SetPoint Learning : Toward an approach to control them all

Joëlle FERZLY [*1] **Boubker TABITI** [*] **Fabien FURFARO** [*1]

Abstract

Deep learning has enabled significant advancements in the field of dynamic system control, particularly in robotics, video games, navigation, and energy management, especially for non-linear systems. However, this type of learning does not allow for generalization to all dynamic systems, and sometimes, controllers based on system modeling (MPC, PID, MOR) are more relevant than methods based on stochastic data, especially for cases where the system is linearizable, and the target setpoint is well-known, which corresponds to most cases in the electronic, mechanical, and thermal domains. We introduce a new learning method that is generalizable to all linear dynamic systems where the setpoint is known and adaptive. We show that when targeting a setpoint, neural networks adapt towards an objective and would be a promising approach for the use of generative diffusion models for direct interaction with multiple real-world environments in parallel. This study has led to the creation of a set of simulation environments that can serve as a basis for other reinforcement or data-based approaches.

Keywords : Reinforcement Learning, Adaptative control, Gym environment, Time invariant linear systems.

1 Introduction

The planning of decision-making in specific control environments is often a relatively simple problem to solve using data-driven approaches and reinforcement learning. In linearizable environments, planning can be reduced to a trajectory optimization to be achieved and can be done using controllers (https://doi.org/10.1016/S0005-1098(98)00178-2). In reinforcement learning, we observe results that surpass linear or data-based control. We can distinguish the DQN algorithms based on value and PPO based on policy.

The previous methods have worked well in specific situations, but they often require a lot of work to study the system and optimize the parameters for each case [Source]. This lack of ability to work in different situations has led to growing interest in approaches that can create agents that can adapt to different environments [Source]. In reinforcement learning, the ability of an agent to perform well in different environments is a big challenge. Agents can do well in specific training setups, but their performance drops a lot when faced with new situations.

For this, several alternatives have been proposed... [Aiding, Navigation with Human Feedloop, Planning diffusion, Google Genie, JEPA https://arxiv.org/pdf/2301.08243]. However, all of these latter techniques are complex and do not allow generalization to all systems.

In this work, we have developed a simple approach, which makes it possible to generalize the control of a linearizable system, as well as a set of control environments to adapt to any type of algorithm. learning. (...) the main hypothesis is that learning models always have the objective of achieving the best results, whereas here we assume that intermediate results where we aim for a lower score, allows us to expect better results as would a human with a trainer who lowers his level for his student. This approach is thus inspired in a certain way by the human learning approach used in educational psychology.

2 Background and contribution

Our approach is a analogue of past work in behavioral synthesis using trajectory optimization (MPC, Witkin Kass, 1988; Tassa et al., 2012). But we propose here an approach where the variables are always the same whatever the physical system and where the objective to be achieved is defined in the controller input. This approach is also compatible for the use of generative models.

∗ Equal contribution. 1 Capgemini Engineering, Meudon, France. Correspondence to: **fabien.furfaro@capgemini.com**.
Arxiv preprint, Paris, 2024.

Code and visualizations of the learned process are available at **setpoint-learning.github.io**.

Our method relies on modifying the states of the simulation environment through the use of a Wrapper or an environment we developed. This enables our approach to interact more dynamically with the system.

3 Problem Setting

Consider a system governed by discrete-time dynamics

$$S_{t+1} = f(S_t, a_t), \tag{1}$$

where S_t is the state of the system at time t, a_t is the control action at time t, and f represents the system dynamics function. For a linear system, the dynamics can be expressed by the following equation:

$$y(t) = s(t) * h(t), \tag{2}$$

where $y(t)$ is the output signal, $s(t)$ is the input action from the controlling system, $h(t)$ represents the impulse response of the system, and $*$ denotes the convolution operation. In this context, the function f in (1) is the convolution operation $f(S_t, a_t) = a_t * h(t)$. The objective is to achieve the optimal control input $u^*(t)$ that minimizes a given cost function $J(S)$:

$$u^*(t) = \arg\min J(S). \tag{3}$$

The main challenge lies in designing a model that can solve problem (1) regardless of the specific physical system.

