Software Impacts 27 (2026) 100829

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

Gym-TORAX: Open-source software for integrating reinforcement learning with plasma control simulators in tokamak research

Antoine Mouchamps *, Arthur Malherbe, Adrien Bolland, Damien Ernst

Montefiore Institute, University of Liège, Liège, Belgium

ARTICLE INFO

Keywords:
Reinforcement learning
Tokamak
Plasma control
Fusion energy
Open-source software

ABSTRACT

This paper presents Gym-TORAX, a Python package to define Reinforcement Learning (RL) environments for plasma control in tokamaks. Gym-TORAX instantiates a Gymnasium environment from an action space, a state-observation space, and a reward function that measures plasma characteristics. The environment computes plasma states using the TORAX plasma simulator and the objective is to maximize the expected sum of rewards. This plasma control formalization is compatible with most RL algorithms and libraries to facilitate RL research and applications. In its current version, one environment is readily available, based on an International Thermonuclear Experimental Reactor (ITER) scenario.

Code metadata

Current code version	v1.0.0
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2025-306
Permanent link to Reproducible Capsule	
Legal Code License	MIT license (MIT)
Code versioning system used	git
Software code languages, tools, and services used	Python, TORAX, Gymnasium
Compilation requirements, operating environments & dependencies	Python 3.10, TORAX 1.0.3, Gymnasium 1.2
If available Link to developer documentation/manual	https://gymtorax.readthedocs.io/latest/
Support email for questions	antoine.mouchamps@gmail.com

1. Introduction

One of the most prominent areas of research in fusion energy is the optimization of the stability and performance of fusion reactors. A notable share of the research effort is dedicated to reactors of the tokamak configuration, a torus-shaped device where fusion conditions are achieved through magnetic confinement [1]. However, the control and design of these devices has proven to be challenging, notably because of the high dimensionality of the problem and the many nonlinearities inherent to plasma control [2].

In recent years, Reinforcement Learning (RL) has emerged as a promising approach to tackle such complex control problems. In addition to successes in domains such as robotics [3,4], electricity markets [5,6], and power system analysis [7,8], RL has also been recently applied to plasma control-related problems. In particular, Degrave et al. [9] focused on maintaining specific plasma shapes using a set of magnetic coils, Char et al. [10] focused on controlling β_N, a plasma stability and performance measure, while Seo et al. [11] trained agents to avoid a certain type of plasma instability. We believe that previous successful applications have been made possible partly thanks to accessible simulation tools and software frameworks that abstract away from the user the underlying physics. This allows RL researchers to focus on optimizing the control strategy.

To support the development of RL applications in plasma control, we present a new Python package, Gym-TORAX, a Gymnasium [12] wrapper around the TORAX simulator [13]. Gym-TORAX allows the creation and utilization of plasma control environments that are ready-to-use by RL algorithms. These environments can be used to represent various operational scenarios such as steady-state control or ramp-up/down.

* Corresponding author.
E-mail addresses: antoine.mouchamps@gmail.com (A. Mouchamps), arthur.m222004@gmail.com (A. Malherbe), adrien.bolland@uliege.be (A. Bolland), dernst@uliege.be (D. Ernst).

https://doi.org/10.1016/j.simpa.2026.100829
Received 27 October 2025; Accepted 16 February 2026

2665-9638/© 2026 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

The remainder of this manuscript is structured as follows. First, the Gym-TORAX package is described in Section 2, along with requirements for the implementation of new environments. Then, the motivation and impacts of our package on plasma control research are discussed in Section 3. Finally, Section 4 concludes this manuscript with some words about future improvements and functionalities.

2. The Gym-TORAX package

Our package relies on the TORAX simulator for plasma dynamics. This section therefore begins with an overview of its core features and assumptions. We then describe the control problems that can be simulated using TORAX. The end of this section explains the procedure to create new Gym-TORAX environments.

2.1. TORAX description

TORAX is an open-source simulator written in Python and using JAX [14] for fast auto-differentiation and runtime.

