Physics-Informed Model and Hybrid Planning for Efficient Dyna-Style Reinforcement Learning

Zakariae El Asri **Olivier Sigaud** **Nicolas Thome**
{zakariae.el_asri, olivier.sigaud, nicolas.thome}@sorbonne-universite.fr
Sorbonne Université, CNRS, ISIR, F-75005 Paris, France

Abstract

Applying reinforcement learning (RL) to real-world applications requires addressing a trade-off between asymptotic performance, sample efficiency, and inference time. In this work, we demonstrate how to address this triple challenge by leveraging partial physical knowledge about the system dynamics. Our approach involves learning a physics-informed model to boost sample efficiency and generating imaginary trajectories from this model to learn a model-free policy and Q-function. Furthermore, we propose a hybrid planning strategy, combining the learned policy and Q-function with the learned model to enhance time efficiency in planning. Through practical demonstrations, we illustrate that our method improves the compromise between sample efficiency, time efficiency, and performance over state-of-the-art methods. Code is available at https://github.com/elasriz/PHIHP/

1 Introduction

Reinforcement learning (RL) has proven successful in sequential decision-making tasks across diverse artificial domains, ranging from games to robotics (Mnih et al., 2015; Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018). However, this success has not yet been evident in real-world applications, where RL is facing many challenges (Dulac-Arnold et al., 2019), especially in terms of sample efficiency and inference time needed to reach a satisfactory performance. A limitation of existing research is that most works address these three challenges – sample efficiency, time efficiency, and performance – individually, whereas we posit that addressing them simultaneously can benefit from useful synergies between the leveraged mechanisms.

Concretely, on one side Model-Free Reinforcement Learning (MFRL) techniques excel at learning a wide range of control tasks (Lillicrap et al., 2016; Fujimoto et al., 2018), but at a high sample cost. On the other side, Model-Based Reinforcement Learning (MBRL) drastically reduces the need for samples by acquiring a representation of the agent-environment interaction (Deisenroth & Rasmussen, 2011; Chua et al., 2018), but requires heavy planning strategies to reach competitive performance, at the cost of inference time.

A recent line of works focuses on combining MBRL and MFRL to benefit from the best of both worlds (Ha & Schmidhuber, 2018; Hafner et al., 2019a; Clavera et al., 2020). Particularly, Byravan et al. (2021); Wang & Ba (2019); Hansen et al. (2022) combine a learned model and a learned policy in planning, this combination helps improve the asymptotic performance but requires more samples, due to the sample cost of learning a good policy.

This paper introduces PhIHP, a **Ph**ysics-**I**nformed model and **H**ybrid **P**lanning method in RL.PhIHP improves the compromise between the three main challenges outlined above – sample efficiency, time efficiency, and performance –, as illustrated in Figure 1. Compared to state-of-the-art MFRL TD3 (Fujimoto et al., 2018) and hybrid TD-MPC (Hansen et al., 2022), we show that PhIHP provides a much better sample efficiency, reaches higher asymptotic performance, and is much faster than TD-MPC at inference.

To achieve this goal, PhIHP first learns a physics-informed model of the environment and uses it to learn an MFRL policy in imagination. This policy is used in a hybrid planning scheme. PhIHP leverages three main mechanisms:

• **Physics-informed model:** We leverage an approximate physical model and combine it with a learned data-driven residual to match the true dynamics. This physical prior boosts the sample efficiency of PhIHP and the learned residual improves asymptotic performance.

• **MFRL in imagination:** we preserve the sample efficiency by training a policy in an actor-critic fashion, using TD3 on trajectories generated from the learned model. The reduced bias in the physics-informed model enables to learn an effective policy in imagination, which is challenging with data-driven models, *e.g.* TD-MPC.

• **Hybrid planning strategy:** We incorporate the learned policy and Q-function in planning with the learned model. A better model and policy learned in imagination improve the performance *vs* inference time trade-off.

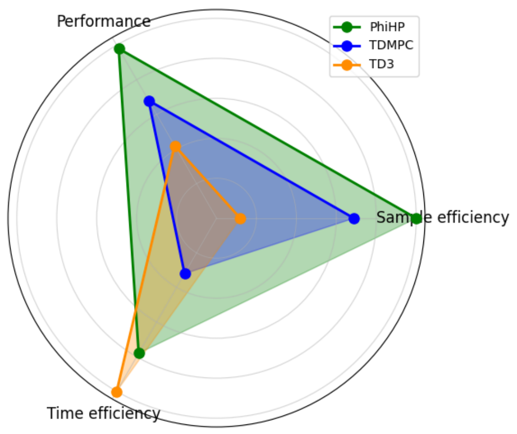


Figure 1: PhIHP includes a Physics-Informed model and hybrid planning for efficient policy learning in RL. PhIHP improves the compromise over state-of-the-art methods, model-free TD3 and hybrid TD-MPC, between sample efficiency, time efficiency, and performance. Results averaged over 6 tasks (Towers et al., 2023).

2 Related work

Our work is at the intersection of Model-based RL, physics-informed methods, and hybrid controllers.

Model-based RL: Since DYNA architectures (Sutton, 1991), model-based RL algorithms are known to be generally more sample-efficient than model-free methods. Planning with inaccurate or biased models can lead to bad performance due to compounding errors, so many works have focused on developing different methods to learn accurate models: PILCO (Deisenroth & Rasmussen, 2011), SVG (Heess et al., 2015), PETS (Chua et al., 2018), PlaNet (Hafner et al., 2019b) and Dreamer (Hafner et al., 2019a; 2020; 2023). Despite the high asymptotic performance achieved by model-based planning, these methods require a large inference time. By contrast, by learning a policy used to sample better actions, we can drastically reduce the inference time.

Physics-informed methods: Recently, a new line of work attempted to leverage the physical knowledge available from the laws of physics governing dynamics, to speed up learning and enhance sample efficiency in MBRL. (Kloss et al., 2017; Ajay et al., 2018; Jeong et al., 2019; Johannink et al., 2019; Zeng et al., 2020; Cranmer et al., 2020; Yin et al., 2021; Yildiz et al., 2021; El Asri et al., 2022; Ramesh & Ravindran, 2023). However, these methods use the learned model in model predictive control (MPC) and suffer from a large inference time. In this work, we efficiently learn an accurate model by jointly correcting the parameters of a physical prior knowledge and learning a data-driven residual using Neural ODEs.

Hybrid controllers: An interesting line of work consists in combining MBRL and MFRL to benefit from the best of both worlds. This combination can be done by using a learned model to generate imaginary samples and augment the training data for a model-free agent (Buckman et al., 2018; Clavera et al., 2020; Morgan et al., 2021; Young et al., 2022). However, the improvement in terms of sample efficiency is limited, since the agent remains trained on real data. Recent hybrid methods enhance the planning process by using a policy (Byravan et al., 2021; Wang & Ba, 2019), or a Q-function (Bhardwaj et al., 2020) with a learned model. More related to our work, TD-MPC (Hansen et al., 2022) combines the last two methods, using a learned policy and a Q-function with a learned data-driven model to evaluate trajectories. TD-MPC jointly trains all components on real samples and learns a latent representation of the world, resulting in improved sample efficiency. However, the need for samples remains significant as they learn a policy from real data. By contrast, we first train a physics-informed model from real samples, and then the policy and the Q-function are trained in imagination. In addition, TD-MPC uses an expensive method to optimize sequences of actions, which impacts inference time. By contrast, accurately learning a policy from the physics-informed model reduces the action optimization budget, thereby enhancing time efficiency.

3 Background

Our work builds on reinforcement learning and the cross-entropy method.

Reinforcement learning: In RL, the problem of solving a given task is formulated as a Markov Decision Process (MDP), that is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, p(s_0))$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{T} =: \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, $\gamma \in [0, 1]$ is a discount factor and ρ_0 is the initial state distribution. The objective in RL is to maximize the expected return $\sum_{t=t_0}^{\infty} \gamma^{t-t_0} r_t$ at each timestep t_0. In model-free RL, an agent learns a policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$ that maximizes this expected return. In contrast, in model-based RL, the agent learns a model that represents the transition function $\mathcal{T}$, then uses this learned model $\hat{\mathcal{T}}_\theta$ to predict the next state $\hat{s}_{t+1} = \hat{\mathcal{T}}_\theta(s_t, a_t)$. The agent maximizes the expected return by optimizing a sequence of actions $A = \{a_{t_0}, ..., a_{t_0+H}\}$ over a horizon H:

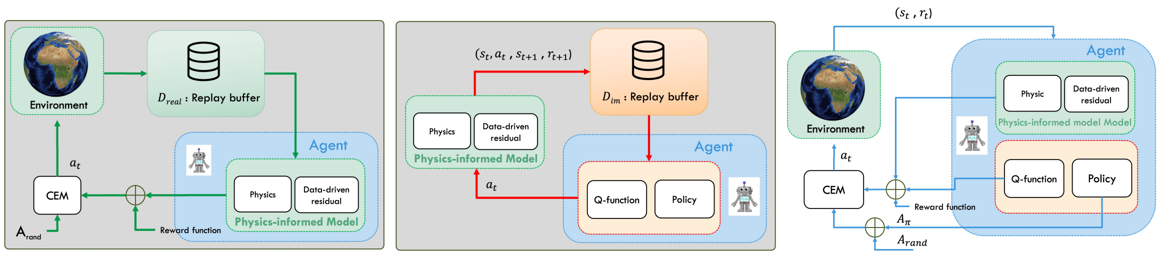
$$A^* = \arg \max_{A \in \mathcal{A}^H} \sum_{t=t_0}^{H} \gamma^{t-t_0} R(s_t, a_t), \quad \text{subject to} \quad s_{t+1} = \hat{\mathcal{T}}_\theta(s_t, a_t). \tag{1}$$

Furthermore, using an inaccurate model can degrade solutions due to compounding errors. So, one often solves this optimization problem at each time step, only executes the first action from the sequence, and plans again at the next time step with updated state information. This is known as model predictive control (MPC).

