Magnetic control of WEST plasmas through deep reinforcement learning

S Kerboua-Benlarbi, R Nouailletas, Blaise Faugeras, E Nardon, P Moreau

Magnetic control of WEST plasmas through deep reinforcement learning

S. Kerboua-Benlarbi, R. Nouailletas, B. Faugeras, E. Nardon, P. Moreau

***Abstract*—Tokamaks require magnetic control across a wide range of plasma scenarios. The coupled behavior of plasma dynamics makes deep learning a suitable candidate for efficient control in order to fulfil these high-dimensional and non-linear situations. For example, on TCV, deep reinforcement learning has already been used for tracking of the plasma's magnetic equilibrium [1]. In this work, we apply such methods to the WEST tokamak, to address control of the plasma's shape, position, and current, in several relevant configurations. To this end, we developed a distributed framework to train an actor-critic agent on a C++ free boundary equilibrium code called NICE, in which resistive diffusion allows a more representative evolution of current profile throughout the simulation. The interface between components was done through UDS protocols for fast, asynchronous and reliable communication. The implemented tool handles feedback control of quantities of interest, with results showing flexibility of the method regarding the use of different training environments.**

***Index Terms*—Reinforcement learning, Neural networks, Plasma control, Distributed computing.**

I. Introduction

MAGNETIC control plays a crucial role in maintaining the stability and performance of plasma's confinement within tokamaks. Control systems actively adjust the voltages applied to the poloidal field coils (PFC), precisely manipulating magnetic fields within the said devices. Such process allows to control quantities intrinsically linked to plasma's behaviour, like position, shape and current, through the use of advanced real-time algorithms. Scientists rely on these tools to study the effects of various configurations on plasma dynamics, such as elongated shapes and their related vertical instabilities[21, 17, 24, 4]. Hence, there is a essential need for flexibility and adaptability of control systems without which no proper plasma could be produced.

WEST is a full tungsten environment superconducting tokamak with a divertor configuration located in France [8, 10]. On such machine, tracking of plasma's shape, position and current is achieved through linear feedback control [14, 15]. Several single-input-single-output PIDs are traditionally built to regulate the said quantities, all of which must be independently designed to not interfere with each other. Plasma's shape and position can not be observed directly, and are instead inferred in real-time from magnetic sensors using reconstruction codes [16, 2]. This overall setup requires substantial engineering effort whenever target configurations undergo variations, and show limits with respect to the coupled behaviour of plasma dynamics. Indeed, linear control laws are suitable for maintaining stability in a narrow operating range within known scenarios, but nonlinear control may be required for more advanced exploration.

Reinforcement Learning (RL) [22] is a machine learning paradigm emerging as an innovative approach to real-time control. An environment is designed as a representation of the physical plant, its state denoted as s_t at each timestep t. An *agent* receives a set of measured *observations* which are function of this state, $o_t = o(s_t)$, and a *reward* r_t. In return, it sends control signals known as *actions* a_t to the environment, according to a control *policy* $\pi(o_t) = a_t$ mapping state space S to action space A. Accordingly to its transition function, the environment evolves to a new state denoted as $s_{t+1} = s(s_t, a_t)$. More precisely, the reward signal $r_t = r(s_t, a_t, s_{t+1})$ is a real-valued function designed by humans which indicates whether last action a_t in state s_t was in line with the overall control objectives. The sequence of triplets $\{s_t, r_t, a_t\}$ is repeated until a terminal condition is reached, which corresponds to a situation that we must avoid within the environment (coils currents saturation, undesired plasma position transient, etc). An *episode* is then formed and the environment is reset to its initial conditions. The goal of RL is then to make the agent learn an optimal policy $\pi^* : S \to A$ which maximizes the discounted cumulative reward over the course of an episode:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}[\sum_{k=0}^{T} \gamma^k r_{t+k+1}] \tag{1}$$

with discount factor $\gamma \in [0, 1]$ working as a penalization term for long-term rewards.

