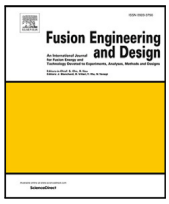
Fusion Engineering and Design 216 (2025) 115064

Contents lists available at ScienceDirect

Fusion Engineering and Design

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



Density regulation with disruption avoidance in next-generation tokamaks using a safe reinforcement learning-based controller

Sai Tej Paruchuri [a,*], Ian Ward [a], Nicholas Rist [a], Vincent Graber [a], Hassan Al Khawaldeh [a], Zibo Wang [a], Eugenio Schuster [a], Andres Pajares [b], June-Woo Juhn [c]

[a] Mechanical Engineering and Engineering Mechanics, Lehigh University, Bethlehem, PA, USA
[b] General Atomics, San Diego, CA, USA
[c] Korean Institute of Fusion Technology, Daejeon, Yuseong-gu, Republic of Korea

ARTICLE INFO

Keywords:
Density control
Density limits
Safe reinforcement learning

ABSTRACT

Achieving high particle density is desirable in fusion reactors because of its direct correlation to the fusion power. However, operational limits constrain the maximum achievable particle density. Several density control algorithms have been designed to inject particles using gas puffing and pellet injection to track carefully selected safe targets. However, a controller unaware of these operational limits may modulate the particle densities beyond the safe limits during transients caused by changing operating conditions. Designing plasma control algorithms that recognize these operational limits and ensure that the particle density remains within the safe operational space is desirable. This work focuses on the safe regulation of deuterium–tritium (DT) particle density. Under the assumption of quasi-neutrality, the total plasma density can be related to the electron density, which is constrained by the well-known Greenwald limit. To regulate the DT density within the safe operational space defined by Greenwald limit, a novel safe reinforcement learning-based controller is developed in this work. Such control-design approach can be critical in designing learning-based safe plasma control algorithms for next-generation tokamaks. The effectiveness of the controller is demonstrated through nonlinear simulations conducted over multiple test cases.

1. Introduction

Maintaining stable and optimal plasma confinement is essential for the future of fusion reactors like ITER. Such confinement requires careful regulation of different plasma properties. One such property is the plasma particle density. Achieving high plasma density is crucial for maximizing fusion power in reactors like ITER, as the fusion power output is proportional to the square of the plasma density. However, increasing plasma density comes with the risk of crossing critical stability limits, which can lead to disruptions [1,2]. Disruptions pose a significant risk to continuous fusion operations. These disruptions can lead to sudden loss of plasma confinement, damaging reactor components and causing unplanned operational downtime. To ensure the long-term viability and efficiency of fusion reactors, preventing disruptions before they occur is essential. Disruption prevention includes three strategies: disruption avoidance, soft landing, and mitigation [3–5]. Disruption avoidance focuses on proactively controlling plasma properties to prevent conditions that lead to disruptions. Soft landing involves steering the plasma into a controlled shutdown when the likelihood of disruption becomes too high. Finally, mitigation refers to actions taken after a disruption has begun or is imminent, such as injecting gas to cool the plasma or applying heating to stabilize it. While crucial, mitigation merely minimizes the impact of disruptions and does not prevent them entirely. On the other hand, regular soft-landing or shutdown is not desirable because it reduces the operational efficiency and the reactor's ability to produce continuous power. Thus, active disruption avoidance is critical to maintain reactor performance.

Disruption avoidance primarily involves tracking proximity to stability limits and taking corrective actions whenever these limits are about to be breached. One approach used for disruption avoidance relies on dynamic pulse scheduling. In this approach, the discharge program follows a discrete sequence of segments that contain reference trajectories for feedback and feedforward control. Whenever predefined conditions (such as proximity to stability limits) are met, the control system switches to a recovery segment. Experiments on ASDEX Upgrade [6–8] have shown that this approach can be effectively applied to prevent disruptions associated with the H-Mode density limit. Reference governors (proximity controllers) [9,10] offer another comparatively well-established, model-based solution with provable

* Corresponding author.
E-mail address: saitejp@lehigh.edu (S.T. Paruchuri).

https://doi.org/10.1016/j.fusengdes.2025.115064
Received 10 January 2025; Received in revised form 28 March 2025; Accepted 7 April 2025
Available online 29 April 2025

