Reinforcement Learning vs Optimal Control: Sparse Nonlinear Dynamical Systems Between Theory and Practice

Davide Maran
Politecnico di Milano, Milan, Italy
davide.maran@polimi.it

Gianmarco Tedeschi
Politecnico di Milano, Milan, Italy
gianmarco.tedeschi@polimi.it

Enea Gusmeroli
Politecnico di Milano, Milan, Italy
enea.gusmeroli@polimi.it

Marcello Restelli
Politecnico di Milano, Milan, Italy
marcello.restelli@polimi.it

Abstract

The recent development of sparse methods for identifying nonlinear dynamical systems has opened new avenues for efficient and interpretable model-based reinforcement learning (RL). In this work, we study online RL in environments where the system dynamics, modeled as $s' = f(s, a)$+noise, is assumed to be sparse with respect to a big feature map, a structural idea inspired by the SINDy framework. We introduce an optimistic algorithm that combines online sparse regression with confidence set construction to guide exploration and planning. Our theoretical contributions are threefold: (i) we provide the first regret bounds for sparse nonlinear dynamics, showing that regret scales with the sparsity level d_0; (ii) we relax standard Gaussian assumptions by allowing general subgaussian noise with bounded variation densities, significantly broadening the class of admissible stochastic systems; and (iii) we extend our theoretical guarantees to misspecified models, where the dynamics are only approximately sparse in the chosen feature space. The algorithm enjoying the regret bound is not computationally efficient, as it builds on a very heavy online regression method. We propose a practical variant using ensemble SINDy in place of the online regression algorithm, and SAC within a Dyna-style framework. Empirical results on classic continuous control tasks demonstrate the practical viability and robustness of our approach.

1 Introduction

Reinforcement learning (RL) Sutton and Barto [2018] is a paradigm of artificial intelligence in which the agent interacts with a system to maximize a cumulative reward over a given number of time-steps. The main challenge to perform this task, which makes it fundamentally different from the bandit problem Lattimore and Szepesvári [2020], is that actions can influence the state of the system according to an unknown *transition function*. Algorithms that try to estimate this transition function to reduce RL to the planning problem are called *model-based*. Model-based Reinforcement Learning and classical continuous control share a fundamental commonality: both seek to design decision-making or feedback policies based on an internal representation of how the system evolves over time Recht [2019]. In continuous control, one typically starts with a (potentially simplified) physics-based model of the system, whose parameters are calibrated to converge to the true model and allow for the design of an improved controller Simpkins [2012]. In model-based RL, the approach likewise involves learning or approximating the transition function from data so that a policy can be optimized against this learned model. However, unlike continuous control, early RL research often focused on tabular or small discrete environments where learning the dynamics model from scratch was feasible without explicit structural assumptions. This gap between the two communities has been greatly reduced recently due to the ever-increasing interest in high-dimensional environments for RL, which has led to the development of model-based algorithms that go well beyond the tabular setting Chua et al. [2018]. Estimating the transition function in these complex domains requires an overwhelming sample complexity if done as in tabular environments. This motivates the need for system identification methods that are both parsimonious, remaining statistically tractable, and powerful, to achieve a satisfactory approximation of the true dynamics.

A prominent example of parsimonious modeling in the control theory community is the Sparse Identification of Nonlinear Dynamics (SINDy) framework Brunton et al. [2016]. SINDy posits that the dynamics of a complex system can often be accurately represented by a small number of terms coming from a redundant feature map (e.g., polynomials, trigonometric terms). In formulas, this corresponds to assuming that the next state s_{h+1} of the system behaves according to

$$s_{h+1} = W_h^\star \phi_h(s_h, a_h) + \eta_h, \tag{1}$$

for a noise variable η_h, where matrix $W_h^\star$ is *sparse*, so that only a few elements of the dictionary (often called *feature map*) $\phi_h(\cdot)$ are in fact relevant. This framework has proven remarkably effective in domains ranging from fluid mechanics Proctor et al. [2016] to biological systems Champion et al. [2019]. By leveraging sparsity, SINDy provides enhanced interpretability and robustness, traits that are increasingly valued in control applications.

Paper structure In the next sections, we first introduce the problem setting and our core assumptions. We then present a theoretically grounded algorithm with regret guarantees under sparsity. This is followed by an extension to misspecified models. Finally, we describe a practical variant based on SINDy and SAC, and validate our approach through experiments on standard control benchmarks. Finally, we report the related works in the appendix A.

2 Setting

We consider the setting of online optimization of a *sparse* dynamical system. Given $\mathcal{S} \subset \mathbb{R}^{p_\mathcal{S}}, \mathcal{A} \subset \mathbb{R}^{p_\mathcal{A}}$ (we set $p = p_\mathcal{S} + p_\mathcal{A}$) the state/action spaces respectively, the system evolves as

$$s_{h+1} = f_h(s_h, a_h) + \eta_h, \tag{2}$$

where $f_h : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is an arbitrary, possibly non-linear function and η_h is random noise with independent realizations at each time-step h. The interaction with this system is divided into $K \in \mathbb{N}$ episodes, with each episode lasting for a total of $H \in \mathbb{N}$ time steps. In each episode, the goal of the agent is to choose actions $\{a_h\}_{h=1}^H$ in order to maximize a cumulative reward function, given by $\sum_{h=1}^H r_h(s_h, a_h), r_h : \mathcal{S} \times \mathcal{A} \to [0, 1]$.

Value functions To better formalize this goal, we call *policy* the selection rule according to which the agent selects their actions at any time step of any episode. Formally, at each episode $k \in [K] := \{1, \dots, K\}$, the agent chooses a policy $\pi^k = \{\pi_h^k\}_{h=1}^H$, which is a sequence of mappings from $\mathcal{S}$ to the probability distributions over $\mathcal{A}$. For each stage $h \in [H]$, the action is chosen according to $a_h \sim \pi_h^k(\cdot|s_h)$, the agent gains reward $r_h(s_h, a_h)$ and the environment transitions according to (2). To measure the expected reward that a policy can achieve, we define the *value function*, i.e., the function $V_h^\pi(s) := \mathbb{E}_\pi \left[\sum_{\ell=h}^H r_\ell(s_\ell, a_\ell) \middle| s_h = s\right]$. This will be compared to the *optimal value function*: $V_h^*(s) = \sup_\pi V_h^\pi(s)$.

Regret As a measure of performance of an agent across the K episodes of interaction with the environment, we define the *regret* as $R_K := \sum_{k=1}^K V_1^*(s_1^k) - V_1^{\pi_k}(s_1^k)$, where s_h^k denotes the state visited at step h of episode k. Note that if $R_K = o(K)$ with some probability, then the value function of the "average played policy" will tend to that of the optimal policy as $K \to \infty$. As the value function is the expected value of the cumulative reward, this corresponds to learning the optimal behavior in the environment.

Environment transition and sparsity To propose an algorithm that is able to deal with an environment of this form, we need some assumptions on its transition (eq. 2).

Assumption 1. *(Sparse feature map decomposition) We assume that*

$$f_h(s,a) = W_h^\star \phi_h(s,a),$$

for some feature map and $\phi_h : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ matrix $W_h^\star \in \mathbb{R}^{p_\mathcal{S} \times d}$. Moreover, each row of $W_h^\star$ has at most d_0 non-zero entries, with $p_\mathcal{S} \ll d_0 \ll d$.

This assumption is the core of the paper, as it represents what is usually meant by a sparse dynamical system Brunton et al. [2016]. Moreover, the assumption that $p_\mathcal{S} \ll d_0 \ll d$ holds whenever the feature map is complex enough to capture general classes of nonlinear functions in s, a. Indeed, polynomials/trigonometric functions of degree N, which have been used as feature maps for nonlinear continuous RL for their approximation power (Maran et al. [2024c] and Maran et al. [2024b] respectively), have a length scaling as $d = \binom{N+p}{p} \approx N^p$. Instead, d_0 is usually smaller than d, but just by a polynomial factor (e.g. $d \approx d_0^2$).

3 Algorithm: Theoretical Foundation

As we said in the introduction, the central challenge in model-based RL is to balance accurate system identification with effective control. On the one hand, a good estimation for matrix $W_h^\star$ (assumption 1) to predict the dynamics of the system. On the other hand, only focusing on estimating the transition function may lead to a bad regret, as it ignores the reward function. In our approach, we leverage the sparsity on the principle of optimism in the face of uncertainty Auer et al. [2008] to balance these two needs. We are going to use some method from online regression Strehl and Littman [2007] to maintain a confidence set in the parameter space, then plan optimistically within that set. This idea draws on principles from both sparse regression and optimistic exploration in model-based RL. Below, we give a high-level overview of the three main ingredients needed to build our algorithm.

First, we need a regression method that, from collected transitions $(s_h^k, a_h^k, s_{h+1}^k)$, and tough an appropriate sparsity-enforcing regularization, yields an estimate $\widehat{s}_{h+1}^k$ of $\mathbb{E}[s_{h+1}^k]$. Crucially, this algorithm must provide a theoretical guarantee that exploits the sparsity of the problem, so that assumption 1 is used at its full. Second, we need a confidence ball that contains $W_h^\star$ with high probability. To do so, we can use the predictions $\widehat{s}_{h+1}^k$ by the regression algorithm in order to build an estimation $\widehat{W}_h$ of the transition matrix, and the statistical complexity guarantee of the same algorithm prove that $\|\widehat{W}_h - W_h^\star\| \le \beta(\delta)$, where δ is the error probability and $\|\cdot\|$ is some norm. In this way, we can define our confidence sets as $\{W : \|\widehat{W}_h - W\| \le \beta(\delta)\}$. Lastly, once the confidence sets for the transition are fixed, we use the reward, which we assume to be known, to plan the best actions. In this way, we can ensure that (as long as the confidence sets hold), the agent plays with the best policy according to the most optimistic transition function in the confidence balls. In this way, in any episode, we either achieve a good return (that means the model is accurate) or we gather information allowing to reduce the size of the confidence balls. The first critical part is to provide a regression method that can deal with the sparse regression case.

