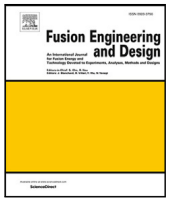
Fusion Engineering and Design 196 (2023) 114008

Contents lists available at ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes



Enhancing deep reinforcement learning with integral action to control tokamak safety factor

Andrea Mattioni [a,*], Samuele Zoboli [b], Bojan Mavkov [c], Daniele Astolfi [b], Vincent Andrieu [b], Emmanuel Witrant [a], Paolo Frasca [d], Christophe Prieur [a]

[a] Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-Lab, F-38000 Grenoble, France
[b] Univ. Lyon, Univ. Claude Bernard Lyon 1, CNRS, LAGEPP UMR 5007, 43 boulevard du 11 novembre 1918, F-69100, Villeurbanne, France
[c] Univ. Côte d'Azur, CNRS, I3S, Nice, France
[d] Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, GIPSA-Lab, F-38000 Grenoble, France

ARTICLE INFO

Keywords:
Deep Reinforcement Learning
Integral control
Safety factor
Magnetic profiles

ABSTRACT

Recent advances in the use of Artificial Intelligence to control complex systems make it suitable for profile plasma control. In this work, we propose an algorithm based on Deep Reinforcement Learning to control the safety factor profile with a feedback design. For this purpose, we first derive a device-specific control-oriented model with fast simulation time. Then, in order to enhance robustness with respect to external disturbances and model errors, we include an error time integrator into the controller. A cascade of the kinetic and magnetic models with the error time integrator is used in the learning procedure of the feedback controller. Finally, to illustrate the efficiency of the proposed design procedure, the obtained controller is tested in a reference plasma simulator, the Raptor simulator.

1. Introduction

Because of the high uncertainties in the measurements and estimations of plasma profiles, as well as in the modelling of kinetic and magnetic dynamics, robust feedback control is crucial to obtain high-performance operations of tokamak reactors. In tokamak reactors, the *safety factor* has been found to be strictly related to Magnetohydrodynamic (MHD) activities [1]: therefore, controlling the safety factor to the desired profile becomes an essential step towards obtaining long-time discharges [2]. In this article, we will consider the safety factor profile control problem during the so-called flat-top phase.

In the plasma control literature, it is common to interchangeably speak about the current profile, safety factor q (and its inverse ι-profile), magnetic flux gradient profile, and magnetic flux profile. From an operative point of view, a lot of contributions have been made by the plasma physics and control communities working together. For instance, an overview of the plasma control in the Tore Supra tokamak can be found in [3]. More specifically, in [4] are shown experimental results using proportional feedback for the control of the internal inductance on Tore Supra, while in [5,6] the authors show the results on the DIII-D and JET tokamaks obtained by using optimal control applied on data-driven models. Subsequently, different strategies using first-principles-driven models have been developed from the control community.

The main challenge in model-based safety factor profile control for advanced mode operations is the derivation of dynamical models that are complex enough to retain the main physical properties and simple enough to be used for feedback design. A first attempt to model and numerically simulate the plasma profile evolution has been made in [7]. Then, a control-oriented model describing the magnetic flux gradient and temperature evolution has been proposed in [8]. Control strategies have been developed using a linear finite-dimensional approximation of the original PDE describing the magnetic flux dynamics [9–11]. An optimal controller designed on a nonlinear model obtained by Galerkin approximation has been proposed in [12], while a backstepping controller has been designed on the nonlinear model obtained by finite differences in [13]. Model predictive control strategies for the safety factor profile control have been presented in [14]. Nonlinear robust safety factor profile control is developed in [15]. Control algorithms for simultaneous control of magnetic and kinetic parameters of tokamak plasmas using finite-dimensional approximation are presented in [16–19].

Recently, much effort has been spent in designing controllers directly on the PDE model of magnetic and temperature diffusion. In [20], a sum-of-square polynomial technique has been used to construct a Lyapunov function to stabilize the closed-loop system, whereas in [21]

* Corresponding author.
E-mail address: andreamattioni2@gmail.com (A. Mattioni).

https://doi.org/10.1016/j.fusengdes.2023.114008
Received 15 April 2023; Received in revised form 4 September 2023; Accepted 15 September 2023
Available online 23 September 2023

the same technique has been used to optimize the bootstrap current. Furthermore, a Lyapunov-based controller has been designed in [22,23]. A Lyapunov-based control technique for the kinetic and magnetic profiles designed on the linearization of the original equations has been proposed in [24].

The seminal work [25] provided a powerful control-oriented fast plasma transport simulator (Raptor) that helps the feedback design study and implementation for kinetic and magnetic profiles control for the TCV tokamak. This enabled the design, simulation and implementation of multiple controllers [26,27]. For a broader overview of emerging and current challenges in tokamak control, we refer the reader to [28,29].

With the rise of Deep Neural Networks (DNNs) as tools for function approximation, the machine learning community took a step towards control problems of physical systems. Recently, Deep Reinforcement Learning (DRL) methods proved to be effective in solving complex nonlinear control problems [30,31]. Guided by an optimization objective, DRL algorithms train a DNN to produce a sequence of almost-optimal inputs (or actions). This sequence of inputs is called *policy*. These algorithms are data-driven, as training evolves according to the interactions with the environment. One major advantage of DRL algorithms is their direct applicability to a large family of complex systems, especially in the case of model-free approaches, e.g. [32–34]. The environment to be controlled is typically considered a black box. In order to estimate the future performances of the policy without knowing the environment, many DRL algorithms exploit an actor-critic structure. This family of methods exploits two or more DNNs (see e.g. [35]). The former is used for approximating the policy, while the latter predicts its performance by estimating the sum of future rewards. This model-agnostic approach enabled DRL algorithms to be applied on a wide variety of complex tasks and, most recently, also in the field of nuclear fusion [36].

In this paper, we propose to design a dynamic DNN controller complemented by a time integral of the error. Such an addition is valuable because, according to systems and control theory, the addition of an integrator in the feedback loop is known to solve the problems of constant reference regulation and constant disturbance rejection [37]. This control strategy has indeed been shown to be effective for the control of some classes of linear and nonlinear Ordinary Differential Equations (ODE) [38], as well as some classes of linear and nonlinear PDEs [39]. In [40], the authors presented a method to solve the regulation problem of linear systems in closed-loop with a neural network controller and an integrator. More recently, control design by RL has been combined with Model Predictive Control to solve the tracking problem of surface vessels [41].

However, few works have been so far dedicated to controlling PDEs using DRL. Because of the spatial differential operators present in PDEs, the state inherits some spatial regularity properties, such as local smoothness. This intrinsic property can be exploited to design specific RL algorithms that are able to deal with very large state spaces [42], e.g. the regularized fitted q-iteration (RFQI) algorithm. Furthermore, this method has been successfully applied to the control design of a multidimensional nonlinear problem such as a heating, ventilating and air conditioning system [42,43], but only with a finite amount of possible actions. The case in which the action space is infinite-dimensional has been investigated in [44]. More recently, a Proximal Policy Optimization (PPO) algorithm has been used to design the controller for congested freeway traffic [45]. DRL algorithms have already been used in the context of nuclear fusion control. In [46], the authors proposed a DRL technique to control the safety factor during the ramp-up phase, while in [47] the authors developed a DRL algorithm to train a feed-forward controller for the kinetic profiles. Recently, a DRL controller has been proposed in [36] for plasma shape control. A recent publication that aligns with our proposed work is [48], where the authors present an RL-based algorithm for simultaneous control of the safety factor and normalized beta in the JT-60SA tokamak. In contrast to their approach, our contribution focuses on emphasizing the utilization of the time integral of the error as a parameter to be fed into the trained neural network. Furthermore, after presenting our control architecture and a novel training procedure, we claim that our proposed control strategy compensates for the disparities between the training model and real-world conditions.

Table 1
Table of symbols.

Symbol	Description
s_j	Environment state at j
a_j	Action applied to the environment at j
V^π	Value function
π	Policy
Q^π	State–action value function
A^π	Advantage function
r_j	Reward at j
R_j	Accumulated reward at j

The contribution of this paper hinges on an original model of the kinetic and magnetic plasma dynamics: the resulting simulator is fast enough to be used for learning purposes by a DRL algorithm. In order to make the latter robust to constant model uncertainties and disturbances, we embed the controller with an error time integrator. To illustrate the robustness of the proposed controller, we test it on a different model than the one used for training, i.e. the Raptor simulator. Several authors proposed the inclusion of an error time integrator in the control loop, which is fundamental in practical operations to compensate for discrepancies between the model employed for control design and the actual plant. Integral action-based strategies appeared for instance in [49], with a modification of an LQR (referred to as LQI), and in [50], where a Lyapunov control strategy is designed based on the linearized partial differential equation (PDE). The main limitation of these works is that they require the knowledge of a linearized model around the desired equilibrium point. Furthermore, such approaches typically limit the applicability of the resulting controller to a small region of initial conditions around the equilibrium. In contrast, our work is based on a feedback design strategy that uses neural networks (NNs) trained on an arbitrarily large region of the state space, thereby yielding a controller applicable from any plant's initialization. Furthermore, we leverage on model-free reinforcement learning algorithms, thereby circumventing the necessity for explicit knowledge of the system's dynamics in proximity to the desired operational point.

The paper is organized as follows: In Section 2 we propose some preliminaries on Reinforcement Learning control and its use in combination with integral action. In Section 3 the plasma model is written as a Markov decision process and the training algorithm is presented. The simulation results of the obtained controller on the Raptor simulator are shown in Section 4. Finally, some concluding remarks and comments on future works are given in Section 5.

In the appendix section, additional context and clarification are provided for the presented material. Specifically, Appendix A introduces the model of the tokamak's kinetic and magnetic dynamics, Appendix B outlines the simulation algorithm, and Appendix C offers a simple example of implementing reinforcement learning control with integral action.

2. Background on Reinforcement Learning control

In this section, we briefly introduce the essential Reinforcement Learning concepts. A thorough discussion can be found in [51]. Justified by Bellman's principle of optimality, Reinforcement Learning aims at optimally solving a problem by learning a sequence of maximum-reward actions (called *policy*). The policy optimization is driven by *value functions*. The *state-value function* $V^\pi(s_j)$ corresponds to the expected total discounted reward R_j starting from state s_j at the j time instance and then following the policy. We remark that the value function profoundly depends on the policy π. If the agent uses a given

policy π to select an action from the state s_j, the value function is given by

$$V^{\pi}(s_t) = \mathbb{E}\left[R_j \mid s_j\right] \tag{1}$$

where $\mathbb{E}[\cdot]$ stands for the expectation. The optimal policy is the policy that corresponds to the maximum value $V^{\star}(s_j)$ of the value function

$$\pi^{\star} = \arg\max_{\pi} V^{\pi}(s_j). \tag{2}$$

Dynamic Programming methods search for the optimal policy using the former equation, but they require knowledge of the model. To enable the concept of model-free Reinforcement Learning, it is necessary to introduce the *state–action value function* or *Q-function*. The Q-function corresponds to the expected total discounted reward when the action a_j is taken in state s_j, and then the policy π is followed henceforth. Therefore, the Q-function is given by

$$Q^{\pi}(s_j, a_j) = \mathbb{E}\left[R_j \mid s_j, a_j\right]. \tag{3}$$

The optimal Q-function is given by

$$Q^{\star}(s_j, a_j) = \max_{\pi} Q^{\pi}(s_j, a_j) \tag{4}$$

and stands for the expected total discounted reward when the agent picks possible non-optimal action a_j in s_j, and then behaves optimally henceforth. The relation between the optimal Q and V function is expressed by

$$V^{\star}(s_t) = \max_{a_j \in \mathcal{A}} Q^{\star}(s_j, a_j). \tag{5}$$

If the optimal Q-function is known, then the optimal action $a_j^{\star}$ can be extracted by choosing the action a_j that maximizes $Q^{\star}(s_j, a_j)$

$$a_j^{\star} = \arg\max_{a_j \in \mathcal{A}} Q^{\star}(s_j, a_j). \tag{6}$$

This is the reason why the knowledge of the Q function enables model-free RL. Finally, the advantage function measures how advantageous a certain action a_j is with respect to the one drawn from the policy

$$A^{\pi}(s_j, a_j) = Q^{\pi}(s_j, a_j) - V^{\pi}(s_j). \tag{7}$$

A summary of the RL notation is given in Table 1.

In RL, as well as in dynamic programming, the action is chosen through a policy that has the objective of maximizing the expected total discounted reward. In the present application, we have a system model with continuous state and action spaces, which inherently comprise an infinite number of elements. Due to this infinite nature, it becomes infeasible to fully explore the whole state and action spaces [51]. Consequently, tabular dynamic programming methods cannot be employed to apply a reinforcement learning algorithm to our fusion control problem. Instead, the typical solution to tackle the continuous time nature of state and action spaces is to employ deep function approximations and estimate the value function and the policy. In our application, we chose to use the PPO algorithm, which is based on Trust Region Policy Optimization (TRPO). These algorithms use an actor-critic approach, where the *actor* is in charge of improving the policy based on the value function that is estimated by the *critic*. The actor and the critic correspond to function approximators parametrized by ϕ and θ, respectively. In particular, in our application case, these function approximators are selected to be DNNs and ϕ, θ are vectors collecting weights and biases.

The *Critic*'s role is to evaluate the current policy prescribed by the actor. At each iteration of an episode p, the tuple $(s_{p,j}, a_{p,j}, r_{p,j+1}, s_{p,j+1})$ is stored in a buffer $\mathcal{B}_p$. Each episode is of length J and each episode-related buffer $\mathcal{B}_p$ is stored in a general buffer $\mathcal{B}$. After a certain number of episodes, the parametrized value function V_{ϕ} is updated to minimize the following loss function

$$\mathcal{L}_V = \hat{\mathbb{E}}[V_{\phi}(s_j) - R_j] \tag{8}$$

where $\hat{\mathbb{E}}$ represents the estimated expectation and can be implemented as the mean

$$\mathcal{L}_V = \frac{1}{N_{sample}} \sum_{p \in \mathcal{B}} \sum_{j=0}^{J} \left(V_{\phi}(s_{p,j}) - R_{p,j}\right) \tag{9}$$

where $R_{p,j}$ is the total discounted reward at iteration j of the pth episode and N_{sample} is the total number of samples. The critic parameters ϕ are updated numerically via gradient descent.

