Curriculum reinforcement learning for tokamak control

Abstract

Tokamaks are the leading candidates to achieve nuclear fusion as a sustainable source of energy, and plasma control plays a crucial role in their operations. However, the complex behavior of plasma dynamics makes control of these devices challenging through traditional methods. Recent works proved the usefulness of reinforcement learning as an efficient alternative, in order to fulfill these high-dimensional and non-linear situations. Despite their performance, controlling relevant plasma configurations requires expensive and long training sessions on simulations. In this work, we leverage the use of a curriculum strategy to achieve significant speed-up in learning a controller for the control coils, which tracks plasma quantities such as shape, position and current. To this end, we developed a fast, asynchronous and reliable framework to enable interactions between a distributed actor-critic and a C++ code simulating the WEST tokamak. By sequentially increasing task complexity, results show a clear reduction in convergence time and training cost. This work is one of the first attempts to enable fast production of robust magnetic controllers, for routine use in the operations of a magnetically confined fusion device.

1 Introduction

Mastering nuclear fusion could significantly impact the world, unlocking the path towards sustainable and attractive means of energy production. With no direct high-level byproducts of the reaction, it has many advantages over conventional energy sources [Ariola and Pironti, 2008]. To harness this potential alternative, tokamaks are promising devices to maintain the stability and performance of plasma's confinement, despite numerous physical and control challenges [Meade, 2009].

Tokamaks are torus-shaped devices which aim at sustaining fusion reactions within a plasma under specific temperature and density conditions [Wesson, 2004]. They rely on magnetic fields generated by both *toroidal* and *poloidal* field coils (PFC) to shape it. Interactions occur at different levels with complex dynamics involved between the plasma and its surroundings. Control systems are then required to adjust the voltages applied to the PFCs (Figure 2), allowing control of quantities intrinsically linked to plasma's evolution, like position of the magnetic center m, *Last Closed Flux Surface* (LCFS), elongation κ and current I_p (Figure 1). To study the effects of various plasma configurations, scientists rely on real-time linear controllers [Nouailletas and et al., 2023], which require substantial engineering effort whenever target scenarios undergo variations. Hence, there is a essential need for flexibility, adaptability and robustness of magnetic control systems through the entirety of the device lifetime, without which no sustained plasma could be produced.

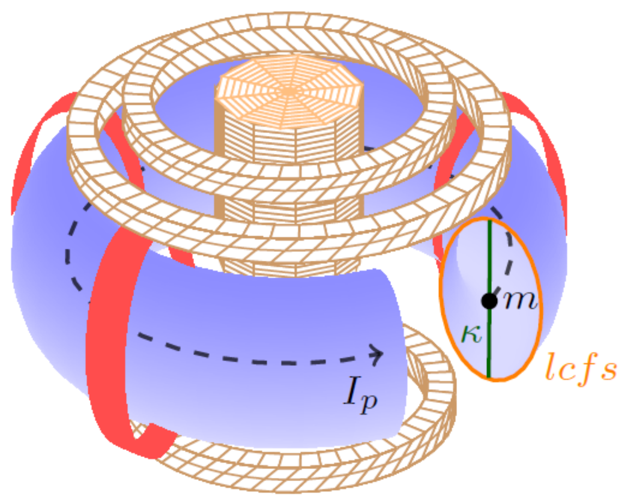


Figure 1: Control quantities of interest with toroidal (red) and poloidal (strided gold) sections.

Reinforcement Learning (RL) [Sutton and Barto, 2018] emerged as an innovative approach to numerous real-time control problems. Despite impressive results in a variety of domains [Han *et al.*, 2023; Brohan and et al., 2023; Kiran and et al., 2022], it usually relies on either fast and precise simulation enabling the collection of vast amount of experiences, or on direct sampling from a physical device. Both cases can not be fulfilled in our context: sampling of experimental data on the plant for the sole purpose of training is impractical, and simulations remain expensive in order to account for the coupled behavior of plasma dynamics. Despite the existence of distributed architectures as powerful tools to compensate for these bottlenecks, training still remains long and costly as the number of parallel environments increases.

In this work, we study the effects of a curriculum strategy on learning a magnetic controller through a distributed reinforcement learning framework. By improving training speed and performance, we intend to accelerate the production of robust magnetic controllers for the operation of WEST, a supraconductive tokamak located at CEA Cadarache[1] in Saint-Paul-lez-Durance, France [Bourdelle and et al., 2015; Bucalossi and et al., 2022]. Indeed, such methodology could assist plasma researchers in quickly obtaining controllers, or adapt existing ones, for each new experimental campaign, hence improving flexibility and adaptability of RL-based magnetic control.

Next sections will be organized as follows. First, we will give an overview of the related work regarding RL for tokamaks, and curriculum strategies in RL. We will then describe the curriculum methodology within plasma magnetic control, and the overall training framework. Finally, experiments are discussed through validation and analysis of the learned policy. The latter will be compared to a baseline agent obtained without the strategies of interest. Finally, we will conclude on this study and its perspectives.

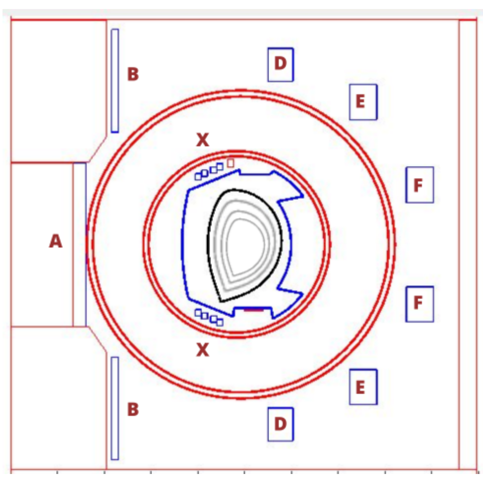


Figure 2: Cross-section with surrounding control coils, namely poloidal field coils.