Regression algorithm Selecting an appropriate regression algorithm is non-trivial. Indeed, the need to work for an arbitrary sequence of input values that may depend on past realizations of the noise (as they depend on the state and the actions) rules out standard Ridge Regression approaches. We adopt algorithm SeqSEW, originally introduced in Gerchinovitz [2013]. While we do not delve into its intricate structure here, we highlight its regret bound. This result, which belongs to the original paper but was restated in a more convenient form by Lattimore and Szepesvári [2020] (theorem 23.6), ensures what follows

Theorem 2. *(Theorem 23.6 from Lattimore and Szepesvári [2020]) Let $\boldsymbol{x}_t \in \mathbb{R}^d, y_t \in \mathbb{R}$ be arbitrary sequences such that $\|\boldsymbol{x}_t\|_2 \le X, |y_t| \le Y$ for $t = 1, \dots T$. Let $\widehat{y}_t$ be the output of* SeqSEW *at step t (therefore, knowing $\boldsymbol{x}_t$ and all previous data for $\tau < t$). Then, for any $\theta \in \mathbb{R}^d$ with $\|\theta\|_0 = \theta_0$, we have*

$$\rho_T(\theta) := \sum_{t=1}^T (y_t - \widehat{y}_t)^2 - (y_t - \boldsymbol{x}_t^\top \theta)^2 \le \rho_{\max} = \widetilde{\mathcal{O}}(X^2 \theta_0), \tag{3}$$

where $\widetilde{\mathcal{O}}$ hides logarithmic factors in $X, Y, T, \theta_0, \|\theta\|_2$.

In our case, the pairs $\{\boldsymbol{x}_t, y_t\}$ correspond to $\boldsymbol{x}_t = \phi_h(s_h^k, a_h^k)$, and $y_t = s_{h+1,\iota}^k$, where ι defines the $\iota-$**th component** of the vector. This is done since s_{h+1}^k is multivariate and we treat its prediction as $p_{\mathcal{S}}$ single regressions. Our algorithm is going to use these predictions by SeqSEW in order to build some confidence balls for the parameters.

Confidence ball The result from Theorem 2 can be easily transformed into the definition of a confidence ball.

Theorem 3. *Fix an error probability $\delta > 0$, $\iota \in \{1, \dots p_{\mathcal{S}}\}$ and $h \in \{1, \dots H\}$. Let $\widehat{y}_{h,\iota}^k$ be the output of* SeqSEW *for the sequences $\phi_h(s_h^k, a_h^k), s_{h+1,\iota}^k$, that are indexed by k. Define the* ***confidence balls***

$$\mathcal{C}_{h,\iota}^k := \left\{ \theta \in \mathbb{R}^d : \sum_{\tau=1}^k (\widehat{y}_{h,\iota}^\tau - \phi_h(s_h^\tau, a_h^\tau)^\top \theta)^2 \le \beta(\delta) + \|W_{h,\iota}^\star\|_2^2 \right\}, \tag{4}$$

where $W_{h,\iota}^\star$ is the $\iota-$th row of matrix $W_h^\star$, and $\beta(\delta) = 1 + 2\rho_{\max} + 32 \log\left(\frac{\sqrt{8}+\sqrt{1+\rho_{\max}}}{\delta}\right) = \widetilde{\mathcal{O}}(\rho_{\max} + \log(1/\delta))$ (where $\rho_{\max}$ comes from equation 3). Then, $\boxed{\mathbb{P}(\exists k : W_{h,\iota}^\star \notin \mathcal{C}_{h,\iota}^k) \le \delta}$.

For the proof, see the appendix C. This confidence set satisfies a very useful property, which is going to be one of the main backbones of the regret bound:

Lemma 1. *For every h, k, ι, let $\Lambda_h^k := I + \sum_{\tau=1}^k \phi_h(s_h^k, a_h^k)\phi_h(s_h^k, a_h^k)^\top$, where I stands for the identity matrix of size d. Moreover, let $\widehat{W}_{h,\iota}^k := (\Lambda_h^k)^{-1} \sum_{\tau=1}^k \widehat{y}_{h,\iota}^k \phi_h(s_h^k, a_h^k)$. Then,*

$$\mathcal{C}_{h,\iota}^k \subseteq \left\{ \theta \in \mathbb{R}^d : \|\theta - \widehat{W}_{h,\iota}^k\|_{\Lambda_h^k}^2 \le \beta(\delta) + \|W_{h,\iota}^\star\|_2^2 \right\}.$$

The previous lemma allows us to give a much better idea of what the confidence ball looks like. In fact, if we see $\widehat{W}_{h,\iota}^k$ as a replacement for the standard OLS solution, the result of lemma 1 is similar to standard confidence bound for linear bandits/MDPs [Jin et al., 2020, Abbasi-Yadkori et al., 2011], just that it scales with the sparsity parameter d_0, instead of d.

Planning To design our algorithm 1, the only missing part is the choice of the policy π_h^k to be played at stage h of episode k. Here, the idea is to leverage our construction of the confidence balls to build an optimistic estimate of the Q_h^*-function of the MDP, which we can obtain iteratively at each step h, starting from $h = H$. This procedure builds on the well-known Value Iteration VI algorithm Sutton and Barto [2018]. In fact, if the matrix $W^\star$ were known, we could explicitly implement VI as follows:

$$V_{h+1}^*(s) = \arg\max_{a \in \mathcal{A}} Q_{h+1}^*(s, a) \quad Q_h^*(s, a) = r_h(s, a) + \underbrace{\int_{\mathcal{S}} V_{h+1}^*(W_h^\star \phi_h(s, a) + \eta) d\mu_{h+1}(\eta)}_{\mathbb{E}[V_{h+1}^k(s')]}. \tag{5}$$

This procedure cannot be directly implemented, as we do not know $W^\star$. Still, knowing that $W^\star \in \mathcal{C}_h^k$ with high probability, we can perform the same algorithm with respect to the most promising matrix $W \in \mathcal{C}_h^k$. This is precisely what we do in lines 11 and 12 of algorithm 1. The only difference w.r.t. VI (equation) is in line 12 where we replace the $\mathbb{E}[V_{h+1}^k(s')]$ with the "expected next state under W", that is $\mathbb{E}[V_{h+1}^k(s')|W] = \int_{\mathcal{S}} V_{h+1}^k(W\phi_h(s, a) + \eta) d\mu_{h+1}(\eta)$. Then, to ensure optimism, we maximize over $W \in \mathcal{C}_h^k$, which, as we said, contains the true value $W^\star$ with high probability. In this way, we are able to ensure optimism, as we prove in the appendix C.

We now explain in more depth the structure of algorithm 1, which combines the previous three steps. The first lines, up to 4 initialize our confidence balls and online regression algorithms SeqSEW Gerchinovitz [2013], one for each time-step h and dimension ι. Then, we start iterating over the episode and, we said, we compute the current policy using VI over the most optimistic model W in lines 11 and 12. This policy is then run over the environment (line 17). Here, there is an additional step compared to standard RL algorithms: once we chosen the action, we query the online learner

Algorithm 1 DYNAMIC PLANNER WITH CONFIDENCE-BASED ESTIMATION (DPCBE)

```
Require: Upper bounds $X, Y$, Initial radius $R$, Reward function $r_h$, Failure probability $\delta$
 1: for $\iota = 1 \dots p_{\mathcal{S}}$ and $h = 1, \dots, H$ do
 2:     Initialize $\mathcal{C}^1_{h,\iota} := R \cdot B_1(0)$
 3:     Initialize ONLINELEARNER$[h, \iota] \leftarrow$ SEQSEW$(X, Y)$
 4: end for
 5: for $k = 1, 2, \dots K$ do
 6:     for $h = 1, 2, \dots H$ do
 7:         $\mathcal{C}^k_h = \times_{\iota \in [p_{\mathcal{S}}]} \mathcal{C}^k_{h,\iota}$
 8:     end for
 9:     $Q^k_H(s, a) = r_H(s, a)$
10:     for $h = H - 1, \dots, 1$ do
11:         $V^k_{h+1}(s) = \arg\max_{a \in \mathcal{A}} Q^k_{h+1}(s, a)$
12:         $Q^k_h(s, a) = \max_{W \in \mathcal{C}^k_{h+1}} r_h(s, a) + \mathbb{E}[V^k_{h+1}(s')|W]$
13:         $\pi^k_h(s) = \arg\max_{a \in \mathcal{A}} Q^k_h(s, a)$
14:     end for
15:     Receive initial state $s^k_1$
16:     for $h = 1, 2, \dots H - 1$ do
17:         Choose action $a^k_h \sim \pi^k_h(\cdot|s^k_h)$
18:         for $\iota = 1, \dots p_{\mathcal{S}}$ do
19:             Input $\phi_h(s^k_h, a^k_h)$ to ONLINELEARNER$[h, \iota]$, receiving $\widehat{s}^k_{h+1,\iota}$
20:             Update $\mathcal{C}^k_{h,\iota}$ with $\widehat{s}^k_{h+1,\iota}$ according to equation (4).
21:         end for
22:         Receive next state from the environment $s^k_{h+1}$
23:         for $\iota = 1, \dots p_{\mathcal{S}}$ do
24:             Update ONLINELEARNER$[h, \iota]$ with $s^k_{h+1,\iota}$
25:         end for
26:     end for
27: end for
```

(line 19) in order to get a prediction that we are then using to update the confidence ball (line 20). After this, we receive the next state (line 22) from the environment. This state is *not* directly used to update the confidence balls, as it usually happens, but to update the online learners (line 24).