The *Actor*'s role is to use the information from the *Critic* to update the current policy. To understand the PPO algorithm we first need to understand the optimization objective of Policy Gradient (PG) methods, defined as follows

$$\mathcal{L}_{PG}(\theta) = \hat{\mathbb{E}}[\log \pi_{\theta}(a_j \mid s_j) \hat{A}(a_j, s_j)] \tag{10}$$

that can be implemented as

$$\mathcal{L}_{PG}(\theta) = \frac{1}{N_{sample}} \sum_{p \in \mathcal{B}} \sum_{j=0}^{J} (\log \pi_{\theta}(a_{p,j} \mid s_{p,j}) \hat{A}(a_{p,j}, s_{p,j})) \tag{11}$$

where $\pi_{\theta}(a_{p,j} \mid s_{p,j})$ is computed using the current neural network and $\hat{A}(a_j, s_j)$ using the Generalized Advantage Estimator (GEA) using the critic value estimator. It can be proven that (10) drives the policy in a gradient ascent fashion, with respect to the objective function [52]. The policy π_{θ} is the current neural network that gives the probability of picking a_j when the environment gives a certain state observation s_j. If for a certain couple (a_j, s_j) the advantage is positive, the policy gradient updates θ to augment the probability of taking the action a_j when the environment is in s_j. Unfortunately, with the previously defined loss function, the parameter θ will often be updated far from the previous policy. To solve this problem, one can use the Trust Region Policy Optimization (TRPO) [53]. The objective of TRPO is to maximize the loss function

$$\mathcal{L}_{TRPO}(\theta) = \hat{\mathbb{E}}\left[\frac{\pi_{\theta}(a_j \mid s_j)}{\pi_{\theta_{old}}(a_j \mid s_j)} \hat{A}(a_j, s_j)\right] \tag{12}$$

subject to the constraint

$$\hat{\mathbb{E}}[KL[\pi_{\theta_{old}}(\cdot \mid s_j), \pi_{\theta}(\cdot \mid s_j)]] \leq \delta \tag{13}$$

where θ_{old} corresponds to the actor parameters before the last update and the KL is the Kullback–Leibler function measuring the difference between the old and current policy. The constraint assures that the new policy does not deviate from the old policy by any more than δ. In this work, we adopt the PPO reinforcement learning algorithm [54], which is based on TRPO. Indeed, while TRPO computes the trust region using second-order information, PPO approximates it via clipping. For PPO the loss function to be maximized is defined as

$$\mathcal{L}_{CLIP}(\theta) = \hat{\mathbb{E}}\left[\min(r_j(\theta)\hat{A}(a_j, s_j), \text{clip}(r_j(\theta), 1-\epsilon, 1+\epsilon)\hat{A}(a_j, s_j))\right] \tag{14}$$

where

$$r_j(\theta) = \frac{\pi_{\theta}(a_j \mid s_j)}{\pi_{\theta_{old}}(a_j \mid s_j)}. \tag{15}$$

The idea is to use probability clipping, which removes the incentives for moving r_j outside of the interval $[1-\epsilon, 1+\epsilon]$. The minimum of the clipped and unclipped objective ensures the final objective is a lower bound (i.e., a pessimistic bound) on the unclipped objective. With this scheme, we only ignore the change in probability ratio when it would improve the objective, and we include it when it deteriorates the objective.

3. Control design

In this section, we propose the design of a dynamic controller enhanced with knowledge about the integral of the error. Inspired by classical control-theoretic solutions (PIs and PIDs) and recent results on total stability [55,56], we embed the stabilization of a discrete-time

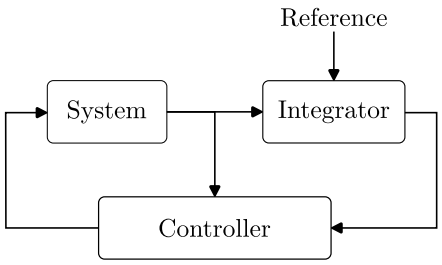


Fig. 1. General overview of a control scheme including a time integrator. In the context of this work, the controller is a DNN trained using a DRL algorithm. For more information regarding the control structure, we refer to Figs. 4 and 6.

integrator in the control objective. As such, we consider the problem of stabilizing the extended cascade system composed of the plant and the integrator, as in Fig. 1. The idea is to embed "memory" in the agent by providing some time-related information. Then, if the integrator's state is stabilized, we guarantee a zero-tracking error with respect to the (constant) reference. Note that such a property is due to the specific structure of the integrator dynamics. Actually, an alternative for embedding memory into the agent is to use Recursive Neural Networks (RNNs), e.g. [57,58]. Yet, aside from the increased complexity in their training due to back-propagation through time, it would not be possible to guarantee perfect asymptotic tracking and (constant) disturbance rejection. Indeed, it is not possible to predict the dynamics information that will be stored in the RNN's latent space and its evolution. As stated above, the proposition of adding an integral state to the controller stems from regulation theory. To explain the need for an integral state, a toy example is considered in Appendix C.

3.1. Integral state for magnetic flux control

Consider $\psi(R,Z)$ the poloidal flux of the magnetic field $B(R,Z)$ passing through a disc centred at the toroidal axis at height Z and with surface $S=\pi R^2$ where R is the large plasma radius. Let the magnetic flux be defined as

$$\psi(R,Z)=\frac{1}{2\pi}\int_S B(R,Z)dS \tag{16}$$

and its dynamics can be expressed by the following reaction–diffusion equation [8]

$$\frac{\partial\psi}{\partial t}(x,t)=\frac{D(x,t)}{a_\rho^2}\frac{\partial^2\psi}{\partial x^2}(x,t)+\frac{G(x,t)}{a_\rho}\frac{\partial\psi}{\partial x}(x,t)+S(x,t). \tag{17}$$

where $x=\rho/a_\rho$ identifies the normalized spatial variable, $D(x,t)$ and $G(x,t)$ are diffusion parameters, while $S(x,t)$ is the source term and are defined in (A.2). In this study, we consider that the magnetic flux is controlled by two Electron Cyclotron Current Drive (ECCD) systems, each characterized by its input power $P_{eccd,i}$, where $i\in 1,2$. Our case of study resembles the scenario presented in [50], where the two inputs are applied to the same spatial point. Specifically, the first antenna $P_{eccd,1}$ has a positive effect on z_j, while the second antenna $P_{eccd,2}$ has a negative effect on it. Thus, from a theoretical point of view, the two inputs can be treated as a single input. The term ρ is the toroidal flux coefficient indexing the magnetic surfaces, defined as $\rho=(2\phi/B_{\phi 0})^{\frac{1}{2}}$, where ϕ is the toroidal magnetic flux and $B_{\phi 0}$ is the value of the toroidal magnetic flux at the plasma center. The spatial index belongs to the interval $\rho\in[0,a_\rho]$ where a_ρ is the minor plasma radius corresponding to the Last Closed Flux Surface (LCFS). An important quantity of plasma control in tokamak devices is the *safety factor*. This distributed variable measures the toroidal over poloidal turns of a field line passing through a point (R,Z) in a toroidal plane. Since the magnetic field lines are assumed to be equal in the same magnetic surface, we can define the safety factor for each magnetic surface indexed by x. In particular, the safety factor is defined as the quotient between the toroidal and poloidal gradient, that using the previous definition of the toroidal magnetic flux, can be defined as

$$q(x,t)=\frac{d\phi}{d\psi}=\frac{\partial\phi/\partial x}{\partial\psi/\partial x}=-\frac{B_{\phi 0}a_\rho^2 x}{\partial\psi/\partial x} \tag{18}$$

where $\phi(x,t)$ is the toroidal flux defined in (A.11). Another important quantity in plasma analysis is the ι-profile, which is also referred to as "rotational transform"

$$\iota(x,t)=\frac{1}{q(x,t)}=\frac{\partial\psi/\partial x}{B_{\phi 0}a_\rho^2 x}. \tag{19}$$

The ι-profile is a more natural control variable since it proportionally depends on the poloidal flux gradient. Additional details about the model are discussed in Appendix A. The following equation gives the plasma thermal energy dynamics

$$\begin{cases}\tau_{th}=e^{-5.7466}P_{oh}^{0.0214}(1+P_{eccd,1})^{0.0426}(1+P_{eccd,2})^{0.0012}\\ \frac{d}{dt}W_{th}=-\frac{1}{\tau_{th}}W_{th}+P_{tot},\quad W_{th}(0)=P_{tot}(0)\tau_{th}(0)\end{cases} \tag{20}$$

where,

$$P_{tot}=\sum_{i=1}^{N_{eccd}}P_{eccd,i}+P_{OH} \tag{21}$$

while P_{OH} identifies the ohmic power and is defined in (A.12). Although temperature regulation is not the primary objective of this work, we still consider thermal dynamics in the model. The reason for including these dynamics is to account for the delay introduced by temperature diffusion. Rather than using a pure delay term, we chose to represent the temperature diffusion by an appropriate dynamic equation for more accurate modelling. At the same time, we made the decision not to incorporate the full distributed thermal diffusion model in order to minimize simulation time and achieve a sufficiently short training time for the RL algorithm. For more details regarding the temperature reconstruction using thermal energy and Ohmic power, we refer to Appendix B.

By utilizing an implicit–explicit time discretization and a fixed-step spatial discretization for Eq. (17), and an implicit–explicit time discretization for Eq. (20), we obtain the difference equation

$$z_{j+1}=f(z_j,a_j) \tag{22}$$

in the state variable $z_j=[\psi_j\ W_{th,j}]\in\mathbb{R}^{N+1}$ and input $a_j\in[0,1]=\mathcal{A}$. The discrete vector field f is defined in (B.5). Here, N is the number of elements used in the spatial discretization of Eq. (17). Hence, at iteration j, the vector $\psi_j\in\mathbb{R}^N$ denotes the discretized magnetic flux in N spatial points, while $W_{th,j}$ represents the thermal energy. The relation between a_j and $P_{eccd,i}$ and further information on the simulation algorithm employed in this study can be found in Appendix B.

In this work, the objective is to regulate the ι-profile $\iota(x,t)$ defined in (19) to a desired $\iota^\star(x)$. As ι depends on the magnetic flux gradient $\frac{\partial\psi}{\partial x}(x,t)$, the control objective can be reformulated as the regulation to a desired magnetic flux gradient profile $\frac{\partial\psi^\star}{\partial x}(x)$. Therefore, we define the discretized version of the magnetic flux gradient

$$\frac{\partial\psi_j}{\partial x}=\begin{bmatrix}\frac{\psi_{j,2}-\psi_{j,1}}{\delta x_1}\\ \frac{\psi_{j,3}-\psi_{j,1}}{\delta x_1+\delta x_2}\\ \vdots\\ \frac{\psi_{j,N}-\psi_{j,N-1}}{\delta x_N}\end{bmatrix}=\overbrace{\begin{bmatrix}-\frac{1}{\delta x_1} & \frac{1}{\delta x_1} & 0 & \cdots & 0 & 0\\ -\frac{1}{\delta x_1+\delta x_2} & 0 & \frac{1}{\delta x_1+\delta x_2} & \cdots & 0 & 0\\ \vdots & \vdots & \vdots & & \vdots & \vdots\\ 0 & 0 & 0 & \cdots & -\frac{1}{\delta x_N} & \frac{1}{\delta x_N}\end{bmatrix}}^{C}\psi_j \tag{23}$$

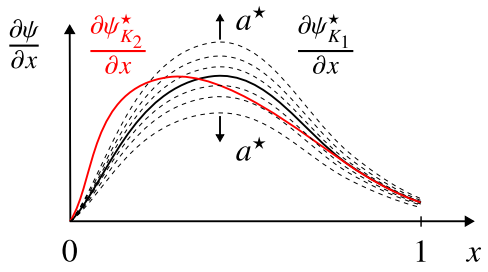


Fig. 2. Examples of different poloidal magnetic flux equilibriums of the reaction–diffusion equation describing the magnetic flux dynamics. In black we show that with a certain set of parameters K_1, changing the applied constant input (ECCD power), the poloidal magnetic flux equilibrium changes. The red profile represent a possible poloidal magnetic flux equilibrium with a different set of parameters K_2.

where $\frac{\partial\psi_j}{\partial x} \in \mathbb{R}^N$ and $C \in \mathbb{R}^{N\times N}$. Then, we select one element of the magnetic flux gradient to define the output $y_s = S\frac{\partial\psi}{\partial x}$, where $S \in \mathbb{R}^{1\times N}$ is the selection matrix with a single element equal to one and zero elsewhere. From now on, we assume that the system's parameters are known to belong to a certain range of values. This means that we do not know the exact parameter's value, but we only have an estimation. Consider k_n a generic system's parameter. According to the previous assumption, we know that $k_n \in [\underline{k}_n, \bar{k}_n]$. We define the vector K as the collection of all the system's parameters $K = [k_1\ k_2\ \dots\ k_{N_k}]^T \in \mathcal{K} = [\underline{k}_1, \bar{k}_1]\times[\underline{k}_2, \bar{k}_2]\times\cdots\times[\underline{k}_{N_k}, \bar{k}_{N_k}] \subset \mathbb{R}^{N_k}$, where N_k is the total number of parameters. We denote by $f_K(z_j, a_j)$ the magnetic flux dynamics with parameters K

$$\begin{cases} z_{j+1} = f_K(z_j, a_j) \\ y_j = \begin{bmatrix} \frac{\partial\psi_j}{\partial x} \\ y_{s,j} \end{bmatrix} = \begin{bmatrix} C\psi_j \\ SC\psi_j \end{bmatrix}. \end{cases} \tag{24}$$

By simulating this system with constant input $a^\star$, it is possible to extract the magnetic flux gradient at witch the system stabilizes $\frac{\partial\psi}{\partial x}^\star$. In particular, the equilibrium magnetic flux gradient is related to the equation's set of parameters K and the applied constant input $a^\star$. It is worth to remark that equilibrium magnetic flux's gradients obtained with the same input $a^\star$ and different set of parameters K_1 and K_2 may be different, as shown in Fig. 2. In real applications, an integral action is vital since we have to compensate for the uncertainties coming from the modelling procedure. At the same time, the integral action allows compensation for the mismatch between Raptor and the training model.