TORAX simulates the evolution of the plasma state. This plasma state is essentially composed of the ion and electron temperatures $T_{i,e}$, the ion, electron, and impurities densities $n_{i,e,imp}$, and the poloidal magnetic flux ψ. Secondary metrics such as the safety factor q, beta coefficient β, and fusion gain Q_{gain}, also considered part of the state, are also computed to evaluate plasma performance. By making use of axi-symmetries, the plasma is reduced to evolving along a single spatial dimension, the normalized radius of the plasma cross-section. All space-dependent variables are discretized along this dimension.

Every TORAX simulation starts with a configuration file, which sets, among other things, the initial conditions of the plasma state and its geometry, physical equations to solve, as well as numerical and solver-related quantities. The configuration file also sets the finite time horizon of the simulation and the numerical discretization scheme (*chi* for adaptive time steps or *fixed* for fixed ones).

In addition to the initial values of the state variables described previously, plasma-related time series are given at each simulation time step. Those time series are related to either the total current I_p or the loop voltage V_{loop}, the ion, impurities, and effective charge numbers $Z_{i,imp,eff}$, and various energy or particle sources. From a control perspective, the variables V_{loop}, I_p, and the energy and particle sources can be regarded as control variables. By default, TORAX therefore operates as an open-loop simulator, where these inputs are predefined for the entire simulation.

In order to compute the plasma-state evolution over time, TORAX solves a system of Partial Differential Equations (PDEs). Among these equations, important ones are so-called transport equations. Transport equations are PDEs that describe the spatial evolution of a given quantity in a moving medium over time. In TORAX, the transport equations related to plasma dynamics are the ion and electron heat transport and electron particle transport equations. In addition to these, other important equations are the plasma composition equations and the current diffusion. From these equations and given initial conditions and time series, the plasma state can be fully determined from one simulation time step to the next. In addition to computing the plasma state, geometric and transport equations related quantities are also updated at every simulation time step. Fig. 1 illustrates this update step, grouping variables into *state variables*, *time series*, and *derived quantities* for clarity.

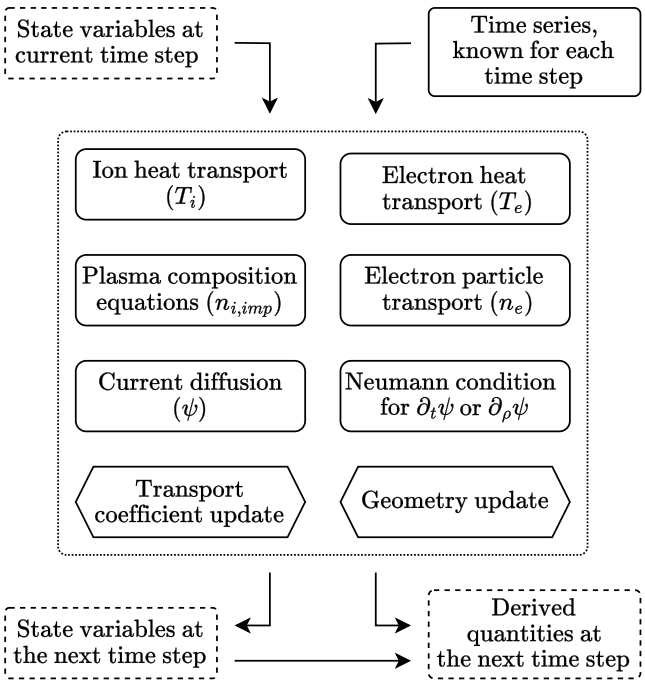

Fig. 1. Update loop in a single TORAX simulation step. Given the plasma state at the current simulation time step and the time series, TORAX solves the system of equations to compute the plasma state at the next time step. Derived quantities, such as performance metrics, are then calculated from this updated state. In the figure, dashed boxes represent variables computed iteratively, while plain boxes represent time series known in advance for the whole simulation.

2.2. Modeling approach

The Gym-TORAX package wraps around the TORAX simulator to implement a closed-loop control environment.