RL is becoming increasingly popular among plasma control research. For example, RL has been used for vertical stabilization [6], for control of β_n [9, 3], to build feedforward trajectories of plasma β [11], or even for safety factor profile control [23]. Recent works [5, 7] designed and optimized a RL-based system which achieved "full" magnetic control of the *Tokamak à Configuration Variable* (TCV). The learned policy output voltages for all 19 magnetic control coils, by observing TCV's raw magnetic measurements (38 magnetic probes and flux loops), demonstrating the capability for RL-based systems to tackle a various set of plasma configurations. These examples highlight an explicit shift of focus from controllers designed with *à priori* constraints on how control should be performed on the plant, to controllers learning by trial-and-error to act on the system based on *what* should be achieved. By leveraging neural networks as powerful function approximators, deep RL's advantages stem from its ability to:

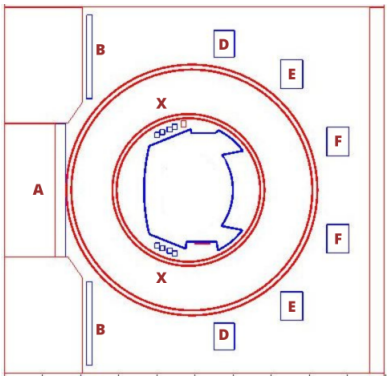


Fig. 1: Cross-section of WEST with its PFCs configuration.

- to fulfil these high dimensional, uncertain and non-linear systems
- avoid the need for reconstruction codes
- explore possible strategies in order to make the control policy more flexible in contrast with the fixed heuristics of classical control

While both need tuning of a set of parameters (gains, reward function), RL is particularly valuable in situations where classical control methods may fall short due to the stated challenges.

In this paper, we apply such methods to the WEST use-case (Figure 1), to address tracking of plasma's shape, position, and current in a relevant baseline scenario.

The main objective of this article is to describe the developed framework, in which domain knowledge of the agent is structured with resistive diffusion inside the simulation, while working on specific reward engineering to account for objectives of interest. Next sections will be organized as follows. First, we will describe how the simulated environment is implemented, then followed by details on how the framework was designed to allow efficient execution of the training loop. The nominal training scenario will then be presented, notably details about the environment's initial conditions, state and reward's specification. Finally, validation and analysis of the learned policy will be performed, before concluding on the study and its perspectives.

II. A GENERAL DISTRIBUTED FRAMEWORK FOR WEST

A. The NICE environment

The NICE code [16] is a C++ free-boundary equilibrium code solving the *Grad-Shafranov* equation, a non-linear 2D elliptic partial differential equation of the magnetic flux ψ in time and space in tokamaks:

$$-\Delta^*\psi = Rp'(\psi) + \frac{1}{\mu_0 R} f f'(\psi) \tag{2}$$

with magnetic permeability μ_0, radius R, pressure $p(\psi)$, flux function $f(\psi) = RB_\phi$. Given prescribed $p(\psi)$ and $f'(\psi)$, we solve it for ψ such that $\psi \to 0$ as $(R, Z) \to \infty$.

One could augment the model described by (2) with resistive diffusion [18] in order to model the dynamics of plasma's magnetic flux profiles. It will help the controller learn to better control plasma's characteristics such as I_p, which variations are modeled differently than alternative methods. Indeed, a lumped-circuit equation [5], or a $0D$ flux consumption model [15] could be of interest, but the use of the present extension shows significant benefits:

- it influences how the current density distributes within the plasma;
- it accounts for how the magnetic field lines evolve over time as they diffuse. This is crucial for better simulation of the plasma's magnetic configuration changes during different phases of a discharge;
- it leads to a more representative evolution of the total plasma current I_p:

$$I_p := \int_{\mathbf{P}} \left(Rp'(\psi) + \frac{1}{\mu_0 R} f f'(\psi) \right) dr dz \tag{3}$$

with $\mathbf{P}$ defining plasma domain.

Overall, it provides a more complete and physically accurate representation of the plasma's behavior. Hence, the set of information sent to the agent is more in line with reality. Finally, with considerations on time, given active coils voltages $V_a(t)$, we solve for $\psi(t)$ and active coils currents $I_a(t)$. This forward evolution mode computes the environment's state at each new training step within one episode. Moreover, power supply and diagnostic models are implemented, to give an accurate representation of the plasma control system on WEST.