Control problem. *Stabilize the magnetic flux gradient $\frac{\partial\psi}{\partial x}$ as close as possible to the (potentially unreachable) desired magnetic flux gradient $\frac{\partial\psi}{\partial x}^\star$, and at the same time make sure that the selected output $y_s = S\frac{\partial\psi}{\partial x}$ converges to the desired value $y_s^\star = S\frac{\partial\psi}{\partial x}^\star$.*

The use of the integral action allows the regulation of the integrated quantity to zero. Having at our disposal one input (the two antennas act in opposite directions on the magnetic flux dynamics, therefore they can be treated as a single input), in case of unreachable equilibrium, using the integral action we can regulate to zero the error $y_{s,j} - y_s^\star$. We define the discrete-time integrator state dynamics as

$$\varepsilon_{j+1} = \varepsilon_j + (y_{s,j} - y_s^\star)\delta t. \tag{25}$$

We define the extended state as

$$s_j = \begin{bmatrix} z_j \\ \varepsilon_j \end{bmatrix} \in \mathcal{S} = \mathbb{R}^{N+2} \tag{26}$$

and therefore the extended dynamics can be defined as

$$s_{j+1} = \begin{bmatrix} f_K(z_j, a_j) \\ \varepsilon_j + (y_{s,j} - y_s^\star)\delta t \end{bmatrix} = g_K(s_j, a_j). \tag{27}$$

Since we do not know a priori the equilibrium of the integral state (since it depends on the unknown parameters K), we fix the integral state reference to zero $\varepsilon^\star = 0$.

3.2. Magnetic flux and temperature dynamics plus integral state as a Markov Decision Process

In this section, we express the tokamak magnetic flux and temperature dynamical model together with the integral state in Markov Decision Process settings. At each time step j, the environment conditions are described by the state vector $s_j \in \mathcal{S}$, while the action corresponding to the power of the ECCD antennas can be picked from the action space $a_j \in \mathcal{A}$. An action a_j is applied to the environment in state s_j at a time j, which evolves to the state s_{j+1} according to the state transition probability $\mathcal{P}(s_{j+1} \mid s_j, a_j)$. In other words, $\mathcal{P}(s_{j+1} \mid s_j, a_j)$ represents the probability of ending in state s_{j+1} when we apply an action a_j in state s_j. In our case, since we have the model of the system, the state transition probability boils down to deterministic dynamics

$$\mathcal{P}(s_{j+1} \mid s_j, a_j) = \delta(s_{j+1} - g_K(s_j, a_j)) \tag{28}$$

where $\delta(\cdot)$ is the Dirac delta function. When applying the action a_j, the agent receives the new state s_{j+1} together with a scalar reward $r_{j+1} = r(s_{j+1}, a_j)$, as shown in Fig. 3. We select the reward such that

$$\begin{aligned} r(s_{j+1}, a_j) = &-\left(\frac{\partial\psi_{j+1}}{\partial x} - \frac{\partial\psi}{\partial x}^\star\right)^T Q \left(\frac{\partial\psi_{j+1}}{\partial x} - \frac{\partial\psi}{\partial x}^\star\right) \\ &- \alpha_3 S\left(\frac{\partial\psi_{j+1}}{\partial x} - \frac{\partial\psi}{\partial x}^\star\right)\varepsilon_{j+1} - \alpha_4\varepsilon_{j+1}^2 - R(a_j - a^\star)^2 \end{aligned} \tag{29}$$

where

$$Q = \begin{bmatrix} \alpha_1 I_{(p_i-1)\times(p_i-1)} & 0 & 0 \\ 0 & \alpha_2 & 0 \\ 0 & 0 & \alpha_1 I_{(N-p_i)\times(N-p_i)} \end{bmatrix} \in \mathbb{R}^{N\times N} \tag{30}$$

where the notation $I_{\alpha\times\alpha}$ stands for an identity matrix of dimensions $\alpha\times\alpha$. The Q matrix is built to have a different cost at the diagonal entry corresponding to the error integrated by the integral state. It possible to notice that the third term in (29) can be rewritten as

$$\alpha_3 S\left(\frac{\partial\psi_{j+1}}{\partial x} - \frac{\partial\psi}{\partial x}^\star\right)\varepsilon_{j+1} = \alpha_3(y_{s,j} - y_s^\star)\varepsilon_{j+1} = \alpha_3\frac{\varepsilon_{j+1} - \varepsilon_j}{\delta t}\varepsilon_{j+1}. \tag{31}$$

Therefore, the presence of this term in the reward allows to penalize the integral state variations. The policy π defines the mapping from the state space to the action space, which the agent modifies during the learning phase, and will be used once the learning procedure is completed. The policy can be deterministic or stochastic. A stochastic policy draws actions from a random distribution, whose state-dependent momenta are learned by the agent. A common choice in DRL algorithms is to draw from Gaussian distributions. Hence, the policy DNN $\pi(s_j) = [\mu(s_j), \sigma(s_j)]^T$ is a function of the state s_j that returns the mean μ and the variance σ. Then, the probability of selecting a_j when the system is in the state s_j is

$$\mathcal{P}(a_j \mid s_j) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{a_j - \mu(s_j)}{\sigma(s_j)}\right)^2}. \tag{32}$$

In the case of deterministic policy, we have that

$$\mathcal{P}(a_j \mid s_j) = \delta(a_j - \mu(s_j)). \tag{33}$$

Hence, in the case of deterministic policies, the policy DNN is usually trained to provide directly the next action. As in the standard DRL framework, we optimize over a discounted objective. Hence, the total discounted reward from time j onward is defined as

$$R_j = \sum_{k=0}^{\infty} \gamma^k r(s_{j+1+k}, a_{j+k}) \tag{34}$$

where $\gamma \in [0, 1]$ is the discount factor. The tokamak's environment, together with the integral action, is sketched in Fig. 4.

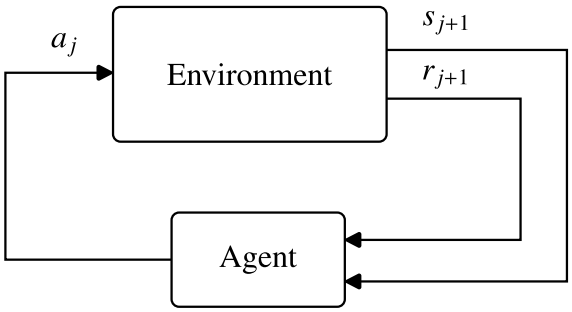


Fig. 3. General representation of the interaction between the *environment* (system to be controlled) and the *agent* (controller) in the RL context.

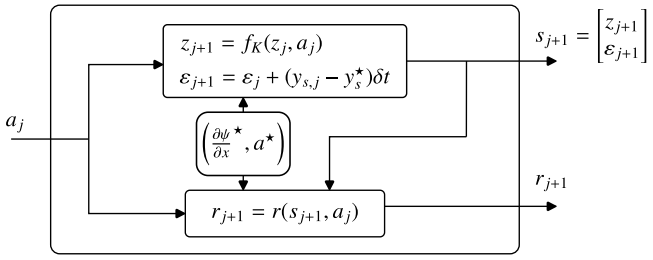


Fig. 4. Graphical representation of the tokamak environment together with the integral state in the RL settings. The action allows to compute the future state. The future state together with the actual input allow to compute the future reward.

3.3. Training algorithm

In this section, we describe the strategy used in the training algorithm in order to make the agent learn how to use the integral state to regulate the desired error to zero. The implementation of the learning algorithm is summarized in Algorithm 1. Firstly, the parameters $K \in \mathcal{K}$ are selected according to the tokamak configuration that we want to control. In the following, we refer to N_E and N_S as the number of episodes and episode steps in the training, respectively. We define the *steady-state output set* $\mathcal{O}$ as

$$\mathcal{O} = \{(\frac{\partial \psi}{\partial x}^\star, a^\star) \in \mathbb{R}^N \times \mathcal{A}\}. \tag{35}$$

At the beginning of every episode, a couple of steady-state output and input $(\frac{\partial \psi}{\partial x}^\star, a^\star) \in \mathcal{O}$ is selected. After analysing numerous equilibrium positions using different plasma parameters, we observed that the variation of the gradient's equilibria was small with respect to plasma parameter variations. In particular, we noticed that by changing the plasma parameters, we could shift the $\frac{\partial \psi}{\partial x}^\star$ maximum value and either flatten or sharpen the shape of the equilibrium function around this point. To avoid the time-consuming task of identifying all possible steady-state outputs with different parameters selection, we observed that a similar outcome could be achieved by adding a Gaussian function perturbation to the calculated equilibrium position computed with parameters

$$\frac{\partial \psi_P}{\partial x}^\star = \frac{\partial \psi}{\partial x}^\star + \frac{c_{\mathcal{N}}}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{36}$$

where $\mu \backsim U([0.4, 0.6])$, $\sigma \backsim U([0.8, 1.2])$ and $c_{\mathcal{N}} \backsim U([-1, +1])$ are continuous uniform random variable selected in different intervals. We chose to use the Gaussian function since it is a function that approaches zero at $x = 1, 0$, as we aimed to preserve the value of $\frac{\partial \psi}{\partial x}^\star$ at the plasma's center and the LCFS. This decision was influenced by the fact that the magnetic flux gradient remains fixed at zero at the center, while the flux gradient at the LCFS is primarily dependent on the magnetic central location, which experiences comparatively minimal variations compared to other plasma parameters.

Algorithm 1 Training Algorithm

Data:

- Initialize parameters for actor θ_0 and critic ϕ_0
- Initialize the evolution parameters $K \in \mathcal{K}$

for $k = 1$ **to** N_E **do**
  Initialize the tokamak's state $z_0 \in Z$
  Initialize the integrator state $x_{i,0} = 0$
  Randomly select a couple of compatible steady-state output and input $(\frac{\partial \psi}{\partial x}^\star, a^\star)$
  Compute the perturbed steady-state output $\frac{\partial \psi_P}{\partial x}^\star$ using (35)
  for $j = 1$ **to** N_S **do**
    The agent draws an action a_j using the current stochastic policy π_{θ_k}
    Update the state $s_{j+1} = g_K(s_j, a_j)$
    Compute the reward $r_{j+1} = r(s_{j+1}, a_j)$ using $(\frac{\partial \psi_P}{\partial x}^\star, a^\star)$
  end
  Compute the total discounted reward R_j
  Compute the Advantage estimate A_j from current critic V_{ϕ_k} using the "Generalized Advantage Estimate" algorithm
  Update actor θ_{k+1} minimizing (14)
  Update critic ϕ_{k+1} minimizing (9).
end

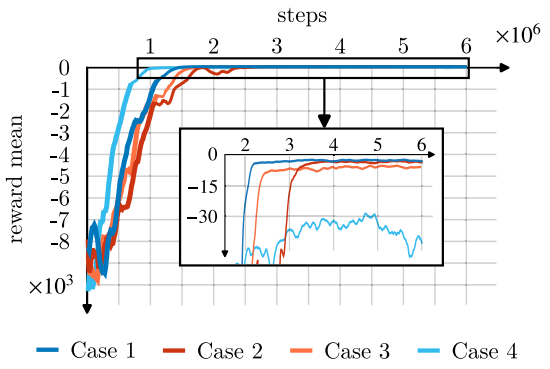
After the modification of the equilibrium's shape, we ask the agent to stabilize the system to this unreachable equilibrium, and therefore we expect the agent to give priority to the regulation of the integrated error at one point of the spatial domain. Then, the learning procedure is carried out following the PPO algorithm described in the previous section. It is worth emphasizing that during the learning procedure, a stochastic policy is employed to improve exploration, whereas in Section 4, a deterministic policy will be utilized at test time.

The selection matrix S, for the definition of the point to be regulated, is selected to be 0 in all entries except for the 5th position, which is set equal to 1. This means that we seek to regulate to zero the error at the $x = 0.2$ position. We select a position near the plasma center since it is in the interval of the spatial domain where the measures are more reliable. Moreover, we are sufficiently near the deposit of the ECCD current that is in $x = 0$.

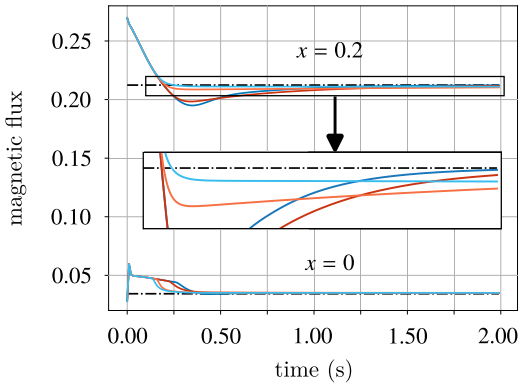
To design the training-based controller, we carry out four different controller training in order to give some hints on the definition of the α_i parameters depending on the desired closed-loop performances. It is worth mentioning that only the ratio between the free parameters matters in the learning of the final controller. Therefore, we arbitrarily fix the value of the α_3, α_4 and R parameters, letting varying only the α_1 and α_2 parameters. In particular, the common cost parameters for the four trainings are

$$\alpha_3 = 0.1, \quad \alpha_4 = 130, \quad R = 0.01, \tag{37}$$

while the α_1 and α_2 parameters (α_1, α_2) are selected in the set of pairs $\{(0.1, 0.05), (1, 0.5), (10, 5), (100, 50)\}$. In Fig. 5(a), we show the episode reward mean (as defined in (29)) of four trainings with different α_1 and α_2. In Fig. 5(b) we show the time-varying profiles at two points of the spatial domain, namely $x = 0$ and $x = 0.2$, of the closed-loop simulations using the four obtained controllers. The simulations are performed on the same model with which the controllers have been trained, while we set an unreachable equilibrium position, in the form of (36), as desired set-point. A proper functioning of the integral action becomes necessary to obtain the convergence towards the equilibrium at the integrator location, *i.e.* at $x = 0.2$. Throughout the different simulations, we employ the same unreachable equilibrium to ensure comparability between the simulations. After analysing Figs. 5(a) and 5(b), we class the four different cases depending on the speed of the integral action,



(a) Reward mean of four different training of the proposed RL algorithm.



(b) Magnetic flux trajectories at two points of the spatial domain ($x = 0$ and $x = 0.2$) of the system in closed-loop with the four controllers obtained with different α_i parameters.