Cross Entropy Method (CEM): Since the dynamics and the reward functions are generally nonlinear, it is difficult to analytically calculate the exact minimum of (1). In this work, we use the derivative-free Cross-Entropy Method (de Boer et al., 2005) to resolve this optimization problem. In CEM, the agent looks for the best sequence of actions over a finite horizon H. It first generates N candidate sequences of actions from a normal distribution $X \sim \mathcal{N}(\mu, \sigma^2)$. Then, it evaluates the resulting trajectories using the learned dynamics model using a reward model and determines the K elite sequences of actions $(K < N)$, that is the sequences that lead to the highest return. Finally, the normal distribution parameters σ and μ are updated to fit the elites. This process is repeated for a fixed number of iterations. The optimal action sequence is calculated as the mean of the K elites after the last iteration. We call CEM budget the size of the population times the number of iterations, this budget being the main factor of inference time in methods that use the CEM.

4 Physics-Informed model for Hybrid Planning

In this section, we describe PhIHP, our proposed Physics-Informed model for Hybrid Planning. PhIHP first learns a physics-informed residual dynamics model (Sec. 4.1), then learns a MFRL agent through imagination (Sec. 4.2), and uses a hybrid planning strategy at inference (Sec. 4.3). PhIHP follows recent hybrid MBRL/MFRL approaches, *e.g.* TD-MPC Hansen et al. (2022), but the physics-informed model brings important improvements at each stage of the process. It brings a more accurate model, which improves predictive performance and robustness with respect to training data distribution shifts. Crucially, it benefits from the continuous neuralODE method (Sec. 4.1) to accurately predict trajectories, enabling to learn a powerful model-free agent in imagination (Sec. 4.2). Finally, it enables to design a hybrid policy learning (Sec. 4.3) optimizing the performance *vs* time efficiency trade-off.



(a) Learn a physics-informed model (b) Learn an actor/critic offline (c) Behaviour at inference time

Figure 2: **Schematic view of PhIHP.** (a) We iteratively learn a physics-informed model from few interactions in the environment. (b) We learn a policy and Q-function from trajectories imagined with the learned model. (c) The agent samples actions from the policy output and random actions and then evaluates the resulting trajectories using CEM, a reward function, and the Q-function.

4.1 Learning a physics-informed dynamics model

Model-based RL methods aim to learn the transition function $\mathcal{T}$ of the world *i.e.* a mapping from (s_t, a_t) to s_{t+1}. However, learning $\mathcal{T}$ is challenging when s_t and s_{t+1} are similar and actions have a low impact on the output, in particular when the time interval between steps decreases. We address this issue by learning a dynamics function $\hat{\mathcal{T}}_\theta$ to predict the state change Δs_t over the time step duration Δt. The next state s_{t+1} can be subsequently determined through integration with an Ordinary Differential Equation (ODE) solver. Thus, we describe the dynamics as a system following an ODE of the form:

$$\frac{\mathrm{d}\boldsymbol{s}_t}{\mathrm{d}t}\bigg|_{t=t_0} = \hat{\mathcal{T}}_\theta(\boldsymbol{s}_{t_0}, \boldsymbol{a}_{t_0}), \quad \text{and} \quad s_{t+1} \simeq \text{ODESolve}\big(\boldsymbol{s}_t, \boldsymbol{a}_t, \hat{\mathcal{T}}_\theta, t, t+\Delta t\big), \tag{2}$$

where $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$ are the state and action vector for a given time t. We assume the common situation where a partial knowledge of the dynamics is available, generally from the underlying physical laws. The dynamics $\hat{\mathcal{T}}_\theta$ can thus be written as $\hat{\mathcal{T}}_\theta = F^p_{\theta_p} + F^r_{\theta_r}$, where $F^p_{\theta_p}$ is the known analytic approximation of the dynamics and $F^r_{\theta_r}$ is a residual part used to reduce the gap between the model prediction and the real world by learning the complex phenomena that cannot be captured analytically. The physical model $F^p_{\theta_p}$ is described by an ODE and the residual part $F^r_{\theta_r}$ as a neural network with respective parameters θ_p and θ_r. We learn the dynamics model in a supervised manner by optimizing the following objective:

$$\mathcal{L}_{pred}(\theta) = \frac{1}{|\mathcal{D}_{re}|} \sum_{(s_t, a_t, s_{t+1}) \in \mathcal{D}_{re}} \|\boldsymbol{s}_{t+1} - \hat{\boldsymbol{s}}_{t+1}\|_2^2 \quad \text{subject to} \quad \frac{\mathrm{d}\hat{\boldsymbol{s}}_t}{\mathrm{d}t}\bigg|_{t=t'} = (F^p_{\theta_p} + F^r_{\theta_r})(\boldsymbol{s}_{t'}, \boldsymbol{a}_{t'}), \tag{3}$$

on a dataset $\mathcal{D}_{re}$ of real transitions (s_t, a_t, s_{t+1}). As the decomposition $\hat{\mathcal{T}}_\theta = F^p_{\theta_p} + F^r_{\theta_r}$ is not unique, we apply an ℓ_2 constraint over the residual part with a coefficient λ to enforce the model $\hat{\mathcal{T}}_\theta$ to mostly rely on the physical prior. The learning objective becomes $\mathcal{L}_\lambda(\theta) = \mathcal{L}_{pred}(\theta) + \frac{1}{\lambda} \cdot \|F^r_{\theta_r}\|_2$. The coefficient λ is initialized with a value λ_0 and updated at each epoch with $\lambda_{j+1} = \lambda_j + \tau_{ph} \cdot \mathcal{L}_{pred}(\theta)$, where λ_0 and τ_{ph} are fixed hyperparameters.

4.2 Learning a policy and Q-function through imagination

Simply planning with a learned model and CEM is time expensive. MFRL methods are generally more time-efficient during inference time than planning methods, since they use policies that directly map a state to an action. However, learning complex policies requires a large amount of training data which impacts sample efficiency. To maintain sample efficiency, a policy can be learned from synthetic data generated by a model. However, an imperfect model may propagate the bias to the learned policy. In this work, we benefit from the reduced bias in the physics-informed model to generate a sufficiently accurate synthetic dataset $\mathcal{D}_{im}$ to train a parametric policy $\pi_\theta(s_t)$ and a Q-function $Q_\theta(s_t, a_t)$, using the TD3 model-free actor-critic algorithm (Fujimoto et al., 2018).

4.3 Hybrid planning with learned model and policy

PhIHP leverages a hybrid planning method that combines a physics-informed model with a learned policy and Q-function. This combination helps overcome the drawbacks associated with each method when used individually. While using a sub-optimal policy in control tasks significantly affects the asymptotic performance, planning with a learned model has a high computational cost: *i*) the planning horizon must be long enough to capture future rewards and *ii*) the CEM budget must be sufficiently large to converge.

We use the learned policy in PhIHP to guide planning. In practice, a CEM-based planner first samples N_π informative candidates from the learned policy outputs $\hat{\pi}(s_t)$ and complements them with N_{rand} exploratory candidates sampled from a uniform distribution $X \sim \mathcal{N}(\mu, \sigma^2)$. These informative candidates help reduce the population size and accelerate convergence. The planner estimates the resulting trajectories using the learned model and evaluates each trajectory using the immediate reward function up to the MPC horizon and the Q-value beyond that horizon.

By using the Q-value, we can evaluate the trajectories over a considerably reduced planning horizon H and we add the Q-value of the last state to cover the long-term reward. Hence, the optimization problem is written as follows:

$$A^* = \quad \arg \max_{A \in \mathcal{A}^H} \Big(\sum_{t=t_0}^{H} \gamma^{t-t_0} R(s_t, a_t) + \alpha \cdot \gamma^{H-t_0} Q(s_H) \Big), \quad \text{subject to} \quad s_{t+1} = \hat{\mathcal{T}}_\theta(s_t, a_t), \tag{4}$$

where the discounted sum term represents a local solution to the optimization problem, while the Q-value term encodes the long-term reward and α balances the immediate reward over the planning horizon and the Q-value.

5 Experiments

We first compare PhIHP to baselines in terms of performance, sample efficiency, and time efficiency. Then we perform ablations and highlight the generalization capability brought by the physics prior. The robustness of PhIHP to hyper-parameter settings is deferred to Appendix E.

5.1 Experimental setup

Environments: We evaluate our method on 6 ODE-governed environments from the gymnasium classic control suite. These include the continuous versions of 3 basic environments: Pendulum, Cartpole, and Acrobot. Additionally, we consider their swing-up variants, where the initial state is "hanging down" and the goal is to swing up and balance the pole at the upright position, similarly to Yildiz et al. (2021). We opted for this benchmark for its challenging characteristics, including tasks with sparse rewards and early termination.

However, to move closer to methods applicable in a real-world situation, we added to the original environments from the gymnasium suite a friction term which is not present in the analytical model of these environments. Thus, the dynamic of each system is governed by an ODE that can be represented as the combination of two terms: a friction-less component F^p and a friction term F^r. Please refer to Appendix B for additional details.

