SetPoint Learning : Toward approach to control them all

JFE [*1] **BTI** [1] **DOBEDO** **F. 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.

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.

[1]Capgemini Engineering, Paris. Correspondence to: fabien.furfaro@capgemini.com, fabien.furfaro@gmail.com.

Arxiv preprint, Paris, 2024.

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]. 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.

2 Background

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.

2.1 Problem Setting

Consider a system governed by a discrete-time dynamics $S_{t+1} = f(S_t, a_t)$. The objective, whatever our system, will be to achieve the optimum:

$$a^*_{0:T} = argmax_{a_{0:T}} \sum_{t=0}^{T} r(s_t, a_t) \tag{1}$$

The difficulty is to have a set of variables as input to a model

which allow the resolution of this equation whatever the physical system.

2.2 Linear system

In the case of a linear system, we have:

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

Which $s(t)$ is an action following the controling system and $h_X(t)$ the randomized linear system.

3 Methods: Setpoint Learning

When we summarize a dynamic system in its trajectory, we observe four fundamental sets of variables which allow the resolution of equation (1) whatever the linear system (2). These variables are:

- a_{t-1} : Action in previous state s_{t-1} doing next state s_t
- s_{t-1} : Previous state
- s_t : Actual state of system
- $(s_c)_{t+1}$: Objective state

In a linear system, the response instantly goes in the direction of the action given at the system input (we don't have delay responce).

$$(S, A) : a_{(i,t)} \rightarrow \left\{a_{(i,t-1)}, s_{(i,t-1)}, s_{(i,t)}, s_{c(i,t+1)}\right\} \tag{3}$$

From this assetion, we can make the hypothesis that only this variable is enough to know what action to take to achieve an objective at the next moment. We can define naive binary controler :

$$a_t = sgn(a_{t-1}) \cdot sgn(s_t - s_{t-1}) \cdot sgn(s_{c,t+1} - s_t) \tag{4}$$

The consequence is that if a learning algorithm has several types of linear and invariant environments as data, it can generalize to any linearizable system and adapt to any level rather than just a specific value.

3.1 General case

Cas général (fleche)

This relation provides a straightforward translation between classifier-guided sampling, used to generate classconditional data

to be defined

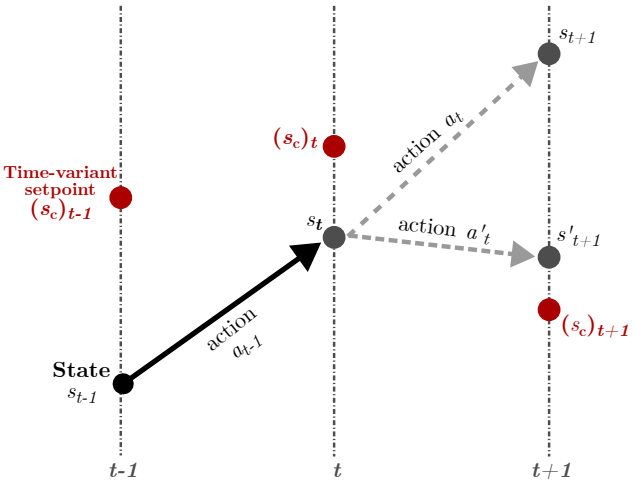


Figure 1. Representative diagram of the target setpoint concept. At time t, the agent in state s_t should choose the action in order to get as close as possible to the target state (setpoint) (s_c) at time $t + 1$.

Algorithm 1 Environment Planning

Require ABCD M_X, guide $\mathcal{J}$, scale α, covariances Σ^i
while not done **do**
  Observe state $\mathbf{s}$; initialize plan $\boldsymbol{\tau}^N \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  for $i = N, \ldots, 1$ **do**
    `// parameters of reverse transition`
    $\mu \leftarrow \mu_\theta(\boldsymbol{\tau}^i)$
    `// guide using gradients of return`
    $\boldsymbol{\tau}^{i-1} \sim \mathcal{N}(\mu + \alpha\Sigma\nabla\mathcal{J}(\mu), \Sigma^i)$
    `// constrain first state of plan`
    $\boldsymbol{\tau}^{i-1}_{\boldsymbol{s}_0} \leftarrow \boldsymbol{s}$
  Execute first action of plan $\boldsymbol{\tau}^0_{\mathbf{a}_0}$

3.2 Reinforcement learning case

Cas renforcement (environnemnt)

Attention entre le point de vue de l'environnement et le point de vue du modele d'apprentissage !

Pour le modele, St correspond à la sortie instantané, mais lorsque l'action est faite, ca devient St+1 pour l'environnement !