4 Setpoint Learning method

In this work, we present a setpoint learning approach designed to improve setpoint tracking by incorporating a target guideline. This method mainly relies on selecting a set of primary input variables from initial observations, according to their relevance in understanding system dynamics and achieving predefined targets. Additionally, this approach uses a target setpoint as a reference to guide the learning process. The key variables include:

- a_{t-1} : the action taken in the preceding state,
- s_{t-1} : the state of the system in the previous timestep,
- s_t : the current state of the system
- $(s_{\text{sp}})_{t+1}$: the setpoint, or target state, to be achieved at the next timestep.

By incorporating the previous action a_{t-1}, we introduce a temporal dimension, allowing the models to learn from past actions and adapt to changing dynamics across different contexts. By directing actions towards the convergence of the current state s_t to the target state s_{sp} at the next timestep, this method ensures goal-oriented behavior while maintaining adaptability, as presented in Figure 1.

The setpoint learning the method relies on selecting $a_{(i,t-1)}$ from the action space A, the states $s_{(j,t-1)}$ and $s_{(j,t)}$ from the state space S, and the target setpoint $s_{c(j,t+1)}$ to be achieved at the next timestep, resulting in the state space $\tilde{S}_t$ at time t, as describes in the following equation

$$\left\{a_{(i,t-1)}, s_{(j,t-1)}, s_{(j,t)}, s_{\text{sp}(j,t+1)}\right\} \longrightarrow \tilde{S}_t \tag{4}$$

Here, i represents the index of the action a within the set A, and j represents the index of the state s within the set S.

This work aims to demonstrate two key points: (i) a model trained on an environment using the variables defined in equation (4) can control any other linear environment that incorporates the same set of variables; and (ii) this model is capable of tracking any given setpoint.

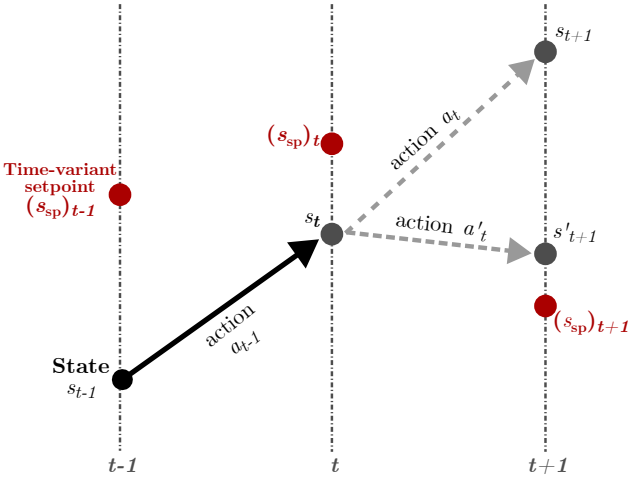

Figure 1. Representative diagram of the target setpoint concept. At time t, the agent in state s_t should choose the action in order to get as close as possible to the target state (setpoint s_{sp}) at time $t+1$.

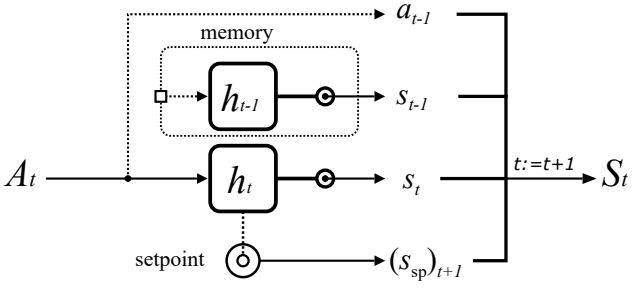

Figure 2. Representative scheme of the setpoint learning environment. Here, A_t represents the action space, h_{t-1} a linearized system or wrapped environment stored in memory and h_t the present system. At $t := t+1$, S_t denotes the resulting state space.

This method is designed to be applied in two distinct frameworks: 1) reinforcement learning, presented in Section

5, where the goal is for the agent to learn to follow a setpoint and adapt to different environments, and 2) diffusion models, detailed in Section 6, where each image represents state-action spaces.