Fig. 5. Episode reward mean during the RL training and Magnetic flux trajectories with the corresponding controllers. Case 1: $\alpha_1 = 0.1$, $\alpha_2 = 0.05$; Case 2: $\alpha_1 = 1$, $\alpha_2 = 0.5$; Case 3: $\alpha_1 = 10$, $\alpha_2 = 5$; Case 4: $\alpha_1 = 100$, $\alpha_2 = 50$.

the overshoot magnitude and the number of steps N_{steps} after which no substantial controller changes were observed:

- Case 1: $\alpha_1 = 0.1$, $\alpha_2 = 0.05$. Big overshoot, fast regulation. $N_{steps} = 3.5 \times 10^6$;
- Case 2: $\alpha_1 = 1$, $\alpha_2 = 0.5$. Medium overshoot, medium speed regulation. $N_{steps} = 4.5 \times 10^6$;
- Case 3: $\alpha_1 = 10$, $\alpha_2 = 5$. No overshoot, slow regulation. $N_{steps} = 4.5 \times 10^6$;
- Case 4: $\alpha_1 = 100$, $\alpha_2 = 50$. No overshoot, no regulation. $N_{steps} = 3.5 \times 10^6$.

It follows from the comparisons of all these cases that reducing the values of α_1 and α_2 with respect to α_4 provides a quick output regulation at the desired point. However, this comes at the expense of a larger magnitude of overshoot. Furthermore, due to the difference in N_{steps} across the various cases, we can formulate some preliminary hypotheses (which necessitate more extensive investigation for validation) concerning the learning behaviour of the RL algorithm. Specifically, in Cases 1 and 4, the α_i parameters "guide" the controller's learning towards distinct objectives: achieving rapid integral action and minimizing overshoot, respectively. In Cases 2 and 3, the α_i parameters steer the controller towards a balance between these two attributes. We infer that this difference is accountable for the larger N_{steps} in Case 2 and 3, where the controller is "asked" to learn two distinct attributes rather than just one.

In the next section, the controller trained with the parameters of Case 1 is tested in closed loop with the Raptor simulator. The reason behind the selection of the controller of Case 1 lies in its sufficiently fast integral action. However, if the priority is to have reduced overshoot at the expense of a slower integral action, an alternative would be to select the controller of Case 3. We remark that the output of the training procedure is a trained neural network that takes as input the state s_j and the reference point $s_K^\star$ and returns the values of the input a_j to be applied.

4. Control results

In this section, we show the closed-loop simulations of the RL feedback control law applied to the Raptor simulator. In the following simulations, the parameters are selected equal to the ones in (B.7)–(B.8). Target ι-profiles, corresponding to steady-state plasma configurations, are generated by applying a constant input for a sufficiently long time in the Raptor simulator. If these profiles are used in the feedback control law without changing the Raptor configurations, we can achieve perfect tracking of the target profiles. Nevertheless, it is difficult to get perfect tracking in practical applications because of the high system uncertainties and the limited degrees of freedom in the available actuators. During the simulations, the ramp-up phase lasts until $t = 0.02$ s bringing the central current from $I_p = 80$ kA to $I_p = 120$ kA, while the feedback controller is activated at $t = 0.1$ s. During the flat-top phase, four different target profiles are given to the controller: the first at $t = 0.1$ s, the second at $t = 2.5$ s, the third at $t = 5$ s, and the last at $t = 7.5$ s. Therefore, to test the robustness of the RL control feedback, we set up three different control scenarios:

- *1st scenario*: RL feedback applied on the Raptor simulator (with and without anti-windup).
- *2nd scenario*: RL feedback applied on the Raptor simulator with input disturbance.
- *3rd scenario*: RL feedback applied on the Raptor simulator with input disturbance and different integration point (no retraining).

1st scenario.

The training procedure is done using the linear integrator dynamic described by (25). In the first simulation, the test is done using the same integrator dynamics as in the training. A strong overshoot can be observed for the target equilibrium corresponding to a constant feedforward near the input's limits. This overshot problem is due to the windup problem caused by the simultaneous presence of the integrator and the input saturation. To solve this problem, we implement an anti-windup scheme modifying the integrator linear dynamics (25) into

$$\varepsilon_{j+1} = \begin{cases} \varepsilon_j & \text{if } a_j = 1 \text{ and } \varepsilon_{new} < \varepsilon_j \\ \varepsilon_j & \text{if } a_j = 0 \text{ and } \varepsilon_{new} \geq \varepsilon_j \\ \varepsilon_{new} & \text{else} \end{cases} \tag{38}$$
$$\varepsilon_{new} = \varepsilon_j + (y_{s,j} - y_s^\star)\delta t.$$

We refer to [59] for a survey on anti-windup techniques for linear, nonlinear, discrete, and continuous time systems. In Fig. 6, we show the controller design for the Raptor simulator. Fig. 7 shows the ι-profile evolution at four locations of the spatial domain $x \in \{0, 0.1, 0.2, 0.35\}$ during a simulation time of length $T_{sim} = 10$ s. In Fig. 7, we compare the simulation results in case the controller is equipped or not with the anti-windup algorithm. We remark that for every ι-profile stabilization, the controller without the anti-windup presents a larger overshoot when

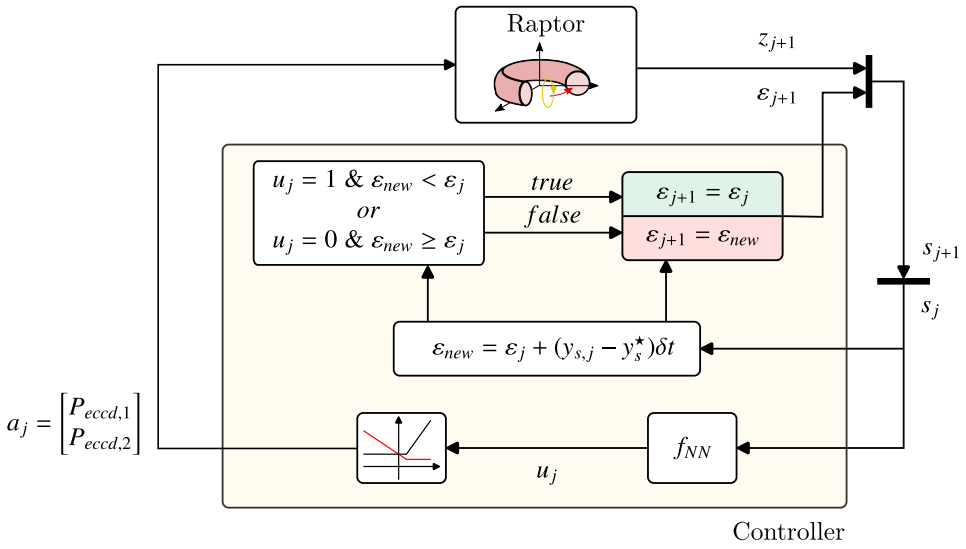


Fig. 6. Graphical representation of the proposed control scheme applied to the Raptor simulator. At each time step the trained neural network (f_{NN}) returns a number (u_j) between 0 and 1. This number is mapped to the ECCD powers to be applied to Raptor. At the same time as the input is applied to Raptor, the integral state is updated.

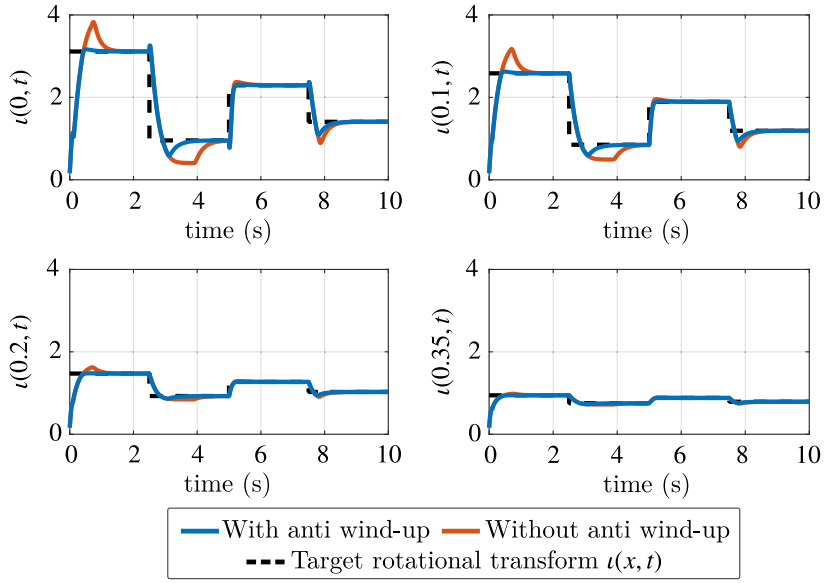
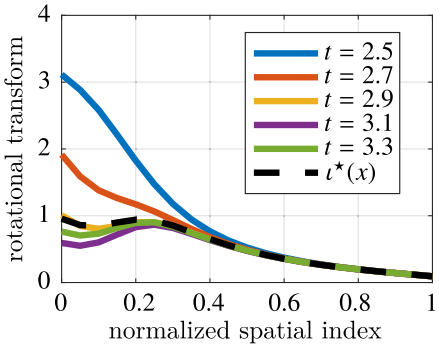


Fig. 7. Rotational transform profiles ($\iota(x,t)$) at different points of the spatial domain $\{0, 0.1, 0.2, 0.35\}$. The orange curves represent the temporal evolution of the profiles without the implementation of the anti-wind-up algorithm, while the blue curves depict the profiles with the utilization of the anti-wind-up algorithm.

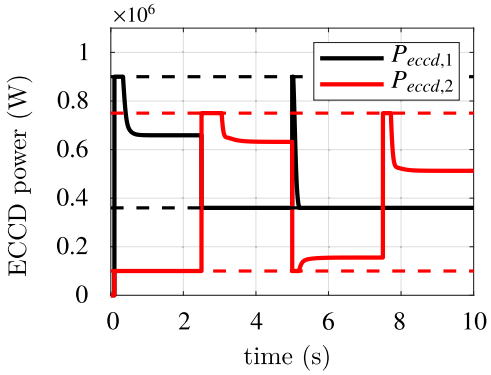
the feed-forward input (connected to the required reference) is closer to the saturation. Using the linear integrator dynamics, the integrator state can continue to integrate (possibly in the wrong direction) when the input is saturated. This means that the integrator will need some additional time to come back to a value in the interval where the input is not saturated. Using the nonlinear integrator dynamics in (38), we prevent the integrator state to vary in the wrong direction in case of input saturation. Notice that the trajectories with the anti-windup implementation present a smaller overshoot for some profiles and no overshot in others. Fig. 8 shows the ι-profiles at different time instants $t \in \{2.5,\ 2.7,\ 2.9,\ 3.1,\ 3.3\}$ and the feedback control input $a_j = \begin{bmatrix} P_{eccd,1} \\ P_{eccd,2} \end{bmatrix}$ during the simulation interval. Comparing Figs. 8(a) and 8(c), we can remark that the ι-profile in the case of anti-windup implementation is much closer to the desired profile than in the case of linear integrator dynamics. Nevertheless, in both cases, the perfect tracking of the desired ι-profile is eventually achieved for all given references.

2nd scenario.

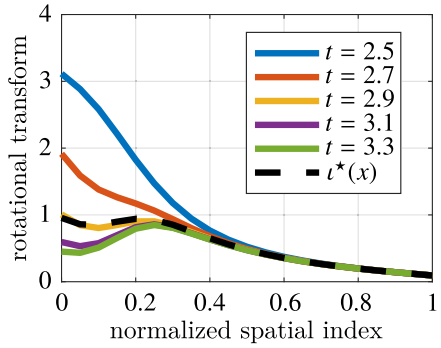
The same controller including the anti-windup integrator as in the last section is evaluated in the second scenario. In this scenario, we introduce an input disturbance at time $t = 5.3$ s, implemented with a Neutral-Beam Injector with $\rho_{dep} = 0.4$ and $w_{cd} = 0.4$. Both parameters ρ_{dep}, w_{cd} are set differently from the ones of the ECCD antenna in (B.8). Therefore, when the input disturbance is activated, the magnetic flux profile is modified and it is not possible to obtain the perfect tracking of the ι-profile reference as in the previous scenario. Fig. 9 represents the ι trajectories in four points of the spatial domain. Starting from the disturbance application, we can remark that the error at the integration point $x = 0.2$ is regulated to zero, while the trajectories in the other locations are not regulated to zero. Fig. 10(a) shows that the ι-profile is stabilized to the desired profile at $t = 5.3$ s, when the disturbance is introduced. Then, the profile is stabilized into a shape that coincides with the desired shape at $x = 0.2$ and it is different at the other locations of the spatial domain.



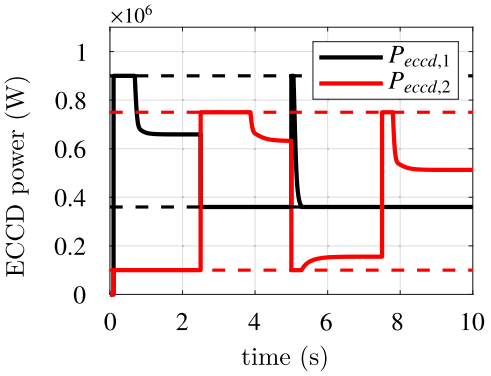
(a) Rotational transform profiles ($\iota(x,t)$) at different time instants with the implementation of the anti-windup algorithm.



(b) Applied ECCD power control input in case of implementation of the anti-windup algorithm.



(c) Rotational transform profiles ($\iota(x,t)$) at different time instants without the implementation of the anti-windup algorithm.



(d) Applied ECCD power control input in case the anti-windup algorithm is not implemented.

Fig. 8. ι-profile and applied input comparison between controller with and without anti-windup.