In fact, our policy is a function of confidence balls, which are only updated based on data that we collect from the online learner. The online learner is the only element that directly uses data from the observed transition. As we shall see, this multi-layered structure is the key to achieving our regret bound in the next section.

Regret bound In this section, we present the guarantees for our algorithm. We start from a very general formulation that holds in a wide generality of environments, and then specialize to a few cases of interest. Apart from assumption 1, we have to make an assumption about the distribution of the noise η_h that influences the transition at any time step. Usually, it is assumed that this noise follows from a multivariate Gaussian distribution Kakade et al. [2020]. Here, we relax this assumption by introducing a general class of distributions for the noise.

Assumption 4. *For any h, the noise η_h is $\sigma-$subgaussian. Moreover, it follows a distribution in $\mathbb{R}^{p_{\mathcal{S}}}$ with density function $\mu_h(\cdot)$ sich that $\mu_h(\cdot) \in BV(\chi_h)$, where $BV(\chi) := \{\|\nabla \mu(\cdot)\|_{\mathcal{M}} \leq \chi\}$.*

Here, ∇ stands for the (weak) gradient operator, and the norm $\|\cdot\|_{\mathcal{M}}$ is defined, for any function $g : \mathbb{R}^p \to \mathbb{R}^p$, as $\|g\|_{\mathcal{M}} = \sup_{f \in \mathcal{C}(\mathbb{R}^p, \mathbb{R}^p),\ \|f\|_{L^\infty} \leq 1} \int_{\mathbb{R}^p} \langle f(x), g(x) \rangle dx$. The generality of this call is going to be explained in the next paragraph. For now, we just give the regret bound under this assumption.

Theorem 5. *Assume 1 and 4 which holds for $\chi_h \leq \chi$ for every $h = 1, \dots H - 1$. Then, with probability at least $1 - \delta$, algorithm 1 achieves the following regret bound* $\boxed{R_K \leq \widetilde{\mathcal{O}}(H^2 \chi p_{\mathcal{S}} \sqrt{d_0 d K})}$.

The most interesting parameters that emerge from this regret bound are d, d_0, and χ, as $p_{\mathcal{S}} \ll d_0$ and H, K are standard in the theoretical RL literature. As we can see, even if the system is sparse, the dependency on d still appears in the regret bound, even if under a square root. This is still an improvement of the state of the art, where the dependency was linear, and cannot be further improved, even in the case of sparse linear bandits (see section 23.3 in Lattimore and Szepesvári [2020]). In fact, our bound scales as the one by Abbasi-Yadkori et al. [2012], which works with sparse linear *bandits*. The other very interesting parameter that appears is χ, which is strictly linked to assumption 4 and represents the maximal total variation of the density function of the noise. Even if it is formally difficult to characterize this parameter in terms of something more interpretable, the next paragraph shows that it is bounded in most real cases.

Bounding χ for common noise classes Before this work, it was known that a) in case of *Gaussian noise*, a regret bound scaling polynomially in H in a model with transition of the form $s_{h+1} = f(s_h, a_h) + \eta$ was possible (Kakade et al. [2020]). b) In general, MDPs with this form may suffer an *exponential lower bound* in H (Maran et al. [2024c] theorem 2). In particular, the lower bound b) leverages a Dirac-type noise at some time steps, which is highly concentrated on a discrete number of points. We can see our assumption 4 as a way to prevent this scenario to happen and see $\chi = \max_h \|\mu_h\|_{BV}$ as a measure of how much the density function of the noise concentrates on one or more single points. In fact, we have, fixing $\mathcal{S} \subset \mathbb{R}$, for clarity,

- $\mu(s) = \delta_0(s)$ leads to $\chi = +\infty$ (Dirac's delta is not in BV).
- $\mu(s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-s^2/(2\sigma^2))$ leads to $\chi = \frac{1}{\sqrt{2\pi\sigma^2}}$ (Gaussian noise is in BV).
- $\mu(s) = \mathbf{1}_{(-a,a)}(s)$ leads to $\chi = 1/(2a)$ (uniform noise is in BV).
- Every bounded $\mu(\cdot)$ with at most m local maxima is in BV with $\chi \leq m \sup_{s\in\mathcal{S}} \mu(s)$.

Moreover, one crucial property of this class of density functions, which makes it very useful, is that it is closed under convex combinations, as results from the following proposition.

Proposition 6. *For any pair of density functions $\mu(\cdot) \in BV(\chi), \widetilde{\mu}(\cdot) \in BV(\widetilde{\chi})$ and any constant $\lambda \in [0, 1]$, their convex combination satisfies* $\boxed{\lambda\mu(\cdot) + (1 - \lambda)\widetilde{\mu}(\cdot) \in BV(\lambda\chi + (1 - \lambda)\widetilde{\chi})}$.

This shows that assuming 4 increases dramatically the level of generality: indeed, any Gaussian Mixture is included in the BV class, which makes this class extremely flexible. We now investigate how a small change in the algorithm can lead to a generalization of theorem 10 to the case where the model 1 is misspecified.

3.1 Extension: misspecified models

For the sake of this section, we relax assumption 1 in the following way:

Assumption 7. *(Sparse misspecified feature map decomposition) For some feature map and $\phi_h : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ matrix $W_h^\star \in \mathbb{R}^{p_\mathcal{S}\times d}$ such that each row of $W_h^\star$ has at most d_0 non-zero entries, with $p_\mathcal{S} \ll d_0 \ll d$, let $\xi_h : \mathcal{S} \times \mathcal{A} \to [0, \infty)$ be the function*

$$\xi_h(\cdot) := \|W_h^\star \phi_h(\cdot) - f_h(\cdot)\|_{L^\infty}.$$

We assume that, for any $h = 1, \dots H$, $\xi_h(\cdot)$ is uniformly bounded by a constant ξ_∞.

We can see that Assumption 1 corresponds to Assumption 7 for $\xi_\infty = 0$. As the next result shows, a modification of algorithm 1 allows us to generalize its regret bound to this, more challenging, setting.

Theorem 8. *Assume 7 and 4 which holds for $\chi_h \leq \chi$ for every $h = 1, \dots H - 1$. Let us modify the construction of the confidence sets in equation 4 by using as confidence radius $\beta'(\delta) = \beta(\delta) + K\xi_\infty^2 \log(1/\delta)$. Then, with probability at least $1 - \delta$, algorithm 1 achieves the following regret bound*

$$\boxed{R_K \leq \widetilde{\mathcal{O}}\Big(H^2 \chi p_\mathcal{S} \sqrt{d_0 d K} + H^2 p_\mathcal{S} \sqrt{d} K \xi_\infty\Big)}.$$

The new regret bound contains a first term $H^2 \chi p_\mathcal{S} \sqrt{d_0 d K}$ which is identical to the previous regret bound. The dependence on the misspecification is all captured by the other term, $H^2 p_\mathcal{S} \sqrt{d} K \xi_\infty$, where some things need to be commented on. First, the linear growth in K is expected: for big ξ_∞, the regret grows linearly, as the approximation power of our feature map is very limited. What is more concerning is the multiplicative factor $\sqrt{d}$, which creates a trade-off: bigger d may reduce the misspecification, but if the reduction is not sufficient, the term $\sqrt{d}\xi_\infty$ gets bigger, actually. This potentially pernicious phenomenon affects many other algorithms for Bandits/Reinforcement Learning with linear function approximation, and cannot be avoided in general Lattimore et al. [2020].

Algorithm 2 OPTIMISTIC SIMULATION WITH CONFIDENCE-BALL REGRESSION (OSCAR)

```
Require: Regularization parameter $\alpha$, Reward function $r$, Basis Functions $\phi$, Batch size $b$, Horizon $H$, Number
    of off-policy iterations $N_e$, Number of on-policy iterations $N_o$, Number of model interactions $N_m$, Policy
    $\pi(\cdot|s_w)$, Ball scale factor $\lambda$
 1: Initialize dataset $D_e \leftarrow \emptyset$
 2: Collect trajectories in $D_e$ using Random Policy for $N_e$ episodes
 3: $SINDy(D_e, \phi)$                                              ▷ Fit SINDy Model
 4: for each $(s, a)$ in $D_e$ do                                   ▷ Set Confidence Ball
 5:     Predict $s'$ = $SINDy(s, a)$
 6:     Update $\mathcal{C}$ with $s'$ according to equation (4)
 7: end for
 8: for $n = N_e + 1, ..., N_o$ do                                  ▷ Start On Policy Interaction
 9:     for $h = 1, 2, ..., H$ do
10:         Observe state $s_h$
11:         Sample action $a_h \sim \pi(\cdot|s_h^{\widehat{W}})$
12:         Predict $s_{h+1} = SINDy(s_h, a_h)$
13:         Update $\mathcal{C}$ with $s_{h+1}$ according to equation (4)
14:         Observe next state $s_{h+1}$
15:         Update $D_e \leftarrow (s_h, a_h, s_{h+1})$
16:     end for
17:     Randomly sample a batch of $b$ states $S = \{s\} \sim D_e$      ▷ Start Model Interaction and Policy Update
18:     for $i = 1, 2, \ldots N_m$ do
19:         Uniformly sample $b$ scale factors $\beta \sim U[0, \lambda]$
20:         Sample $b$ models $M \sim N(\widehat{W}, \beta(\Lambda + \alpha \cdot I)^{-1})$
21:         Uniformly sample $b$ scale factors $\beta' \sim U[0, \lambda]$
22:         Sample $b$ models $M' \sim N(\widehat{W}, \beta'(\Lambda + \alpha \cdot I)^{-1})$
23:         Initialize $A \leftarrow \emptyset$, $S' \leftarrow \emptyset$, $K \leftarrow \emptyset$
24:         for each $(s, m, m')$ in $S \times M \times M'$ do
25:             Sample action $a \sim \pi(\cdot|s^{m'})$
26:             Observe $s' = m \cdot \phi(s, a)$
27:             Collect $k = r(s, a)$
28:             Update $A \leftarrow (a), S' \leftarrow (s'), K \leftarrow (k)$
29:         end for
30:         Perform a SAC update using batch $(S \times M, A, K, S' \times M)$
31:     end for
32:     $SINDy(D_e, \phi)$                                          ▷ Update SINDy Model
33: end for
```