5 Reinforcement learning case

The proposed approach can be employed in a reinforcement learning framework, leading to the Sepoint Reinforcement Learning approach (SPRL). It mainly enables agents to adapt to different setpoints across various environments without the need for specific trainings.

The environment integrating the SPRL approach is presented in Figure 3.

The reward system is redefined based on the deviation from the setpoint to precisely adjust the agent's objective. Thus, the trained agent can choose an action a_t based on the current state s_t, aiming to closely approach the target setpoint s_{sp} at time $t+1$, as illustrated in Figure 1.

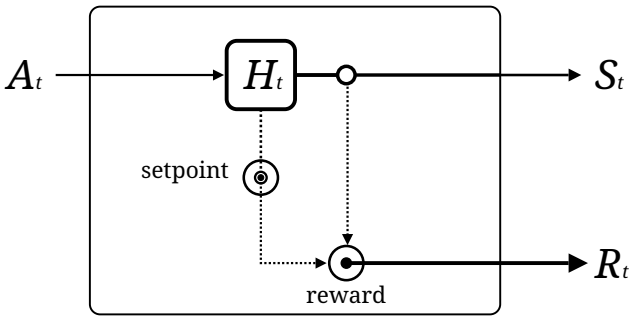

Figure 3. Scheme of the Setpoint Reinforcement Learning environment (SPRL). Here, A_t represents the action space, H_t a linearized system or wrapped environment. At $t := t+1$, R_t denotes the reward received (calculated based on the system's response and the setpoint), and S_t the resulting state space.
As already mentioned,

5.1 Application to the CartPole problem

To validate our approach on a simple model before tackling more complex problems, we first apply it to the inverted pendulum problem (CartPole). This environment is chosen due to its straightforward reward system and its number of independent variables, which are comparable to those in our proposed method.

5.1.1 Cartpole environment

We briefly recall the structure of the Cartpole environment and then present the modified environment. pourquoi modified env

Classical Cartpole environment
The classical CartPole environment as introduced in (Car) provides four observations that allow us to define the set of states S_{clas} at time t as

$$S_{\text{clas},t} = \{x_t, v_t, \theta_t, \omega_t\}, \tag{5}$$

where x_t denotes the cart position, v_t the cart velocity, θ_t the pole angle, and ω_t the pole angular velocity. The reward function at time t is defined as follows:

$$R_{\text{clas},t}(\theta_t) = 1 \text{ if the episode is not terminated and } \theta_t \text{ is in the range } (-0.2095, 0.2095) \text{ rad.} \tag{6}$$

Modified Cartpole environment
In the design of our modified environment, the pole angle θ is selected as the primary variable due to its key role in maintaining equilibrium in the classical CartPole setup. Focusing on the angle provides the agent with relevant information to capture system dynamics and adjust its actions accordingly. The following table summarizes the observations provided by the modified environment.

Variable	Observation	min	max
a_{t-1}	Previous action	0	1
θ_{t-1}	Previous pole angle	-0.2095 rad	0.2095 rad
θ_t	Pole angle	-0.2095 rad	0.2095 rad
$(\theta_{\text{sp}})_{t+1}$	Target pole angle	-0.2095 rad	0.2095 rad

Table 1. Observations provided by the modified CartPole environment.

This leads us to define the set of states S_{SPRL} of the modified environment at time t as follows:

$$S_{\text{SPRL},t} = \{a_{t-1}, \theta_{t-1}, \theta_t, (\theta_{\text{sp}})_{t+1}\}, \tag{7}$$

where θ_{sp} is the target angle we aim to reach at time $t+1$.

Moreover, the introduction of a reward function based on the setpoint reinforces the understanding of the objective by concentrating high rewards around the setpoint, as shown in Figure 4. This reward function at time t is defined as follows, with $Z = \left(\theta_{\text{sp}} - \frac{\theta_\Omega}{2}, \theta_{\text{sp}} + \frac{\theta_\Omega}{2}\right)$ rad :

$$R_{\theta_{\text{sp}}}(\theta_t) = \begin{cases} 1 & \text{if } \theta_t \text{ is within the range } Z \\ 0.1 & \text{if } \theta_t \text{ is not in } Z \text{ and} \\ & \theta_t \text{ is within } \left(-(\theta_{\text{sp}})_{\max}, (\theta_{\text{sp}})_{\max}\right) \text{ rad.} \end{cases} \tag{8}$$