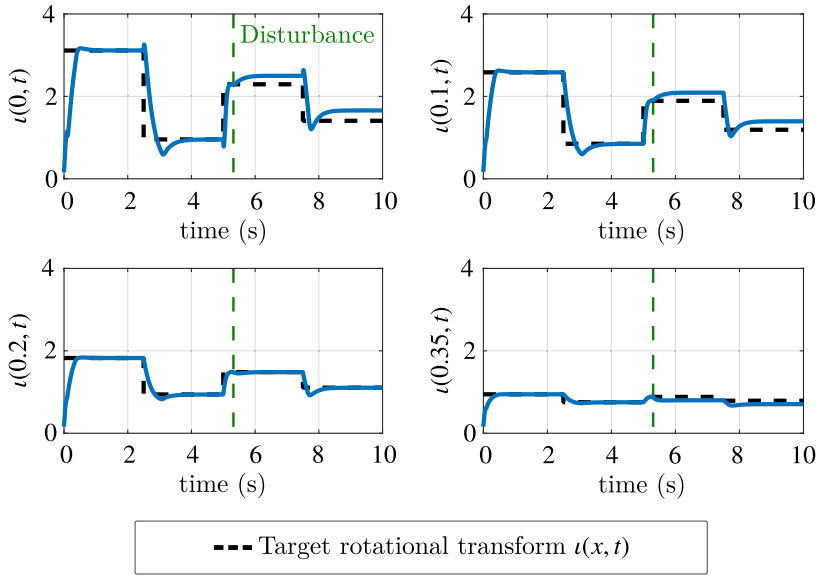
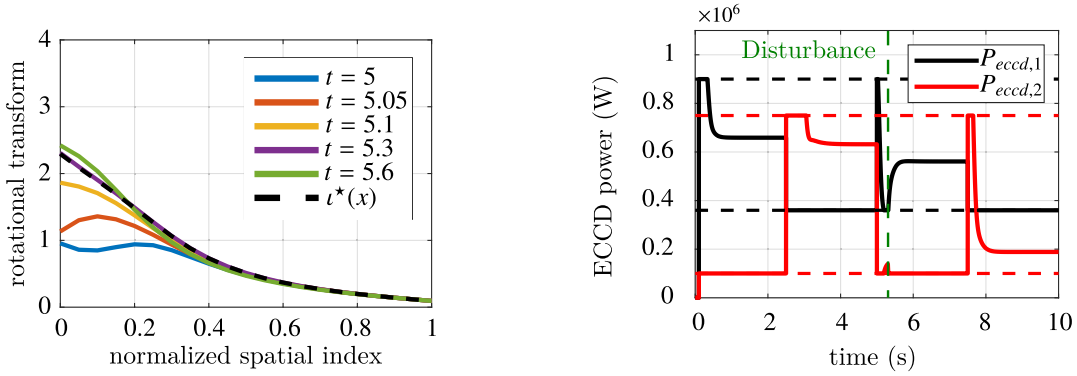


Fig. 9. Rotational transform profiles ($\iota(x,t)$) at different points of the spatial domain $\{0, 0.1, 0.2, 0.35\}$ for a time period of 10 seconds. A disturbance, implemented as a Neutral-Beam Injector placed in a different place from the ECCD antenna control input, is applied at time $t = 5.3$ (s).

3rd scenario.

In this third scenario, we use the same anti-windup controller used in the previous scenarios. In this scenario, the integration position is changed from $p_i = 5$ to $p_i = 3$. This means that the position where we want to regulate the error to zero is $x = 0.1$ instead of $x = 0.2$. We remark that the neural network has not been retrained in a different integration position, therefore with this simulation, we want to test the robustness of our control algorithm with respect to the integration point. Fig. 11 shows that before the disturbance application,



(a) Rotational transform profiles ($\iota(x,t)$) at different time instants after the disturbance application.

(b) Applied ECCD power control input before and after the disturbance application.

Fig. 10. ι-profiles and applied ECCD power ($P_{ECCD,i}$) for the two antennas in case of external disturbance application.

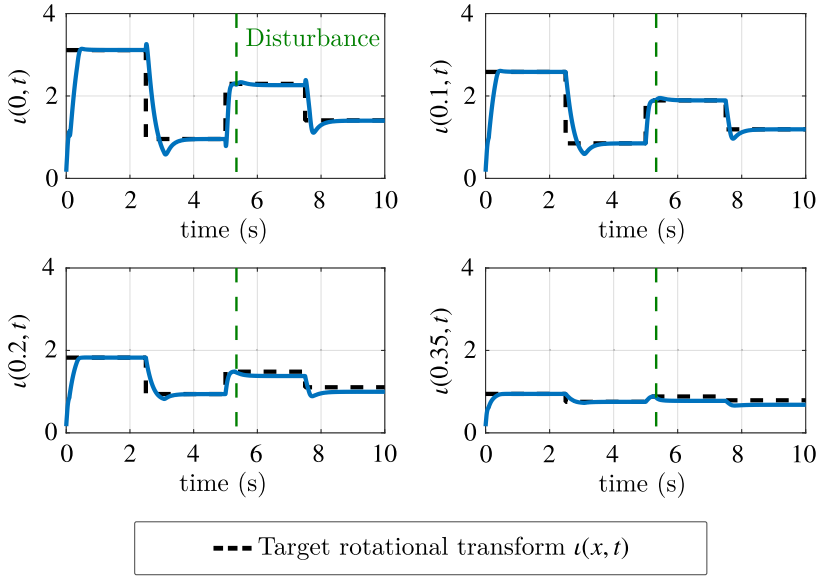


Fig. 11. Rotational transform profiles ($\iota(x,t)$) at different points of the spatial domain $\{0, 0.1, 0.2, 0.35\}$ for a time period of 10 seconds. A disturbance, implemented as a Neutral-Beam Injector placed in a different place from the ECCD antenna control input, is applied at time $t = 5.3$ (s). In this simulation the position at which we want to regulate the error to zero is $x = 0.1$, while the controller was trained to regulate the error at $x = 0.2$.

the controller is able to perfectly track the desired profiles. While after the disturbance application, the error is regulated to zero at $x = 0.1$ and not in the other points. In Fig. 12(a) we remark that after the disturbance application, the ι-profile is stabilized to a shape where the error is zero at $x = 0.1$ position.

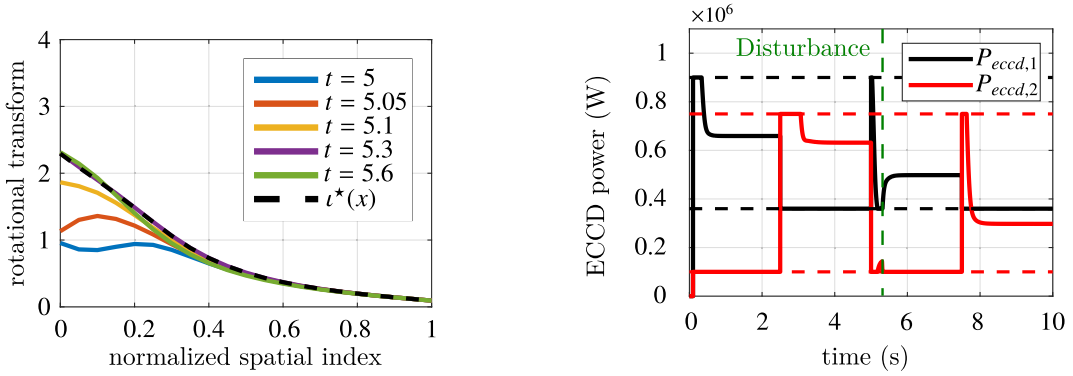
5. Conclusions

In this article, we have developed a dynamic Deep Neural Network (DNN) controller enhanced with an integral action using a Deep Reinforcement Learning (DRL) algorithm to regulate the plasma safety factor in a tokamak. Firstly, we provided an overview of the current state-of-the-art in reinforcement learning control. Subsequently, we derived a simplified, finite-dimensional nonlinear discrete system from the original distributed magnetic and kinetic plasma equations. Next, a DRL training algorithm has been proposed to design the controller for the plasma dynamics in cascade with the temporal integrator of the error to be regulated. The proposed simulator has been implemented in Python to be able to train the DNN controller using state-of-the-art DRL algorithms. Finally, we evaluated the performance of the obtained controller on the Raptor simulator under standard conditions, in the presence of external disturbances and when the time integrator integrates an error that differs from the one used during training.

A possible extension to this work is the realization of a second loop of optimization that learns how to select the best α_i parameters of the reward function to optimize a cost function as the one proposed in Section 3.3 of [60]. Another future research is to use the same RL algorithm to achieve simultaneous stabilization of the temperature and the safety factor profile. Additionally, an important extension would be to obtain at least local stability guarantees for the closed-loop system with the proposed dynamic DNN controller.

CRediT authorship contribution statement

Andrea Mattioni: Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft, Writing – review & editing. **Samuele Zoboli:** Conceptualization, Methodology, Software, Validation, Investigation, Writing – review & editing. **Bojan Mavkov:** Software, Validation, Resources. **Daniele Astolfi:** Conceptualization, Writing – review & editing, Supervision. **Vincent Andrieu:** Conceptualization, Supervision. **Emmanuel Witrant:** Resources, Supervision.



(a) Rotational transform profiles ($\iota(x,t)$) at different time instants after the disturbance application.

(b) Applied ECCD power control input before and after the disturbance application.

Fig. 12. ι-profiles and applied ECCD power ($P_{ECCD,i}$) for the two antennas in case the point to be regulated is different from the one the controller has been trained with and external disturbance application.

Paolo Frasca: Conceptualization, Writing – review & editing, Supervision, Funding acquisition. **Christophe Prieur:** Conceptualization, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

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.

Data availability

Data will be made available on request.

Acknowledgements

This work has been received funding from MIAI@Grenoble Alpes project, France (contract ANR-19-P3IA-0003) and the ANR DELICIO project, France (contract ANR-18-CE40-0010).

Appendix A. Safety factor and thermal energy control model

Consider the reaction–diffusion equation describing the magnetic flux dynamics (see Table A.2)

$$\frac{\partial \psi}{\partial t}(x,t) = \frac{D(x,t)}{a_\rho^2}\frac{\partial^2 \psi}{\partial x^2}(x,t) + \frac{G(x,t)}{a_\rho}\frac{\partial \psi}{\partial x}(x,t) + S(x,t). \tag{A.1}$$

For a summary of the notation regarding the fusion equations, we refer to Table A.2. The coefficients $D(x,t)$, $G(x,t)$, and $S(x,t)$ can be computed by following [61, eq. III-34] as

$$D(x,t) = \frac{\eta_\parallel C_2}{\mu_0 C_3} \quad G(x,t) = \frac{\eta_\parallel F a_\rho}{\mu_0 C_3}\frac{\partial}{\partial \rho}\left(\frac{C_2}{F}\right) \quad S(x,t) = L(\rho,t) j_{ni}$$
$$L(\rho,t) = \frac{\eta_\parallel V' B_{\phi 0}}{F C_3} \tag{A.2}$$

where $\eta_\parallel(\rho,t)$ is the resistivity, μ_0 is the permeability of the free space, F is the diamagnetic function, $V(\rho,t)$ is the plasma volume, $V' = \frac{\partial V}{\partial \rho}$ is the volume spatial derivative while C_2 and C_3 are space varying parameters depending on the considered plasma geometry configuration. $j_{ni}(x,t)$ is the non-inductive current source and includes the bootstrap current j_{bs} as well as the ECCD density currents j_{eccd}

$$j_{ni} = j_{bs} + j_{eccd}. \tag{A.3}$$

In this work, the bootstrap currents are computed according to [25]

$$j_{bs} = -\frac{k_{bs}}{\partial \psi / \partial \rho}\left(\mathcal{L}_{31}\frac{\partial \ln(n_e)}{\partial \rho} + R_{pe}(\mathcal{L}_{31}+\mathcal{L}_{32})\frac{\partial \ln(T_e)}{\partial \rho} + (1-R_{pe})(\mathcal{L}_{31}+\alpha\mathcal{L}_{34})\frac{\partial \ln(T_i)}{\partial \rho}\right) \tag{A.4}$$

where T_e is the electronic temperature, $T_i(x,t) \approx \alpha_{Ti}(t)T_e(x,t)$ is the ions temperature, n_e is the electron density, α is a constant parameter while k_{bs}, $\mathcal{L}_{31}$, $\mathcal{L}_{32}$, $\mathcal{L}_{34}$, R_{pe} are space varying parameters depending on the electronic and ion temperatures and on the plasma geometric configuration. The ion-to-electron temperature ratio can be fixed to $\alpha_{Ti} = 0.7$. The electron density can be approximated by

$$n_e(x,t) \approx \frac{\gamma_n + 1}{\gamma_n}(1 - x^{\gamma_n})\bar{n}_e \tag{A.5}$$

where $\bar{n}_e$ is the electron line average density, that in our case has been considered to be constant $\bar{n}_e = 1 \times 10^{-19}$.

An appropriate and effective choice used in control-oriented plasma-dynamics simulators is to approximate the current density by a

Table A.2
Table of symbols and corresponding units.

Symbol	Description	Unit
a_ρ	Radius of the LCFS	m
B	Magnetic field	T
B_ϕ	Toroidal magnetic field	T
$\eta_\parallel$	Resistivity	$\Omega \times$ m
e	Electron charge 1.6022×10^{-19}	C
F	Diamagnetic function	T × m
ι	Rotational transform	
I_p	Total plasma current	A
j_{ni}	Non-inductive current	J/m^2
j_{bs}	Bootstrap current	J/m^2
j_{eccd}	Electron Ciclotron Current Drive density current	J/m^2
j_{tor}	Toroidal density current	J/m^2
μ_0	Permeability of the free space $4\pi \times 10^{-7}$	H/m
n_e	Electron density profile	m^{-3}
n_i	Ion density profile	m^{-3}
ϕ	Magnetic flux of the toroidal field	T/m^2
P_{eccd}	ECCD power	W
P_{OH}	Total ohmic power	W
ψ	Magnetic flux of the poloidal field	T/m^2
q	Safety factor	
R	Major plasma radius	m
R_0	Magnetic center location	m
ρ	Spatial index	
τ_{th}	Thermal energy confinement time	s
T_e	Electronic temperature	eV
T_i	Ion temperature	eV
U_{pl}	Toroidal loop voltage	
V	Plasma volume	m^3
W_{th}	Plasma thermal energy	J
x	Normalized spatial index	
χ_e	Electron thermal diffusivity	m^2/s
Z_{eff}	Effective Plasma charge	C

weighted Gaussian [8]. According to [25], the ECCD efficiency can be modelled heuristically as

$$j_{eccd,i}(\rho,t) = c_{cd,i} e^{\rho^2/0.5^2} \frac{T_e}{n_e} e^{-4(\rho-\rho_{dep,i})^2/w_{cd,i}^2} P_{eccd,i}(t) \tag{A.6}$$

where w_{dep} is the deposition width and ρ_{dep} is the location of the peak of the deposition, while $P_{eccd,i}$ is the power associated with the ith antenna. The parameter c_{cd} is a machine-dependent parameter that can be chosen to scale the expression to the experimentally obtained current drive values. The total ECCD current is obtained as the sum of the different antennas, that in this work are considered to be two

$$j_{eccd}(\rho,t) = j_{eccd,1}(\rho,t) + j_{eccd,2}(\rho,t). \tag{A.7}$$

According to [62], the conductivity can be computed as

$$\eta_{\|} = \frac{1}{\sigma_{\|}} = \frac{1}{\sigma_{sptz} c_{neo}}. \tag{A.8}$$

The Spitzer conductivity σ_{sptz} depends on the electron temperature and on the effective value of the plasma charge Z_{eff}. This last parameter may in general vary spatially, but it is chosen here to be a fixed quantity for the whole plasma $Z_{eff} = 3.5$. The neoclassical correction c_{neo} depends on the electron and ion collisionality parameters as well as on Z_{eff}. Both σ_{sptz} and c_{neo} are space and time-varying.