2 Background

2.1 Reinforcement learning for tokamaks

A classical RL framework sets an agent which interacts with an environment formalized as a *Markov Decision Process (MDP)* denoted $\mathcal{M}$. This MDP is defined by a state space $\mathcal{S}$, an action space $\mathcal{A}$, its state transition distribution $P(s'|s,a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$, an initial state distribution $P^0(s) : \mathcal{S} \to [0,1]$, and a reward signal $R(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

Starting from state $s_0 \sim P^0(.)$, the agent must learn an optimal policy $\pi^* : \mathcal{S} \times \mathcal{A} \to [0,1]$, which maximizes the discounted cumulative reward, or *return*, over the course of an episode, i.e a trajectory over states and actions from the interactions with the environment:

$$\pi^* = \underset{\pi_\theta}{\operatorname{argmax}}\, \mathbb{E}_{(s_0,a_0,\dots,s_t,a_t)}[\sum_{k=0}^{\infty} \gamma^k r_{t+k}] \qquad (1)$$

with discount factor $\gamma \in [0,1]$ working as a penalization term for long-term rewards, and $r_t = R(s,a) = \mathbb{E}[r_{t+1}|s_t = s, a_t = a]$. Most importantly, the reward function is a scalar feedback signal which indicates how well the agent performs with respect to the overall objectives, hence the importance of its design. The feedback loop between the agent and the environment ends once a terminal condition is reached, like a situation that we want to avoid, or a threshold on simulated time. As a side note, the policy can be deterministic, assigning a probability of 1 to the same action for each observed state. Moreover, it can be a parametrized function, like a neural network. In such cases, it is usually denoted by π_θ, where θ are the weights of the said model.

Fundamental definitions arise with the value function $V_{\pi_\theta}(s) = \mathbb{E}_{\pi_\theta}[\sum_{k=0}^{\infty} \gamma^k r_{t+k}|s_t = s]$, and the action-value function $Q_{\pi_\theta}(s,a) = \mathbb{E}_{\pi_\theta}[\sum_{k=0}^{\infty} \gamma^k r_{t+k}|s_t = s, a_t = a]$. It is worth mentioning that relying on the first is difficult in real-world applications such as fusion, since they do not exhibit proper knowledge of the probability transition function P. Because of that, making actions explicit is an interesting way of computing the expected return, as state-action pairs can be easily sampled throughout learning. Over the past years, the use of neural networks (NN) as powerful action-value and policy approximators lead to major advancements in continuous control problems. Deep RL algorithms such as ones from the actor-critic family kept increasing in efficiency, leading to precise control in several high-dimensional and non-linear control problems [Grondman *et al.*, 2012], both in on-policy [Schulman and et al., 2015; Schulman and et al., 2017; Mnih and et al., 2016] and off-policy settings [Haarnoja *et al.*, 2018; Fujimoto *et al.*, 2018; Lillicrap and et al., 2015].

Consequently, deep reinforcement learning is becoming increasingly popular among the plasma control community. For example, RL has been used for model-based control [Char and et al., 2023], for vertical stabilization [Dubbioso *et al.*, 2023; De Tommasi *et al.*, 2022], to build feedforward trajectories of plasma parameters [Seo and et al., 2021], for temperature and profile control [Wakatsuki and et al., 2019; Wakatsuki *et al.*, 2021], or even for tearing instability control and disruption avoidance [Seo *et al.*, 2024]. Recent works [Degrave and et al., 2022] designed a RL-based system which achieved magnetic control of the *Tokamak à Configuration Variable* (TCV), in Lausanne, Switzerland. The learned controller demonstrates the capability for RL-based systems to tackle various complex plasma configurations while tracking many quantities of interest at the same time. A similar procedure was proposed by [Kerboua-Benlarbi *et al.*, 2024], with the same limitations of the initial proposal, while refining the simulation on which magnetic controllers were trained.

These examples highlight a shift of focus from classical controllers, designed using prior knowledge on how control

[1]French Alternative Energies and Atomic Energy Commission

should be performed with respect to physical properties of the dynamical system, to controllers learning by trial-and-error to act on the system based on what should be achieved in terms of final objectives. In summary, deep RL advantages over classical tokamak control stem from its ability to: fulfil these high dimensional, uncertain and non-linear systems; explore possible strategies in order to make the control policy more flexible in contrast with the fixed heuristics of classical control; learn from raw magnetic signals using neural networks, since plasma quantities can not be measured directly, and are instead usually inferred in real-time from reconstruction codes [Faugeras, 2020; Carpanese, 2021] for use by classical controllers.

2.2 Curriculum learning for reinforcement learning

Curriculum learning (CL) [Bengio *et al.*, 2009] is a methodology to optimize the order in which experiences are processed by an agent over the course of training. From the early stages of human development to adulthood, learning is structured and organized sequentially, so that the knowledge acquired over time facilitates the understanding of new notions or tasks that occur later to us. Therefore, a sequence of increasingly difficult tasks implicitly builds a curriculum, as knowledge is transferred from one intermediate objective to the other. Scheduling and designing such strategy helps in acquiring transferable skills to guide exploration during training, with the premise of increasing performance and reduce convergence time towards a final set of tasks.