5.1.2 Experimental Setup

In order to evaluate the adaptability of the SPRL approach, we train agents using the SPRL approach with different target setpoint value θ_{sp} (zero, random constant X, and

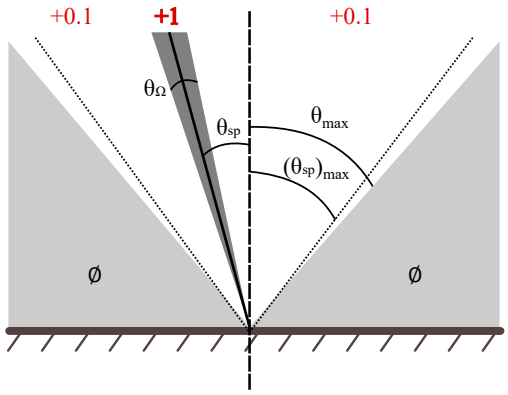

Figure 4. Distribution of the reward (8) in the modified CartPole environment. Around the setpoint, in the range Z (dark gray zone), the reward is equal to 1. Otherwise, the reward is 0.1 within the range $(-(\theta_{\mathrm{sp}})_{\max}, (\theta_{\mathrm{sp}})_{\max})$ rad.

variable random function $f(X)$). We also consider three different scenarios as references trainings, each distinguished by the space state (5) or (7) and the reward function (6) or (8). The configuration of the total six scenarios is resumed in Table 2.

We use the "CartPole-v1" version for agent training as well as for the tests. The constraint on the cart's position is removed, allowing the agent to learn to follow setpoints with more or less off-centered angles. The only constraint remains the pole angle. The episode is terminated when one of the stopping conditions is met, and truncated if the episode ends before the threshold defined by the version of the environment used. For more details, refer to (Car).

The training was realized over 1000 episodes using the Dual Double DQN algorithm, implemented in the RLlib library (RLl). During the training, we save the point at which the moving average of the last 100 scores is maximal, enabling us to compare the performance of the different models as shown in Figure 5.

Training model	Environment	Space state	Reward	Setpoint θ_{sp}	Goal
d $\text{T}_{S_{\text{SPRL}},R_{\theta_0}}$	SPRL	S_{SPRL} (7)	$R_{\theta_{\text{sp}}}$ (8)	0	Train SPRL approach with three different setpoints
$\text{T}_{S_{\text{SPRL}},R_{\theta_X}}$	SPRL	S_{SPRL} (7)	$R_{\theta_{\text{sp}}}$ (8)	random variable, constant per episode	
$\text{T}_{S_{\text{SPRL}},R_{\theta_{f(X)}}}$	SPRL	S_{SPRL} (7)	$R_{\theta_{\text{sp}}}$ (8)	random, time-varying function	
$\text{T}_{S_{\text{clas}},R_{\text{clas}}}$	classical	S_{clas} (5)	R_{clas} (6)		Reference training
$\text{T}_{S_{\text{SPRL}},R_{\text{clas}}}$	SPRL	S_{SPRL} (7)	R_{clas} (6)	0	
$\text{T}_{S_{\text{clas}},R_{\theta_0}}$	classical	S_{clas} (5)	$R_{\theta_{\text{sp}}}$ (8)		

Table 2. Overview of the six training models, each defined by the approach used (classical or SPRL), the reward function (R_{class} or $R_{\theta_{\text{sp}}}$), and the associated setpoint.

5.1.3 Results

We begin by presenting training results for both the SPRL and classic CartPole environments. Figure 5 illustrates agent performance, displaying rewards trajectories across episodes for each model detailed in Table 2.