4 Algorithm: Practical Variant

The algorithmic solution presented in Section 3 relies on the usage of SEQSEW [Gerchinovitz, 2013] as a regressor to build the confidence balls for the parameters. However, a computationally feasible version of SEQSEW is still an open problem (see Lattimore and Szepesvári [2020] 23.5 Note 1). To design a practical version, we employed the SINDy framework as a regressor Brunton et al. [2016].

SINDy SINDY is a model discovery framework that identifies the governing equations of a dynamical system directly from measured data. Specifically, SINDY can extract differential equations or data given a collection of possible features ϕ_h. SINDY sparsity refers to selecting a minimal set of functions to represent the dynamics, while its non-linearity indicates that these functions can include nonlinear combinations of state variables. The sparsity assumption 1 motivates the choice of SINDY for the identification of the dynamics, since it relies on the Sequentially Threshold Least Squares with RIDGE (STLRidge) algorithm which iteratively performs RIDGE regression forcing to zero coefficients below a certain threshold. This process is repeated until convergence, progressively refining the model by discarding less relevant terms. To improve robustness in the presence of noise and enhance performance on real-world data, Ensemble-SINDY [Fasel et al., 2022, E-SINDY] has been proposed as an extension of the standard algorithm.

We now explain the structure of algorithm 2 in more detail. Initially, a random policy is exploited to explore the environment and collect useful data for the initialization of the SINDY model. For each state-action pair collected from the environment, the next state prediction is used to update the confidence ball according to Lemma 1. The center of the confidence ball $\widehat{W}$ can be computed online, initializing the matrix Λ and a vector P to zero value. For each new tuple state, action, and next state, they are updated with the following rules:

$$\Lambda_{h+1} \leftarrow \Lambda_h + \phi_h(s,a)\phi_h(s,a)^T \qquad P_{h+1} = P_h + s'\phi_h(s,a)$$

So that each row of the matrix $\widehat{W}$ can be computed as $\widehat{W}_{h,\iota} = (\Lambda_{h+1} + \alpha \cdot I)^{-1}P$. with α being an hyperparameter for regularization of the SINDy model and I the identity matrix. After this setup phase, the main algorithm follows a Dyna-style learning approach, alternating between interactions with the real and surrogate environment, where the real environment is used only for data gathering.

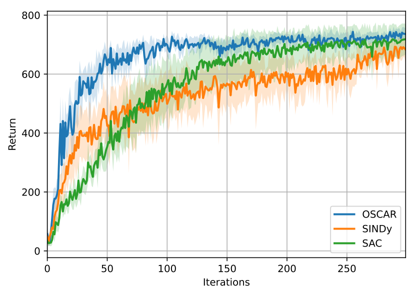
The original algorithm involves a nested optimization problem over actions and models (line 12). To address this in practice, we define an extended state s_w which includes both the observed environment state and the dynamical system parameters, enabling evaluation across all models within the confidence ball. To evaluate the *state-model-action* space, we adopt a Dyna-style approach with Soft Actor-Critic [Haarnoja et al., 2018, SAC], considering models randomly sampled from the confidence ball at each step. Model sampling is done using *Thompson Sampling* with posterior reshaping, drawing from a multivariate normal distribution $N(W^\star, \beta(\Lambda + \alpha \cdot I)^{-1})$ with $\beta \sim U[0, \lambda]$ with λ being a hyperparameter controlling the maximum scale of the covariance matrix Λ, to sample inside the confidence ball uniformly. To avoid storing model parameters in the reply buffer, SAC updates are performed online by simultaneously simulating batches of trajectories. For this reason, the neural networks of SAC are extended, including layer normalization as done in PQN Gallici et al. [2025] to stabilize learning. Model exploration is done by randomly sampling a vector of models from the confidence ball at each step, with each sampled state evaluated under its corresponding model. Action exploration is guided not only by SAC but also by enforcing actions that are optimal in similar models, i.e., taking the optimal actions for the sampled states under different models randomly sampled from the confidence ball. Finally, we collect the reward, update the policy based on the SAC algorithm, and update the SINDy model accordingly. This approach can also be parallelized to reduce computing time, considering the set of random models as a vectorized environment.

4.1 Experiments

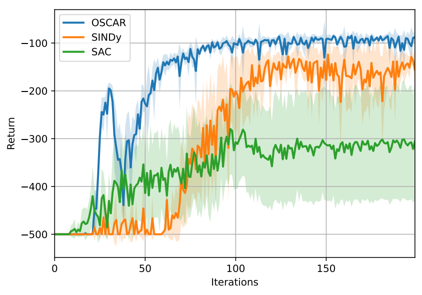
In this section, we present the results obtained by adopting our approach over different RL benchmarks. We conducted experiments using three environments of increasing complexity from the Gymnasium suite, Acrobot, Continuous Mountain Car [Towers et al., 2024], and the dm_control Swing Up Tassa et al. [2018]. We evaluate the algorithm by studying the effects of the confidence ball with respect to classical Soft Actor Critic and model-based Soft Actor Critic with SINDy. In each environment, the confidence ball dimension is obtained scaling $(\Lambda + \alpha \cdot I)^{-1}$ for a fixed hyperparameter λ. All the results presented are averaged over 10 different seeds. Appendix D provides additional experimental details. The code to reproduce the experiments is available in the supplementary material.

Swing-Up The Cartpole Swing-Up task involves swinging up and balancing a pole attached to a cart that moves horizontally. The state includes the cart position x, the cosine and sine of the pole angle θ, the cart velocity $\dot{x}$, and the pole angular velocity $\dot{\theta}$. The reward function incentivizes the agent to keep the pole up, keeping the cart in a central position. The environment considers episodes of 1000 time steps. Initially, a single episode of data is collected for SINDy and confidence ball initialization. Subsequently, for each episode of data collection in the real environment, 4 episodes are taken in the surrogate one for a total of $N_m = 4 \cdot 1000$. The batch size $b = 256$ and the scaling factor $\lambda = 0.01$. We can see in Figure 1a how, among the three approaches presented, OSCAR outperforms both methods, managing to converge to the optimal policy with significantly fewer iterations.

Acrobot The Acrobot environment consists of two linked pendulum-like arms, with the second joint actuated. The goal is to swing the free end upward to reach a target height starting from the downward position. The state is represented by the cosine and sine of the two angles θ_1, θ_2, considering that the angle of the second joint is relative to the first one, and their angular velocities $\dot{\theta}_1, \dot{\theta}_2$. The agent always receives a negative reward unless the target position is reached. Each episode lasts 500 time steps. Initially, a single episode of data is collected for SINDy and confidence ball initialization. Then, for each trajectory collected in the real environment, we collect one episode in the surrogate environment. The batch size is $b = 256$ and the scaling factor is $\lambda = 1.0$. Figure 1b shows the learning curves for all three approaches. OSCAR clearly outperforms the baselines, converging faster and reaching the optimal policy.

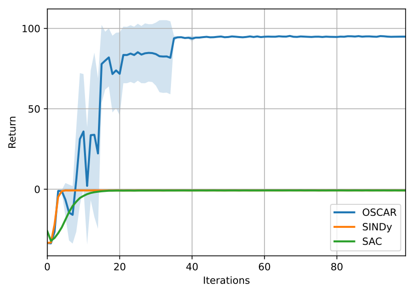


(a) Learning Curve for Swing Up

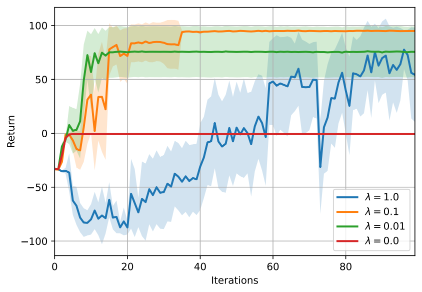


(b) Learing Curve for Acrobot

Figure 1: Learning Curves for Swing Up (a) and Acrobot (b)



(a) Learning Curve



(b) Ball dimension analysis

Figure 2: Learning Curves for Mountain Car

Mountain Car The Mountain Car environment objective is to bring a car to the top of the hill starting from the bottom of a sinusoidal valley. The state is represented by the position of the cart x and its velocity $\dot{x}$. The agent receives positive feedback only if the car reaches the goal position, making exploration essential to solve the task. The environment considers episodes of 1000 time steps and terminates if the goal is reached. Initially, a single episode of data is collected for SINDY and confidence ball initialization. Subsequently, for each episode of data collection in the real environment, 5 episodes are taken in the surrogate for a total of $Nm = 5 \cdot 1000$. The batch size $b = 512$ and the scaling factor $\lambda = 0.1$. We show in Figure 2a the learning curves between the three proposed methods. We notice that OSCAR is the only approach that manages to reach the goal of the task. In tasks where the exploration is crucial, the optimistic behavior of OSCAR permits a more efficient exploration, exploiting the model ensemble. Instead, approaches like SAC or model-based SAC with SINDY end up in a suboptimal behavior due to a lack of exploration. Furthermore, in Figure 2b we present an analysis over the λ parameter, which scales the radius of the confidence ball used. A larger λ corresponds to a more optimistic selection strategy, as the ball includes a wider range of models. However, we can notice that, while $\lambda = 1.0$ incentivizes a broader exploration, it slows down the learning of the agent due to querying over uncertain models. On the other hand, $\lambda = 0.01$ leads to a conservative behavior that limits exploration, resulting in suboptimal performance. Eventually, we notice that an intermediate, $\lambda = 0.1$ value manages to balance the trade-off between exploration and model reliability, achieving the optimal performance.