Recent works classified the taxonomy of existing methods [Soviany *et al.*, 2022] as well a mathematical framework for curriculum learning in reinforcement learning domains using graphs [Narvekar *et al.*, 2020]. In most cases, we consider different MDPs between each task and three main concepts arise with which CL methods can be classified: the intermediate task generation, the partial ordering on the obtained set of tasks and how knowledge could be shared between its elements. Considering the importance of human intuition to define simple tasks [Bengio *et al.*, 2009], domain experts could efficiently make a distinction between objectives that are neither "too easy" or "too hard". Indeed, task generation and sequencing of the latter could be handcrafted from human operators [MacAlpine and Stone, 2018; Stanley *et al.*, 2005], but both concepts could be built up automatically as part of the curriculum learning procedure [Graves *et al.*, 2017; Wu and Tian, 2017; Ivanovic and et al., 2019]. Transfer learning methods are required to share knowledge representation at each step of the curriculum, and concern several elements of the training loop, such as entire policies and value functions, rewards, etc [Zhu *et al.*, 2023]. Care must be taken while choosing the right combinations of methods, to avoid negative transfer which could harm controllers performance [Wołczyk *et al.*, 2022].

3 Approach

3.1 Motivation

RL is still an emerging field within plasma magnetic control, and few applications are observable. It can take several days of training for efficiency on relative simple plasma scenarios [Degrave and et al., 2022; Kerboua-Benlarbi *et al.*, 2024]. Nevertheless, the routine operation of a tokamak requires flexibility over the design of controllers. Minimum engineering efforts should be targeted to adapt and fine-tune the controllers with respect to the objectives of each new experimental campaign.

For this reason, this study aims at assessing the effects of CL in the context of fusion, where poor reward signal and state representation at the beginning of learning, can destabilize the whole training process. We do not specifically intend to reach a new general performance threshold, but look for increased performance at start of each new task, specializing exploration as training evolves towards its final goal. Considering the cost of data sampling using WEST simulations, yet in the real world, curriculum learning could be of great help to stabilize the entire procedure, and reduce convergence time by several orders of magnitude. Furthermore, each new experimental campaign on WEST requires the definition of multiple control scenarios. The latter might have shared plasma states, and overall control objectives. This means that the same events can be used within different scenarios, especially while choosing initial conditions or terminal ones. Since a scenario is a sequence of events, their ordering already defines a curriculum in an implicit manner, as plasma equilibriums must follow each other in a realistic and feasible way. Moreover, one could go further by explicitly building a curriculum on the reward function, considering a sequence on its definition, i.e directly on the explicit control objective which might be similar between scenarios. A simple reward on the shape for example could be used as a starter, latter including the elongation, etc. Both ideas lead to the same conclusion regarding CL in fusion:

- curriculum generation and ordering could describe tasks as events, or intermediate reward definitions;
- the two approaches shows that a curriculum working for one plasma scenario, could be intuitively generalizable with little effort on similar ones, enhancing production of controllers for several cases during experimental campaigns.

It is worth noticing that [Tracey *et al.*, 2023] addressed the initial drawbacks of the method described by [Degrave and et al., 2022], i.e. training speed and steady-state performance of the controller. Their approach resembles curriculum learning by borrowing its codes. Researchers partitions a target scenario in smaller chunks, each related to one part of the general task. Distributed environments are then divided into subsests of different cardinalities, each linked to one of the said chunks. Experiences are accumulated from MDPs that differs implicitly in their underlying dynamics. Sampled experiences are more diverse, and mix multiple levels of difficulty inside the same training procedure. This procedure already reduced training time by a factor of 4. However, despite different initial state distributions, the overall task remain the same between chunks, and no proper curriculum is defined, i.e no knowledge transfer is present and task ordering is not specifically mentioned.

3.2 Curriculum definition

Formalism Let τ be a set of tasks with m_i : $(\mathcal{S}, \mathcal{A}, P_i, R_i) \in \tau$, all sharing the same state and action space. Moreover, we denote $\mathcal{D}^\tau$, the set of all transitions belonging to τ, so that $\mathcal{D}^\tau = \{(s, a, r, s') \,|\, \exists\, m_i \in \tau, s \in \mathcal{S}, a \in \mathcal{A}, s' \sim P_i(.|s, a), r = R_i(s, a)\}$. A curriculum $\mathcal{C}$ can then be defined as a direct acyclic graph $(\mathcal{V}, \varepsilon, H, \tau)$, with $\mathcal{V}$ vertices, ε edges, $H : \mathcal{V} \to \mathcal{P}(\mathcal{D}^\tau)$, connecting $v \in \mathcal{V}$ to a subset of samples of $\mathcal{D}^\tau$. An edge $< v_j, v_k >$ of $\mathcal{C}$ links two tasks, using all samples associated by H to v_j before transferring to v_k. For each $m_i \in \tau$, we have $\mathcal{D}_i^\tau = \{(s, a, r, s') \,|\, s \in \mathcal{S}_i = \mathcal{S}, a \in \mathcal{A}_i = \mathcal{A}, s' \sim P_i(.|s, a), r = R_i(s, a)\}$. We need to associate all $v \in \mathcal{V}$ with corresponding m_i and $\mathcal{D}_i^\tau$, meaning that each path on the graph directly influences how $H : \mathcal{V} \to \{\mathcal{D}_i^\tau | m_i \in \tau\}$ filter knowledge transfer between tasks, with edges built on properties of the samples associated with successive nodes. Indeed, tasks must be ordered properly so that π_i^* is useful for acquiring good samples at each transition to the current vertex. In our case, a task is associated with only one vertex, and each intermediate vertex sinks in only one node until the final one is reached, .i.e the final task [Narvekar *et al.*, 2020]. This defines an oriented sequence of tasks, similar to what was previously stated in terms of curriculum learning.