B. Communication protocols

While NICE is written in C++, optimized deep learning libraries benefit from the use of Python, e.g. TensorFlow [12]. Therefore, there is a need for a proper communication protocol, to let these building bricks from different languages interact with each other. Several communication tools could have been used, each one of them with its advantages and drawbacks. To choose properly between these options, one must take into account the specifications of a RL training loop for plasma control. Given the computing timescales of NICE, communication must be fast and reliable, so that training duration would be optimized without modifying in depth the numerical solver, and to avoid loosing any valuable information in the observational data.

Such needs confirm sockets as suitable platforms to connect these logical components, not only considering their flexibility, but also because of the vast set of involved protocols. A first attempt relied on TCP protocols which helped in creating a development routine for further extensions. Unix domain sockets (UDS) prove to be a valuable enhancement, analyzing how transfer is observed with sending times kept at 14 microseconds in average (compared to TCP's 49 microseconds) between the agent set up as a server, and the environment as a client, with serialized structures weighting approximately $6ko$ for states. By this mean, time taken for communication is almost 4 times faster than within previous procedure.

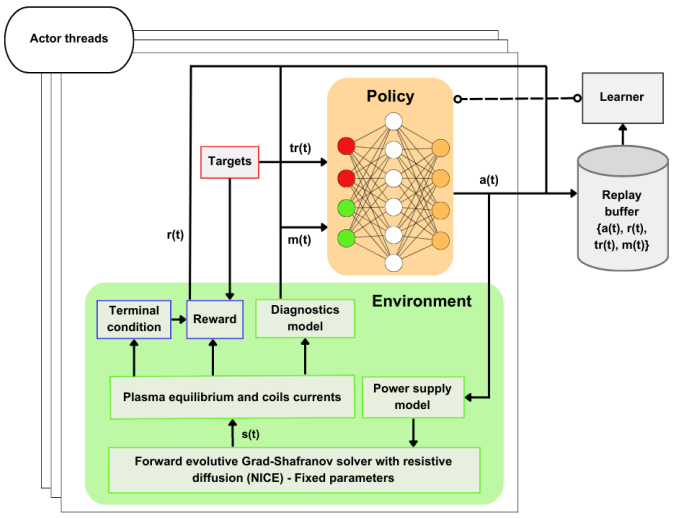


Fig. 2: Framework's overview. Diagnostics and power supply model take into account bias, delays and offsets.

C. Overview

From previous considerations, a general framework based on [5, 13] has been developed in order to build the interaction loop inherent to RL setups. The agent's architecture is based on actor-critic models [20], which is here turned into a distributed configuration. The interaction loop can be described as follows:

- A learner worker contains policy and critic stored neural networks (NN). It uses information gathered within a replay buffer, which works as a dataset filled in an online manner, to optimize weights of said NNs;
- Actor threads works independently from each other. Each thread spans a client-server interface, in which a policy network interacts with its own instance of NICE, sending relevant tuples to the replay buffer asynchronously;
- Each actor updates its control policy by copying weights periodically from the learner stored policy, retrieving the best behaviour obtained so far.

This results in a fast and reliable, multi-language and multi-threaded framework, running numerous instances of the NICE environment in parallel to learn a control policy in Python. Graphical processing units (GPU) were intensively used to increase computing performance of learner steps. But policy networks were all restricted to CPU, in order to lower simulation to reality gaps, i.e. transfer of the control policy to the plasma control system. Every aspect of the framework then ensures a correct representation of the real control system, so that training can put the agent in the most realistic conditions with regards to the machine's usual operation.

D. Nominal experiment

The objective was to follow a defined plasma trajectory in limiter configuration, and maintain an initial Ip over periods of 350 milliseconds, with a simulation timestep set to 10^{-3} second. This duration was chosen to follow typical transition times on WEST, while shorter intervals made convergence of the algorithm difficult to reach. Figure 2 gives an overview of the interaction loop, which was tested against this baseline configuration.