The control problem is modeled as a finite-time deterministic Markov Decision Process (MDP) $(S, \mathcal{A}, f, r, s_0, \gamma, T)$ with state space S, action space $\mathcal{A}$, deterministic transition function $f : S \times \mathcal{A} \to S$, deterministic reward function $r : S \times \mathcal{A} \to \mathbb{R}$, initial state $s_0 \in S$, discount factor $\gamma \in [0, 1]$, and time horizon $T \in \mathbb{N}_0^+$.

To implement an MDP based on the TORAX simulator, we introduce a two-level discretization. The first level is the temporal discretization corresponding to the reinforcement learning interaction cycle: at time t, the agent receives a complete observation of the plasma state $s_t \in S$. From s_t, the agent determines the action $a_t \in \mathcal{A}$ to apply to the environment. The transition function f applies the action and returns the next state $s_{t+1} \in S$. An application-dependent reward $r_t \in \mathbb{R}$ is afterward computed.

The second level of discretization concerns the evaluation of the transition function using TORAX. Each transition corresponds to solving the previously discussed PDEs for K time-steps with TORAX. The final state of each TORAX simulation is returned as s_{t+1} to the environment. In our package, two numerical discretization options are available: *auto* and *fixed*. Using the *auto* option, the total number of simulation time steps K_t within an environment transition from t to $t+1$ will be dynamic and automatically defined by TORAX. On the contrary, the *fixed* option enforces a constant number of time steps K for each simulation run.

An agent selects an action based on its policy π. A policy is a distribution of actions, usually conditioned on the Markov state. The objective of RL is to maximize its expected return, given by

$$J(\pi) = \mathop{\mathbb{E}}_{\pi}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right] \quad . \tag{1}$$

The reward function is task-specific (e.g., stability, power generation, etc.) and is designed by the user such that the expected return is maximized when the task is achieved.

All variables given as time series in TORAX can be used as actions by the agent and shall then not be specified beforehand except for initial conditions. Those not controlled by the agent must still be specified as time series over the entire time horizon of the environment.

2.3. Design a Gym-TORAX environment

New environments are created by extending the `BaseEnv` class. In order to complete the definition of the MDP, the four following abstract methods of the `BaseEnv` class have to be implemented:

- `_get_torax_config()` defines the simulation discretization (see Section 2.2) and provides the TORAX configuration file (e.g., initial conditions, physical models, etc.) from which are extracted the initial state s_0 and the time horizon T.
- `_define_action_space()` specifies the subset of TORAX actions that are controlled by the agent and optional ramp rate limits on these actions from one time step to the next;
- `_define_observation_space()` selects which TORAX variables and derived quantities are included in the agent observation, potentially making the MDP partially observable;
- `_compute_reward()` defines the reward function r of the environment.

To simplify environment design, the `_define_action_space()` and `_define_observation_space()` methods rely on configurable abstract classes: the `Action` class for action definitions (to specify which variable is controllable and how it relates to the TORAX configuration file) and the `Observation` class for observation content. Multiple helper methods are also available in a dedicated `reward` file for reward definitions.

In case the TORAX simulation returns an error or the state becomes unfeasible, the simulation terminates. Consequently, the episode is terminated using the dedicated Gymnasium flag, and a large negative reward $r_t = -1000$ is returned to the agent. If the action a_t falls outside the action space $\mathcal{A}$ or if the (optional) ramp rate constraints are violated, the action is clipped before being applied to the environment, and a flag is raised in the `info` return dictionary of the Gymnasium `step()` function.

3. Motivation and software impact

Although several plasma simulators are available to study fusion reactors, many of them are not openly accessible and require restrictive licenses, such as RAPTOR [15] or JOREK [16]. Other simulators, such as EFIT [17], are widely used for equilibrium reconstruction but do not provide dynamic plasma analyses, which makes them unsuitable for closed-loop control studies. More broadly, all of these simulators have been primarily designed for plasma physicists, which makes them hard to use by RL researchers who do not have much related expertise. Additionally, none of them provides any interface for control-oriented applications. Although TORAX [13] is open-source and lightweight, it suffers from the same limitations regarding control applications.