Specific boundary conditions have to be considered both at the center and on the LCFS of the plasma. At the plasma center, the spatial variation of the flux is zero

$$\frac{\partial \psi}{\partial x}(0,t) = 0, \tag{A.9}$$

while at the LCFS, we consider a Neumann boundary condition

$$\frac{\partial \psi}{\partial x}(1,t) = -\frac{R_0 \mu_0 I_p(t)}{2\pi}. \tag{A.10}$$

where I_p is the total plasma current. The toroidal flux $\phi(x,t)$ is defined as the magnetic flux passing through a poloidal surface centered at R_0 and with normalized radius x. Assuming that the toroidal magnetic field remains constant, it is possible to obtain an explicit formula for the toroidal flux [8]

$$\phi(x,t) = \frac{1}{2\pi}\int_{S_{pol}} B(R,Z) dS_{pol} = -\frac{1}{2\pi}\int_{S_{pol}} B_\phi dS_{pol} \approx -\frac{B_{\phi 0} a_\rho^2 x^2}{2}. \tag{A.11}$$

An instrumental quantity for the temperature dynamics is the ohmic power

$$P_{OH} = \int_0^1 \frac{1}{2\pi R_0} U_{pl} j_{tor} dx \tag{A.12}$$

In the previous equation, U_{pl} identifies the toroidal loop voltage while j_{tor} corresponds to the toroidal density and they can be computed as

$$U_{pl} = \frac{\partial \psi}{\partial t} \quad j_{tor} = \frac{1}{\eta_{\|}}\left(\frac{D(x,t)}{a_\rho^2}\frac{\partial^2 \psi}{\partial x^2} + \frac{G(x,t)}{a_\rho}\frac{\partial \psi}{\partial x}\right). \tag{A.13}$$

The temperature diffusion equation writes

$$\frac{3}{2}\frac{\partial n_e T_e}{\partial t} = \frac{1}{\rho}\frac{\partial}{\partial \rho}\left(\rho n_e \chi_e(\rho,t)\frac{\partial T_e}{\partial \rho}\right) - \frac{3 n_e T_e}{2\tau_d} + S_T(\rho,t) \tag{A.14}$$

where $\chi_e(\rho,t)$ is the electron thermal diffusivity, τ_d is the time-varying damping modelling the losses and $S_T(\rho,t)$ is the source term. In our specific application, where we consider two ECCD inputs, we have

$$S_T(\rho,t) = S_{T,eccd,1}(\rho,t) + S_{T,eccd,2}(\rho,t). \tag{A.15}$$

It is worth remarking that for $i \in \{1,2\}$ the source term has an amplitude such that

$$\int_0^1 S_{T,eccd,i}(x,t) dx = P_{eccd,i}. \tag{A.16}$$

Because of the high uncertainty of the proposed temperature model and in order to effectively diminish the simulation time, we choose to use an empirical reduced-order model that approximates the actual temperature dynamics, similar to the one proposed in [63]. This model is composed of an ordinary differential equation representing the evolution of the thermal energy W_{th} and a Neural network that takes as input the total power and the thermal energy and returns the distributed temperature profile. The plasma thermal energy is defined as

$$W_{th} = W_e(t) + W_i(t) = \frac{3e}{2}\int_V (n_e T_e + n_i T_i) dV = \frac{3e}{2}\int_V (1 + \alpha_{Ti}\alpha_{ni}) n_e T_e dV \tag{A.17}$$

where $n_i \approx \alpha_{ni} n_e(x,t)$ is the ions density, e is the electron charge and W_e, W_i are the electrons and ions energy, respectively. The density ratio can be approximately computed as $\alpha_{ni} \approx (7 - Z_{eff})/6$.

The exponents in the τ_{th} expression in (20) have been obtained by applying linear regression on data obtained from Raptor simulations. In particular, the collected data together with the applied open-loop input, are organized in the vectors X and Y as following

$$X = \begin{bmatrix} 1 \\ \log(P_{tot}) \\ \log(1 + P_{eccd,1}) \\ \log(1 + P_{eccd,2}) \end{bmatrix} \quad Y = \tau_{th}. \tag{A.18}$$

Linear regression is then applied to the couple (X,Y) to obtain the power constant values k_0, k_1, k_2, k_3 such that

$$\tau_{th} = e^{k_0} P_{tot}^{k_1} (1 + P_{eccd,1})^{k_2} (1 + P_{eccd,2})^{k_3}. \tag{A.19}$$

The temperature profile is obtained as the output of an artificial Neural Network

$$T_e(x,t) = f_{NN}(P_{tot}, W_{th}). \tag{A.20}$$

The neural network has been trained using a set of temperature profiles associated with the total power and the thermal energy obtained by some Raptor simulations. For both the τ_{th} linear regression and the NN training, the Raptor simulations have been obtained by applying different constant open-loop inputs to the system and extracting the total power, the thermal energy, τ_{th} and the temperature profiles.

Appendix B. Simulation algorithm

Employing a combination of implicit–explicit time discretization and fixed-step spatial discretization, as outlined in [8, Appendix A], system (17) can be approximated by the difference equation

$$\psi_{j+1} = B_j^{-1} A_j \psi_j + B_j^{-1} S_j \tag{B.1}$$

where $\psi_j, S_j \in \mathbb{R}^N$ are N-dimensional vectors of the magnetic flux and the source term at N different point of the spatial domain at the j time iteration. The matrices $A_j \in \mathbb{R}^{N\times N}$ and $B_j \in \mathbb{R}^{N\times N}$ depend on the plasma physical parameters and change at every iteration j. The time discretization step is fixed at $\delta t = 0.01$, with an implicit–explicit ratio of $h = 0.45$, and the total simulation time is referred to as T_{sim}. The space domain is divided into $N = 21$ discretization elements, with a fixed spatial discretization step of $\delta x_i = 0.05$. Similarly, the thermal energy dynamics can be approximated by the difference equation

$$W_{th,j+1} = d_j W_{th,j} + s_j P_{tot,j} \tag{B.2}$$

where $W_{th,j}, P_{tot,j} \in \mathbb{R}$ are the thermal energy and the total power at the j time iteration. d_j and s_j are coefficients depending on $\tau_{th,j}$. It is worth remarking that our study case is similar to the one considered in [50], where the two available inputs act on the same spatial point: the first antenna $P_{eccd,1}$ acts positively on z_j while the second $P_{eccd,2}$ acts negatively. The two input powers have limited maximum power $P_{eccd,i} \in [\underline{P}_{eccd,i}, \bar{P}_{eccd,i}]$. To control the magnetic flux gradient, the control action corresponds to the difference between the two antennas' power. Inversely, the control action for the temperature profile corresponds to the sum of the two antennas' power. Since in this work we

are interested in magnetic control, in the following we define a function mapping from the desired power difference to the value of each antenna power. Firstly, we define the control input $a_j \in [0, 1]$ that is mapped to the desired difference $\alpha_j \in [\underline{\alpha}, \bar{\alpha}]$ between the two ECCD powers applied at the j time iteration

$$\alpha_j = \underline{\alpha} + a_j(\bar{\alpha} - \underline{\alpha}) = P_{eccd,1,j} - P_{eccd,2,j}, \tag{B.3}$$

where $\underline{\alpha} = \underline{P}_{eccd,1} - \bar{P}_{eccd,2}$ and $\bar{\alpha} = \bar{P}_{eccd,1} - \underline{P}_{eccd,2}$. Given a desired power difference α_j, the control input powers are mapped to minimize their sum $P_{eccd,1,j} + P_{eccd,2,j}$. The mapping can be expressed by

$$\begin{cases} P_{eccd,1,j} = \underline{P}_{eccd,1} & \text{if} \quad \alpha_j < \underline{P}_{eccd,1} - \underline{P}_{eccd,2} \\ P_{eccd,2,j} = \underline{P}_{eccd,1} - \alpha_j & \\ P_{eccd,1,j} = \alpha_j + \underline{P}_{eccd,2} & \text{if} \quad \alpha_j \geq \underline{P}_{eccd,1} - \underline{P}_{eccd,2}. \\ P_{eccd,2,j} = \underline{P}_{eccd,2} & \end{cases} \tag{B.4}$$

After the spatial discretization of the magnetic flux dynamics and the temporal discretization of both the magnetic flux and thermal energy dynamics, we obtain the difference equation

$$z_{j+1} = \begin{bmatrix} B_j^{-1} A_j & 0 \\ 0 & d_j \end{bmatrix} z_j + \begin{bmatrix} B_j^{-1} S_j \\ s_j P_{tot,j} \end{bmatrix} = f(z_j, a_j). \tag{B.5}$$

in the state variable

$$z_j = \begin{bmatrix} \psi_j \\ W_{th,j} \end{bmatrix} \in \mathbb{R}^{N+1}. \tag{B.6}$$

The steps for the plasma magnetic flux and temperature simulation are listed in Algorithm 2. The constant parameters are fixed as follows

$$\begin{array}{llllll} B_{\phi 0} = 1.44, & R_0 = 0.88, & a_\rho = 0.25, & Z_{eff} = 3.5, & \delta_0 = 0.3, \\ \bar{n}_e = 1 \times 10^{-19}, & \alpha_{T_i} = 0.7, & \mu_0 = 4\pi \times 10^{-7}, & \gamma_n = 2. & \end{array} \tag{B.7}$$

In the current experiment, we assume that the two ECCD actuators, described by the injected current density in (A.6), have the following parameters

$$\begin{array}{lll} c_{cd,1} = 1, & \rho_{dep,1} = 0, & w_{cd,1} = 0.35, \\ & \bar{P}_{eccd,1} = 900 \text{ (MW)}, & \underline{P}_{eccd,1} = 360 \text{ (MW)}, \\ c_{cd,2} = -1, & \rho_{dep,2} = 0, & w_{cd,2} = 0.35, \\ & \bar{P}_{eccd,2} = 750 \text{ (MW)}, & \underline{P}_{eccd,2} = 100 \text{ (MW)}. \end{array} \tag{B.8}$$

It is worth noticing that the Ohmic power at time instant $j + 1$ is computed with the variables $\eta_{\parallel,j}$, $T_{e,j}$, $T_{i,j}$, u_j, belonging to the time instant j, as well as ψ_{j+1}, belonging to the time instant $j + 1$. With this simulation procedure, it is not possible to only use variables belonging to the time instant $j + 1$ for the $P_{OH,j+1}$ calculation because $P_{OH,j+1}$ itself is needed to compute $T_{e,j+1}$ and $T_{i,j+1}$. This is an intrinsic property of this simulation procedure, introduced in [8], that avoids the implementation of a fixed point iteration research to find all the states at the time step $j+1$. The nonlinear components of the model are delayed by one sample while an implicit–explicit scheme is used for the linear components, thus avoiding the fixed-point iteration algorithm to obtain a faster simulation.

To test the simulator's precision with respect to a certain tokamak configuration, we compare the trajectories obtained with the application of some constant open-loop controls with the ones obtained through the application of the same controls with the Raptor simulator. The Raptor simulator is a real-time predictor of the Ψ and T_e profiles used as an observer in the TCV control environment [25]. The kinetic and magnetic profiles are obtained by simulating two coupled nonlinear reaction–diffusion PDEs. Therefore, the Raptor simulator provides fairly precise simulation results that can be taken as a reference for our simulator. Fig. B.13 shows the $\frac{\partial \psi}{\partial \rho}(\rho, t)$ trajectories with the application of the open loop input $a = 0.15$. The initial condition for both the Raptor and the training simulator corresponds to the steady state with

Algorithm 2 Simulation Algorithm

Data:

- Initialization of ψ_0, $W_{th,0}$ and $P_{OH,0}$ from Raptor simulation after the ramp-up phase
- Initialization of the constant physical parameters
- Initialization of the constant simulation parameters
- Initialization of the open-loop input U
- Simulation initialization $j = 0$

while $j < T_{sim}/\delta t$ **do**

1. Input extraction : $a_j = U[j]$
2. Temperature: $T_{e,j}, T_{i,j} \leftarrow f_{temp}(a_j, P_{OH,j}, W_{th,j})$ with (21), (A.20) and (B.3)–(B.4)
3. Resistivity: $\eta_{\parallel,j} \leftarrow f_{resistivity}(T_{e,j}, \psi_j)$ with (A.8)
4. Bootstrap current: $j_{bs,j} \leftarrow f_{boostrap}(T_{e,j}, T_{i,j}, \psi_j)$ with (A.4)
5. ECCD deposit: $(j_{eccd,1})_j, (j_{eccd,2})_j \leftarrow f_{eccd}(a_j, T_{e,j})$ with (A.6)
6. Non-inductive currents: $j_{ni,j} \leftarrow j_{bs,j} + (j_{eccd,1})_j + (j_{eccd,2})_j$
7. Diffusion coefficients: $D_{i,j}, G_{i,j}, L_{i,j} \leftarrow f_{coeff}(\eta_{\parallel,j})$ with (A.2)
8. Magnetic flux: $\psi_{j+1} \leftarrow f_\psi(\psi_j, D_{i,j}, G_{i,j}, L_{i,j}, j_{ni,j})$ with (B.5)
9. Thermal energy: $W_{th,j+1} \leftarrow f_{thermal}(u_j, P_{OH,j}, W_{th,j})$ with (B.5)
10. Ohmic Power: $P_{OH,j+1} = f_{ohmic}(\eta_{\parallel,j}, T_{e,j}, T_{i,j}, a_j, \psi_{j+1})$ with (A.12)–(A.13)

$j \leftarrow j + 1$

end

the constant open loop input $a = 0.7$. In Fig. B.13(a) are shown the trajectories of four points of the spatial domain $x = 0.05,\ 0.35,\ 0.5,\ 0.75$ of both the Raptor and training simulator. While in Fig. B.13(b) are shown the $\frac{\partial \psi}{\partial \rho}(\rho, t)$ profiles at time instants $t = 0.1,\ 0.2,\ 1.5$. We remark that there exists a visible difference between the profiles obtained with the proposed simulation algorithm and the Raptor simulator. Nevertheless, we can observe similar trends:

- Small values of the input result to small values of the magnetic gradient peak,
- Small values of the input result to a right-shift of the magnetic gradient peak.