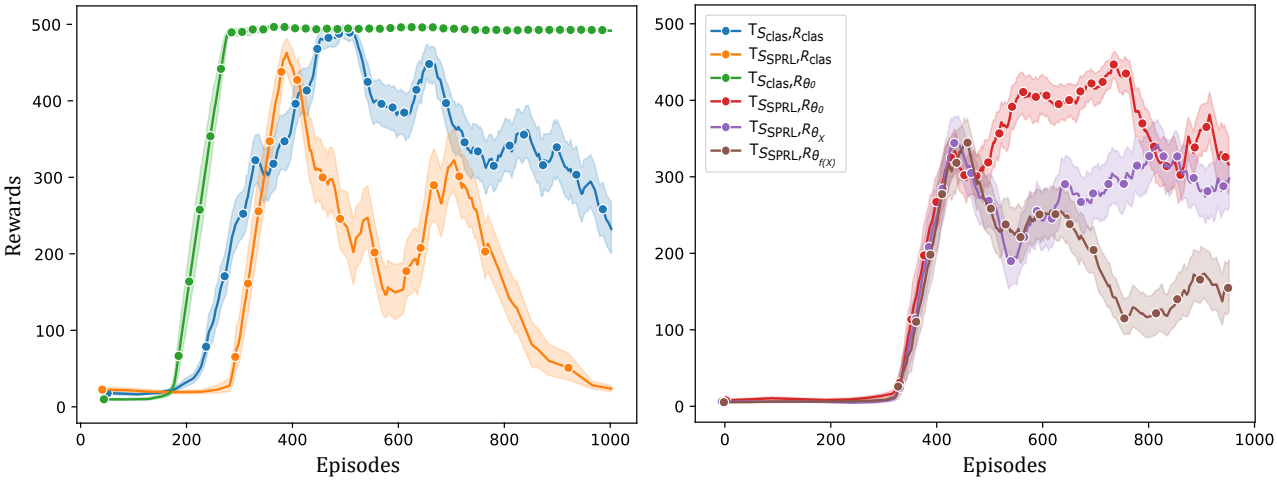

Figure 5. Performance of the six trained agents following the models described in Table 2.

Tests \ Trainings	$\text{T}_{S_{\text{clas}},R_{\text{clas}}}$	$\text{T}_{S_{\text{clas}},R_{\theta_0}}$	$\text{T}_{S_{\text{SPRL}},R_{\text{clas}}}$	$\text{T}_{S_{\text{SPRL}},R_{\theta_0}}$	$\text{T}_{S_{\text{SPRL}},R_{\theta_X}}$	$\text{T}_{S_{\text{SPRL}},R_{\theta_{f(X)}}}$
R_{θ_0}	0.910542	0.994389	0.993675	**0.997996**	0.758116	0.894289
R_{θ_X}	0.151506	0.100000	0.109036	0.100000	0.737048	**0.869880**
$R_{\theta_{f(X)}}$	0.134337	0.100000	0.403614	0.437952	0.682530	**0.859036**

Table 3. Adaptability test results for agents trained in three distinct test environments (null setpoint, random constant per episode, and random time-varying function), expressed by median over 100 iterations.

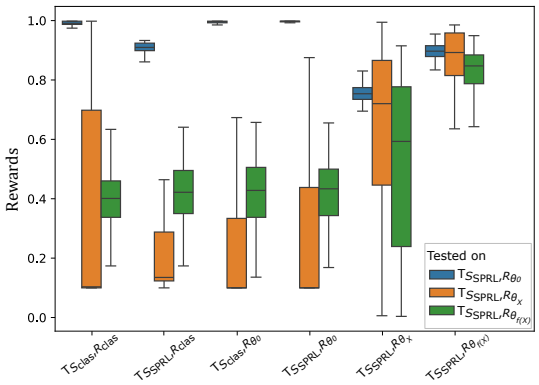

Figure 6. Box plot of the six models tested on the three scenarios respectively integrating a zero setpoint (blue), random constant (orange), and random function (green).

The results of this synthesis, as depicted in table 3, highlight the remarkable adaptability of the $T_{S_{\text{SPRL}},R_{\theta_{f(X)}}}$ model. Evaluating $T_{S_{\text{SPRL}},R_{\theta_{f(X)}}}$ across three tests shows good performance in following the setpoint, whether constant or varying over time. This performance even surpassed that of the $T_{S_{\text{SPRL}},R_{\theta_X}}$ model, specifically trained to follow a constant setpoint X.

5.2 Linear Time-invariant system

In order to improve generalization performance of the approach and evaluate its adaptability, we apply it to linear time-invariant (LTI) systems, which are widely used in engineering and physical sciences.