Gym-TORAX addresses these limitations by building upon TORAX to obtain a closed-loop, RL-compatible, and open-source framework. By encapsulating the plasma physics behind a classical Gymnasium API, our package lowers the entry barrier for RL researchers interested in plasma control. In this way, it fosters collaborations between both communities, allowing each one to focus on their respective expertise.

Looking forward, Gym-TORAX enables the study of new plasma control strategies and, with future developments of the software, new plasma configurations. From an RL standpoint, it allows to discover or study algorithms that are well-suited for plasma control-related problems. Even though the hypotheses inherent to the TORAX plasma simulator limit its use to preliminary investigations, its simplicity and fast execution make it a suitable starting point for advanced studies.

To support these directions, Gym-TORAX already includes a fully implemented environment based on the ITER hybrid ramp-up scenario, described in Appendix. Although the reward function is simple, it is ready to be used for RL training, and results can already be compared to the baselines presented in Appendix A.3. We encourage users to test or improve this environment and to create new ones.

4. Future works and conclusion

In this paper, we introduced Gym-TORAX, an open-source framework to create Gymnasium-compatible reinforcement learning environments based on the TORAX plasma simulator. By abstracting the underlying plasma physics, Gym-TORAX enables reinforcement learning researchers to design and evaluate control strategies without requiring expert knowledge in fusion science. At the same time, it provides plasma physicists with a flexible way to implement and test control scenarios.

Future developments of Gym-TORAX will focus on expanding its flexibility and scope. A first step will be to provide tools to parameterize the plasma and tokamak geometry directly at environment creation, introducing a new dimension to the reinforcement learning problem. In addition, dedicated utilities will be added to handle specific physics-related events, such as the timing of the so-called *LH* transition (the transition between *L* and *H* mode, which are two distinct confinement regimes; see Appendix for more details), which plays a critical role in plasma dynamics. Additionally, any new features extending the capabilities of the TORAX simulator can enhance the capabilities of Gym-TORAX itself.

CRediT authorship contribution statement

Antoine Mouchamps: Conceptualization, Formal analysis, Methodology, Software, Validation, Visualization, Writing – original draft. **Arthur Malherbe:** Conceptualization, Formal analysis, Methodology, Software, Validation, Visualization, Writing – original draft. **Adrien Bolland:** Formal analysis, Methodology, Validation, Writing – review & editing. **Damien Ernst:** Funding acquisition, Project administration, Writing – review & editing, Methodology, Validation.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work the authors used ChatGPT in order to correct and improve the readability, grammar, and spelling of the manuscript. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Package illustration

To illustrate the capabilities of our package, in this appendix we compare the performance of three different policies in a custom environment based on an example configuration file provided in TORAX.

This appendix is organized into three parts. First, a description of the simulation environment is given in Appendix A.1. Then, the three policies that will be evaluated are presented in Appendix A.2. Finally, Appendix A.3 comments on the results.

A.1. Environment description

The environment used for this case study is the ITER hybrid ramp-up scenario, provided as an example configuration file in TORAX. This scenario is based on the work of Citrin et al. [18] and consists of a power ramp-up phase of 100 s, followed by a nominal phase lasting 50 s. The first phase of 100 s takes place in so-called *L-mode* (low-confinement regime), while the nominal phase occurs in *H-mode*

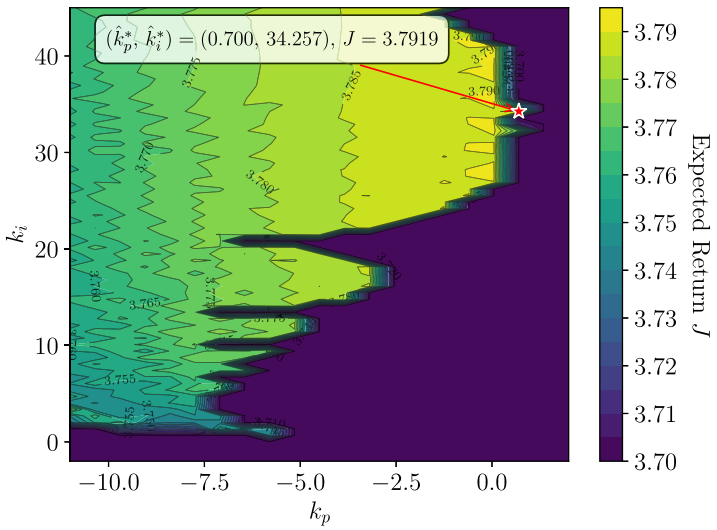

Fig. 2. Heatmap of the expected return J over a subset of the full parameter space. To improve the clarity of the figure, the scale used to represent J values is clipped at a minimum of 3.7.