Evaluation metrics. In all experiments, we use three main metrics to compare methods:
• Asymptotic performance: we report the episodic cumulated reward on each environment.

• Sample efficiency: we define the sample efficiency of a method as the minimal amount of samples required to achieve 90% of its maximum performance.

• Inference time: we report the wall-clock time taken by the agent to select an action at one timestep.

Design choice for PhIHP: We learn the model by combining an approximate ODE describing frictionless motion with a data-driven residual model parameterized as a low-dimension MLP. We use TD3 (Fujimoto et al., 2018) for the model-free component of our method, *i.e.* the policy and Q-function. We found it beneficial to modify the original hyperparameters of TD3 to resolve the friction environments. For planning, we use CEM-based MPC. Please refer to Appendix C for additional details.

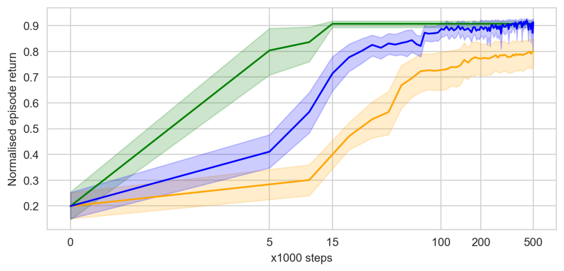
5.2 Comparison to state of the art:

We compare PhIHP to the following state-of-the-art methods:

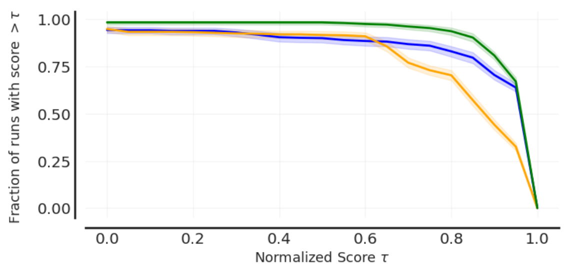
• **TD-MPC** (Hansen et al., 2022), a state-of-the-art hybrid MBRL/MFRL algorithm shown to outperform strong state-based algorithms whether model-based *e.g.* LOOP (Sikchi et al., 2022) and model-free *e.g.* SAC (Haarnoja et al., 2018) on diverse continuous control tasks.

• **TD3** (Fujimoto et al., 2018), a state-of-the-art model-free algorithm. In addition to its popularity and strong performance on continuous control tasks, TD3 is a backbone algorithm for our method to learn the policy and Q-function. We used the same hyperparameters as in PhIHP.

• **CEM-oracle:** a CEM-based controller with the ground-truth model.



(a) Learning curves, the x-axis uses a symlog scale.



(b) Performance profiles.

Figure 3: Comparison of PhIHP *vs* baselines aggregated on 6 control tasks (10 runs). a) PhIHP shows excellent sample efficiency and better asymptotic performance. b) Performance profiles are obtained with rliable (Agarwal et al., 2021). PhIHP shows better performance profiles which indicates a better robustness to outliers. Comparison on individual environments are shown in Appendix D.

	Asymptotic performance				**Sample efficiency ($\times 10^3$ samples)**			**Inference time (in milliseconds)**			
	PhIHP	TD-MPC	TD3	CEM-oracle	PhIHP	TD-MPC	TD3	PhIHP	TD-MPC	TD3	CEM-oracle
Pendulum	-263 ±144	-276 ±301	-229 ±155	-228 ±71	**2**±0	26±24	86±40	6.32±0.02	39.56±0.28	**0.11**±0.0	18.89±0.26
Pendulum-sw	**-356** ±13	**-395**±324	-368±14	-597 ±6	**5**±0	28±12	57±16	6.37±0.01	39.6±0.54	**0.11**±0.0	18.87±0.05
CartPole	**500**±0	432 ±129	464±80	453±24	**5**±0	23±10	108±27	7.43±0.02	39.3±0.07	**0.11**±0.0	33.22±0.03
CartPole-sw	453 ±8	**460** ±4	354 ±113	446±5	**5**±0	76±27	27±10	10.13±0.06	39.36±0.05	**0.11**±0.0	33.73±0.03
Acrobot	**-138** ±122	-249±168	-237±183	-500±0	**5**±0	10±5	233±110	11.14±0.02	39.38±0.09	**0.12**±0.0	59.83±0.1
Acrobot-sw	**371** ±52	**373** ±127	119 ±71	349 ±5	**15**±0	135±123	500±0	9.12±0.02	39.39±0.06	**0.12**±0.0	58.50±0.27

Table 1: Return of PhIHP and baselines on 6 classic control tasks. Mean and std. over 10 runs

In Tab. 1, Figure 4 and Figure 3a, we show that PhIHP outperforms the baselines with a large margin in at least one of the metrics without being worse on the others. Specifically, PhIHP is far more sample efficient than TD3 and it generally shows 5-15 times better sample efficiency than TD-MPC, except on Acrobot where they are comparable. Figure 3a further illustrates this excellent sample efficiency of PhIHP and how TD3 stacks on sub-optimal performance. This enhanced sample efficiency of PhIHP results from training the model-free policy on imaginary trajectories generated by the learned model, as opposed to using real samples in the baselines. Besides, PhIHP demonstrates superior performance in sparse-reward early-termination environment tasks (Cartpole and Acrobot) compared to TD-MPC, and PhIHP outperforms TD3 with a large margin in Cartpole-swingup,

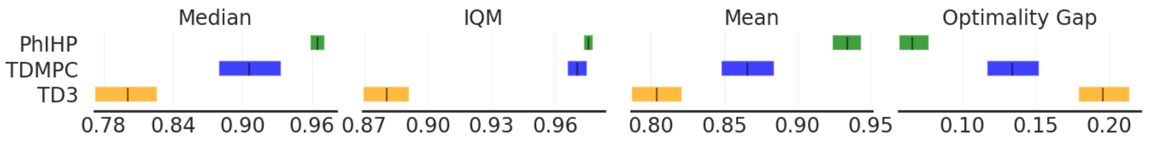


Figure 4: Agregated median, interquartile median (IQM), mean performance, and optimality gap of PhIHP and baselines on 6 tasks (10 runs). Higher mean, median, and IQM performance and lower optimality gaps are better. Confidence intervals are estimated using the percentile bootstrap with stratified sampling (Agarwal et al., 2021). PhIHP outperforms baselines in all metrics.

Acrobot, and Acrobot-swingup. Figure 4 in Appendix D.1 shows how TD3 stacks on lower asymptotic performance for the aforementioned tasks. It also shows that TD-MPC performance drops in sparse-reward early-termination environments *e.g.* Cartpole and Acrobot. It also illustrates that, since CEM-oracle uses the reward function to evaluate trajectories within a limited horizon, it manages to solve both tasks with smooth reward functions, and tasks with sparse reward where the goal is to maintain an initial state (*i.e.* Cartpole), but it fails to solve sparse reward problems where the goal is to reach a position out of the planning horizon (*i.e.* Acrobot).

Finally, Figure 3b shows that PhIHP has better performance profiles compared to baselines which indicates better robustness to outliers in PhIHP.

Tab. 1 also reports the time needed for planning at each time step, obtained with an Apple M1 CPU with 8 cores. It is noteworthy that PhIHP significantly reduces the inference time when compared to TD-MPC. The inference time is still larger than that of TD3 since the latter is a component of our method, but it meets the real-time requirements of various robotics applications.

5.3 Ablation study

In this section, we study the impact of each PhIHP component to illustrate the benefits of using an analytical physics model, imagination learning, and combining CEM with a model-free policy and Q-function for planning. To illustrate this, we compare PhIHP to several methods:

• **TD-MPC*:** our method without physical prior and without imagination. It is similar to TD-MPC since the model is data-driven and it is learned with the policy from real trajectories. But learning the model and the policy are separated.
• **Ph-TD-MPC*:** our method without learning in imagination, thus a physics-informed TD-MPC*.
• **dd-CEM:** our method without physical prior nor policy component, thus a CEM with a data-driven model learned from real trajectories.
• **Ph-CEM:** our method without the policy component, thus a simple CEM with a physics-informed model learned from real trajectories.

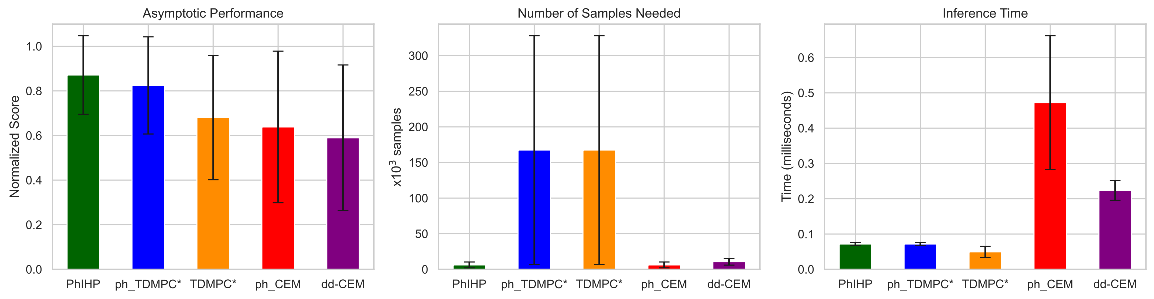


Figure 5: Comparison of PhIHP and its variants on the 3 main metrics. The figures illustrate the aggregated results of running all algorithms on 6 classic control tasks. Histograms and bars represent mean and std. over 10 runs.