5.2.1 Developed environment

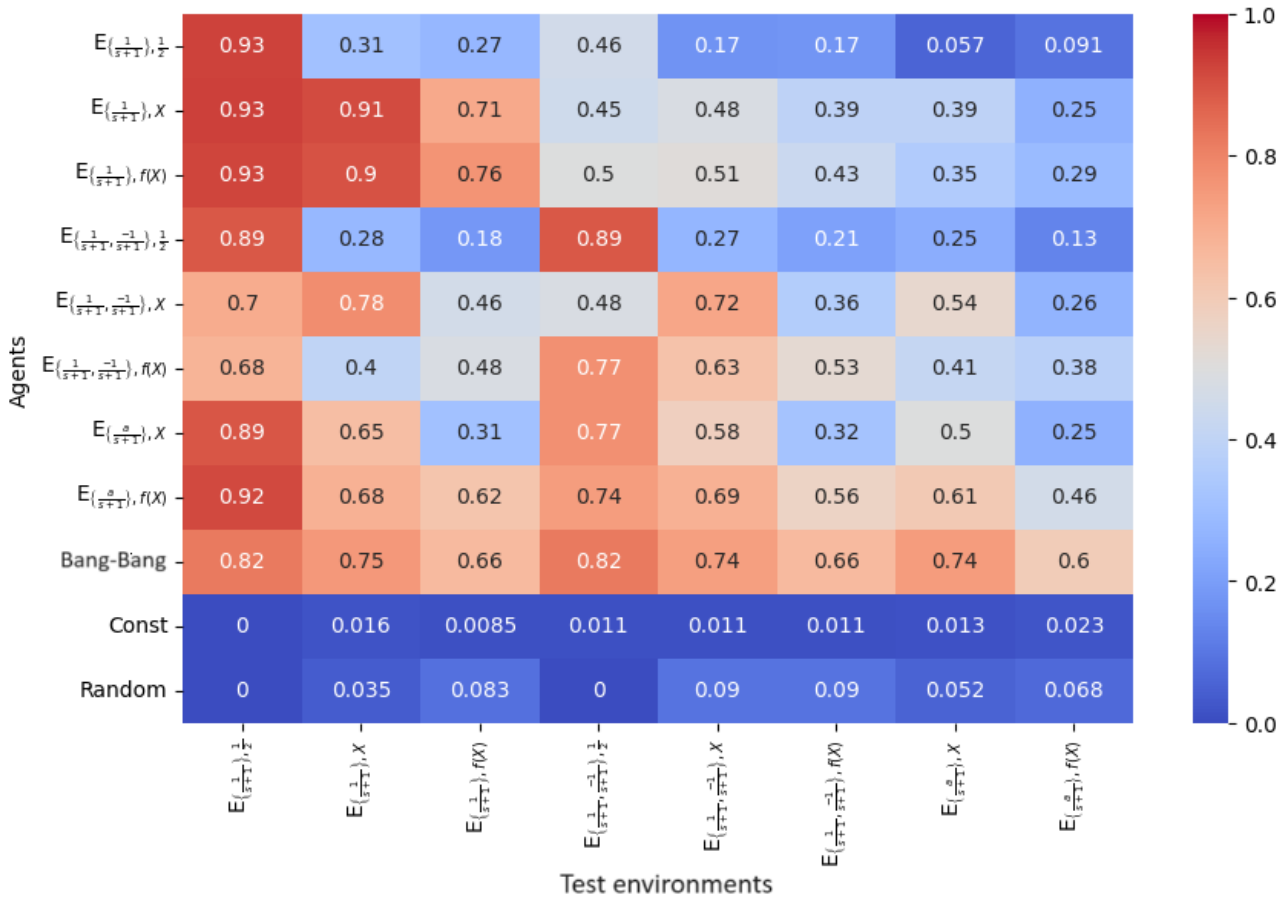

Figure 7. Comparison of agent performance as a function of evaluation environment over 100 episodes, expressed by normalized means.

6 Diffusion learning case

Attention ici, comme c'est le resultat de la sortie, At donne St+1p.