5 Conclusions

We have presented a unified theoretical and practical framework for RL in environments with sparse nonlinear dynamics. Our key theoretical contribution is algorithm 1 (DPCBE), which is endowed with a regret bound that scales with sparsity, a more general noise condition and misspecification, significantly extending existing guarantees for model-based RL. To bridge theory and application, we introduced algorithm 2 (OSCAR), a practical algorithm that uses SINDY to overcome the computational burden of DPCBE. Our experiments demonstrate that this approach not only maintains the theoretical spirit of optimism-driven exploration but also performs competitively or superiorly across classic continuous control benchmarks, especially in tasks where exploration is key.

References

Yasin Abbasi-Yadkori and Csaba Szepesvári. Regret bounds for the adaptive control of linear quadratic systems. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 1–26. JMLR Workshop and Conference Proceedings, 2011.

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24, 2011.

Yasin Abbasi-Yadkori, David Pal, and Csaba Szepesvari. Online-to-confidence-set conversions and application to sparse stochastic bandits. In *Artificial Intelligence and Statistics*, pages 1–9. PMLR, 2012.

Luigi Ambrosio, Nicola Fusco, and Diego Pallara. *Functions of bounded variation and free discontinuity problems*. Oxford university press, 2000.

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

Christopher G Atkeson and Juan Carlos Santamaria. A comparison of direct and model-based reinforcement learning. In *Proceedings of international conference on robotics and automation*, volume 4, pages 3557–3564. IEEE, 1997.

Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21, 2008.

Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin Yang. Model-based reinforcement learning with value-targeted regression. In *International Conference on Machine Learning*, pages 463–474. PMLR, 2020.

Steven J Bradtke, B Erik Ydstie, and Andrew G Barto. Adaptive linear quadratic control using policy iteration. In *Proceedings of 1994 American Control Conference-ACC'94*, volume 3, pages 3475–3479. IEEE, 1994.

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.

Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45): 22445–22451, 2019.

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.

Brian M. de Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J. Nathan Kutz, and Steven L. Brunton. Pysindy: A python package for the sparse identification of nonlinear dynamics from data, 2020. URL `https://arxiv.org/abs/2004.08424`.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

Jialin Dong and Lin Yang. Does sparsity help in learning misspecified linear bandits? In *International Conference on Machine Learning*, pages 8317–8333. PMLR, 2023.

Kenji Doya. Reinforcement learning in continuous time and space. *Neural computation*, 12(1): 219–245, 2000.

U. Fasel, J. N. Kutz, B. W. Brunton, and S. L. Brunton. Ensemble-sindy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 478(2260), April 2022. ISSN 1471-2946. doi: 10.1098/rspa.2021.0904. URL `http://dx.doi.org/10.1098/rspa.2021.0904`.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning, 2025. URL `https://arxiv.org/abs/2407.04811`.

Sébastien Gerchinovitz. Sparsity regret bounds for individual sequences in online linear regression. *The Journal of Machine Learning Research*, 14(1):729–769, 2013.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on learning theory*, pages 2137–2143. PMLR, 2020.

Sham Kakade, Akshay Krishnamurthy, Kendall Lowrey, Motoya Ohnishi, and Wen Sun. Information theoretic regret bounds for online nonlinear control. *Advances in Neural Information Processing Systems*, 33:15312–15325, 2020.

Alan Kaptanoglu, Brian de Silva, Urban Fasel, Kadierdan Kaheman, Andy Goldschmidt, Jared Callaham, Charles Delahunt, Zachary Nicolaou, Kathleen Champion, Jean-Christophe Loiseau, J. Kutz, and Steven Brunton. Pysindy: A comprehensive python package for robust sparse system identification. *Journal of Open Source Software*, 7(69):3994, January 2022. ISSN 2475-9066. doi: 10.21105/joss.03994. URL `http://dx.doi.org/10.21105/joss.03994`.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Tor Lattimore, Csaba Szepesvari, and Gellert Weisz. Learning with good feature representations in bandits and in rl with a generative model. In *International conference on machine learning*, pages 5662–5670. PMLR, 2020.

Davide Maran, Alberto Maria Metelli, Matteo Papini, and Marcello Restelli. Local linearity: the key for no-regret reinforcement learning in continuous mdps. *arXiv preprint arXiv:2410.24071*, 2024a.

Davide Maran, Alberto Maria Metelli, Matteo Papini, and Marcello Restelli. Projection by convolution: Optimal sample complexity for reinforcement learning in continuous-space mdps. *arXiv preprint arXiv:2405.06363*, 2024b.

Davide Maran, Alberto Maria Metelli, Matteo Papini, and Marcello Restelli. No-regret reinforcement learning in smooth mdps. In *Forty-first International Conference on Machine Learning*, 2024c.

Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):253–279, 2019.

Alex Simpkins. System identification: Theory for the user, (ljung, l.; 1999)[on the shelf]. *IEEE Robotics & Automation Magazine*, 19(2):95–96, 2012.

Alex Simpkins, Raymond De Callafon, and Emanuel Todorov. Optimal trade-off between exploration and exploitation. In *2008 American control conference*, pages 33–38. IEEE, 2008.

Alexander Strehl and Michael Littman. Online linear regression and its application to model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 20, 2007.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018. URL `https://arxiv.org/abs/1801.00690`.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments, 2024. URL https://arxiv.org/abs/2407.17032.

Gellért Weisz, András György, and Csaba Szepesvári. Online rl in linearly q^π-realizable mdps is as easy as in linear mdps if you learn what to ignore. *Advances in Neural Information Processing Systems*, 36:59172–59205, 2023.

Runzhe Wu, Ayush Sekhari, Akshay Krishnamurthy, and Wen Sun. Computationally efficient rl under linear bellman completeness for deterministic dynamics. *arXiv preprint arXiv:2406.11810*, 2024.

Andrea Zanette, Alessandro Lazaric, Mykel Kochenderfer, and Emma Brunskill. Learning near optimal policies with low inherent bellman error. In *International Conference on Machine Learning*, pages 10978–10989. PMLR, 2020.

Nicholas Zolman, Urban Fasel, J Nathan Kutz, and Steven L Brunton. Sindy-rl: Interpretable and efficient model-based reinforcement learning. *arXiv preprint arXiv:2403.09110*, 2024.

Despite the synergy between model-based RL and continuous control, little work has explored how SINDy-like sparse assumptions might inform theoretical developments in RL. While some existing results in RL address linear or factored representations of the transition dynamics, these models are fundamentally different and not well-suited for physical environments. Seminal papers Jin et al. [2020], Ayoub et al. [2020], which try, under different assumptions, to enforce linearity of the transition function with respect to a feature map, do not trivially work under an assumption like (1). Even if there is a small literature in RL Kakade et al. [2020] that deals with an assumption similar to equation (1), to the best of our knowledge, there is still no theoretical work that enforces sparsity on the model parameters. In fact, the only works that study model-based RL under a sparse dynamical system assumption Arora et al. [2022], Zolman et al. [2024] are very recent, and mostly empirical.

A Related Works

While the majority of RL papers focus on a transition function of the form $s_{h+1} \sim P_h(\cdot|s_h, a_h)$, there is still a huge literature that, lying at the intersection between RL and control theory, considers systems where $s_{h+1} = f_h(s_h, a_h) + \eta_h$, as in this paper. Most of these works assume that the additive noise term η_h is Gaussian, which we do not. This assumption is classical in the control community, appearing in early works on adaptive control and reinforcement learning for linear systems with Gaussian disturbances (e.g., Bradtke et al. [1994], Abbasi-Yadkori and Szepesvári [2011]). In the nonlinear setting, Doya [2000] and Simpkins et al. [2008] model continuous-time systems subject to Brownian or Gaussian process noise, and derive actor-critic or Bayesian optimal policies accordingly.

From a more practical point of view, although models of this type have been known for a long time Atkeson and Santamaria [1997], recent model-based RL approaches "re-discovered" this assumption to capture uncertainty in learned dynamics. For instance, PILCO Deisenroth and Rasmussen [2011] and PETS Chua et al. [2018] explicitly posit $s' \sim \mathcal{N}(\phi(s, a), \Sigma)$ and exploit the additive structure to propagate uncertainty during planning. These approaches, however, do not typically impose or exploit sparsity in the transition model.

SINDy On the point of view of the application, the works that are closer to our paper are the ones that try to leverage the powerful SINDy algorithms on top of RL methods. The usage of SINDy has been paired with a classical model-based approach. In Arora et al. [2022], the authors propose a Dyna-style learning algorithm adopting SINDy as a method to estimate the surrogate environment paired with the state-of-the-art model-free algorithm. Despite the positive practical results, the authors have not presented any theoretical guarantee on the approach. A more complex approach has been proposed in the literature, which involves the usage of Deep Reinforcement Learning (DRL) and dictionary learning based on the sparse identification provided by SINDy Zolman et al. [2024]. However, unlike the cited methodologies, our approach also provides theoretical guarantees on the algorithm, paired with competitive practical results.