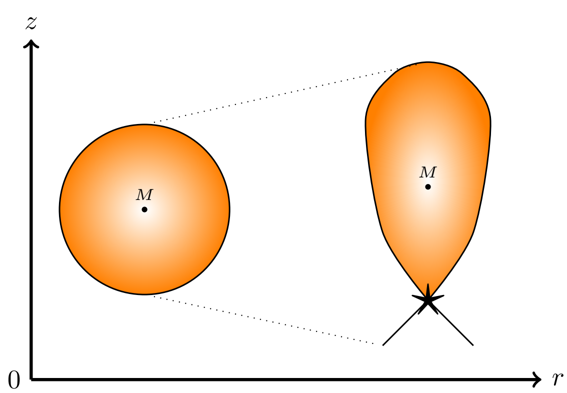
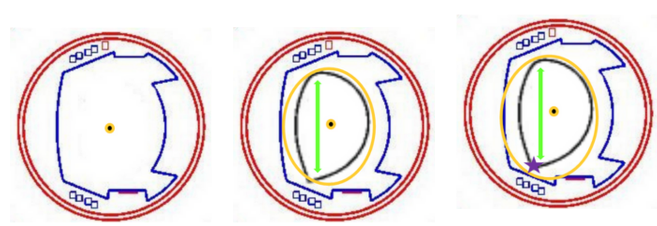


Figure 3: Schematic view of the scenario of interest. It starts from a limiter configuration, and ends up by stabilizing an elongated plasma in an x-point configuration.

Tasks In this work, we consider only one of the two possibilities mentioned earlier. Indeed, tasks are defined on the reward function, and only one scenario is considered for learning a controller. We focus on transitioning from a "circular" shaped plasma in limiter configuration, to an elongated plasma in X-point configuration, i.e $\kappa > 1$ (Figure 3). Elongated configurations have improved thermal confinement properties compared to limiter plasmas, at the cost of developing growing vertical instabilities which make control more difficult. Once formed, the *Last Closed Flux Surface* (LCFS) defines the plasma boundary and the X-point appears at its intersection. The chosen curriculum is entirely conditioned by a set of predefined rewards R_i. This means that while it could have been defined automatically, the uncertainty around tokamak dynamics makes the choice for a handcrafted sequence of tasks quite straightforward for this first application. Prior control experience on the device informs on which tasks could be considered easier than others. This work then relies on human experts for both determining τ, as well as the resulting sequence order based on $\mathcal{V}$ and ε. More precisely, the curriculum has been built from physical intuition around several key control challenges studied for all tokamaks (Figure 4):

1. vertical stabilization of elongated plasmas while tracking plasma current is a well-known control problem. Using classical feedback control, simple proportional-integral-derivative (PID) controllers [Ang *et al.*, 2005] can stabilize plasma's magnetic center (m_r, m_z), as well as plasma current I_p. Their relative simplicity are not far from a basic RL-based solution, as a naive agent can be summarized as proportional-integral control which reduces errors between measurements and targets. The initial reward function then includes targets for the two elements of interest. Hence, handling such classical problem is a good start in order to build strong foundations for the next tasks;
2. tracking the entire plasma boundary becomes more challenging, as approaches from classical control often relies on more advanced methods to synthesize efficient controllers. Since the difficulty becomes more important, we add the LCFS as well as the elongation to the initial targets. This creates a way to guide the agent towards an elongated shape, properly positioning it before the final task;
3. finally, once the plasma is set up towards its X-point configuration, we modify the reward to include targets on the X-point itself (distance, magnetic flux, etc). This could be considered as a fine-tuning exploration, since the agent must have already positioned the plasma boundary according to the final configuration. Nevertheless, we must proceed with caution, in order to avoid loosing accuracy on previous tasks through catastrophic forgetting [Goodfellow *et al.*, 2015].



$R_1 : (r_M, z_M), I_p$ $\quad$ $R_2 : R_1, \kappa$, LCFS $\quad$ R_3 : R_2, X-point targets

Figure 4: Curriculum overview. We start from a simple vertical control stabilization problem with a free plasma current, to a complex one involving shape and X point.

Transfer learning We transfer the policy and the action-value function between tasks, as both of them are neural networks. The parameters of $\mathcal{Q}_i$ learned during an intermediate task, serves as initialization for the parameters of the next action-value function $\mathcal{Q}_j$, without any freezing procedure which could negatively impact transfer [Wołczyk *et al.*, 2022]. Doing so bias the agent towards more efficient exploration in the next domain. The policy's weights are also used to initialize the parameters of the new one, again without any freezing procedure. One could have incrementally frozen layers between tasks in order to keep previous representations learned by the controller. However, we empirically observed that it is not necessary for the curriculum learning to work well in practice. Furthermore, it limits the amount of tasks present in the curriculum, as the number of layers is bounded. We further use *potential-based advice* reward shaping (PBARS) so that $R'_j(s,a) = R_i(s,a) + F(s,a) + R_j(s,a)$ with $F(s,a,s',a') = Q_i(s',a') - Q_i(s,a)$. R_i retains knowledge from the source task and F encourages exploration from states that were valuable and overlap with the target j. They form the potential-based bonus with guarantees that it will not change the optimal policy [Harutyunyan *et al.*, 2015],.