6.1 Multiple LTI system data generation

On retrouve 4 configurations, (a), (b), (c) et (d) see figures **??**, où la (b) correspond à la division en $(N * N)/2^{2i}$ fonction de transfert de la (a) (car image). Les entrees et variable d'etat initiale, peuvent egalement divisé de cette facons indépendaement des configurations.

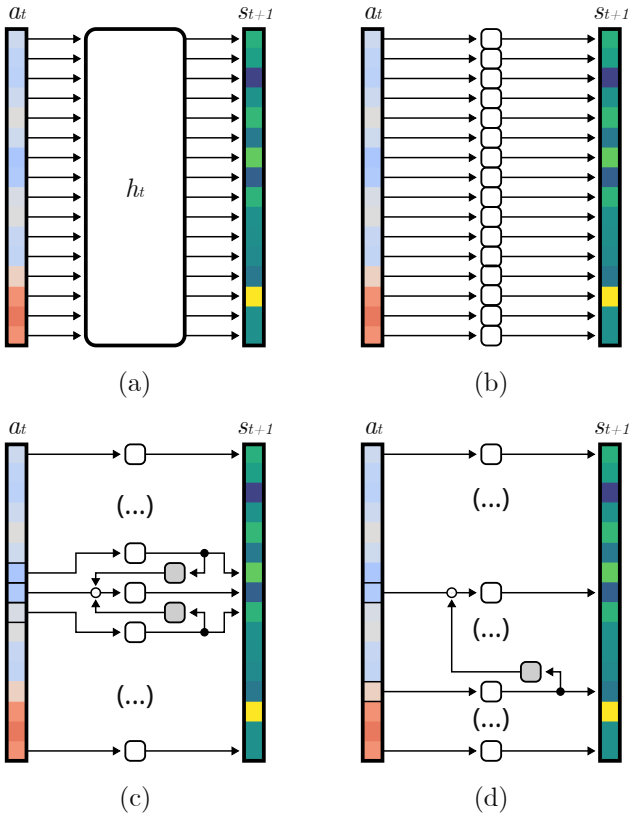

Figure 8. Algorithm scheme of DDPMS-Approach. In A, you have minimal RL system exprimed with transfert fonction h_t, it's can be a linearized system, or wrapped environment. In B, the algorithm scheme of environment with reward calculation. Je separerais les figures ? pour bien seperarer la partie "génération des environnement multiple" et l'algorithme proposé pour la génération de la base de données...

6.2 Multiple LTI training

The objective is to test our approach with an $N \times N$ environment. For this, we used a data-driven, diffusive method following three configurations:

- Direct Generation from Complete Noise : In this configuration, environment configurations are directly generated from complete noise. (classic)

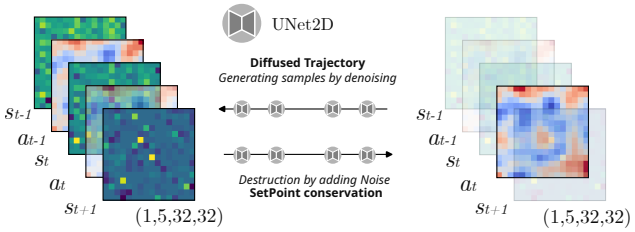

Figure 9. Algorithm scheme of DDPMS-Approach. In A, you have minimal RL system exprimed with transfert fonction h_t, it's can be a linearized system, or wrapped environment. In B, the algorithm scheme of environment with reward calculation. Je separerais les figures ? pour bien seperarer la partie "génération des environnement multiple" et l'algorithme proposé pour la génération de la base de données...

- Generation with a Fixed "Target" Channel : Here, environment configurations are generated with a fixed "target" channel, meaning that this channel remains unchanged throughout the generation process. (setpoint)

- Generation with a Noisy "Target" Channel : In this last configuration, environment configurations are generated with a noisy "target" channel, meaning that this channel is perturbed by noise before the generation process. (setpoint-inverted)

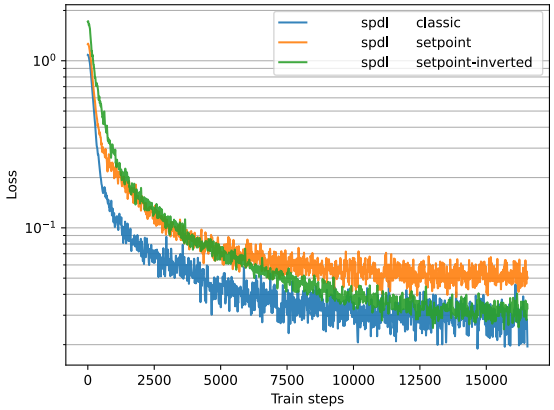

Figure 10. Train curves of DDPMS-Approach. In A...