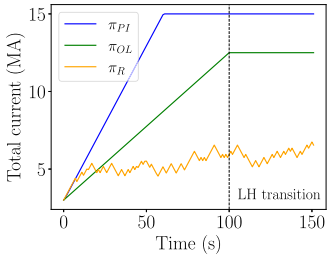

Fig. 3. Comparison of one action (total current) trajectory for each policy.

Table A.1
Expected return of the three studied policies, using a discount factor $\gamma = 1$.

Policy	Expected return
π_{OL}	3.40
π_R	−10.79
π_{PI}	3.79

(high-confinement regime). These are two distinct plasma confinement regimes with different physical properties.

The environment is named `IterHybridEnv`. The environment features three actions: `IpAction` for the total current, `NbiAction` for NBI (Neutral Beam Injection), and `EcrhAction` for ECRH (Electron-Cyclotron Resonance Heating), both representing external energy sources. Its action space $\mathcal{A}$ is bounded, and a ramp-rate limit is imposed on the total current. By default, the environment is fully-observable and uses the `AllObservation` class with custom bounds applied to certain variables.

The reward function is a linear combination of four elements:

$$r = \alpha_Q \cdot g_Q + \alpha_{q_{min}} \cdot g_{q_{min}} + \alpha_{q_{95}} \cdot g_{q_{95}} + \alpha_{\mathrm{H98}} \cdot g_{\mathrm{H98}} \quad . \tag{2}$$

In this equation, α_i and g_i, with $i \in \{Q, q_{min}, q_{95}, \mathrm{H98}\}$, represent weights and functions, respectively. These are related to the fusion gain Q, the minimum q_{min} and edge q_{95} safety factors, and the H-mode confinement quality factor H98, respectively.

A.2. Policies

We consider three different policies: an open-loop policy, a random policy, and one Proportional Integral (PI) controller-based policy. The open-loop policy π_{OL} uses a predetermined set of actions that directly follows the action trajectories of the initial scenario given in TORAX. This serves as the reference scenario for the two other policies. The random policy π_R selects the actions uniformly at random. The PI controller-based policy π_{PI} controls the total current action using a PI controller and uses the same predetermined trajectories as the open-loop policy for the last two actions, NBI and ECRH. The PI controller is used to follow a prescribed linear increase (from $0.6\,\mathrm{MA/m^2}$ to $2\,\mathrm{MA/m^2}$) of the current density at the center of the plasma during the ramp-up phase of 100 s. Once the ramp-up has been performed, action values are kept constant from the last action given by the PI controller until the end of the episode (for the last 49 s).

The proportional k_p and integral k_i gains of the PI controller are optimized to maximize the expected return $J(\pi)$. The optimization is carried out using a grid search. First, we define a subspace $\mathcal{K}_p \times \mathcal{K}_i \subset \mathbb{R}^2$ of the full parameter space. This subspace is determined so as to avoid regions of large reward penalties and regions where the expected return does not evolve anymore. The parameter space $\mathcal{K}_p \times \mathcal{K}_i$ is then discretized into $n \cdot m$ discrete points by using n and m values for k_p and k_i, respectively. An approximation of the optimal parameters can then be computed by selecting the point, among all, whose corresponding policy maximizes the expected return. Running this algorithm using $n = 20$ and $m = 60$ as hyperparameters, we obtain the expected return topology depicted in Fig. 2, from which we obtain the following estimations for the optimal parameters: $\hat{k}_p^* = 0.700$ and $\hat{k}_i^* = 34.257$.

A.3. Results

The expected return defined in Eq. (1) obtained for each policy is given in Table A.1.