In the following sections, we show that having a model that keeps the same trends as the "real" system is enough to design a controller with a DRL algorithm.

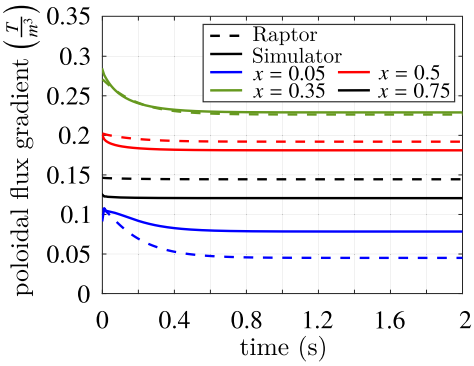
Appendix C. Reinforcement Learning with integral action on a toy model

Consider a continuous-time mass–spring–damper system with mass m, spring constant k^0, damping c, and position denoted by x. The toy model's dynamic equations can be expressed as

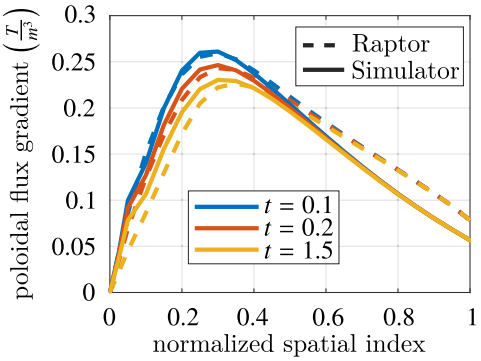
$$\ddot{x}(t) = -\frac{k^0}{m} x(t) - \frac{c}{m} \dot{x}(t) + \frac{1}{m} a(t) \tag{C.1}$$

where $a(t)$ is the control force (action) applied to the system. The state space representation, with $z = [x\ \dot{x}]^T$, corresponds to

$$\dot{z}(t) = \overbrace{\begin{bmatrix} 0 & 1 \\ -\frac{k^0}{m} & -\frac{c}{m} \end{bmatrix}}^{A} z(t) + \overbrace{\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}}^{B} a(t) \tag{C.2}$$



(a) Trajectories of poloidal flux gradient ($\partial\psi/\partial x$) over time at various points of the normalized spatial index $\{0.05, 0.35, 0.5, 0.75\}$. The dashed lines represent trajectories obtained from an open-loop Raptor simulation, while the solid lines depict trajectories resulting from the simulator proposed in this study.



(b) Poloidal flux gradient ($\partial\psi/\partial x$) profiles at different time instants $\{0.1, 0.2, 1.5\}$. The dashed lines represent trajectories obtained from an open-loop Raptor simulation, while the solid lines depict trajectories resulting from the simulator proposed in this study.

Fig. B.13. Comparison between Raptor and training model open-loop simulations.

Using an implicit–explicit discretization scheme, we are able to obtain the difference equation corresponding to the mass–spring–damper system

$$z_{j+1} = \overbrace{(I - \delta t(1-h)A)^{-1}(I + h\delta tA)}^{A_D} z_j + \overbrace{(I - \delta t(1-h)A)^{-1}B}^{B_D} a_j, \tag{C.3}$$

where δt is the discretization time step. The control objective is to find a control law capable of stabilizing the system to a desired position $x^\star$. Hence, we aim at steering the system to $z^\star = [x^\star \ 0]^\top$. Moreover, we want such a stabilization property to be robust, namely, we want the controller to stabilize the system even in presence of (constant) disturbances. An example of such disturbances may be the imperfect knowledge of the system's parameters, e.g., the spring's constant $k = k^0 + \delta k$ with $\delta k \in [-\Delta, \Delta]$. Hence, we model the true plant to be controlled as

$$z_{j+1} = (A_D + \tilde{A}_D)z_j + B_D a_j, \tag{C.4}$$

where $z_j = [x_j \ \dot{x}_j]^T$ and $\tilde{A}_D$ embeds the constant unknown spring variation δk. Instead of using a more conventional control approach (e.g. Lyapunov-based, forwarding, etc.) we now want to use an RL algorithm to learn the policy (controller) for the former system. To do that, let us define the optimal problem via the reward

$$r_j = (z_j - z^\star)^T Q (z_j - z^\star) + R(a_j - a^\star)^2. \tag{C.5}$$

where $a^\star = k^0 x^\star$ is the steady state input for obtaining the desired equilibrium with a spring constant $k = k^0$. The training is performed on a system with unitary parameters $m = c = 1$ and spring constants $k^0 = 1$ and $\Delta = 0.2$. The cost matrices are defined as $Q = 0.001I$ and $R = 0.01$. In this example, we use a 2 layers neural network with 32 nodes for both the actor and the critic. The learning rate is set equal to $\gamma = 0.001$. Training is performed with a 2×10^6 total number of steps, while each episode has 500 steps. The time step is set equal to $\delta t = 0.1$. The training is performed using 8 environments in parallel using the PPO algorithm. At each episode, the spring parameter variation is selected randomly in the interval $[-\Delta, \Delta]$. Fig. C.14 shows the evolution of the episode reward mean over time. In Fig. C.15, we show the mass–spring–damper system in a closed loop with the trained control law in case $k = 1$ with initial conditions set to $x_0 = -4, \dot{x}_0 = -5$. We can see that the control law drives the system towards an equilibrium position, with a constant offset from the desired position $x^\star = 1$.

Now, let us add the integrator dynamics and consider the stabilization of the extended system

$$\begin{cases} z_{j+1} &= (A_D + \tilde{A}_D)z_j + B_D a_j, \\ \epsilon_{j+1} &= \epsilon_j + (x_j - x^\star)\delta t. \end{cases} \tag{C.6}$$

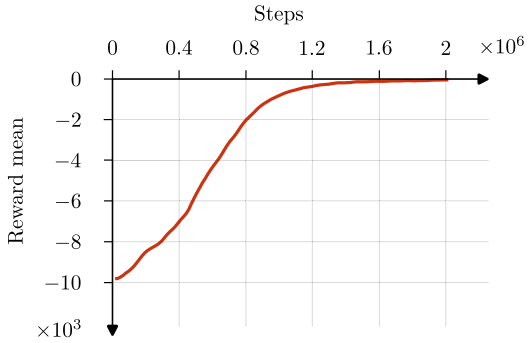


Fig. C.14. Reward mean over steps of the mass–spring–damper controller training without the integral action.

Then, if the extended system is stabilized to some (z_e, η_e) by a feedback control law $a(t) = u(z, \eta)$ such that $u(z_e, \eta_e) = u_e$, we have

$$\begin{cases} 0 &= (A_D + \tilde{A}_D)z_e + B_D u_e, \\ 0 &= (x_e - x^\star)\delta t. \end{cases} \tag{C.7}$$

As such, the second equation implies $x_e = x^\star$. Hence, $x^\star$ is reached even in presence of constant unknown variation of the spring constant. We rewrite the open-loop system with state $s_j = [x_j \ \dot{x}_j \ \epsilon_j]^T$ as

$$s_{j+1} = \begin{bmatrix} A_D + \tilde{A}_D & 0 \\ \delta t \ \ 0 & 1 \end{bmatrix} s_j + \begin{bmatrix} 0 \\ \frac{1}{m} \\ 0 \end{bmatrix} a_j + \begin{bmatrix} 0 \\ 0 \\ -x^\star \delta t \end{bmatrix} \tag{C.8}$$

We define

$$s^\star = \begin{bmatrix} 0 \\ x^\star \\ 0 \end{bmatrix} \tag{C.9}$$

and the reward as

$$r_j = (s_j - s^\star)^T Q_e (s_j - s^\star) + R(a_j - a^\star)^2 \tag{C.10}$$

where Q_e is the new extended state cost matrix and $a^\star = k^0 x^\star$ as before. We perform the same training as the one done for the system without the integrator state. In this case we fix

$$Q_e = \begin{bmatrix} 0.001 & 0 & 0.0005 \\ 0 & 0.001 & 0 \\ 0.0005 & 0 & 0.001 \end{bmatrix} \quad R = 0.01. \tag{C.11}$$

In Fig. C.16 is depicted the evolution of the episode reward mean. In Fig. C.17 we show the extended system in closed-loop with the

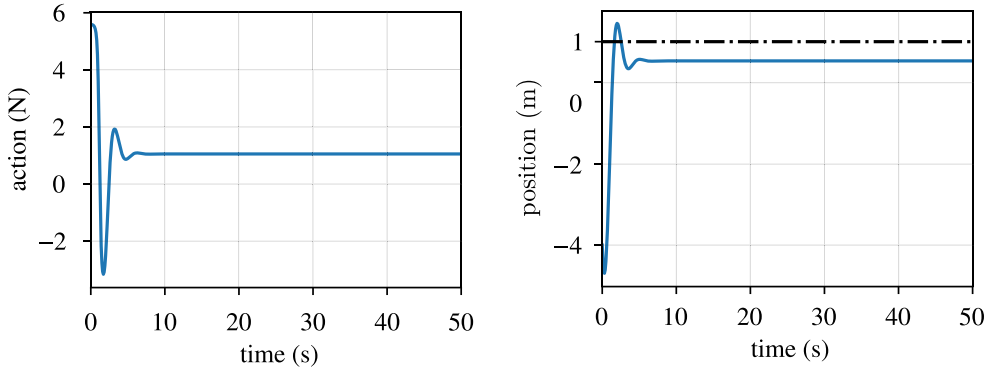


Fig. C.15. Control action and position of the mass–spring damper system in closed loop with a controller without integral state that has been trained on a model with a different spring coefficient.

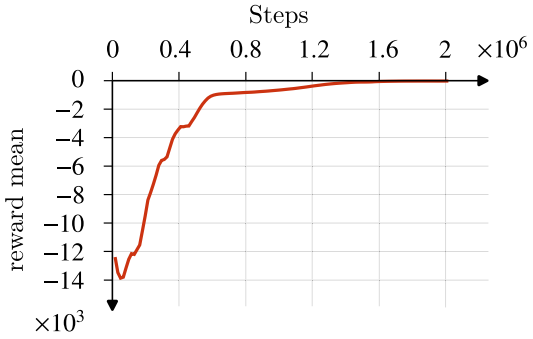


Fig. C.16. Reward mean over steps of the mass–spring–damper controller training with integral action.

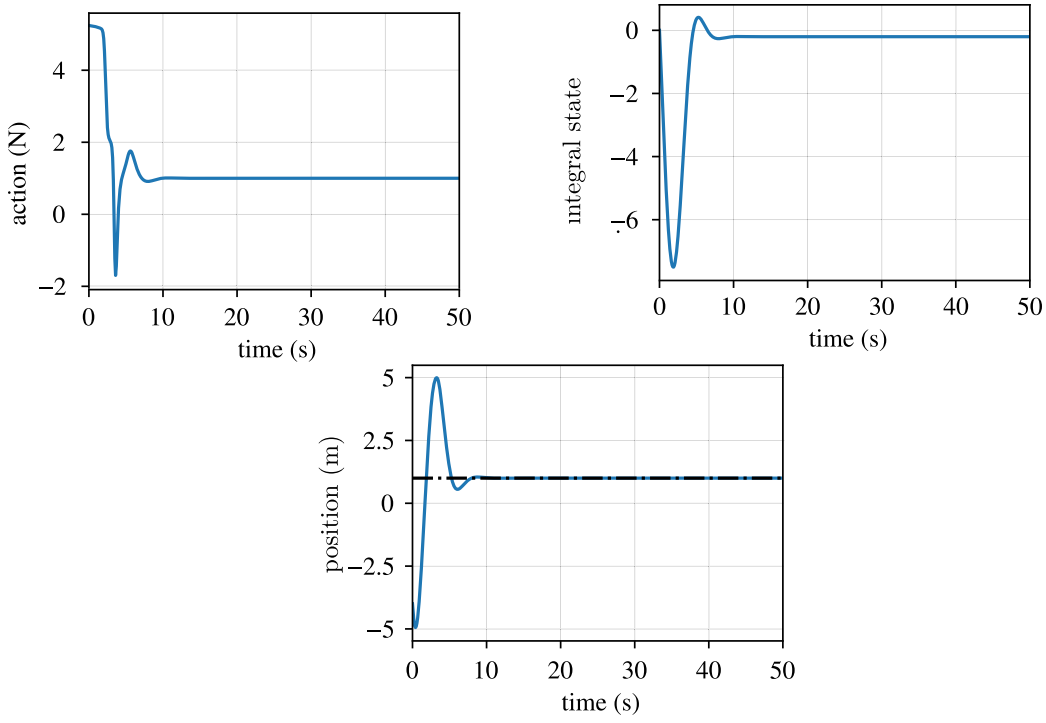


Fig. C.17. Control action, integral state and position of the mass–spring damper system in closed loop with a controller with integral state that has been trained on a model with a different spring coefficient.

trained control law in case $k = 1$. We can appreciate that in this case, the position converges to the desired equilibrium. Therefore, we have trained a robust controller that makes use of an integrator state to be robust with respect to constant parameter variation. It is possible to show that the same controller is robust also with respect to constant disturbances.

References

[1] J. Wesson, D.J. Campbell, Tokamaks, Vol. 149, Oxford University Press, 2011.