Figure 5 shows the impact of the quality of the model on the final performance in MBRL. Precisely, leveraging a physical prior in Ph-CEM and Ph-TD-MPC* shows improvements compared to full data-driven methods, i.e. dd-CEM and TD-MPC*. We also illustrate that planning with a model, a Q-function, and a policy leads to better performance compared to planning only with the model. For instance Ph-TD-MPC* outperforms Ph-CEM and TD-MPC* outperforms dd-CEM. However, this gain in performance comes with a significant cost in samples, because the agent needs a large amount of data to learn a good policy and Q-function.
Figure 5 illustrates the trade-off between asymptotic performance, sample efficiency, and inference time in RL. On one hand, methods that learn a model and directly plan with it (*e.g.* dd-CEM and ph-CEM) do not need many samples to achieve sufficiently good performance, but they are too expensive at inference time. On the other hand, methods that learn to plan with a model, Q-function, and policy plan fast but require many samples to train their policies and Q-functions. PhIHP is the only method that achieves good asymptotic performance with low cost in sample efficiency due to learning in imagination and a good inference time due to hybrid planning.

5.4 Generalization benefits of the physics prior

In this section, we highlight the key role of incorporating physical knowledge into PhIHP in finding the better compromise between asymptotic performance, sample efficiency, and time efficiency illustrated in Figure 5. Actually, learning a policy and Q-function through imagination leads to

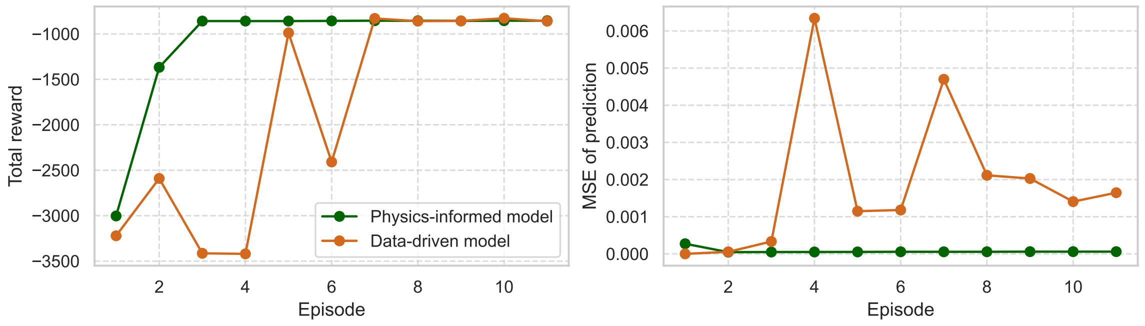


Figure 6: A data-driven model still poorly predicts the next states even when its asymptotic performance matches that of the physics-informed model. Figure obtained with 10 episodes of model training on Pendulum swingup.

superior performance only when the model used to generate samples is accurate enough. Figure 5 in Appendix D.3 shows that an agent trained on imaginary trajectories generated with a physics-informed model largely outperforms the same agent using a fully data-driven model and matches the performance of TD3 which is trained on real trajectories. This highlights the capability of the physics-informed model to immediately generalize to unseen data, in contrast to the data-driven model, which poorly predicts trajectories in unseen states. Figure 6 illustrates this faster generalization capability, showing that the agent with a data-driven model still poorly predicts trajectories even when it meets the asymptotic performance of the agent with the physics-informed model.

6 Conclusion

We have introduced PhiHP, a novel approach that leverages physics knowledge of system dynamics to address the trade-off between asymptotic performance, sample efficiency, and time efficiency in RL. PhIHP enhances the sample efficiency by learning a physics-informed model that serves to train a model-free agent through imagination and uses a hybrid planning strategy to improve the inference time and the asymptotic performance. In the future, we envision to apply PhIHP to more challenging control tasks where there is a larger discrepancy between the known equations and the real dynamics of the system.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Anurag Ajay, Jiajun Wu, Nima Fazeli, Maria Bauza, Leslie P Kaelbling, Joshua B Tenenbaum, and Alberto Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3066–3073. IEEE, 2018.

Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, pp. 834–846, 1983.

Mohak Bhardwaj, Sanjiban Choudhury, and Byron Boots. Blending mpc & value function approximation for efficient reinforcement learning. *arXiv preprint arXiv:2012.05909*, 2020.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *Advances in neural information processing systems*, 31, 2018.

Arunkumar Byravan, Leonard Hasenclever, Piotr Trochim, Mehdi Mirza, Alessandro Davide Ialongo, Yuval Tassa, Jost Tobias Springenberg, Abbas Abdolmaleki, Nicolas Heess, Josh Merel, et al. Evaluating model-based planning and planner amortization for continuous control. *arXiv preprint arXiv:2110.03363*, 2021.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Ignasi Clavera, Violet Fu, and Pieter Abbeel. Model-augmented actor-critic: Backpropagating through paths. *arXiv preprint arXiv:2005.08068*, 2020.

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

P. T. de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134:19–67, 2005.

Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *ICML*, 2011.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Zakariae El Asri, Clément Rambour, Vincent Le Guen, and Nicolas THOME. Residual model-based reinforcement learning for physical dynamics. In *3rd Offline RL Workshop: Offline RL as a"Launchpad"*, 2022.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

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

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b.

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

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *ICML*, 2022.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. *Advances in neural information processing systems*, 28, 2015.

Rae Jeong, Jackie Kay, Francesco Romano, Thomas Lampe, Tom Rothorl, Abbas Abdolmaleki, Tom Erez, Yuval Tassa, and Francesco Nori. Modelling generalized forces with reinforcement learning for sim-to-real transfer. *arXiv preprint arXiv:1910.09471*, 2019.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029. IEEE, 2019.

Alina Kloss, Stefan Schaal, and Jeannette Bohg. Combining learned and analytical models for predicting action effects. *arXiv preprint arXiv:1710.04102*, 11, 2017.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Andrew S Morgan, Daljeet Nandha, Georgia Chalvatzaki, Carlo D'Eramo, Aaron M Dollar, and Jan Peters. Model predictive actor-critic: Accelerating robot skill acquisition with deep reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6672–6678. IEEE, 2021.

Adithya Ramesh and Balaraman Ravindran. Physics-informed model-based reinforcement learning. In *Learning for Dynamics and Control Conference*, pp. 26–37. PMLR, 2023.

Harshit Sikchi, Wenxuan Zhou, and David Held. Learning off-policy with online planning. In *Conference on Robot Learning*, pp. 1622–1633. PMLR, 2022.

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

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL https://zenodo.org/record/8127025.

Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. *arXiv preprint arXiv:1906.08649*, 2019.

Chris Xie, Sachin Patil, Teodor Moldovan, Sergey Levine, and Pieter Abbeel. Model-based reinforcement learning with parametrized physical models and optimism-driven exploration. In *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 504–511. IEEE, 2016.

Cagatay Yildiz, Markus Heinonen, and Harri Lähdesmäki. Continuous-time model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 12009–12018. PMLR, 2021.

Yuan Yin, Vincent Le Guen, Jérémie Dona, Emmanuel de Bézenac, Ibrahim Ayed, Nicolas Thome, and Patrick Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124012, 2021.

Kenny Young, Aditya Ramesh, Louis Kirsch, and Jürgen Schmidhuber. The benefits of model-based generalization in reinforcement learning. *arXiv preprint arXiv:2211.02222*, 2022.

Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4): 1307–1319, 2020.

A Comparison to existing methods

In this section, we present a conceptual comparison of PhIHP and existing RL methods. Figure 7 illustrates the general scheme of existing RL methods and the possible connections between learning and planning. We highlight in Figure 8 the origin of the well-known drawbacks in RL: *i*) learning a policy on real data (arrow 1) impacts the sample efficiency, *ii*) learning a policy from a data-driven learned model (arrow 3) impacts the asymptotic performance due to the bias in the learned model, *iii*) model-based planning (arrow 4) impacts the inference time.

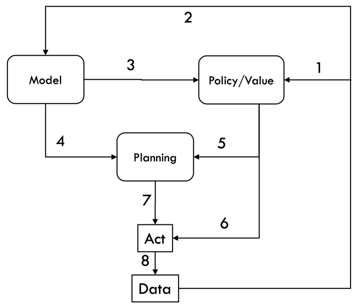
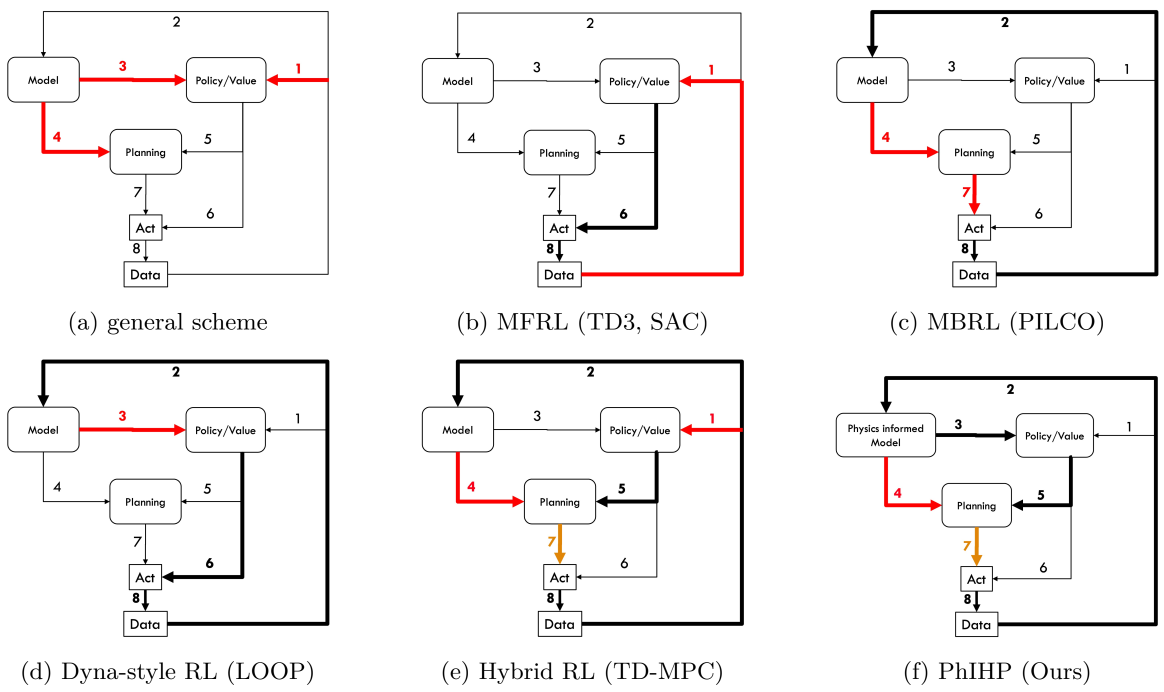