The NICE environment is initialized to a limited shaped plasma, extracted from a recent discharge, with plasma's resistivity η and the non-inductive current density J_{ni} set as fixed parameters. An inverse problem is solved at each new training episode, to get optimized currents from the initial shape. This procedure ensures a stable starting point for the simulation, so that training can be performed with realistic outputs, and smooth convergence. The error tolerance on the residual of Newton solver was set to $1e-5$ instead of the usual value of $1e-10$, to speedup training without significant loose of accuracy in NICE outputs. The state's environment was defined as:

$$s(t) = \{y(t), I_a(t), m(t)\} \tag{4}$$

with $y(t)$ the plasma equilibrium information, $I_a(t)$ the currents in the active control coils, and $m(t)$ the raw magnetic measurements. Termination is triggered if thresholds are reached on active coils currents or safety factor, to avoid any damage on the device.

Rewards are computed from $s(t)$, with target references $tr(t)$ linearly interpolated from a set of snapshots extracted from experimental data, in order to make sure that the relevant scenario exists within the operational domain of the device. The chosen reward is a normalized combination of error signals, each one of them focusing on a specific sub-task of magnetic control (shape, plasma current, etc).
For a description of each component's weight and parameters, please refer to table I.

Each reward component c is computed as the difference between its target value $tr_c(t)$ and the retrieved one from the NICE environment $v_c(t)$. To get a proper functional reward,

Algorithm 1 Reward calculation pseudo-code

C, set of reward components, TR set of corresponding targets, W set of corresponding weights.

SOFTPLUS(E)
  $E \leftarrow \xi(\frac{E-bad}{good-bad})$
  $E \leftarrow \sigma(E)$
  return $min(max(2 \cdot E, 0), 1)$

SMOOTHMAX(R, W)
  return $\frac{\sum w_i R_i e^{\alpha R_i}}{\sum w_i e^{\alpha R_i}}$

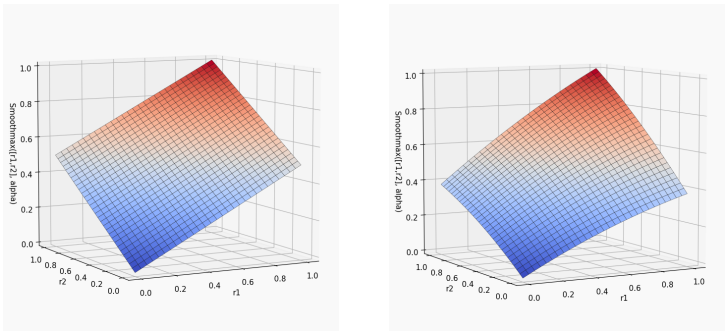
COMPUTE(C)
$R \leftarrow \{\}$
for each $c \in \mathcal{C}$ **do**
  if c scalar **then**
    $E \leftarrow |v_c(t) - tr_c(t)|$
    $R_c \leftarrow SOFTPLUS(E)$
  else
    $R_c \leftarrow SMOOTHMAX(\{R_{c_i}\}_{1 \leq i \leq size(c)}, 1)$
  end if
  APPEND(R, R_c)
end for
return $SMOOTHMAX(R, W)$

TABLE I: 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.

Component	Description	Good	Bad	α
LCFS [m]	Distance between 32 points from current and target LCFS	0.0	0.04	-1
Magnetic center (r,z) [m]	Distance from current and target magnetic center location	0.005	0.05	x
Radius [m]	Difference in minor radius	0.002	0.02	x
Elongation	Distance between computed and target κ	0.002	0.02	x
Triangularity	Distance between computed and target δ	0.005	0.02	x
Ip [kA]	Difference in plasma current	0.5	30	x
Final combiner: *Smoothmax(α = -0.5)*				

scaling is applied to each c in order to get intermediate values between 0 and 1. If the component is made out of several targets, like shape control using multiple last closed flux surface (LCFS) reference points, they are combined with *Smoothmax* function to get a scalar value within the wanted interval. The same function is used to combine all reward components into a final reward value in $[0, 1]$. Algorithm 1 shows the reward computation more precisely, following several standards proposed by [5].

