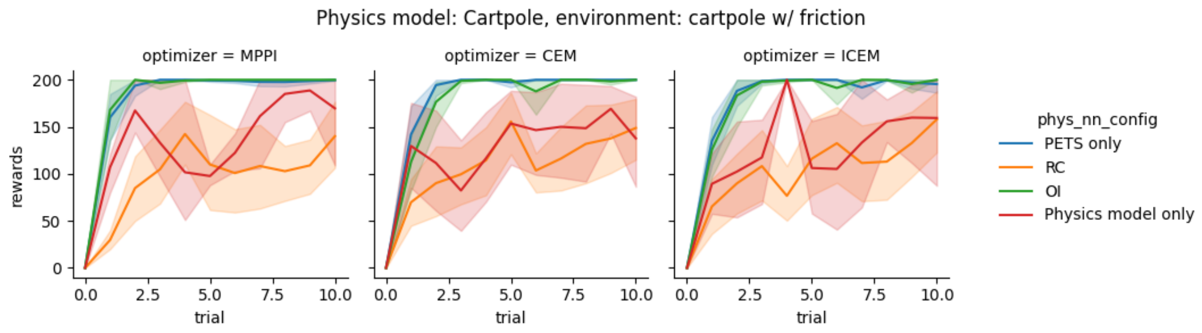

Figure 5: Reward vs trial plot for the different configurations. Here the physics model doesn't include friction and is an inaccurate representation of the actual environment. Despite having an incorrect physics model, the Opinion Input configuration is able to perform as well as PETS.

Specifically in this setting we increase the number of steps in a trial from the default 200 to 2000 steps. We compare the SINDY-PETS in configuration 2, which performed best in previous experiments, against PETS, which provides the state of the art baseline. Furthermore, SINDY-PETS was tested with two different settings: different length of the initial replay buffer (i.e. how many samples are used to pre-train SINDY: 10, 50, and 100) and with/without synthetic training option, which augments the pre-training of SINDY. The augmentation routine takes the SINDY model pretrained on the initial replay buffer, and simulates a buffer which is then used to augment the initial replay buffer dataset.

The results of these experiments are shown in Figure 8. As expected, the models don't max out the rewards, which indicates that the models are not general enough. On average SINDY models marginally outperform PETS in the initial trials. Furthermore, on average, the performance of both models slowly improves in each trial, which is expected as the model is retrained on the replay buffer after each trial.

Applying the synthetic training augmentation routine does not yield any significant change in performance. Interestingly, decreasing the pretraining buffer size, also doesn't lead to any significant loss in performance, which mean that SINDY can be trained on as few as 10 observations, and lead to the same performance. Overall, the differences between PETS and SINDY-PETS observed in these experiments are insufficient to conclude any advantage of one or the other. For that, more extensive experiments experiments are needed.

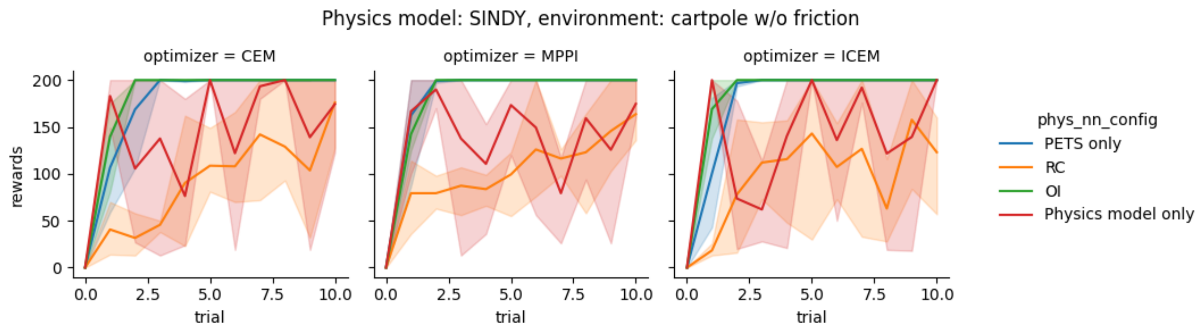

Figure 6: Reward vs trial plot for the different configurations, where the physics model is SINDy and there is no friction in the environment.