Figure 7: Overview of existing scheme of learning/planning in RL. 1- learn a policy/value function from real data. 2- learn a model from real data. 3- learn a policy/value function from imaginary data. 4- plan with a learned model. 5- plan with a learned policy/value function. 6- act based on a policy output. 7- act based on the planning outcome. 8- collect data from the interaction with the real world.



(a) general scheme
(b) MFRL (TD3, SAC)
(c) MBRL (PILCO)
(d) Dyna-style RL (LOOP)
(e) Hybrid RL (TD-MPC)
(f) PhIHP (Ours)

Figure 8: Conceptual comparison of PhIHP and existing methods based on the general scheme in Figure 7. Thick lines are used by a method, red lines indicate the origin of the main drawbacks: 1- learning on real data impacts **the sample efficiency**, 3- bias introduced by the data-driven model impacts **the asymptotic performance**, 4- planning with a model impacts the **inference time**.

PhIHP benefits from the good sample efficiency of model-based learning methods (arrow 2) and from the physical knowledge to reduce the bias in the learned model. The accurately learned model generates good trajectories to train the policy/value networks (arrow 3). When interacting with the environment, PhIHP uses a hybrid planning strategy (arrows 4 & 5) to improve asymptotic performance and time efficiency.

B Environments

In this section, we give a comprehensive description of the environments employed in our work. Across all environments, observations are continuous within $[-S_{box}, S_{box}]$ and actions are continuous and restricted to a $[-a_{max}, a_{max}]$ range. An overview of all tasks is depicted in Figure 9 and specific parameters are outlined in Table 2.

Pendulum: A single-linked pendulum is fixed on one end, with an actuator on the joint. The pendulum starts at a random position and the goal is to swing it up and balance it at the upright position. Let θ be the joint angle at time t and $\dot{\theta}$ its velocity, the observation at time t is $(\theta, \dot{\theta})$.

Pendulum-Swingup: the version of Pendulum where it is started at the "hanging down" position.

Cartpole: A pole is attached by an unactuated joint to a cart, which moves along a horizontal track. The pole is started upright on the cart and the goal is to balance the pole by applying forces in the left and right direction on the cart.

Cartpole-Swingup: the version of Cartpole where the pole is started at the "hanging down" position.

Acrobot: A pendulum with two links connected linearly to form a chain, with one end of the chain fixed. Only the joint between the two links is actuated. The goal is to apply torques on the actuated joint to swing the free end of the linear chain above a given height.

Acrobot-Swingup: For the swingup task, we experiment with the fully actuated version of the Acrobot similarly to (Yildiz et al., 2021; Xie et al., 2016). Initially, both links point downwards at the "hanging down" position. The goal is to swing up the Acrobot and balance it in the upright position. Let θ_1 be the joint angles of the first fixed to a hinge at time t and θ_2 the relative angle between the two links at time t. The observation at time t is $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$.

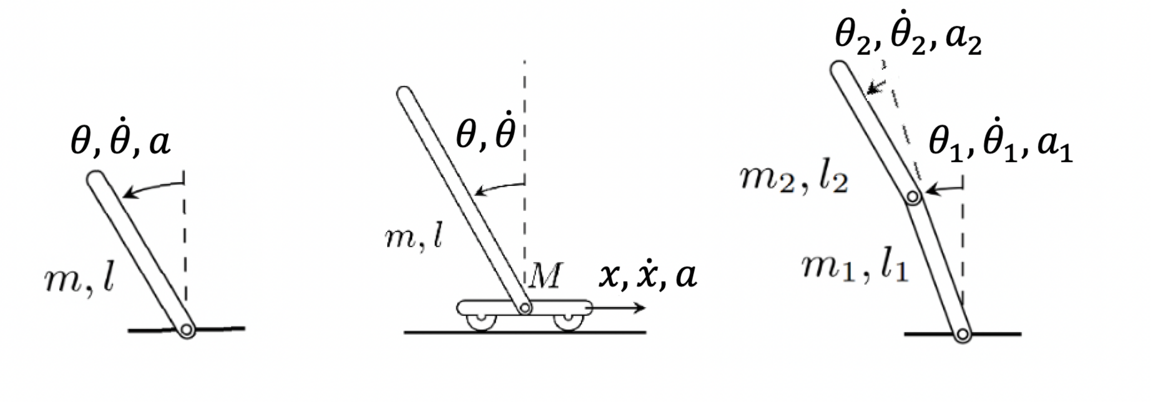

$\theta, \dot{\theta}, a$
m, l
$\theta, \dot{\theta}$
m, l
M
$x, \dot{x}, a$
$\theta_2, \dot{\theta}_2, a_2$
$\theta_1, \dot{\theta}_1, a_1$
m_2, l_2
m_1, l_1

Figure 9: Experimental tasks : Pendulum & Pendulum-swingup (left), Cartpole & Cartpole-swingup (center), Acrobot & Acrobot-swingup(right). The Acrobot-swingup is fully actuated while Acrobot is only actuated at the joint between the two links, thus $a_2 = 0$.

	Environments					
Parameters	Pendulum	Pendulum-SU	Cartpole	Cartpole-SU	Acrobot	Acrobot-SU
Reward type	Smooth	Smooth	Sparse	Smooth	Sparse	Smooth
Early termination	No	No	Yes	No	Yes	No
State space	$\mathbb{R}^2$		$\mathbb{R}^4$		$\mathbb{R}^4$	
States	$[\theta, \dot{\theta}]$		$[x, \dot{x}, \theta, \dot{\theta}]$		$[\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]$	
Observation space	$\mathbb{R}^3$		$\mathbb{R}^5$		$\mathbb{R}^6$	
Observations	$[cos(\theta), sin(\theta), \dot{\theta}]$		$[x, \dot{x}, cos(\theta), sin(\theta), \dot{\theta}]$		$[cos(\theta_1), sin(\theta_1), cos(\theta_2), sin(\theta_2), \dot{\theta}_1, \dot{\theta}_2]$	
Actions space	$\mathbb{R}^1$	$\mathbb{R}^1$	$\mathbb{R}^1$	$\mathbb{R}^1$	$\mathbb{R}^1$	$\mathbb{R}^2$
a_{max}	[2.0]	[2.0]	[10.0]	[10.0]	[1.0]	$[1.0, 1.0]$
Length of the rollout	200	500	500	500	500	500
Δt	0.05		0.02		0.2	

Table 2: Environment specifications

B.1 Dynamic functions

In this section, we provide details of the dynamic functions. For each task, the dynamic function consists of a frictionless component and a friction term.

Pendulum and Pendulum Swingup: Let $s_t = (\theta, \dot{\theta})$ be the state and a_t the action at time t. The dynamic of the pendulum is described as:

$$F(\boldsymbol{s}_t, a_t) = \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ C_g \cdot sin(\theta) + C_i \cdot a_t + C_{Fr} \cdot \dot{\theta} \end{bmatrix} \tag{5}$$

where C_g is the gravity norm, C_i is the inertia norm and C_{Fr} is the Friction norm.