Weights in the *Smoothmax* definition affects the importance of each reward component, while the α defines focus balance between components of different control complexities (Figure 3). A negative value will give rewards close to the least performing component, leaving other ones loosely explored, while positive values will exclude the said worst components. Such trade-offs are important, since the closer α gets to 0, the more all components will be treated equally, e.g. scenarios like X-point formation where a component on the X-point location would be closely related to an LCFS tracking one. In our case, weigths are kept at 1 for all components, except for the LCFS one which is equal to 3. As for the α parameter, we emphasize all LCFS components almost equally by setting it to -1, and -0.5 is chosen for the final smoothmax combination. Indeed, our scenario should consider all objectives "almost" equally looking for small exploration, since initial conditions are similar but not that close to targets.



(a) $\alpha = 0$ (b) $\alpha = -0.5$

(c) $\alpha = -1$ (d) $\alpha = -5$

Fig. 3: $Smoothmax(r_1, r_2, \alpha)$, with r_1, r_2 scaled reward components in $[0, 1]$. Focus is directed towards the worst component as $\alpha \to -\infty$. Such non-linear scaling allows to refine objectives specification during training.

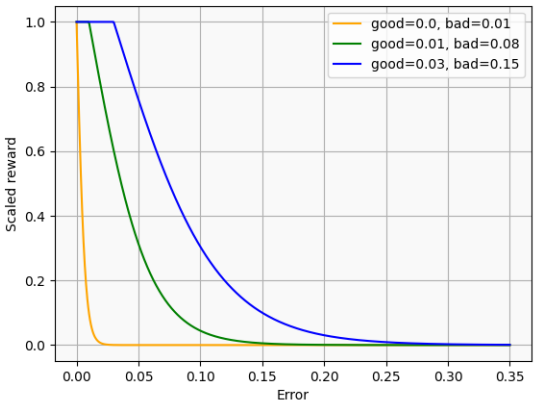


Fig. 4: *Softplus* behaviour. If the error is smaller than the good parameter, the reward will saturate to 1. If it is worse than bad, reward decays to 0. ξ describes scaling "steepness" between the two anchor points, and is fixed at $log(-19)$ [5].

Good and *bad* formulation in the *Softplus* formulation scales the reward signal according to region of interest in the reward space (Figure 4). Tight values in both parameters will lead to higher focus on the component to achieve high related reward, making it difficult to get valuable signals when the training scenario involves strong variations and exploration should occur. On the other hand, smoother values will make the components easier to satisfy, helping exploration at the cost of precise control. Considering our baseline, which initial conditions are similar to the final targets, choosing relatively tight values is suitable since the region of interest does not need extensive exploration. *Sigmoid* and asymptotic scaling functions were tested but the use of the *Softplus* function happened to be more suitable. Reward undergo a final scaling, so that the maximum cumulated reward for 100 ms equals 10.

The agent is a distributed Maximum à posteriori Policy Optimization agent [1] (MPO) implemented in Python with 95 multi-layered perceptrons and a recurrent learner (Long Short Term Memory - LSTM). NN architectures and weights initialization follow [5] considering that variations from the