Transfer metrics While final performance on the target task will be analyzed, our main objective is to observe how CL could produce RL-based magnetic controllers faster, for routine use on WEST. Metrics must be chosen accordingly in order to measure by how much it speeds up training, compared to the vanilla method where the agent learn directly on the final task. We will refer to this question with two tools: the *jumpstart* and the *Time to threshold* (TTT). The former measures the initial performance increase at the beginning of each new task either for a unique task, or as a result of transfer; the latter checks how much faster the agent learns the policy which achieves a threshold on the episode return, with or without curriculum. Each intermediate task is caped to a maximum duration of 60 episodes, mostly to stay in line with empirical observations regarding MPO's warmup phase, i.e the phase during which NN do not undergo real variations.

4 Experiments

4.1 Setup

The NICE code The environment is based on the NICE C++ code [Faugeras, 2020], which solves the *Grad-Shafranov* equation [Wesson, 2004] for the plasma domain, with resistive diffusion [Heumann, 2021] and transport equation enabled. We use its forward evolution mode which computes the environment's state at each timestep. Moreover, power supply and diagnostic models are implemented in order to account for bias, delays and offsets of actuators. Overall, it gives an accurate representation of the plasma, as well as the WEST control system. NICE is safely initialized to a limiter shaped plasma extracted from recent experimental data, and whose internal profiles are randomized to promote diversity among examples. The relative error of the Newton solver is increased to accelerate execution without significant loose of accuracy in its outputs. Termination is triggered if thresholds are reached on active coils currents or safety factor (proportional to the geometry of the plasma and its current), to avoid any damage on the device. Episodes typically last for 500ms, as it appeared enough for generalization on longer shots.

State and Action spaces The environment's state is defined as $s = \{y, I_a, m\}$ with y the plasma equilibrium information, I_a the currents in the active control coils, and m the raw magnetic measurements. y typically contains all quantities of interest described in the curriculum definition. It is usually difficult to observe the entirety of s in real-time. To overcome this issue, the learned policy is restricted to a *Partially Observable MDP* (POMDP) where the state space is limited to the observation space $\mathcal{O}$. As such, we have $o(s) = \{tr, m_b, fl, I_a, \frac{dm_b}{dt}\}$, with tr target references, $\{m_b, fl\}$ magnetic probes and flux loops raw measurements, and $\frac{dm_b}{dt}$, temporal derivatives of magnetic probes signals. Noise is injected in observations at each timestep from Gaussian laws with parameters identified from WEST plasma discharges database, as well as delays to model real data acquisition from sensors. For actions, voltages are sampled from Gaussian distributions which parameters are the outputs of the control policy, and then supplied to each of the 11 PFCs circumventing the device (Figure 1 - Naming conventions stated in Figure 2). After exploring possible outcomes during training, only the mean of each distribution is kept at inference to predict optimal actions. Offsets, bias and delays are part of the power supply model within NICE to ensure correct handling of WEST actuators in the real world.

Component	Good	Bad	α	weight
LCFS [m]	0.005	0.1	-1	3.
Magnetic center [m]	0.002	0.03	x	1.
κ	0.005	0.03	x	1.
I_p [kA]	0.5	20	x	3.
X point distance [m]	0.01	0.15	x	2.
Flux at current x point	0.	1.	x	2.
Flux at target x point	0.	0.08	x	2.
Flux gradient at target x point	0.	4.	x	1.
Final combiner: *Smoothmax*(α = *-0.5*)				

Table 1: Reward components description with dimensions. Scaling to $[0, 1]$ range is performed, before combination to a final scalar value. Alpha is specified for each component if it has multiple targets. Flux setpoints are set to 1 since their measure is normalized , while flux gradient must tend towards zero.

Rewards Each reward R_i is a normalized weighted combination of error signals, extended with PBARS. Each component c_i^j is computed as the difference E_j between its target value and the one retrieved from the environment, then scaled to $[0,1]$ with *Softplus*(E_j) $:= min(max(2 \cdot \sigma(-\xi(\frac{E_j - good}{bad - good})), 0), 1)$. They are then combined into a final scalar within the same interval using the function *Smoothmax*(α, R_i, W) $:= \sum_j w_j R_i^j e^{\alpha R_i^j} / \sum_j w_j e^{\alpha R_i^j}$. If one component is made out of several targets, an intermediate combination using the latter is also performed. *Good* and *bad* parameters in the *Softplus* formulation, scales the reward signal according to regions of interest in the reward space. Tight values in both parameters will lead to higher focus on the component to achieve high reward. Smoother values will help exploration at the cost of precise control. Weights in the *Smoothmax* definition affects the importance of each reward component, while the α defines focus balance between them. Specifically, we combine all 32 distances of the LCFS with $w = 1$ and $\alpha = -1$. Reward undergo a final scaling, so that the maximum cumulative reward for 500 ms equals 50. For a description of each component's weight and parameters, please refer to table 1.

Agent In this work, a distributed *Maximum à Posteriori Policy Optimization* (MPO) [Abdolmaleki and et al., 2018a; Abdolmaleki and et al., 2018b] is used, which have shown strong empirical results on a wide range of control problems, including fusion. It is part of a recent interpretation of RL as probabilistic inference [Levine, 2018]. Since our environment is computationally expensive, such paradigm is useful to enhance sample-efficiency and reach faster convergence compared to a variety of policy gradient methods, while avoiding the use of on-policy algorithm such as *Proximal Policy Optimization* (PPO) [Schulman and et al., 2017]. Our implementation is composed of 95 multi-layered perceptrons for the actors and a LSTM for the critic. Specifically, we use stochastic policies which predict a mean and a standard deviation for each of the 11 control coils. Once training is completed, exploring possible outcomes is not needed anymore. As a consequence, only the mean of each distribution is kept at inference to predict optimal actions. Sequences were partitioned so that a *burn-in* phase would take place at each learner step, i.e. part of each input sequence sampled from the replay buffer is used to initialize the LSTM core [Kapturowski and et al., 2018]. Adam optimizer was used both in the critic and the actor networks. Specific hyperparameters chosen for NNs definition can be found in table 2, with others as well as initialization practices following [Kerboua-Benlarbi *et al.*, 2024].