The open-loop policy yields an average return of 3.40. As expected, the random policy performs worse, with an average return of −10.79. The best-performing policy is the PI controller-based one, with an average return of 3.79. This result is an improvement over the reference scenario and can serve as a baseline for more sophisticated policies.

A representation of an action (total current) trajectory for each policy is given in Fig. 3. This figure shows the erratic evolution of the total current of the random policy, which is somewhat mitigated by the ramp-rate constraints imposed in the environment. Regarding the PI policy, the trajectory of the current increases steadily and levels off at 15 MA, the maximum value allowable in the environment. This behavior is consistent with the fact that higher values of total current

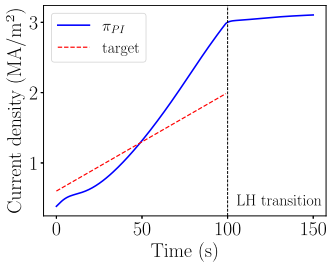

Fig. 4. Evolution of the current density with respect to the target.

can generally be associated with improved confinement and overall better performance [19].

Fig. 4 represents the target evolution of the PI controller and the action taken by the PI controller-based policy. Note that the parameters were optimized to maximize the expected return, rather than having actions close to the target, which can be observed in the figure.

References

[1] L. Artsimovich, Tokamak devices, Nucl. Fusion 12 (1972) 215, http://dx.doi.org/10.1088/0029-5515/12/2/012.

[2] M. Walker, D. Humphreys, D. Mazon, D. Moreau, M. Okabayashi, T. Osborne, E. Schuster, Emerging applications in tokamak plasma control, IEEE Control Syst. Mag. 26 (2006) 35–63, http://dx.doi.org/10.1109/MCS.2006.1615272.

[3] J. Kober, J.A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey, Int. J. Robot. Res. 32 (2013) 1238–1274, http://dx.doi.org/10.1177/0278364913495721.

[4] A. Louette, G. Lambrechts, D. Ernst, E. Pirard, G. Dislaire, Reinforcement learning to improve delta robot throws for sorting scrap metal, 2024, URL: https://arxiv.org/abs/2406.13453, http://arxiv.org/abs/2406.13453arXiv:2406.13453.

[5] I. Boukas, D. Ernst, T. Théate, A. Bolland, A. Huynen, M. Buchwald, C. Wynants, B. Cornélusse, A deep reinforcement learning framework for continuous intraday market bidding, Mach. Learn. 110 (2021) 2335–2387, http://dx.doi.org/10.1007/s10994-021-06020-8.

[6] S. Aittahar, A. Bolland, G. Derval, D. Ernst, Optimal control of renewable energy communities subject to network peak fees with model predictive control and reinforcement learning algorithms, 2024, URL: https://arXiv.org/abs/2401.16321, http://arxiv.org/abs/2401.16321arXiv:2401.16321.

[7] R. Henry, D. Ernst, Gym-ANM: Open-source software to leverage reinforcement learning for power system management in research and education, Softw. Impacts 9 (2021) 100092, http://dx.doi.org/10.1016/j.simpa.2021.100092.

[8] R. Henry, D. Ernst, Gym-ANM: Reinforcement learning environments for active network management tasks in electricity distribution systems, Energy AI 5 (2021) 100092, http://dx.doi.org/10.1016/j.egyai.2021.100092.

[9] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de las Casas, C. Donner, L. Fritz, C. Galperti, A. Huber, J. Keeling, M. Tsimpoukelli, J. Kay, A. Merle, J.-M. Moret, S. Noury, F. Pesamosca, D. Pfau, O. Sauter, C. Sommariva, S. Coda, B. Duval, A. Fasoli, P. Kohli, K. Kavukcuoglu, D. Hassabis, M. Riedmiller, Magnetic control of tokamak plasmas through deep reinforcement learning, Nature 602 (2022) 414–419, http://dx.doi.org/10.1038/s41586-021-04301-9.