We observe a convergence of the 3 training curves in Figure 10, the complete generation is more efficient. Channel generation training has equivalent convergence to full generation, but takes more train steps. training with generation of all channels except one has lower convergence to full generation, for the same training time.

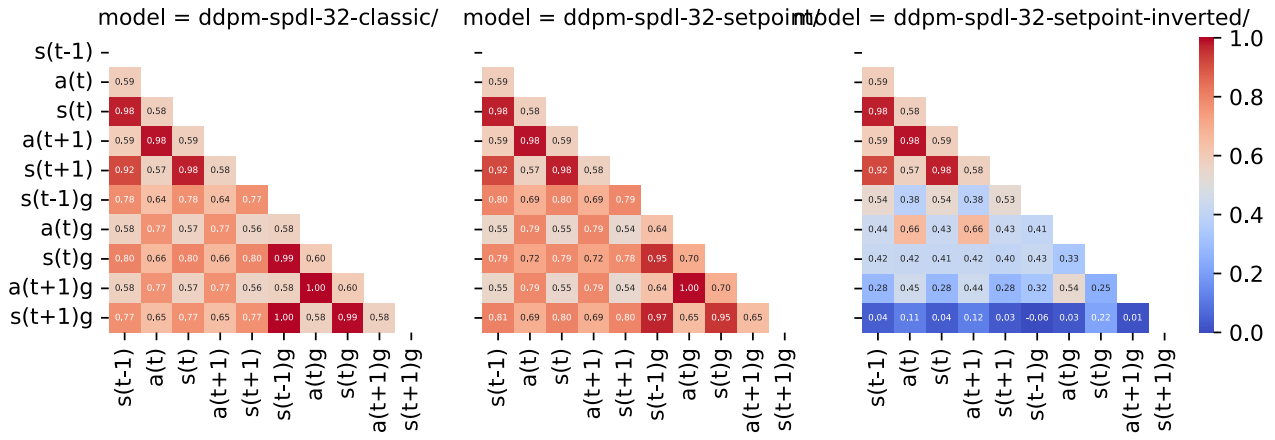

Figure 11. Correlation matrix of DDPMS prediction. In A... METTRE LES VALEUR DES CORRELATION DANS LE TABLEAUX

A positive correlation score is measured for each generation. However, the precision 80% is not high (It's equivalent to bang-bang control), which does not allow the use of this algorithm directly for a efficient multiple control problem. (juste décrire, interpretation en conclusion)

7 Conclusions and perspective

lien vers le code

Utilisation de l'environnement construit dans ce papier pour construire des modele fondation avec plus d'hyperparametre

Utilisation de modèle Transformer pour predire l'action (TD) d'un ensemble d'environnement wrappé

Les DDPMS pas tres efficace, utilisation de modèle SSM pour remedier au probleme ? (S4 https://arxiv.org/pdf/2111.00396 https://arxiv.org/abs/2405.21060 / Vision-MamBa) pour predire l'action d'une sequence d'image en limitant les calculs. Limitation : https://arxiv.org/abs/2404.08819 (illusion, repétition et permutation-composition)

Decision Mamba / Transformer https://arxiv.org/pdf/2403.19925

Alternative pour control End-to-End des environnement physique par des LLM plutot que des technique de function calling qui demande beaucoup de puissance de calcul (Eureka). Exemple avec les detections d'objects via Vision-GPT-Yolo https://arxiv.org/abs/2403.12415

References

Gymnasium documentation. URL https://gymnasium.farama.org/environments/classic_control/cart_pole/.

Rllib documentation. URL https://docs.ray.io/en/latest/rllib/index.html.