Hyperparameter	Chosen value
Batch size	256
Discount factor	0.99
Sequence length for critic	64
Burn-in length critic	10
π_σ	0.5
ϵ	0.5
ϵ_μ	9.09e-5
ϵ_σ	9.09e-8
learning rate	3e-4
dual learning rate	1.5e-2

Table 2: Agent's hyperparameters.

Training framework The interaction loop can be described as follows: a learner worker uses information gathered within a replay buffer to optimize policy and critic NNs; actor threads work independently from each other. Each thread spans a UDS protocol client-server interface with its own random seed, in which the policy interacts with an instance of NICE, sending data to the replay buffer asynchronously; each actor updates its control policy by copying weights periodically from the learner. Evaluation is performed on a separate thread during training using only the mean of the current policy as stated before. This results in a fast and reliable, multi-language, multi-threaded and multi-GPU framework, running numerous instances of the NICE environment in parallel to learn a control policy in Python (Figure 5). Policy networks were all restricted to CPU, in order to lower simulation to reality gaps. Every aspect of the framework then ensures that training can put the agent in realistic conditions with regards to the machine's usual operation. Experiments are performed on a NVIDIA® Tesla™ V100S for the learner, and Intel® Cascade Lake® 6248 at 2.50GHz for the C++ environments. As a side note, the framework is flexible enough to allow fast update or addition of new control scenarios.

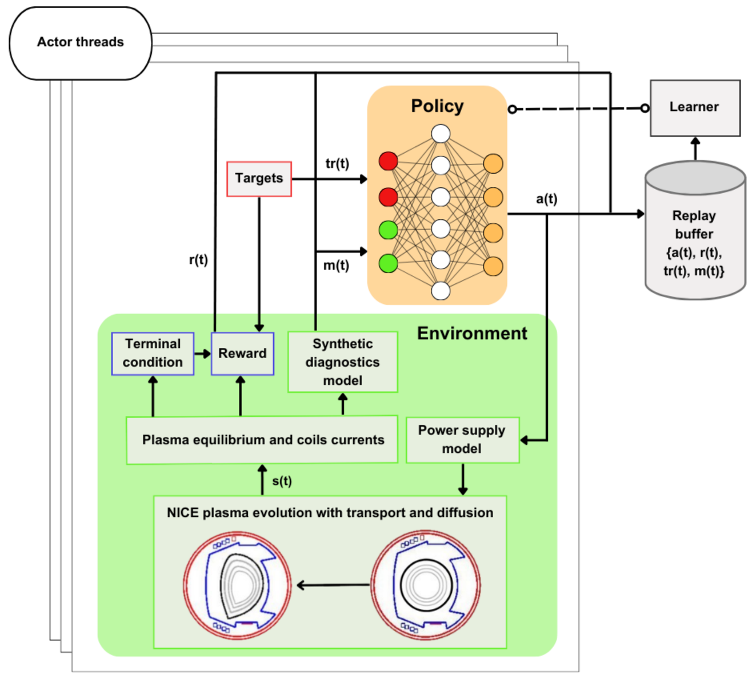


Figure 5: Framework's overview.

4.2 Results

Training results are averaged over 3 different seeds of the evaluator thread. The reward threshold for the TTT is set to 20, as control starts to perform well in such conditions.

Firstly, we know that an environment's step within NICE lasts for about 13 seconds on average during exploration, since the plasma reaches locations of the vacuum chamber in which convergence of the simulation is more difficult. This means that in the complex case, where poor reward signals are common, exploration is long and tedious, increasing computing time of an episode up to 2 hours. Based on this idea, the monitored training time for the vanilla method easily reaches the symbolic threshold of an entire week. Moreover, the reward never exceeds 10 in average, even with training outside the 60 episodes cap scope, which is way under our expectations regarding TTT (Figure 7 - upper). One could mention the fact that we could have undergo further hyperparameters search on the reward definition. However, we kept it general enough to avoid overspecializing the method towards one scenario, leaving more room for adaptation. On the other hand, the CL procedure implicitly leads to reachable states that are easier at the beginning of the initial task. As a consequence, the duration of a simulation's step in this case is shorter in average, and the simulation converges to an equilibrium in about 2 seconds. Next tasks follow on top of this idea, which leads to 10 seconds in average for what is remaining from the curriculum. This leads to episodes computed at worst in 1 hour for complex tasks, which is already an interesting outcome. With that in mind, the reward threshold is reached in about 100 episodes, and the TTT is reduced to approximately 60 hours. As a matter of fact, we observe a clear reduction in convergence time towards the reward threshold, sufficient to gain proper control of the plasma in the configuration of interest (Figure 6a). We stopped training before 60 episodes for the final task, since the return was stable above 20.

If we look at the jumpstart using the total number of episodes, CL actually performs equally, if not worse, than the vanilla method for each curriculum steps (Figure 7 - upper). A simple explanation comes from the fact that the added reward complexity inevitably drops the initial return. Another explanation could arise from so-called catastrophic forgetting. After those sudden drops, the agent fails its first attempt, especially on the last task, but ends up recovering. Recall that we are not stopping previous tasks based on performance, but rather constraining the entire training time to 60 episodes. So, this situation is not entirely surprising, since no optimal behavior was guaranteed at the end of each intermediate curriculum step. Moreover, MPO requires several initial exploratory episodes, in order for training to start concretely. This means that the overall method could also be analyzed without those warm-up interactions, restricting the figure to the last 20 meaningful episodes for example (Figure 7 - lower). In this case, both metrics gives better results, as only improved behaviors are taken into account: the jumpstart is significantly higher, despite the last drop for the last transition, and the time to threslhold is even lower. Actually, drooping the warm-up interactions becomes even more meaningful if we extend transfer to the overall MPO's internal mechanism. A such, exploration would not be as strong as at MPO's initilization, and fine-tuning would be predominant throughout the reward function.