Acrobot and Acrobot Swingup: Let $s_t = (\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$ be the state and $a_t = (a_1, a_2)$ ($a_1 = 0$ for the Acrobot environment) the action at time t. The dynamic of the system is similar to (Yildiz et al., 2021) described as:

$$F(\boldsymbol{s}_t, a_t) = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \frac{-(\alpha_0 + d_2 + \ddot{\theta}_2 + \Sigma 1)}{d_1} \\ \frac{\alpha_1 + \frac{d_2}{d_1} \cdot \Sigma_1 - m_2 \times l_1 \cdot lc_2 \times \dot{\theta}_1^2 \cdot \sin\theta_2 - \Sigma_2}{m2 \cdot {lc_2}^2 + I_2 - \frac{{d_2}^2}{d_1}} \end{bmatrix} \tag{6}$$

where:

$\alpha_0 = a_1 - C_{fr1} \cdot \dot{\theta}_1$ such as C_{fr1} is the friction norm in the first joint ,
$\alpha_1 = a_2 - C_{fr2} \cdot \dot{\theta}_2$ such as C_{fr2} is the friction norm in the second joint ,
m_1 and m_2 the mass of the first and second links,
l_1 and l_2 the length of the first and second links,
lc_1 and lc_2 the position of the center of mass of the first and second links,
I_1 and I_2 the moment of inertia of the first and second links,

and

$$d_1 = m_1 \cdot {lc_1}^2 + m_2 \cdot ({l_1}^2 + {lc_2}^2 + 2 \cdot l_1 \cdot lc_2 \cdot \cos(\theta_2)) + I_1 + I_2$$

$$d_2 = m_2 \cdot ({lc_2}^2 + l_1 \cdot lc_2 \cdot \cos(\theta_2)) + I_2$$

$$\Sigma_2 = m_2 \cdot lc_2 \cdot g \cdot \cos(\theta_1 + \theta_2 - \tfrac{\pi}{2})$$

$$\Sigma_1 = m_2 \cdot l_1 \cdot lc_2 \cdot \ddot{\theta}_2 \cdot \sin(\theta_2) \cdot (\ddot{\theta}_2 - 2 \cdot \ddot{\theta}_1) + (m_1 \cdot lc_1 + m_2 \cdot l_1) \cdot g \cdot \cos(\theta_1 - \tfrac{\pi}{2}) + \Sigma_2.$$

Cartpole and Cartpole Swingup: Let $s_t = (\boldsymbol{x}, \dot{x}, \theta, \dot{\theta})$ be the state and a_t the action at time t. The dynamic of the system is based on (Barto et al., 1983) and described as:

$$F(\boldsymbol{s}_t, a_t) = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \Sigma - m_p \cdot l \cdot \ddot{\theta} \cdot \frac{\cos(\theta)}{m_{total}} \\ \dot{\theta} \\ \frac{g \cdot \sin(\theta) - (\cos(\theta) \cdot \Sigma) - \frac{Fr_p \dot{\theta}}{m_p \cdot l}}{l \cdot [\frac{4}{3} - \frac{m_p \cdot \cos(\theta)^2}{m_{total}}]} \end{bmatrix}, \tag{7}$$

where:

Fr_c is the friction norm in the contact between the cart and the ground,
Fr_p is the friction norm in the joint between the cart and the pole,
l is the length of the pole,
$m_{tot} = m_c + m_p$ and m_p, m_c are the mass of the pole and the cart respectively,
$\Sigma = \frac{1}{m_{total}} \cdot (a + m_p \cdot l \cdot \dot{\theta}^2 \cdot \sin(\theta) - (Fr_c \cdot \mathrm{sgn}(\dot{x}))$.

B.2 Reward Functions

The reward function encodes the desired task. We adopt the original reward functions in the three main environments. For the swingup variants, we choose functions that describe the swingup task: we adopt the same function as Pendulum for Pendulum swingup. For Cartpole swingup, we set a reward function as the negative distance from the goal position $s_{goal} = (x = 0, y = 1)$. For Acrobot swingup, we take the height of the pole as a reward function.

Environment	Reward function
Pendulum	$-\theta^2 - 0.1 \cdot \dot{\theta}^2 - 0.001 \cdot a^2$
Pendulum swingup	$-\theta^2 - 0.1 \cdot \dot{\theta}^2 - 0.001 \cdot a^2$
Cartpole	+1 for every step until termination
Cartpole swingup	$\exp(\lVert \boldsymbol{s} - s_{goal} \rVert_2^2)$
Acrobot	-1 for every step until termination
Acrobot swingup	$-\cos(\theta_1) - \cos(\theta_1 + \theta_2)$

Table 3: Reward functions for each environment.

C Implementation details

In this section, we describe the experimental setup and the implementation details of PhIHP. We first learn a physics-informed residual dynamics model, then learn an MFRL agent through imagination, and use a hybrid planning strategy at inference.
To learn the model, we first use a pure exploratory policy during T timesteps to collect the initial samples to fill $\mathcal{D}_{re}$, then we perform stochastic gradient descent on the loss function (Eq. 3 in Sec. 4.1) to train F_θ. The learned model $\hat{F}$ is used with CEM to perform planning and gather new T samples to add to $\mathcal{D}_{re}$. To improve the quality of the model, the algorithm iteratively alternates between training and planning for a fixed number of iterations.

To train the model-free component of PhIHP, the training dataset $\mathcal{D}_{im}$ is initially filled with T' samples generated from the learned model $\hat{F}$ and random actions from a pure exploratory policy, π_θ and Q_θ are trained on batches from $\mathcal{D}_{im}$ which is continuously filled by samples from the learned model $\hat{F}$.

We list in Tab. 4 the relevant hyperparameters of PhIHP and baselines. and we report in Tab. 5 the task-specific hyperparameters for PhIHP.
We adopted the original implementation and hyperparameters of TD-MPC. However, we needed to adapt it for early termination environments (*i.e.* Cartpole and Acrobot) to support episodes of variable length, and we found it beneficial for TD-MPC to set the critic learning rate at 1e-4 in

these two tasks.
Fot TD3, we tuned the original hyperparameters and used the same for the TD3 baseline and the model-free component of PhIHP.

Hyperparameter	PhIHP	TD-MPC	TD3	CEM-oracle
		Model learning		
Model	ODE + MLP	MLP	-	Ground truth
Activation	Relu	ELU	-	-
MLP size	2 x 16	2 x 512	-	-
Learning rate	1e-3	1e-3	-	-
		Policy/Value learning		
Batch size	64	512	64	-
Critic size	3 x 200	2 x 512	3 x 200	-
Actor size	2 x 300	2 x 512	2 x 300	-
Activation	Relu	ELU	Relu	-
Critic learning rate	1e-4	1e-3	1e-4	-
Actor learning rate	1e-3	1e-3	1e-3	-
Soft update coefficient τ	0.05	0.01	0.05	-
Policy update frequency	2	2	2	-
Discount factor	0.99	0.99	0.99	-
Exploratory steps	10000	5000	10000	-
Replay Buffer size	1e6	1e6	1e6	-
Sampling technique	Uniform	PER ($\alpha = 0.6, \beta = 0.4$)	Uniform	-
		Planning		
Planner	CEM	MPPI	-	CEM
Exploratory population size	200	512	-	700
Policy population size	20	25	-	-
Elite	10	64	-	20
CEM iterations I	3	6	-	3
Update distribution	mean and std.	weighted mean and std.	-	mean and std.
Planning horizon H	4	5	-	30
Receding horizon RH	1	1	-	5

Table 4: PhIHP and baselines hyperparameters. We emphasize that we use the same hyperparameters for TD3 in the baseline and the model-free component of PhIHP.

Hyperparameter	Pendulum	Pendulum swingup	Cartpole	Cartpole swingup	Acrobot	Acrobot swingup
			Model learning			
MLP size	2 x 16	2 x 16	2 x 16	2 x 16	3 x 16	3 x 16
Loss initial coefficient λ_0	1e3	1e3	1e3	1e3	1e2	1e3
Loss update coefficient τ_{ph}	1e3	1e3	1e5	1e5	1e5	1e5
Samples needed	2000	5000	5000	5000	5000	15000
			Planning			
Planning horizon H	5	5	4	6	4	3
Reward coefficient α	1.5	1.5	0.2	0.03	0.8	0.8

Table 5: Task-specific hyperparameters of PhIHP.

D Comparison to state of the art

We compare PhIHP to baselines on individual tasks, we present both statistical results and a qualitative analysis.

D.1 Learning curves

We provide learning curves of PhIHP and baselines on individual tasks. PhIHP outperforms baselines by a large margin in terms of sample efficiency. Figure 10 shows that TD3, even when converging early in Cartpole-swingup, achieves sub-optimal performance and fails to converge within 500k steps in Acrobot-swingup.

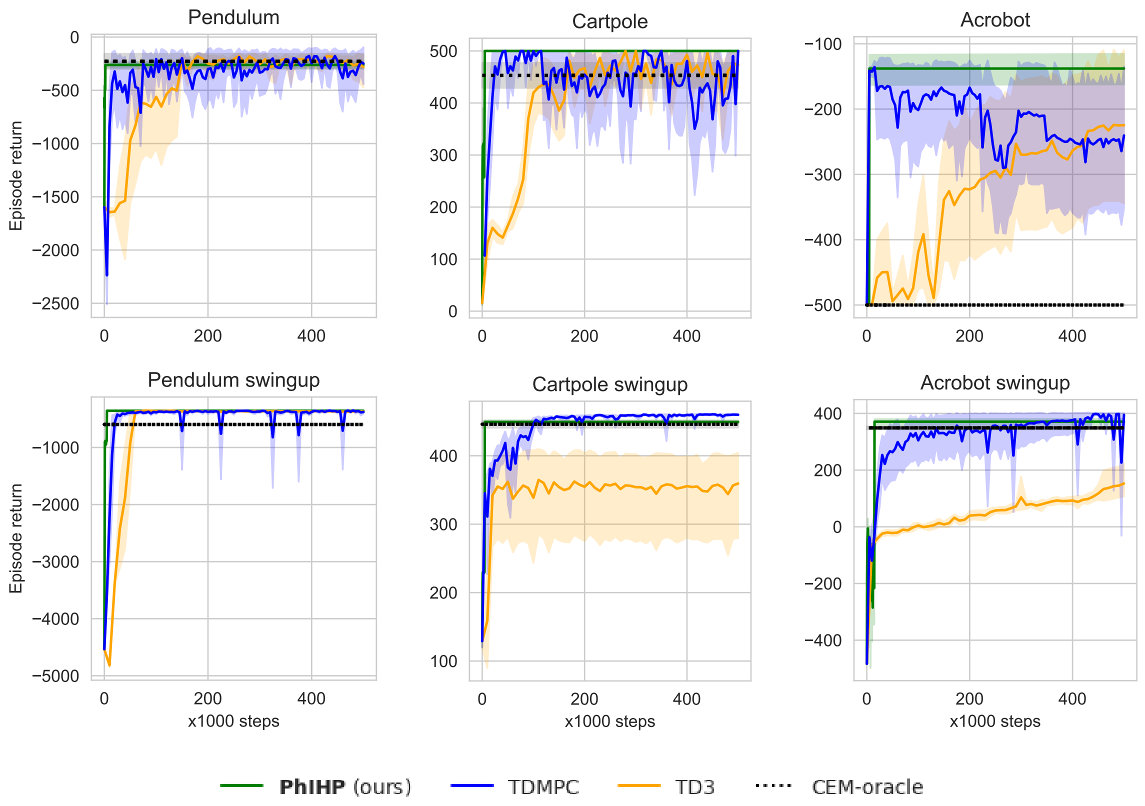


Figure 10: Return of PhIHP and baselines on the gymnasium classic control tasks. Mean and std. over 10 runs. PhIHP outperforms or matches the baselines.