[10] I. Char, J. Abbate, L. Bardoczi, M. Boyer, Y. Chung, R. Conlin, K. Erickson, V. Mehta, N. Richner, E. Kolemen, J. Schneider, Offline model-based reinforcement learning for tokamak control, in: N. Matni, M. Morari, G.J. Pappas (Eds.), Proceedings of the 5th Annual Learning for Dynamics and Control Conference, in: Proceedings of Machine Learning Research, vol. 211, PMLR, 2023, pp. 1357–1372, URL: https://proceedings.mlr.press/v211/char23a.html.

[11] J. Seo, S. Kim, A. Jalalvand, R. Conlin, A. Rothstein, J. Abbate, K. Erickson, J. Wai, R. Shousha, E. Kolemen, Avoiding fusion plasma tearing instability with deep reinforcement learning, Nature 626 (2024) 746–751, http://dx.doi.org/10.1038/s41586-024-07024-9.

[12] M. Towers, A. Kwiatkowski, J. Terry, J.U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. K.G., R. Perez-Vicente, A. Pierré, S. Schulhoff, J.J. Tai, H. Tan, O.G. Younis, Gymnasium: a standard interface for reinforcement learning environments, 2024, http://dx.doi.org/10.48550/arXiv.2407.17032.

[13] J. Citrin, I. Goodfellow, A. Raju, J. Chen, J. Degrave, C. Donner, F. Felici, P. Hamel, A. Huber, D. Nikulin, D. Pfau, B. Tracey, M. Riedmiller, P. Kohli, TORAX: a fast and differentiable tokamak transport simulator in JAX, 2024, http://dx.doi.org/10.48550/arXiv.2406.06718, arXiv preprint arXiv:2406.06718.

[14] J. Bradbury, R. Frostig, P. Hawkins, M.J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of Python+NumPy programs, 2018, URL: http://github.com/jax-ml/jax.

[15] F. Felici, Real-Time Control of Tokamak Plasmas: from Control of Physics to Physics-Based Control, (Ph.D. thesis), École Polytechnique Fédérale de Lausanne, 2011, http://dx.doi.org/10.5075/epfl-thesis-5203.

[16] M. Hoelzl, G. Huijsmans, S. Pamela, M. Bécoulet, E. Nardon, F. Artola, B. Nkonga, C. Atanasiu, V. Bandaru, A. Bhole, D. Bonfiglio, A. Cathey, O. Czarny, A. Dvornova, T. Fehér, A. Fil, E. Franck, S. Futatani, M. Gruca, H. Guillard, J. Haverkort, I. Holod, D. Hu, S. Kim, S. Korving, L. Kos, I. Krebs, L. Kripner, G. Latu, F. Liu, P. Merkel, D. Meshcheriakov, V. Mitterauer, S. Mochalskyy, J. Morales, R. Nies, N. Nikulsin, F. Orain, J. Pratt, R. Ramasamy, P. Ramet, C. Reux, K. Särkimäki, N. Schwarz, P. Singh Verma, S. Smith, C. Sommariva, E. Strumberger, D. van Vugt, M. Verbeek, E. Westerhof, F. Wieschollek, J. Zielinski, The JOREK non-linear extended MHD code and applications to large-scale instabilities and their control in magnetically confined fusion plasmas, Nucl. Fusion 61 (2021) 065001, http://dx.doi.org/10.1088/1741-4326/abf99f.

[17] L. Lao, H. St. John, R. Stambaugh, A. Kellman, W. Pfeiffer, Reconstruction of current profile parameters and plasma shapes in tokamaks, Nucl. Fusion 25 (1985) 1611, http://dx.doi.org/10.1088/0029-5515/25/11/007.

[18] J. Citrin, J. Artaud, J. Garcia, G. Hogeweij, F. Imbeaux, Impact of heating and current drive mix on the ITER hybrid scenario, Nucl. Fusion 50 (2010) 115007, http://dx.doi.org/10.1088/0029-5515/50/11/115007.

[19] ITER Physics Expert Group on Confin Transport, ITER Physics Expert Group on Confin Database, Editors, Chapter 2: Plasma confinement and transport, Nucl. Fusion 39 (1999) 2175–2249, http://dx.doi.org/10.1088/0029-5515/39/12/302.