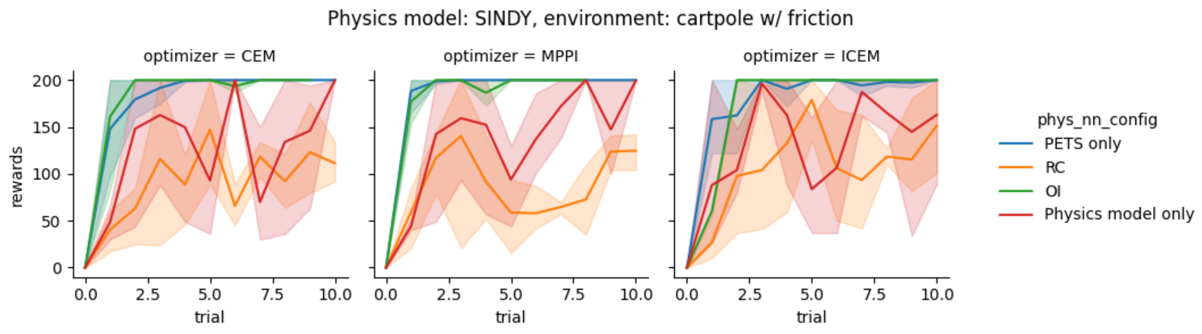

Figure 7: Reward vs trial plot for the different configurations, where the physics model is SINDy and there is friction in the environment.

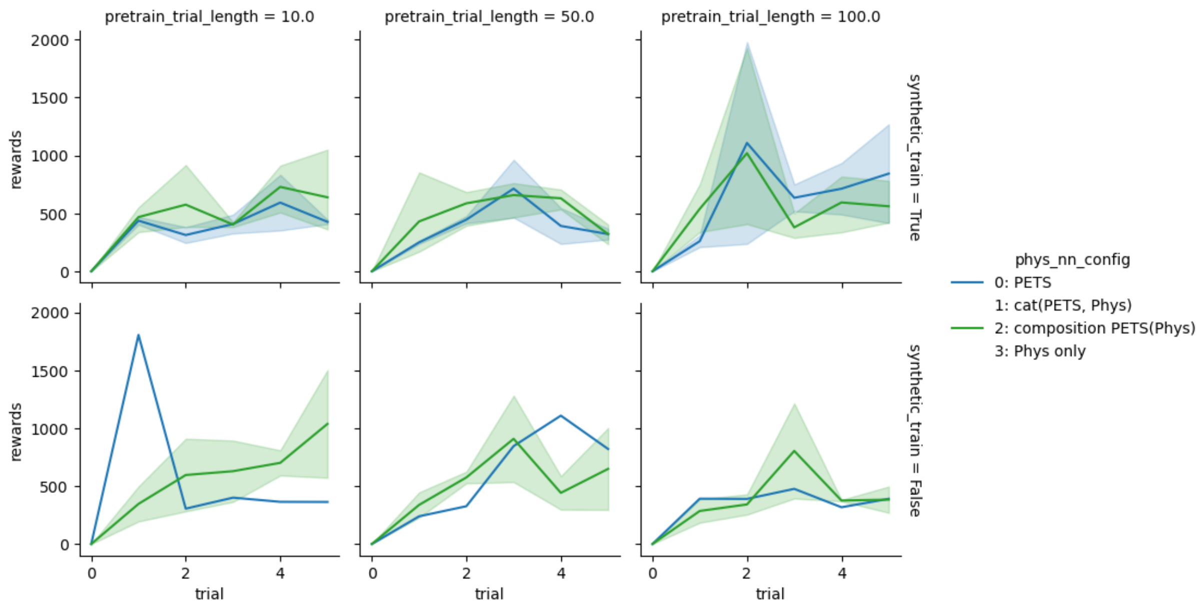

Figure 8: Trained with ICEM optimizer, with default settings (10 iterations, initial population of 200, decay factor=2)