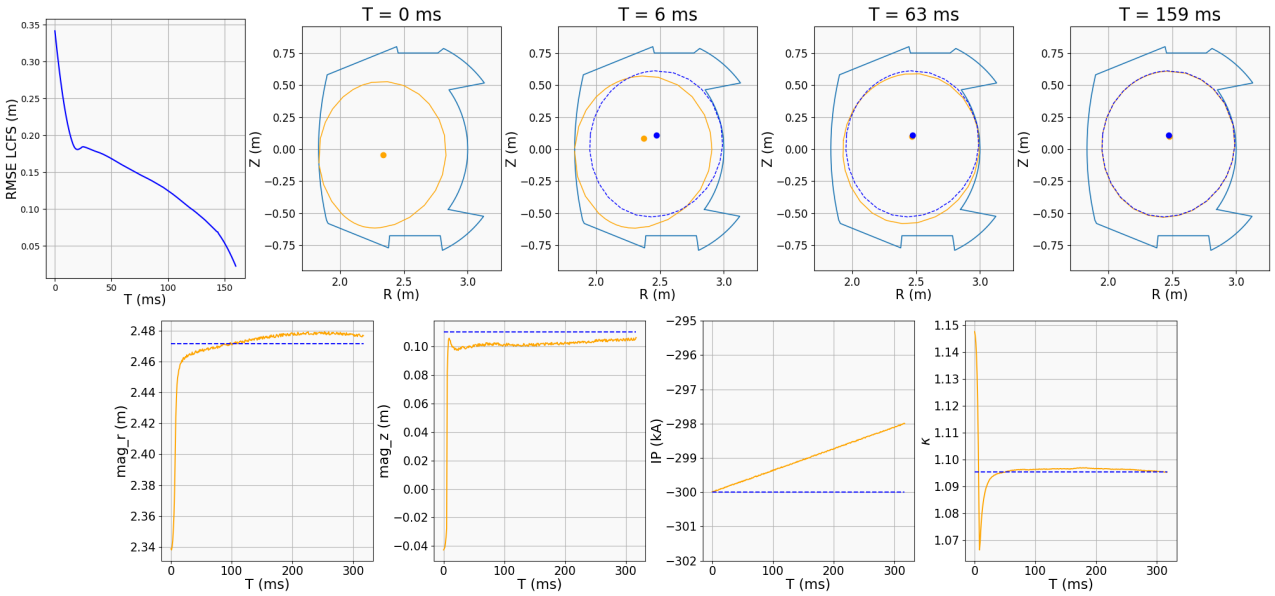


Fig. 5: Tracking of reference (in blue) plasma's shape and position is achieved without significant bias regarding observed quantities (in orange), except for I_p which is about $2kA$ from the setpoint.

latter were benchmarked and lead to worse performance, with use of specialized CUDA RNN unroll operations. 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 used to initialize LSTM core [19]. Adam optimizer was used both in the critic and the actor networks. Specific hyperparameters chosen for NNs definition can be found in table II.

TABLE II: MPO's hyperparameters.

Hyperparameter	Chosen value
Batch size	256
Discount factor	0.99
Sequence length for critic	80
Burn-in length critic	15
Actor architecture	(256, 256, 256, 256, 22)
Critic architecture (LSTM, MLP)	(256 units, 256, 256, 1)

The framework feeds each policy network with augmented observations o(t) from the replay buffer:

$$o(t) = \{tr(t), m_b(t), fl(t), I_a(t), \frac{dm_b(t)}{dt}\} \tag{5}$$

with $\{m_b(t), fl(t)\}$ magnetic probes and flux loops raw measurements, and $\frac{dm_b(t)}{dt}$, temporal derivatives of magnetic probes signals. Specifically, they are chosen as pairs measuring the poloidal magnetic field in both tangent and normal directions at given locations. 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. Predicted voltages supplied to each PFC are sampled from distributions defined by outputs of the control policy. Indeed, to favor exploration throughout learning, mean and standard deviation of each of the 22 active coil distributions are learnable parameters, and only the mean is kept at inference. Offsets, bias and delays are part of the power supply model to ensure correct handling of WEST actuators in the real world.

III. Training and Evaluation

Training was performed on an hybrid architecture with an NVIDIA® Tesla™ V100S with 32GB of available memory operating at 1230 MHz and several Intel® Xeon Gold® at 2.10GHz (160 logical « cores »). Training time took about 2 days for proper convergence of the algorithm.

Firstly, control of the plasma shape, through the LCFS, is carried out correctly (Figure 5). The mean square error over all the points of interest decreases substantially when considering the final target shape. It should be noted that the weight placed on the LCFS reward component is greater than those placed on the other components. As for the control of the magnetic centre and κ, it is carried out without producing an excessive stabilisation bias.

Nevertheless, there are steady-state errors on the center position. Within the reward, this component considers a wide interval between good and bad parameters. Choosing a stricter interval could be more effective, but despite several tests, a bias is still present and could be explained by how the magnetic center is computed. Indeed, its coordinates are not interpolated, but only taken from the nodes of the mesh. However, it is insufficient to entirely account for the displacements of the point of interest, and another explanation could relate to the lack of proper integral control.