D.2 Statistical Comparison: PhIHP vs. Baselines

To ensure a robust and statistically sound comparison with the results previously reported in Table 1 in Sec. 5.2, we conducted Welch's t-test to statistically compare the performance of PhIHP *vs* baselines across individual tasks. We set the significance threshold at 0.05, and calculated p-values to determine whether observed differences in performance were statistically significant. Tab. 6 shows that PhIHP is equivalent to all baselines in Pendulum, and it significantly outperforms TD3 on the remaining tasks. Moreover, PhIHP outperforms TD-MPC in sparse-reward early-termination environment tasks (Cartpole and Acrobot), while they demonstrate equivalent performance in Pendulum, Pendulum swingup, and Acrobot swingup.

	TD3	**TD-MPC**	**CEM-oracle**	**TD3**	**TD-MPC**	**CEM-oracle**	**TD3**	**TD-MPC**	**CEM-oracle**
	Pendulum			**Cartpole**			**Acrobot**		
T-statistic	-1.41	0.52	-1.18	4.66	5.47	25.75	4.39	5.35	29.78
P-value	0.16	0.61	0.26	9.92e-06	3.40e-07	9.69e-10	1.96e-05	2.58e-07	3.30e-51
Significant difference	**No**	**No**	**No**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
	Pendulum swingup			**Cartpole swingup**			**Acrobot swingup**		
T-statistic	6.35	1.19	6.47	8.41	-7.59	1.65	27.49	-0.10	4.02
P-value	1.48e-09	0.24	1.15e-4	2.70e-13	9.01e-12	0.11	3.54e-66	0.92	1.09e-4
Significant difference	**Yes**	**No**	**Yes**	**Yes**	**Yes**	**No**	**Yes**	**No**	**Yes**

Table 6: Statistical Comparison of PhIHP *vs* Baselines across individual tasks: we present the Welch's t-test results including T-statistics and P-values, to assess the significance of performance differences. **Yes** denotes a statistically significant difference (p-value < 0.05), with green **Yes** indicating PhIHP outperforming the baseline (T-statistics > 0), and red **Yes** indicating the baseline performing better (T-statistics < 0). **No** indicates no significant difference between PhIHP and the baseline (p-value > 0.05).

D.3 Imagination learning for model-free TD3

We provide learning curves of TD3 through imagination on individual tasks in Figure 11. TD3-im-ph is a component of PhIHP, it is a TD3 agent learned on trajectories from a physics-informed model. It largely outperforms TD3-im-dd, a TD3 learned on trajectories from a data-driven model. we limited the training budget for TD3-re, trained on real trajectories, at 500k real samples in all tasks.

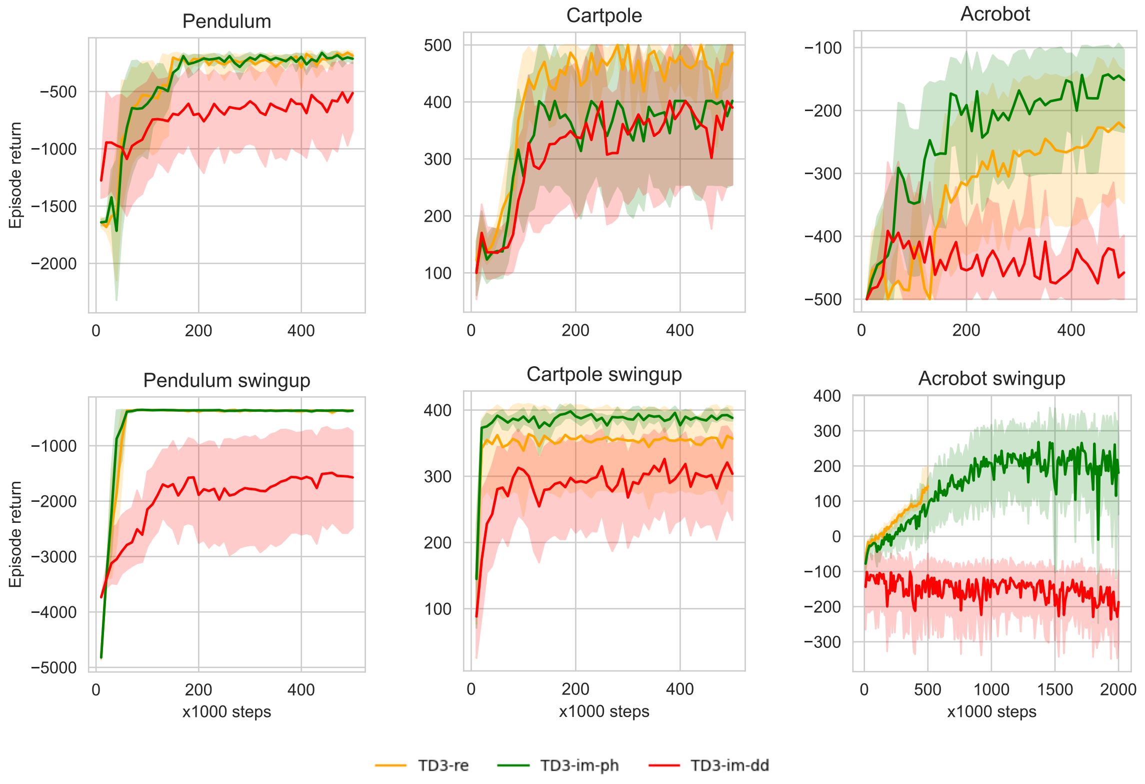
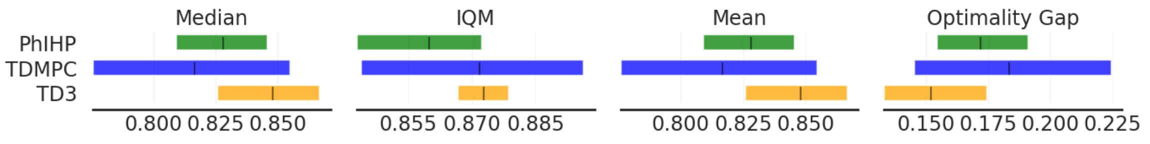


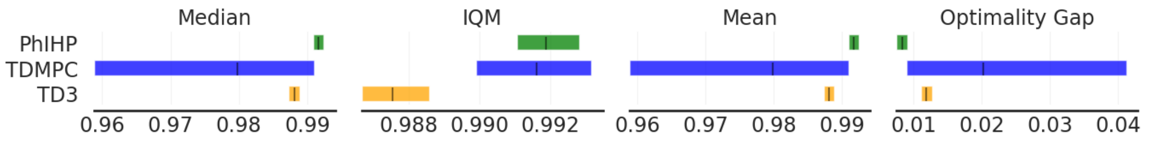
Figure 11: Learning curve of TD3 on classic control tasks, mean and std. over 5 runs. **TD3-re** (orange curve) is a TD3 agent trained on real trajectories, **TD3-im-ph** (green curve) and **TD3-im-dd** (red curve) are TD3 agents trained on imaginary trajectories respectively from a physics-informed model and data-driven model.

D.4 Qualitative comparison

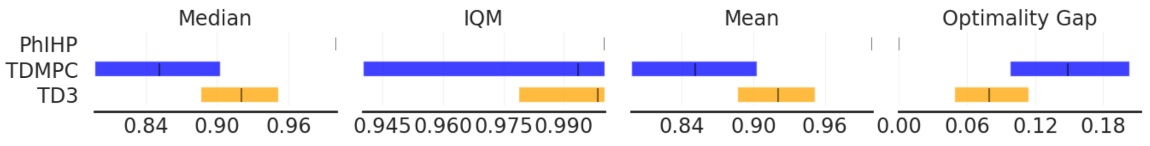
In this section, we compare performance metrics on individual classic control tasks. We estimate confidence intervals by using the percentile bootstrap with stratified sampling (Agarwal et al., 2021).
We show in Figure 12 a comparison of the median, interquartile median (IQM), mean performance, and optimality gap of PhIHP and baselines. PhIHP matches or outperforms the performance of TD-MPC and TD3 in all tasks except in Cartpole swingup. PhIHP shown to be robust to outliers compared to TD-MPC with shorter confidence intervals.
Moreover, Figure 13 shows the performance profiles of PhIHP and baselines. PhIHP shows better robustness to outliers.