L'idée est, sachant que je suis à l'etat "t", quel action pour aller à l'etat "t+1" correspondant à Sc, mais une fois l'action faite, ca devient St pour l'environnement

3.3 Diffusion learning case

Cas diffusion (image-multi-env)

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

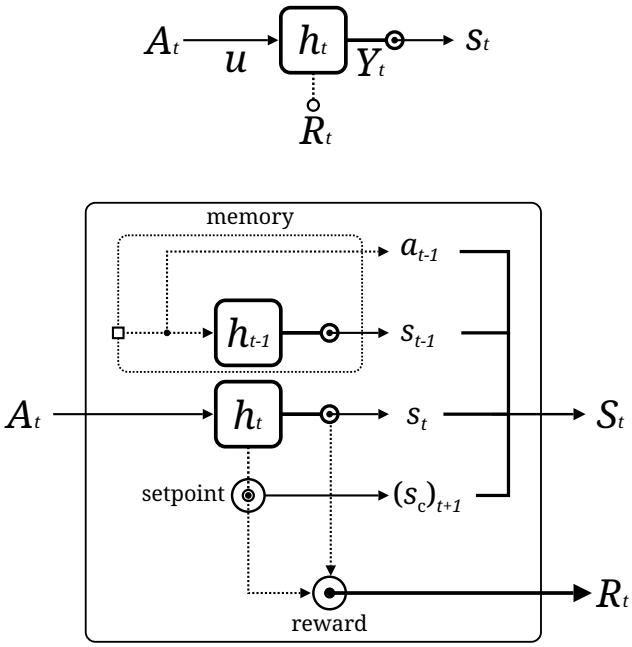


Figure 2. Algorithm scheme of RL-Approach. In A, you have minimal RL system exprimed with transfert fonction h_t, it's can be a linearized system, or wrapped environment. In B, the algorithm scheme of environment with reward calculation.

4 Results

We are the

4.1 CartPole

The objective is to test our approach on a simple use case. We chose CartPole because it is a small angle linear approximation system that also includes a response delay.

4.2 Linear Time-invariant system

LTI

4.3 Multiple Linear Time-invariant system

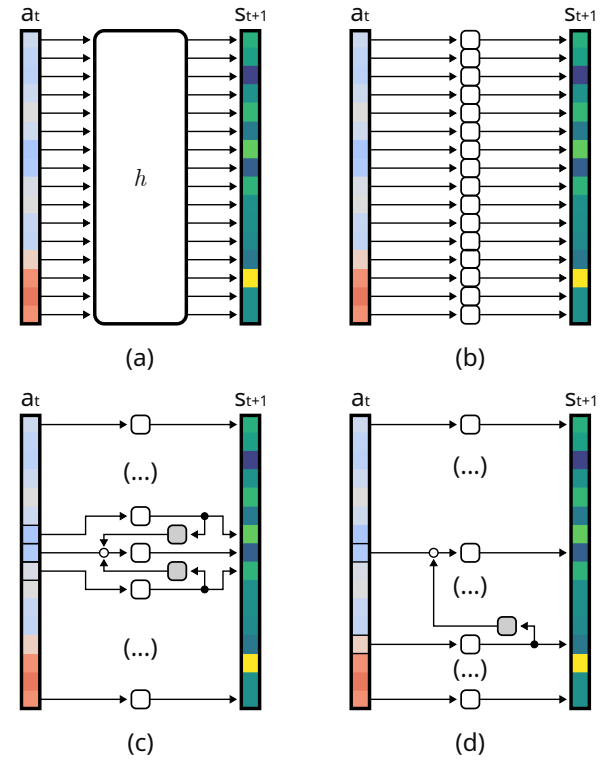
multi-LTI

5 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 control-Net pour generer des actions plus complexe (https://ctrl-adapter.github.io/)



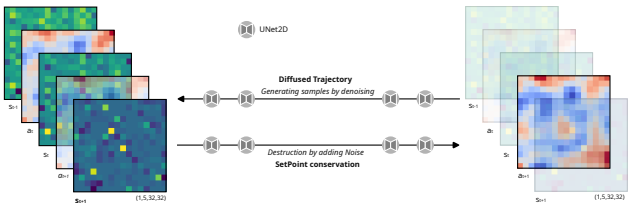


Figure 3. Algorithm scheme of DDPMS-Approach. In A, you have minimal RL system exprimed with transfert fonction h_t, it's can be a linearized system, or wrapped environment. In B, the algorithm scheme of environment with reward calculation.