In parallel, plasma current stays within $2kA$ from initial conditions. We initially observed numerical instabilities within the resisitive diffusion mode of NICE, causing unrealistic surges of plasma current up to $100kA$ in a few milliseconds. This unrealistic behaviour was put in the perspective of other controlled characteristics, and stabilization of the solver was required. Plasma control then becomes more precise, considering that we might still be within a transient phase. Increasing

scenario duration is a short-term perspective of this work to check Ip control.

IV. Conclusion and perspectives

This study presented a fast, reliable and maintainable multi-language and multi-GPU framework. It allows training of RL agents, which achieved accurate tracking of plasma's shape and position. Without any reconstruction step, non-linear control was performed thanks to RL principles, demonstrating its usefulness in the WEST use-case. Despite lack of efficient plasma current control, several mechanisms are already identified to reduce involved bias. Counting on reward engineering and integral control, the framework will be extended for better convergence of the control policy.

Once the controller will successfully tackle the baseline scenario, the framework will be used to train the agent on more complex tasks, e.g. X-point transitions. NICE will be coupled with a transport model like METIS, to enhance NICE performance, notably which new information will help initialization of resistive diffusion. Finally, the control policy will be evaluated first on a "flying" simulator before deployment on WEST.

V. Acknowledgments

The authors would like to thank Federico Felici from the SPC at EPFL, and the Fusion team at DeepMind, for their valuable advice and expertise regarding the use of reinforcement learning for magnetic control. This work has been supported by Capgemini Engineering, and the Provence Alpes Côte d'Azur region.

References

[1] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

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

[3] I. Char, Y. Chung, M. Boyer, E. Kolemen, and J. Schneider. A Model-Based Reinforcement Learning Approach for Beta Control. In *APS Division of Plasma Physics Meeting Abstracts*, volume 2021 of *APS Meeting Abstracts*, 2021.

[4] G. Cunningham. High performance plasma vertical position control system for upgraded mast. *Fusion Engineering and Design*, 88(12):3238–3247, 2013.

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

[6] S. Dubbioso, G. De Tommasi, A. Mele, G. Tartaglione, M. Ariola, and A. Pironti. A deep reinforcement learning approach for vertical stabilization of tokamak plasmas. *Fusion Engineering and Design*, 194:113725, 2023.

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

[8] C. Bourdelle et al. West physics basis. *Nuclear Fusion*, 55(6):063017, may 2015.

[9] I. Char et al. Offline model-based reinforcement learning for tokamak control. In N. Matni, M. Morari, and G. J. Pappas, editors, *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*, volume 211, pages 1357–1372. PMLR, 2023.

[10] J. Bucalossi et al. *Nuclear Fusion*, 62(4):042007, feb 2022.

[11] J. Seo et al. Feedforward beta control in the KSTAR tokamak by deep reinforcement learning. *Nuclear Fusion*, 61(10):106010, 2021.

[12] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[13] M. W. Hoffman et al. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.

[14] R. Nouailletas et al. The west plasma control system: Integration, commissioning and operation on first experimental campaigns. *Fusion Engineering and Design*, 146:999–1002, 2019.

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

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

[17] Y. Gribov, A. Kavin, V. Lukash, R. Khayrutdinov, G.T.A. Huijsmans, A. Loarte, J.A. Snipes, and L. Zabeo. Plasma vertical stabilisation in iter. *Nuclear Fusion*, 55(7):073021, jun 2015.

[18] H. Heumann. A galerkin method for the weak formulation of current diffusion and force balance in tokamak plasmas. *Journal of Computational Physics*, 442:110483, 2021.

[19] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2018.

[20] V. Konda and J. Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[21] F. Pesamosca, F. Felici, S. Coda, C. Galperti, and the TCV Team. Improved plasma vertical position control on tcv using model-based optimized controller synthesis. *Fusion Science and Technology*, 78(6):427–448, 2022.

[22] R. S. Suttion and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[23] T. Wakatsuki, T. Suzuki, N. Hayashi, N. Oyama, and S. Ide. 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.

[24] M.L. Walker and D.A. Humphreys. On feedback stabilization of the tokamak plasma vertical instability. *Automatica*, 45(3):665–674, 2009.