[2] F. Imbeaux, M. Lennholm, A. Ekedahl, P. Pastor, T. Aniel, S. Brémond, J. Decker, P. Devynck, R. Dumont, G. Giruzzi, P. Maget, D. Mazon, A. Merle, D. Molina, P. Moreau, F. Saint-Laurent, J. Ségui, D.Z. and, Real-time control of the safety factor profile diagnosed by magneto-hydrodynamic activity on the Tore Supra tokamak, Nucl. Fusion 51 (7) (2011) 073033.

[3] P. Moreau, S. Bremond, D. Douai, A. Geraud, P. Hertout, M. Lennholm, D. Mazon, F. Saint-Laurent, Tore Supra Team, Plasma control in Tore Supra, Fusion Sci. Technol. 56 (3) (2009) 1284–1299.

[4] T. Wijnands, D.V. Houtte, G. Martin, X. Litaudon, P. Froissard, Feedback control of the current profile on Tore Supra, Nucl. Fusion 37 (6) (1997) 777–791.

[5] D. Moreau, D. Mazon, M. Ariola, G. De Tommasi, L. Laborde, F. Piccolo, F. Sartori, T. Tala, L. Zabeo, A. Boboc, et al., A two-time-scale dynamic-model approach for magnetic and kinetic profile control in advanced tokamak scenarios on JET, Nucl. Fusion 48 (10) (2008) 106001.

[6] D. Moreau, M. Walker, J. Ferron, F. Liu, E. Schuster, J. Barton, M. Boyer, K. Burrell, S. Flanagan, P. Gohil, R. Groebner, C. Holcomb, D. Humphreys, A. Hyatt, R. Johnson, R.L. Haye, J. Lohr, T. Luce, J. Park, B. Penaflor, W. Shi, F. Turco, W.W. and, Integrated magnetic and kinetic control of advanced tokamak plasmas on DIII-D based on data-driven models, Nucl. Fusion 53 (6) (2013) 063020.

[7] J. Blum, Numerical Simulation and Optimal Control in Plasma Physics, John Wiley and Sons Inc., New York, NY, 1989.

[8] E. Witrant, E. Joffrin, S. Bremond, G. Giruzzi, D. Mazon, O. Barana, P. Moreau, A control-oriented model of the current profile in tokamak plasma, Plasma Phys. Control. Fusion 49 (2007) 1075–1105.

[9] L. Laborde, D. Mazon, D. Moreau, A. Murari, R. Felton, L. Zabeo, R. Albanese, M. Ariola, J. Bucalossi, F. Crisanti, M. de Baar, G. de Tommasi, P. de Vries, E. Joffrin, M. Lennholm, X. Litaudon, A. Pironti, T. Tala, A. Tuccillo, A model-based technique for integrated real-time profile control in the JET tokamak, Plasma Phys. Control. Fusion 47 (1) (2004) 155–183.

[10] Y. Ou, C. Xu, E. Schuster, Robust control design for the poloidal magnetic flux profile evolution in the presence of model uncertainties, IEEE Trans. Plasma Sci. 38 (3) (2010) 375–382.

[11] J.E. Barton, E. Schuster, F. Felici, O. Sauter, Closed-loop control of the safety factor profile in the TCV tokamak, in: 53rd IEEE Conference on Decision and Control, IEEE, 2014, pp. 5660–5665.

[12] Y. Ou, C. Xu, E. Schuster, T.C. Luce, J.R. Ferron, M.L. Walker, D.A. Humphreys, Optimal tracking control of current profile in tokamaks, IEEE Trans. Control Syst. Technol. 19 (2) (2011) 432–441.

[13] M.D. Boyer, J. Barton, E. Schuster, M.L. Walker, T.C. Luce, J.R. Ferron, B.G. Penaflor, R.D. Johnson, D.A. Humphreys, Backstepping control of the toroidal plasma current profile in the DIII-D tokamak, IEEE Trans. Control Syst. Technol. 22 (5) (2014) 1725–1739.

[14] E. Maljaars, F. Felici, M. De Baar, J. Van Dongen, G. Hogeweij, P. Geelen, M. Steinbuch, Control of the tokamak safety factor profile with time-varying constraints using MPC, Nucl. Fusion 55 (2) (2015) 023001.

[15] A. Pajares, E. Schuster, Nonlinear robust safety factor profile control in tokamaks via feedback linearization and nonlinear damping techniques, in: 2018 IEEE Conference on Control Technology and Applications, CCTA, IEEE, 2018, pp. 306–311.

[16] J.E. Barton, W.P. Wehner, E. Schuster, F. Felici, O. Sauter, Simultaneous closed-loop control of the current profile and the electron temperature profile in the TCV tokamak, in: 2015 American Control Conference, ACC, IEEE, 2015, pp. 3316–3321.

[17] D. Moreau, J. Artaud, J.R. Ferron, C.T. Holcomb, D.A. Humphreys, F. Liu, T.C. Luce, J.M. Park, R. Prater, F. Turco, et al., Combined magnetic and kinetic control of advanced tokamak steady state scenarios based on semi-empirical modelling, Nucl. Fusion 55 (6) (2015) 063011.

[18] H. Wang, W.P. Wehner, E. Schuster, Combined current profile and plasma energy control via model predictive control in the EAST tokamak, in: 2018 26th Mediterranean Conference on Control and Automation, MED, IEEE, 2018, pp. 1–9.

[19] S. Wang, E. Witrant, D. Moreau, Robust control of q-profile and βp using data-driven models on EAST, Fusion Eng. Des. 162 (2021) 112071.

[20] A. Gahlawat, M.M. Peet, E. Witrant, Control and verification of the safety-factor profile in tokamaks using sum-of-squares polynomials, IFAC Proc. Vol. 44 (1) (2011) 12556–12561, 18th IFAC World Congress.

[21] A. Gahlawat, E. Witrant, M.M. Peet, M. Alamir, Bootstrap current optimization in tokamaks using sum-of-squares polynomials, in: 2012 IEEE 51st IEEE Conference on Decision and Control, CDC, 2012, pp. 4359–4365.

[22] F. Bribiesca Argomedo, C. Prieur, E. Witrant, S. Bremond, A strict control Lyapunov function for a diffusion equation with time-varying distributed coefficients, IEEE Trans. Automat. Control 58 (2) (2013) 290–303.

[23] F.B. Argomedo, E. Witrant, C. Prieur, S. Brémond, R. Nouailletas, J.-F. Artaud, Lyapunov-based distributed control of the safety-factor profile in a tokamak plasma, Nucl. Fusion 53 (3) (2013) 033005.

[24] B. Mavkov, E. Witrant, C. Prieur, Distributed control of coupled inhomogeneous diffusion in tokamak plasmas, IEEE Trans. Control Syst. Technol. 27 (1) (2019) 443–450.

[25] F. Felici, Real-Time Control of Tokamak Plasmas: from Control of Physics to Physics-Based Control (Ph.D. thesis), EPFL, Lausanne, 2011.

[26] N.T. Vu, R. Nouailletas, E. Maljaars, F. Felici, O. Sauter, Plasma internal profile control using IDA-PBC: Application to TCV, Fusion Eng. Des. 123 (2017) 624–627, http://dx.doi.org/10.1016/j.fusengdes.2017.02.074, Proceedings of the 29th Symposium on Fusion Technology (SOFT-29) Prague, Czech Republic, September 5–9, 2016.

[27] E. Maljaars, F. Felici, T. Blanken, C. Galperti, O. Sauter, M. De Baar, F. Carpanese, T. Goodman, D. Kim, S. Kim, et al., Profile control simulations and experiments on TCV: a controller test environment and results using a model-based predictive controller, Nucl. Fusion 57 (12) (2017) 126063.

[28] M.L. Walker, P. De Vries, F. Felici, E. Schuster, Introduction to tokamak plasma control, in: 2020 American Control Conference, ACC, 2020, pp. 2901–2918.

[29] M.L. Walker, D. Humphreys, D. Mazon, D. Moreau, M. Okabayashi, T. Osborne, E. Schuster, Emerging applications in tokamak plasma control, IEEE Control Syst. Mag. 26 (2) (2006) 35–63.

[30] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Process. Mag. 34 (6) (2017) 26–38.

[31] Y. Li, Deep reinforcement learning: An overview, 2017, arXiv preprint arXiv:1701.07274.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, 2013, arXiv preprint arXiv:1312.5602.

[33] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.

[34] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al., Soft actor-critic algorithms and applications, 2018, arXiv preprint arXiv:1812.05905.

[35] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning, PMLR, 2018, pp. 1587–1596.

[36] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas, et al., Magnetic control of tokamak plasmas through deep reinforcement learning, Nature 602 (7897) (2022) 414–419.

[37] H.K. Khalil, Nonlinear Systems, Prentice Hall, 2002.

[38] F. Mazenc, L. Praly, Adding integrations, saturated controls, and stabilization for feedforward systems, IEEE Trans. Automat. Control 41 (11) (1996) 1559–1578.

[39] N. Vanspranghe, L. Brivadis, Output regulation of infinite-dimensional nonlinear systems: a forwarding approach for contraction semigroups, 2022, arXiv.

[40] P. Pauli, J. Köhler, J. Berberich, A. Koch, F. Allgöwer, Offset-free setpoint tracking using neural network controllers, in: Proceedings of the 3rd Conference on Learning for Dynamics and Control, in: Proceedings of Machine Learning Research, vol. 144, PMLR, 2021, pp. 992–1003.

[41] A.B. Martinsen, A.M. Lekkas, S. Gros, Reinforcement learning-based NMPC for tracking control of ASVs: Theory and experiments, Control Eng. Pract. 120 (2022) 105024.

[42] A. Farahmand, S. Nabi, P. Grover, D.N. Nikovski, Learning to control partial differential equations: Regularized fitted Q-iteration approach, in: 2016 IEEE 55th Conference on Decision and Control, CDC, 2016, pp. 4578–4585.

[43] A. Farahmand, S. Nabi, D.N. Nikovski, Deep reinforcement learning for partial differential equation control, in: 2017 American Control Conference, ACC, 2017, pp. 3120–3127.

[44] Y. Pan, A. Farahmand, M. White, S. Nabi, P. Grover, D. Nikovski, Reinforcement learning with function-valued action spaces for partial differential equation control, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 3986–3995.

[45] H. Yu, S. Park, A. Bayen, S. Moura, M. Krstic, Reinforcement learning versus PDE backstepping and PI control for congested freeway traffic, IEEE Trans. Control Syst. Technol. 30 (4) (2022) 1595–1611.

[46] T. Wakatsuki, T. Suzuki, N. Hayashi, N. Oyama, S. Ide, Safety factor profile control with reduced central solenoid flux consumption during plasma current ramp-up phase using a reinforcement learning technique, Nucl. Fusion 59 (6) (2019) 066022.

[47] J. Seo, Y.-S. Na, B. Kim, C. Lee, M. Park, S. Park, Y. Lee, Feedforward beta control in the KSTAR tokamak by deep reinforcement learning, Nucl. Fusion 61 (10) (2021) 106010.

[48] T. Wakatsuki, M. Yoshida, E. Narita, T. Suzuki, N. Hayashi, Simultaneous control of safety factor profile and normalized beta for JT-60SA using reinforcement learning, Nucl. Fusion 63 (7) (2023) 076017.

[49] M.D. Boyer, J. Barton, E. Schuster, T.C. Luce, J.R. Ferron, M.L. Walker, D.A. Humphreys, B.G. Penaflor, R.D. Johnson, First-principles-driven model-based current profile control for the DIII-D tokamak via LQI optimal control, Plasma Phys. Control. Fusion 55 (10) (2013) 105007.

[50] B. Mavkov, E. Witrant, C. Prieur, E. Maljaars, F. Felici, O. Sauter, Experimental validation of a Lyapunov-based controller for the plasma safety factor and plasma pressure in the TCV tokamak, Nucl. Fusion 58 (5) (2018) 056011.
[51] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[52] R.S. Sutton, D. McAllester, S. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, Adv. Neural Inf. Process. Syst. 12 (1999).
[53] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: International Conference on Machine Learning, PMLR, 2015, pp. 1889–1897.
[54] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.
[55] D. Astolfi, L. Praly, Integral action in output feedback for multi-input multi-output nonlinear systems, IEEE Trans. Automat. Control 62 (4) (2017) 1559–1574.
[56] S. Zoboli, D. Astolfi, V. Andrieu, Total stability of equilibria motivates integral action in discrete-time nonlinear systems, Automatica 155 (2023).
[57] J. Schmidhuber, Making the world differentiable: On using self-supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments, Inst. Inf. 126 (1990).
[58] M. Igl, L. Zintgraf, T.A. Le, F. Wood, S. Whiteson, Deep variational reinforcement learning for POMDPs, in: International Conference on Machine Learning, PMLR, 2018, pp. 2117–2126.
[59] S. Tarbouriech, M. Turner, Anti-windup design: an overview of some recent advances and open problems, IET Control Theory Appl. 3 (1) (2009) 1–19.
[60] E. Maljaars, F. Felici, T. Blanken, C. Galperti, O. Sauter, M. de Baar, F. Carpanese, T. Goodman, D. Kim, S. Kim, M. Kong, B. Mavkov, A. Merle, J. Moret, R. Nouailletas, M. Scheffer, A. Teplukhina, N. Vu, The EUROfusion MST1-team, The TCV-team, Profile control simulations and experiments on TCV: a controller test environment and results using a model-based predictive controller, Nucl. Fusion 57 (12) (2017) 126063.
[61] R. Brégeon, Évolution résistive du profil de courant dans les tokamaks, application à l'optimisation des d'echarges de Tore Supra (Ph.D. thesis), Université de Provence (Aix-Marseille I), CEA - Cadarache, 1998.
[62] O. Sauter, C. Angioni, Y.R. Lin-Liu, Neoclassical conductivity and bootstrap current formulas for general axisymmetric equilibria and arbitrary collisionality regime, Phys. Plasmas 6 (7) (1999) 2834–2839.
[63] E. Witrant, S. Brémond, Shape identification for distributed parameter systems and temperature profiles in tokamaks, in: 2011 50th IEEE Conference on Decision and Control and European Control Conference, 2011, pp. 2626–2631, http://dx.doi.org/10.1109/CDC.2011.6160523.