CL does clearly improve the average performance on the final task (Figure 6b), as it performs better than the vanilla policy (Figures 7 - both). It enhances magnetic control, showing that the method does not induce any training instabilities, apart from potential catastrophic forgetting.

Method	Jumpstart on the final task	TTT
Vanilla	4.3	180h
CL	-10.2	60h

(a) Transfer metrics.

	Episode mean reward	Error margin
Vanilla	5.2	± 3.65
CL	18.4	± 4.23

(b) Mean error for each component.

Figure 6: Analysis of the vanilla control policy against the CL method.

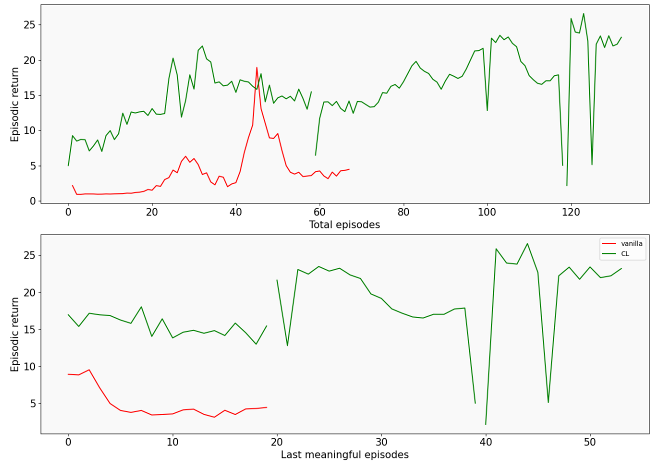


Figure 7: Episodic return for both methods (vanilla - red, CL - green). Since MPO takes several hours to properly start learning, we consider the last episodes that were meaningful regarding reward convergence.

5 Conclusion and perspectives

Curriculum learning displays interesting results in terms of convergence time, while reaching higher levels of performance that a controller exhibits when trained from scratch. Through the simple definition of a sequence of tasks in terms of reward functions, robust magnetic controllers are obtained three times faster than baseline training which requires at least a wee.=k.

This work is one of the first attempts along with [Tracey *et al.*, 2023] to look for practical means of speeding up training of RL-based magnetic controllers. The two methods are also not orthogonal, and combining them could lead to training times even shorter. Moreover, we fixed the action space between tasks, but using the 11 coils might not be useful all the time. Same goes for the magnetic measurements, since nothing indicates that all sensors are useful all the time. Automatic sequencing of the action and state spaces definitions could help in improving the curriculum generation.

A clear limitation of the method comes from the risk of catastrophic forgetting, since we transfer without freezing procedure. A perspective lies in the use of *Progressive Neural Networks* (PNN)[Rusu *et al.*, 2016b], which are not affected by catastrophic forgetting and are theoretically capable of handling complete different tasks. However, big architectures can not efficiently work on real-time control systems due to predictions slower than the timescale of many plasma events. One solution could come from *Policy Distillation* [Rusu *et al.*, 2016a]. By training PNNs through curriculum learning, powerful expert policies could be obtained quickly, and distilled into a smaller network in line with our operational constraints.

References

[Abdolmaleki and et al., 2018a] Abbas Abdolmaleki and Jost Tobias Springenberg et al. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

[Abdolmaleki and et al., 2018b] Abbas Abdolmaleki and Jost Tobias Springenberg et al. Relative entropy regularized policy iteration. *arXiv preprint arXiv:1812.02256*, 2018.

[Ang *et al.*, 2005] Kiam Heong Ang, G. Chong, and Yun Li. Pid control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 13(4):559–576, 2005.

[Ariola and Pironti, 2008] Marco Ariola and Alfredo Pironti. *Magnetic Control of Tokamak Plasmas*. Springer London, 2008.

[Bengio *et al.*, 2009] Yoshua Bengio, Jerome Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, page 41–48. Association for Computing Machinery, 2009.

[Bourdelle and et al., 2015] Clarisse Bourdelle and Jean-François Artaud et al. West physics basis. *Nuclear Fusion*, 55(6):063–017, may 2015.

[Brohan and et al., 2023] Anthony Brohan and Noah Brown et al. Rt-1: Robotics transformer for real-world control at scale. 2023.

[Bucalossi and et al., 2022] Jerome Bucalossi and Joelle Achard et al. Operating a full tungsten actively cooled tokamak: overview of west first phase of operation. *Nuclear Fusion*, 62(4):042007, feb 2022.

[Carpanese, 2021] Francesco Carpanese. Development of free-boundary equilibrium and transport solvers for simulation and real-time interpretation of tokamak experiments. page 238, 2021.

[Char and et al., 2023] Ian Char and Joseph Abbate et al. Offline model-based reinforcement learning for tokamak control. volume 211 of *Proceedings of Machine Learning Research*, pages 1357–1372. PMLR, 15–16 Jun 2023.

[De Tommasi *et al.*, 2022] Gianmaria De Tommasi, Sara Dubbioso, and Yao Huang et al. A rl-based vertical stabilization system for the east tokamak. In *2022 American Control Conference (ACC)*, pages 5328–5333, 2022.