RL theory with linear function approximation From the point of view of theory, we are the first paper providing a regret bound in our setting under the sparsity assumption. While regret bounds for the LQR are already well-known, papers generalizing these results to an arbitrary non-linear feature map of the state-action pair (like in our case (2)) are relatively novel Kakade et al. [2020]. With respect to the latter paper, we generalize over three sides: we use a sparsity assumption, we extend the family of the noise allowed for the model, from Gaussians to any density of bounded variation, and we allow for model misspecification. Related to the first point, the work of Abbasi-Yadkori et al. [2012] is worth mentioning, as it solves the problem of sparse linear bandits. The bounded variation assumption on the density of the noise is, to the best of our knowledge, novel in the RL setting, while this class of functions plays a central role in variational calculus and partial differential equations Ambrosio et al. [2000]. Lastly, introducing the misspecification into a linear function approximation scheme for RL is known to introduce one problem that we briefly mention in subsection 3.1: the influence of ξ_∞ in the regret bound is multiplied by $\sqrt{d}$. As Lattimore et al. [2020] argues, this problem arises from the fact that, in bandit-RL settings, the agent needs uniform control over the error of the model on the state-action space, and is not solvable in general. In fact, several recent seminal works in RL with linear function approximation Zanette et al. [2020] Weisz et al. [2023] Wu et al. [2024] provide regret bounds that suffer from misspecification amplification. To deal with this problem was the main goal of multiple papers, like Maran et al. [2024a], which removes the amplification factor in case of a locally linear feature map, and Dong and Yang [2023], which improves the $\sqrt{d}$ amplification in case of sparsity, but just in case of bandits, not in RL. None of these works is able to tackle the problem in our setting, so that, at the current state-of-the-art, misspecification amplification remains one big open problem in RL with linear function approximation.

B Notation

In this section, we leave, for the reader's convenience, a table of the notation used in this paper.

$\mathcal{S}$	State space $\subset \mathbb{R}^{p_\mathcal{S}}$
$\mathcal{A}$	Action space $\subset \mathbb{R}^{p_\mathcal{A}}$
H	Time horizon
f_h	Transition function $\mathcal{S} \times \mathcal{A} \to \mathcal{S}$ at stage $h \in \{1, \dots, H\}$
η_h	Noise at step h
μ_h	Density function of the noise at step h
r_h	Reward function $\mathcal{S} \times \mathcal{A} \to [0, 1]$
K	Number of episodes
π_h^k	Policy played at stage h of episode k
s_h^k	state visited at stage h of episode k
a_h^k	action played at stage h of episode k
V_h^*	Optimal value function
R_K	Regret
$W_h^\star$	System matrix (see equation 2)
ϕ_h	Feature map (see equation 2)
d	Dimension of the feature maps ϕ_h
d_0	Sparsity parameter (see assumption 1)

C Proofs

Theorem 3. *Fix an error probability $\delta > 0$, $\iota \in \{1, \dots p_\mathcal{S}\}$ and $h \in \{1, \dots H\}$. Let $\widehat{y}_{h,\iota}^k$ be the output of* SEQSEW *for the sequences $\phi_h(s_h^k, a_h^k), s_{h+1,\iota}^k$, that are indexed by k. Define the* ***confidence balls***

$$\mathcal{C}_{h,\iota}^k := \left\{ \theta \in \mathbb{R}^d : \sum_{\tau=1}^k (\widehat{y}_{h,\iota}^\tau - \phi_h(s_h^\tau, a_h^\tau)^\top \theta)^2 \le \beta(\delta) + \|W_{h,\iota}^\star\|_2^2 \right\}, \tag{4}$$

where $W_{h,\iota}^\star$ is the $\iota-$th row of matrix $W_h^\star$, and $\beta(\delta) = 1 + 2\rho_{\max} + 32 \log\left(\frac{\sqrt{8}+\sqrt{1+\rho_{\max}}}{\delta}\right) = \widetilde{\mathcal{O}}(\rho_{\max} + \log(1/\delta))$ (where $\rho_{\max}$ comes from equation 3). Then, $\boxed{\mathbb{P}(\exists k : W_{h,\iota}^\star \notin \mathcal{C}_{h,\iota}^k) \le \delta}$.

Proof. In this proof, we fix h, ι and call

$$\boldsymbol{x}_k = \phi_h(s_h^k, a_h^k) \qquad \widehat{y}_k = \widehat{y}_{h,\iota}^k.$$

Under our transition (2)

$$y_k = \boldsymbol{x}_k^\top \theta^\star + \eta_k \qquad \theta^\star := W_{h,\iota}^\star.$$

At this point applying Theorem 23.4 from Lattimore and Szepesvári [2020] gives, with probability $1-\delta$, $\theta^\star \in \mathcal{C}^k$, at the same time for every $k = 1, \dots K$

$$\mathcal{C}^k := \left\{ \theta \in \mathbb{R}^d : \|\theta\|_2^2 + \sum_{\tau=1}^{k} (\widehat{y}_\tau - \boldsymbol{x}_\tau^\top \theta)^2 \leq \beta(\delta) + \|\theta^\star\|_2^2 \right\},$$

with our definition of $\beta(\delta)$. Erasing $\|\theta\|_2^2$, which is non-negative, makes only the set bigger. This ends the proof. □

Lemma 1. *For every h, k, ι, let $\Lambda_h^k := I + \sum_{\tau=1}^k \phi_h(s_h^k, a_h^k)\phi_h(s_h^k, a_h^k)^\top$, where I stands for the identity matrix of size d. Moreover, let $\widehat{W}_{h,\iota}^k := (\Lambda_h^k)^{-1} \sum_{\tau=1}^k \widehat{y}_{h,\iota}^k \phi_h(s_h^k, a_h^k)$. Then,*

$$\mathcal{C}_{h,\iota}^k \subseteq \left\{ \theta \in \mathbb{R}^d : \|\theta - \widehat{W}_{h,\iota}^k\|_{\Lambda_h^k}^2 \leq \beta(\delta) + \|W_{h,\iota}^\star\|_2^2 \right\}.$$

Proof. Since, also in this case, h, ι are fixed, we keep calling

$$\boldsymbol{x}_k = \phi_h(s_h^k, a_h^k) \qquad \widehat{y}_k = \widehat{y}_{h,\iota}^k,$$

and

$$y_k = \boldsymbol{x}_k^\top \theta^\star + \eta_k \qquad \theta^\star = W_{h,\iota}^\star \qquad \widehat{\theta}_k = \widehat{W}_{h,\iota}^k \qquad \Lambda_k = \Lambda_{h,\iota}^k.$$

In this notation, and using theorem 3, we have

$$\mathcal{C}^k = \left\{ \theta \in \mathbb{R}^d : \|\theta\|_2^2 + \sum_{\tau=1}^{k} (\widehat{y}_\tau - \boldsymbol{x}_\tau^\top \theta)^2 \leq \beta(\delta) + \|\theta^\star\|_2^2 \right\},$$

and we want to show it is contained in

$$\widetilde{\mathcal{C}}^k = \left\{ \theta \in \mathbb{R}^d : \|\widehat{\theta}_k - \theta\|_{\Lambda_k}^2 \leq \beta(\delta) + \|\theta^\star\|_2^2 \right\}.$$

This from the identity (see Exercise 23.5 from Lattimore and Szepesvári [2020])

$$\|\widehat{\theta}_k - \theta\|_{\Lambda_k}^2 + \|\widehat{\theta}_k\|_2^2 + \sum_{\tau=1}^{k} (\widehat{y}_\tau - \boldsymbol{x}_\tau^\top \widehat{\theta}_k)^2 = \|\theta\|_2^2 + \sum_{\tau=1}^{k} (\widehat{y}_\tau - \boldsymbol{x}_\tau^\top \theta)^2,$$

as we are adding two terms that are nonnegative to the LHS. □

C.1 Regret bound

The regret bound builds on the following fundamental properties of bounded variation noises.

Proposition 9. *Let $\mu(\cdot) \in BV(\chi)$. Then, for any $x_0 \in \mathbb{R}^p$,*

$$TV(\mu(\cdot), \mu(\cdot + x_0)) \leq \chi \|x_0\|.$$

where TV denotes the total variation distance.

Proof. By definition,

$$\mathrm{TV}(\mu(\cdot), \mu(\cdot + x_0)) = \sup_{f \in \mathcal{C}(\mathbb{R}^p),\ \|f\|_{L^\infty} \leq 1} \int_{\mathbb{R}^p} (f(y) - f(y - x_0))\, \mu(y) dy.$$

Now, fix a function f. By Lagrange's theorem,

$$\int_{\mathbb{R}^p} (f(y) - f(y - x_0))\ \mu(y)dy \leq \|x_0\| \sup_{x \in \mathbb{R}^p} \left\| \nabla_x \int_{\mathbb{R}^p} f(y - x)\ \mu(y)dy \right\|.$$

At this point, we have

$$\begin{aligned}
\|x_0\| \sup_{x \in \mathbb{R}^p} \left\| \nabla_x \int_{\mathbb{R}^p} f(y - x)\ \mu(y)dy \right\| &= \|x_0\| \sup_{x \in \mathbb{R}^p} \left\| \nabla_x \int_{\mathbb{R}^p} f(y)\ \mu(y + x)dy \right\| \\
&= \|x_0\| \sup_{x \in \mathbb{R}^p} \left\| \int_{\mathbb{R}^p} f(y)\ \nabla_x \mu(y + x)dy \right\| \\
&\leq \|x_0\| \sup_{x \in \mathbb{R}^p} \|\nabla \mu(\cdot)\|_{\mathcal{M}} \leq \|x_0\|\chi.
\end{aligned}$$

The second passage comes from the fact that, using the weak gradient, we can bring it under the integral sign under the condition that the integral is finite, which holds since f is bounded μ is a probability density function. □

Theorem 10. *Assume 1 and 4 which holds for $\chi_h \leq \chi$ for every $h = 1, \dots H - 1$. Then, with probability at least $1 - \delta$, algorithm 1 achieves the following regret bound* $\boxed{R_K \leq \widetilde{\mathcal{O}}(H^2 \chi p_{\mathcal{S}} \sqrt{d_0 d K})}$.