## 5 Discussion

In this study, we have presented a modular physics-based approach for predicting a wide range of rigid body systems. Our approach is built on the flexible and fast-growing pySINDy library and offers several key advantages. Firstly, our model is agnostic to state-space size, enabling the learning of physics models at various state space dimensions. This feature promotes an intuitive understanding of the conservation mechanics and the inherent benefits they confer. Additionally, our approach allows for easy modification of the sparsity of the physics model and differentiation methods during training. The implementation of online learning for physics-based control further enhances the model's ability to learn in situations with control inputs, increasing sample efficiency. Lastly, our method can be extended to discontinuous environments where weak-form PDEs are required.

Using physics-based models has demonstrated increased rollout length in data-sparse environments, enabling better extrapolation into the future. In challenging environments, where a large number of samples is necessary for learning, our physics-informed model-based reinforcement learning (RL) approach outperforms state-of-the-art model-free RL algorithms such as Soft Actor-Critic, as well as highly data-efficient algorithms like PILCO and PETS. This performance improvement is achieved by generating accurate imaginary data.

In chaotic systems, PETs-SINDy exhibits the capability to accurately represent the attractors and stability criteria of a system. Knowledge of general stability is particularly crucial in environments with high inherent sensitivity to initial conditions.

Our work further extends the utility of the pySINDy library by incorporating GPU parallelization, enabling its use in conjunction with PyTorch. The model also allows for noise injection, integration of physics models, and learned physics models, while being compatible with Hydra-configurations.

However, certain challenges arise when using the Cross-Entropy Method (CEM) as an optimizer. Exploding particle rollouts may slow down learning, though pretraining can help mitigate this issue. CEM is also a computationally inefficient process, necessitating tens of thousands of simulations between each iteration.

### 5.1 Future Work

There are several potential avenues for future research that can further enhance our modular physics-based approach:

- Our current framework offers the ability to extract equations and operate within a variety of frameworks, such as Physics-Informed Neural Networks (PINNs), which typically constrain the loss with ODE/PDE residuals. Integrating this extension into our framework may benefit the agent in environments where different sparsity thresholds are required for SINDy regression, enabling the implementation of non-linear dynamics. While SINDy was trained offline after each epoch in our method, and the SINDy model itself was not involved in the backpropagation process during the training of the probabilistic ensemble, future work should explore concurrently training both models to achieve more relevant gradients.

- Extending our approach to Bayesian SINDy, where probabilities over the initial physics model are learned, can aid in generating high-quality imagined rollouts with confidence. This extension may provide a more robust and data-efficient framework for reinforcement learning tasks.

- Investigating the use of physics-based model-based reinforcement learning (MBRL) in partially observed Markov decision processes (POMDPs) remains an open area of exploration. Discovering physics dynamics in partially observed state spaces can be challenging, as vanilla SINDy struggles in high dimensions. However, extensions using autoencoders have demonstrated the ability to extrapolate well. Future work could focus on filtering the state space before entering the model to reduce noise, which may help the model learn the underlying dynamics more effectively.

## References

[1] Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. 2011.

[2] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[3] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[4] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[5] Xin Yang Liu and Jian Xun Wang. Physics-informed Dyna-style model-based deep reinforcement learning for dynamic control. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2255), 2021.

[6] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-efficient machine learning workshop, ICML*, volume 4, page 25, 2016.

[7] Rushiv Arora, Bruno Castro da Silva, and Eliot Moss. Model-based reinforcement learning with sindy. *arXiv preprint arXiv:2208.14501*, 2022.

[8] Megan R. Ebers, Katherine M. Steele, and J. Nathan Kutz. Discrepancy modeling framework: Learning missing physics, modeling systematic residuals, and disambiguating between deterministic and random effects. 2022.

[9] Kurtland Chua, Roberto Calandra, Rowan Mcallister, and Sergey Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. *arxive*, 2018.

[10] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[11] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, 2nd edition, 2018.