[Degrave and et al., 2022] Jonas Degrave and Federico Felici et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

[Dubbioso *et al.*, 2023] Sara Dubbioso, Gianmaria De Tommasi, and Adriano Mele et al. A deep reinforcement learning approach for vertical stabilization of tokamak plasmas. *Fusion Engineering and Design*, 194:113725, 2023.

[Faugeras, 2020] Blaise Faugeras. An overview of the numerical methods for tokamak plasma equilibrium computation implemented in the nice code. *Fusion Engineering and Design*, 160:112020, 2020.

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

[Goodfellow *et al.*, 2015] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2015.

[Graves *et al.*, 2017] Alex Graves, Marc G. Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks, 2017.

[Grondman *et al.*, 2012] Ivo Grondman, Lucian Busoniu, Gabriel A. D. Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

[Haarnoja *et al.*, 2018] 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.

[Han *et al.*, 2023] Dong Han, Beni Mulyana, Vladimir Stankovic, and Samuel Cheng. A survey on deep reinforcement learning algorithms for robotic manipulation. *Sensors*, 23(7), 2023.

[Harutyunyan *et al.*, 2015] Anna Harutyunyan, Sam Devlin, Peter Vrancx, and Ann Nowe. Expressing arbitrary reward functions as potential-based advice. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Feb. 2015.

[Heumann, 2021] H. Heumann. A galerkin method for the weak formulation of current diffusion and force balance in tokamak plasmas. *Journal of Computational Physics*, 442, 2021.

[Ivanovic and et al., 2019] Boris Ivanovic and James Harrison et al. Barc: Backward reachability curriculum for robotic reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 15–21. IEEE, 2019.

[Kapturowski and et al., 2018] Steven Kapturowski and Georg Ostrovski et al. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2018.

[Kerboua-Benlarbi *et al.*, 2024] S. Kerboua-Benlarbi, R. Nouailletas, B. Faugeras, E. Nardon, and P. Moreau. Magnetic control of west plasmas through deep reinforcement learning. *IEEE Transactions on Plasma Science*, pages 1–0, 2024.

[Kiran and et al., 2022] B. Ravi Kiran and Ibrahim Sobh et al. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2022.

[Levine, 2018] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018.

[Lillicrap and et al., 2015] Timothy P. Lillicrap and Jonathan J. Hunt et al. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[MacAlpine and Stone, 2018] Patrick MacAlpine and Peter Stone. Overlapping layered learning. *Artificial Intelligence*, 254:21–43, 2018.

[Meade, 2009] Dale Meade. 50 years of fusion research. *Nuclear Fusion*, 50(1):014004, dec 2009.

[Mnih and et al., 2016] Volodymyr Mnih and Adria Puigdomenech Badia et al. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[Narvekar *et al.*, 2020] Sanmit Narvekar, Bei Peng, and Matteo Leonetti et al. Curriculum learning for reinforcement learning domains: A framework and survey. *The Journal of Machine Learning Research*, 21(1):7382–7431, 2020.

[Nouailletas and et al., 2023] Rémy Nouailletas and Philippe Moreau et al. West plasma control system status. *Fusion Engineering and Design*, 192:113582, 2023.

[Rusu *et al.*, 2016a] Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation, 2016.

[Rusu *et al.*, 2016b] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

[Schulman and et al., 2015] John Schulman and Sergey Levine et al. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[Schulman and et al., 2017] John Schulman and Filip Wolski et al. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Seo and et al., 2021] Jaemin Seo and Yong-Su Na et al. Feedforward beta control in the KSTAR tokamak by deep reinforcement learning. *Nuclear Fusion*, 61(10):106010, 2021.

[Seo *et al.*, 2024] Jaemin Seo, SangKyeun Kim, Azarakhsh Jalalvand, Rory Conlin, Andrew Rothstein, Joseph Abbate, Keith Erickson, Josiah Wai, Ricardo Shousha, and Egemen Kolemen. Avoiding fusion plasma tearing instability with deep reinforcement learning. *Nature*, 626:746–751, 02 2024.

[Soviany *et al.*, 2022] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565, 2022.

[Stanley *et al.*, 2005] Kenneth O. Stanley, Bobby D. Bryant, and Risto Miikkulainen. Evolving neural network agents in the nero video game. 2005.

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

[Tracey *et al.*, 2023] Brendan D. Tracey, Andrea Michi et al., and The TCV Team. Towards practical reinforcement learning for tokamak magnetic control. *ArXiv*, abs/2307.11546, 2023.

[Wakatsuki and et al., 2019] Takuma Wakatsuki and T. Suzuki et al. Safety factor profile control with reduced central solenoid flux consumption during plasma current ramp-up phase using a reinforcement learning technique. *Nuclear Fusion*, 59(6):066022, 2019.

[Wakatsuki *et al.*, 2021] Takuma Wakatsuki, T. Suzuki, N. Oyama, and N. Hayashi. Ion temperature gradient control using reinforcement learning technique. *Nuclear Fusion*, 61(4):046036, mar 2021.

[Wesson, 2004] John Wesson. Tokamaks 3rd edition. *Journal of Plasma Physics*, 71(3):377–377, 2004.

[Wołczyk *et al.*, 2022] Maciej Wołczyk, Michał Zajac, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Disentangling transfer in continual reinforcement learning, 2022.

[Wu and Tian, 2017] Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. In *International Conference on Learning Representations*, 2017.

[Zhu *et al.*, 2023] Zhuangdi Zhu, Kaixiang Lin, Anil K. Jain, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):13344–13362, 2023.