Proof. We work under the event defined by theorem 3, which has probability at least $1 - \delta$. Under this even, the true matrix $W^\star_{h,\iota}$ of the transition at step h (component ι) belongs to $\mathcal{C}^k_{h,\iota}$ at any episode. Since algorithm 1 chooses its policy by maximizing over the possible transitions in $\mathcal{C}^k_{h,\iota}$. We the following bound on the estimated value function at each episode

$$V_1^*(s) \leq V_1^k(s) \qquad \forall s_1 \in \mathcal{S}.$$

Thus, the regret can be written in the following form

$$\begin{aligned}
R_K &= \sum_{k=1}^K V_1^*(s_1^k) - V_1^{\pi_k}(s_1^k) &(6)\\
&\leq \sum_{k=1}^K V_1^k(s_1^k) - V_1^{\pi_k}(s_1^k) &(7)\\
&\leq \sum_{k=1}^K \sum_{h=1}^H \mathbb{E}_{s_h^k, a_h^k}[(\widehat{P}_h^k(\cdot|s_h^k, a_h^k) - P_h(\cdot|s_h^k, a_h^k))(V_h^{\pi_k})], &(8)
\end{aligned}$$

where step 8 comes from the performance difference lemma and we have defined $\widehat{P}_h^k(\cdot|s, a)/P_h(\cdot|s, a)$ to be the estimated/true distribution of s' following s, a. Indeed, by equation 2, these correspond to

$$\mu_h(\cdot - W_h^\star \phi_h(s, a)) \qquad \mu_h(\cdot - \widehat{W}_h^k \phi_h(s, a)).$$

Using the fact that $V_h^{\pi_k}$ is bounded by H, we can proceed as follows.

$$\begin{aligned}
\text{R.H.S.} &\leq H \sum_{k=1}^K \sum_{h=1}^H \mathbb{E}_{s_h^k, a_h^k}[\text{TV}((\widehat{P}_h^k(\cdot|s_h^k, a_h^k) - P_h(\cdot|s_h^k, a_h^k))] \\
&\leq H \sum_{k=1}^K \sum_{h=1}^H \sum_{\iota=1}^{p_{\mathcal{S}}} 2 \wedge \mathbb{E}_{s_h^k, a_h^k}[|W^\star_{h,\iota} \phi_h(s_h^k, a_h^k)) - \widehat{W}^k_{h,\iota} \phi_h(s_h^k, a_h^k)|],
\end{aligned}$$

where in the second passage we have used proposition 9, dividing the norm into the $p_\mathcal{S}$ component of the state (indeed $\|x\| \le \|x\|_1 \le |x_1| + |x_2| + \dots$). The presence of 2 takes into account that the total variation distance can never exceed 2. We then proceed as follows:

$$\begin{aligned}
\text{R.H.S.} &\le H \sum_{k=1}^{K} \sum_{h=1}^{H} \sum_{\iota=1}^{p_\mathcal{S}} 2 \wedge \mathbb{E}_{s_h^k, a_h^k}[\|W_{h,\iota}^\star - \widehat{W}_{h,\iota}^k\|_{\Lambda_{h,\iota}^k} \|\phi_h(s_h^k, a_h^k)\|_{(\Lambda_{h,\iota}^k)^{-1}}] && (9)\\
&\le H^2 \mathbb{E}\left[\sum_{k=1}^{K} \sum_{\iota=1}^{p_\mathcal{S}} 2 \wedge \|W_{h,\iota}^\star - \widehat{W}_{h,\iota}^k\|_{\Lambda_{h,\iota}^k} \|\phi_h(s_h^k, a_h^k)\|_{(\Lambda_{h,\iota}^k)^{-1}}\right] && (10)\\
&= H^2 \mathbb{E}\left[\sum_{\iota=1}^{p_\mathcal{S}} \sum_{k=1}^{K} 2 \wedge \|W_{h,\iota}^\star - \widehat{W}_{h,\iota}^k\|_{\Lambda_{h,\iota}^k} \|\phi_h(s_h^k, a_h^k)\|_{(\Lambda_{h,\iota}^k)^{-1}}\right] && (11)\\
&\le H^2 \mathbb{E}\left[\sum_{\iota=1}^{p_\mathcal{S}} \sqrt{K} \sqrt{\sum_{k=1}^{K} 4 \wedge \|W_{h,\iota}^\star - \widehat{W}_{h,\iota}^k\|_{\Lambda_{h,\iota}^k}^2 \|\phi_h(s_h^k, a_h^k)\|_{(\Lambda_{h,\iota}^k)^{-1}}^2}\right] && (12)\\
&\le H^2 \mathbb{E}\left[\sum_{\iota=1}^{p_\mathcal{S}} \sqrt{K} \sqrt{\sum_{k=1}^{K} 4(\beta_T(\delta) + m_2)\left(1 \wedge \|\phi_h(s_h^k, a_h^k)\|_{(\Lambda_{h,\iota}^k)^{-1}}^2\right)}\right] && (13)\\
&= H^2 \mathbb{E}\left[\sum_{\iota=1}^{p_\mathcal{S}} \sqrt{4K(\beta_K(\delta) + m_2)} \sqrt{\sum_{k=1}^{K} \left(1 \wedge \|\phi_h(s_h^k, a_h^k)\|_{(\Lambda_{h,\iota}^k)^{-1}}^2\right)}\right] && (14)\\
&\lesssim H^2 p_\mathcal{S} \sqrt{(\beta_K(\delta) + m_2)K} \sqrt{d \log(K)}. && (15)
\end{aligned}$$

In the previous derivation, we have used the shortcut $m_2 = \max_{h,\iota} \|W_{h,\iota}^\star\|_2^2$. Step 12 comes from the Cauchy-Schwartz inequality. Step 13 comes from our lemma 1. Step 15 comes from the elliptical potential lemma (Lemma 11 in Abbasi-Yadkori and Szepesvári [2011]). Replacing the definition of $\beta_K(\delta)$ ends the proof. □

Theorem 8. *Assume 7 and 4 which holds for $\chi_h \le \chi$ for every $h = 1, \dots H-1$. Let us modify the construction of the confidence sets in equation 4 by using as confidence radius $\beta'(\delta) = \beta(\delta) + K\xi_\infty^2 \log(1/\delta)$. Then, with probability at least $1-\delta$, algorithm 1 achieves the following regret bound*

$$\boxed{R_K \le \widetilde{\mathcal{O}}\Big(H^2 \chi p_\mathcal{S} \sqrt{d_0 d K} + H^2 p_\mathcal{S} \sqrt{d} K \xi_\infty\Big)}.$$

Proof. The proof of this result is very similar to the one of the previous theorem 10. First, we have to show that, letting $\beta'(\delta) = \beta(\delta) + K\xi_\infty^2 \log(1/\delta)$, the optimism is maintained, even with misspecification. This time, we cannot use directly the result by Lattimore and Szepesvári [2020] and we need to derive the result from theorem 2, assuming that the samples take the form

$$y_t = \boldsymbol{x}_t^\top \theta + \xi(\boldsymbol{x}_t) + \eta_t,$$

where η_t is an i.i.d. noise. Our goal is to bound

$$Q_T := \sum_{t=1}^{T} (x_t^\top \theta - \widehat{y}_t)^2, \tag{16}$$

so that, for $\theta = \theta^\star$, we can use this bound to build a confidence ball. Indeed, theorem 2, (which makes no assumptions of y_t being well-specified), gives, for each $\theta \in \mathbb{R}^d$

$$\rho_T(\theta) := \sum_{t=1}^{T} (y_t - \widehat{y}_t)^2 - (y_t - \boldsymbol{x}_t^\top \theta)^2 \le \rho_{\max}.$$

We have, by definition, calling $\zeta_t := \eta_t + \xi(x_t)$,

$$\begin{aligned}
\sum_{t=1}^{T}(x_t^\top \theta - \widehat{y}_t)^2 &= \rho_T(\theta) + \sum_{t=1}^{T}(x_t^\top \theta - \widehat{y}_t)^2 - (y_t - \widehat{y}_t)^2 + (y_t - x_t^\top \theta)^2 \\
&= \rho_T(\theta) + \sum_{t=1}^{T} x_t^\top \theta^2 + \widehat{y}_t^2 - 2x_t^\top \theta \widehat{y}_t - y_t^2 - \widehat{y}_t^2 + 2\widehat{y}_t y_t + x_t^\top \theta^2 - 2y_t x_t^\top \theta + y_t^2 \\
&= \rho_T(\theta) + \sum_{t=1}^{T} 2x_t^\top \theta^2 - 2x_t^\top \theta \widehat{y}_t + 2\widehat{y}_t y_t - 2y_t x_t^\top \theta \\
&= \rho_T(\theta) + \sum_{t=1}^{T} 2x_t^\top \theta^2 - 2x_t^\top \theta \widehat{y}_t + 2\widehat{y}_t(x_t^\top \theta + \zeta_t) - 2(x_t^\top \theta + \zeta_t)x_t^\top \theta \\
&= \rho_T(\theta) + \sum_{t=1}^{T} 2\widehat{y}_t \zeta_t - 2\zeta_t x_t^\top \theta \\
&= \rho_T(\theta) + \sum_{t=1}^{T} 2(\widehat{y}_t - x_t^\top \theta)\zeta_t.
\end{aligned}$$