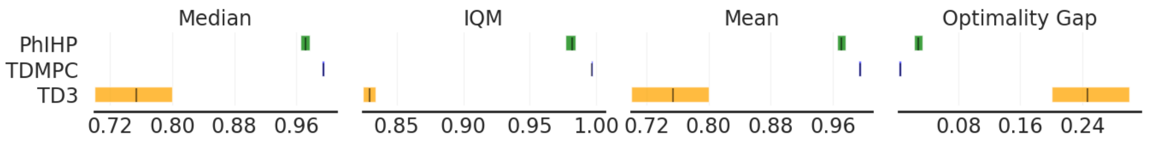
(a) **Pendulum**. PhIHP matches the performance of TD-MPC and TD3.



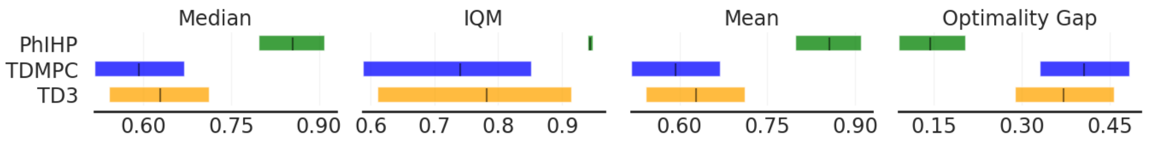
(b) **Pendulum swingup**. PhIHP outperforms TD-MPC and TD3, and PhIHP shows to be robust to outliers compared to TD-MPC.



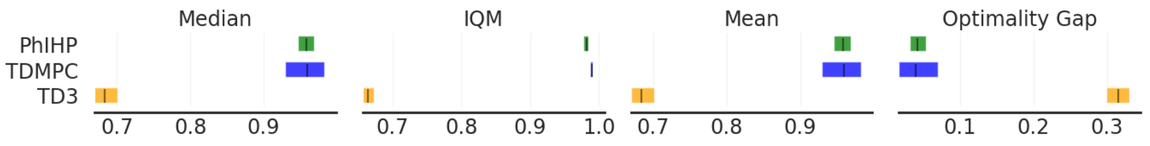
(c) **Cartpole**. PhIHP largely outperforms TD-MPC and TD3.



(d) **Cartpole swingup**. PhIHP outperforms TD3 and shows slightly less performance than TD-MPC.



(e) **Acrobot**. PhIHP largely outperforms TD3 and TD-MPC.



(f) **Acrobot swingup**. PhIHP outperforms TD3 and matches the performance of TD-MPC.

Figure 12: Median, interquartile median (IQM), mean performance, and optimality gap of PhIHP and baselines on individual classic control tasks (10 runs). Higher mean, median, and IQM performance and lower optimality gap are better. Confidence intervals (CIs) are estimated using the percentile bootstrap with stratified sampling (Agarwal et al., 2021).

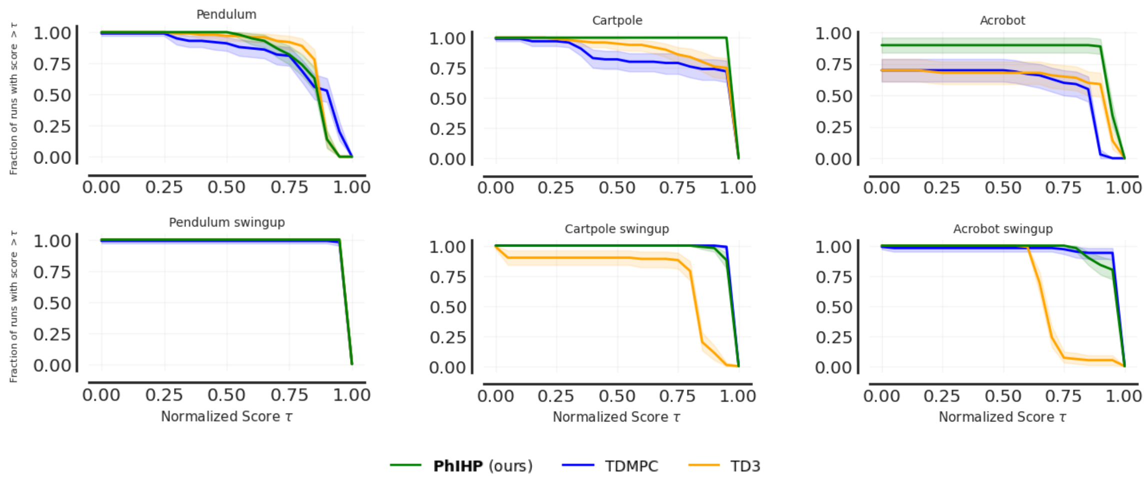


Figure 13: Performance profiles of PhIHP and baselines on individual tasks (10 runs). Confidence intervals are estimated using the percentile bootstrap with stratified sampling (Agarwal et al., 2021). PhIHP shows a better robustness to outliers.

E Hyperparameter sensitivity analysis

We investigate the impact of varying controller hyper-parameters on the performance and inference time of PhIHP. We first study the impact of varying planning horizons and receding horizons (from 1 to 8). We note that planning over longer horizons generally leads to better performance, however, the performance slightly drops in Acrobot-swingup for planning horizon $H > 4$ (Figure 14). We explain this by the compounding error effect on complex dynamics. Unsurprisingly, lower receding horizons always improve the performance because the agent benefits from replanning.
For the impact of the population size, Figure 14 shows that excluding the policy (policy-population = 0) from planning degrades the performance, and increasing it under 10 does not have a significant impact. Moreover, excluding random actions (random-population = 0) from planning degrades the performance.
Unsurprisingly, the inference time increases with an increase in both the planning horizon and the population size. Conversely, it decreases when the receding horizon increases.

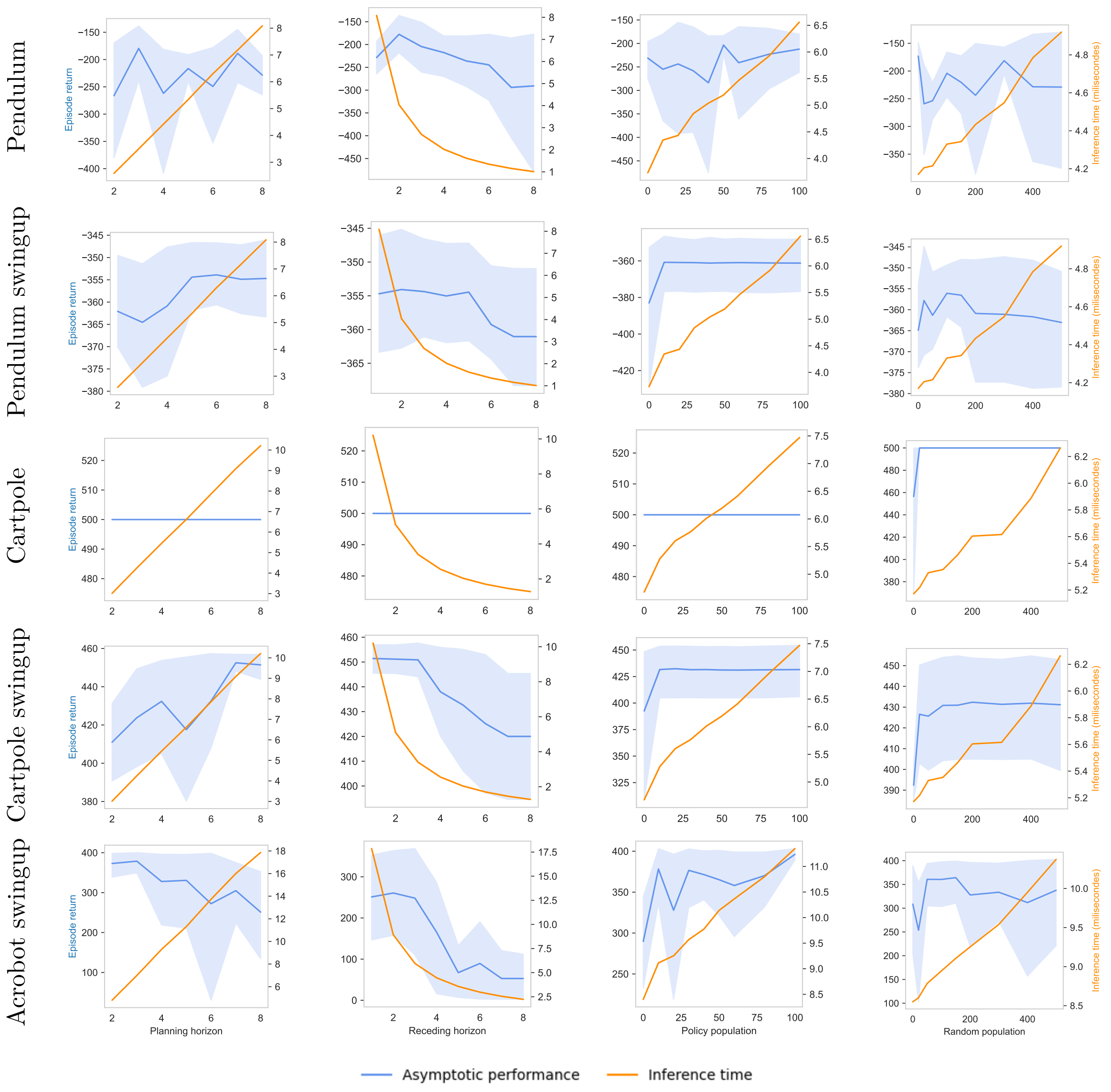


Figure 14: Impact of varying planning hyperparameters on asymptotic performance and inference time on individual tasks.