We can rearrange the last result as follows:

$$\sum_{t=1}^{T} 2(\widehat{y}_t - x_t^\top \theta)\zeta_t = \sum_{t=1}^{T} 2(\widehat{y}_t - x_t^\top \theta)\xi(x_t) + \sum_{t=1}^{T} 2(\widehat{y}_t - x_t^\top \theta)\eta_t.$$

These two terms can be bounded by means of Q_T (equation (16)), as follows: i) the random variable $\sum_{t=1}^{T} 2(\widehat{y}_t - x_t^\top \theta)\eta_t$ is $\sigma-$subgaussian with $\sigma = \sqrt{Q}_t$. Therefore, by Hoeffding's inequality, w.p. $1-\delta$,

$$\sum_{t=1}^{T} 2(\widehat{y}_t - x_t^\top \theta)\eta_t \leq 2\sqrt{Q_T \log(T/\delta)}.$$

ii) By Cauchy-Schwartz inequality:

$$\sum_{t=1}^{T}(\widehat{y}_t - x_t^\top \theta)\xi(x_t) \leq \sqrt{Q_T \sum_{t=1}^{T} \xi(x_t)^2} \leq \sqrt{Q_T T \xi_\infty^2}.$$

Putting everything together, we have shown the following bound on Q_T (eq. 16):

$$Q_T \leq \rho_{\max} + \sqrt{Q_T(2\log(T/\delta) + T\xi_\infty^2)},$$

which is satisfied by $Q_T = \beta'(\delta) = \beta(\delta) + T\xi_\infty^2 \log(1/\delta) = \widetilde{\mathcal{O}}(\rho_{\max} + T\xi_\infty^2 \log(1/\delta))$. This proves that

$$\sum_{t=1}^{T}(x_t^\top \theta^\star - \widehat{y}_t)^2 \leq \beta'(\delta).$$

Replacing back $x_t = \phi_h(s_h^k, a_h^k), y_t = s_{h+1,\iota}^k, \theta^\star = W_{h,\iota}^\star$, and following the same passages of the proofs of theorem 3 and lemma 1, this leads to

$$\mathcal{C}_{h,\iota}^k \subseteq \left\{\theta \in \mathbb{R}^d : \|\theta - \widehat{W}_{h,\iota}^k\|_{\Lambda_h^k}^2 \leq \beta'(\delta) + \|W_{h,\iota}^\star\|_2^2\right\},$$

with probability $1-\delta$. This proves the optimism, and allows us to bound the regret as

$$R_K = \sum_{k=1}^{K} V_1^*(s_1^k) - V_1^{\pi_k}(s_1^k).$$

The sequent passages follow as in the proof of theorem 10, until we get

$$H \sum_{k=1}^{K} \sum_{h=1}^{H} \mathbb{E}_{s_h^k, a_h^k}[\mathrm{TV}((\widehat{P}_h^k(\cdot|s_h^k, a_h^k) - P_h(\cdot|s_h^k, a_h^k))].$$

This term is bounded by

$$H \sum_{k=1}^{K} \sum_{h=1}^{H} \sum_{\iota=1}^{p_{\mathcal{S}}} \left(2 \wedge \mathbb{E}_{s_h^k, a_h^k}[|W_{h,\iota}^{\star} \phi_h(s_h^k, a_h^k)) - \widehat{W}_{h,\iota}^k \phi_h(s_h^k, a_h^k)|] + 2\xi_\infty\right),$$

where the first part is bounded as in the previous theorem, giving $H^2 p_{\mathcal{S}} \sqrt{(\beta_K'(\delta) + m_2)K} \sqrt{d \log(K)}$, and the second one gives

$$H \sum_{k=1}^{K} \sum_{h=1}^{H} \sum_{\iota=1}^{p_{\mathcal{S}}} 2\xi_\infty \leq 2H^2 K p_{\mathcal{S}} \xi_\infty.$$

Since $\beta_K'(\delta) = \widetilde{\mathcal{O}}(\rho_{\max} + K\xi_\infty^2)$, we get the statement. $\square$

D Experimental Settings and Additional Results

In this appendix, we discuss the experimental setting for the simulations provided in the main paper, and we provide further results.

D.1 Experimental setting

Algorithm Setting For each test, the policy adopted is parametrized using a neural network with 2 hidden layers and 256 nodes per layer for both the actor and the critic using a ReLu as a non-linear layer. The model-based SINDY version adopts two different reply buffers, one filled with data collected from the real environment and one filled with data from the surrogate one. Differently, for our implementation, a single reply buffer is employed since data generated using the confidence ball is consumed directly from the network for a gradient step. We use PySINDy's E-SINDy [de Silva et al., 2020, Kaptanoglu et al., 2022] implementation with an ensemble of 20 models with STLRidge and the coefficients of all the models are *ensembled*, taking the median of the coefficients. All the trained models use a threshold of 1×10^{-3} and RIDGE regularization of 1×10^{-3} for all the experiments.

Experiments have been conduced with a discount factor $\gamma = 0.99$ and a learning rate of $\eta_a = 3 \cdot 10^{-4}$ for the actor and $\eta_c = 1 \cdot 10^{-3}$ for the critic except for the Acrobot environment where the learning rate of the critic is lowered to $\eta_c = 3 \times 10^{-4}$. Since the model error accumulates at each step the horizon of trajectories in the surrogate environment needs to be tuned in relation to the quality of the model. Following [Zolman et al., 2024, Appendix D], rollouts in the surrogate environment are truncated, and the surrogate environment is reset every time the predicted state exceeds certain thresholds. We assume that physical limitations of the environment are well known a priori, i.e., limits that could damage the physical assets, and we used those as boundaries, as shown in Table 1.

Computational Resources. All the experiments are run on a server equipped as follows:

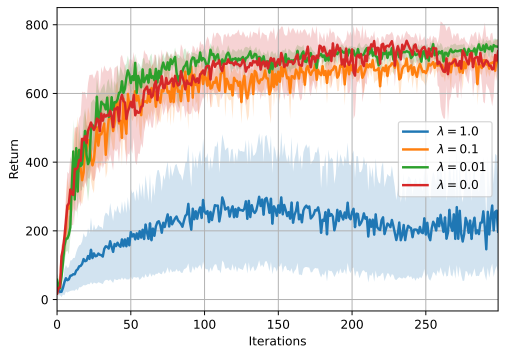
- CPU: Intel Xeon Gold 6238R (112 cores, 2.20 GHz);
- RAM: 256GB GB;

In particular, $N = 300$ trajectories for each environments with $H = 1000$ scored ≈ 58.82 iterations per second for SAC and ≈ 12.71 iterations per second with model-based SINDY, while ≈ 8.92 iterations per second for OSCAR. All the performance are run over a single CPU core.

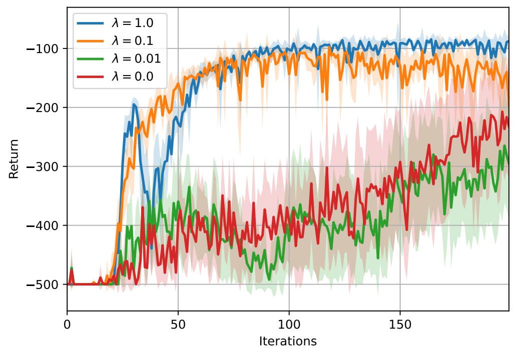
Environment	Threshold
Swing-up	$\lvert x\rvert < 5$ $\lvert\dot{x}\rvert < 10$ $\lvert\cos\theta\rvert < 1.1$ $\lvert\sin\theta\rvert < 1.1$ $\lvert\dot{\theta}\rvert < 10$
Mountain Car	$x < 0.45$ $x > -1.2$ $\lvert\dot{x}\rvert < 0.07$
Acrobot	$\lvert\cos\theta_1\rvert < 1.1$ $\lvert\sin\theta_1\rvert < 1.1$ $\lvert\cos\theta_2\rvert < 1.1\lvert\sin\theta_2\rvert < 1.1$ $\lvert\dot{\theta}_1\rvert < 4\pi$ $\lvert\dot{\theta}_2\rvert < 9\pi$

Table 1: Model Thresholds

Additional Results In Figure 3 we present an analysis of the performance of OSCAR w.r.t. the scale factor λ in the Swing-Up and Acrobot environment. In Figure 3a, we can notice how small values of λ manage to converge to the optimal behavior, while forcing a broader exploration ($\lambda = 1$) leads us to a suboptimal solution. Since the environment provides dense reward that incentivizes keeping the pole upright and the cart centered, policies that focus their exploration around the optimal region (i.e., the upright position) receive more consistent learning signals. An excessive exploration can thus lead to policies that swing the pole but fail to stabilize it. On the other hand, in environments like Acrobot, with sparse reward signals, wider exploration helps find the optimal solution. We can, in fact, notice, in Figure 3b, how with higher values of λ we manage to converge faster to the optimal solution.



(a) Swing-Up



(b) Acrobot

Figure 3: Ball Dimension Analysis