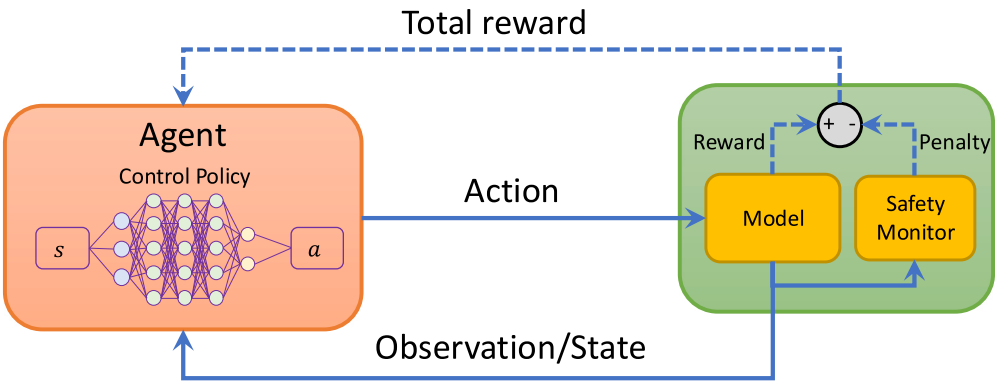


Fig. 1. Illustration of the agent-environment interaction in a typical RL framework. Note that adding a penalty term to the reward does not alter the overall structure of the RL framework.

convergence guarantees. These methods continuously adapt the plasma reference trajectories in real time based on their proximity to the stability limits, using either optimization or predefined rules. The updated reference trajectories are then fed to a feedback control system that adjusts the plasma state to track the new reference.

Another class of control algorithms directly regulates the plasma state to proactively avoid stability limits. These methods directly enforce stability limits on the plasma state and can be broadly categorized as either model-based or learning-based. A widely used model-based approach in plasma control is model predictive control (MPC) [11,12], which performs real-time optimization to determine the optimal input for achieving a desired target. In MPC, stability limits are imposed as safety constraints during the optimization. However, the computational cost of MPC can be significantly high, making it challenging to implement in the plasma control system (PCS). In contrast, a computationally inexpensive model-based approach employs control barrier functions [13,14]. Barrier functions provide a control-theoretic framework for enforcing safety constraints by employing a mathematical barrier around unsafe regions of the plasma state space. Algorithms based on these methods automatically adjust the control input to keep the system within safe operating bounds as the plasma state approaches a safety limit. Despite these advantages, barrier methods can be challenging to design when safety limits are dynamic.

Safe reinforcement learning (SRL) is a learning-based approach that provides an effective alternative to the model-based methods discussed above [15,16]. Although rigorous mathematical proofs of convergence and closed-loop stability in SRL remain challenging, and the training process can be computationally expensive, the resulting SRL-based controllers are computationally efficient at runtime and adaptable to dynamic environments. In SRL, an agent learns a policy that maximizes long-term performance while adhering to safety constraints through continuous interactions with its environment. Unlike traditional reinforcement learning, which focuses solely on optimizing cumulative rewards, SRL integrates safety criteria directly into the learning process. This work aims to leverage SRL to develop a controller that regulates the deuterium–tritium (DT) density while actively avoiding a specified density stability limit.

1.1. A brief review of density limits

One of the most well-known density limits is the Greenwald density limit [17], which correlates the maximum allowable density to the plasma current and the cross-sectional area of the plasma, given by

$$n_{e,20} \leq n_{gw} := \frac{I_{p,MA}}{\pi a^2}, \tag{1}$$

where $n_{e,20}$ is the line-averaged density in 10^{20} m^{-3}, $I_{p,MA}$ is the plasma current in MA, and a is the plasma minor radius in m. Breaching this limit increases the risk of magnetohydrodynamic (MHD) instabilities, which can ultimately trigger a disruption. In addition to the Greenwald limit, other density-related limits also play a critical role. For instance, a limited set of disruptions have been observed in some tokamaks at density levels distinct from the Greenwald limit. This may be attributed to the violation of a density limit for the plasma's edge region, as studied in [18–20]. In addition to the aforementioned limits, other density limits must be considered during the ramp-up and ramp-down phases of the discharge [21,22]. Consequently, a density regulation and limit avoidance algorithm may have to monitor multiple safety thresholds simultaneously and regulate the density so that no specific limit is approached. However, as a preliminary step, we limit the safety criterion for the controller to the Greenwald limit in this work. Due to the adaptable nature of SRL methods, this control synthesis and training methodology can be readily adapted to incorporate other density limits mentioned above.

1.2. Control objective statement

The objective of this work is to design a computationally efficient control policy μ using safe reinforcement learning (SRL) to regulate the deuterium–tritium particle density n_{DT} in the flattop phase of an ITER scenario to track a given target $\bar{n}_{DT}$ while maintaining the electron density below the Greenwald limit n_{gw} defined in (1).

1.3. Paper organization

This paper is organized as follows. Section 2 introduces safe reinforcement learning and formalizes the reinforcement learning framework. Section 3 describes the environment model. Details on training the reinforcement learning agent are provided in Section 4. Section 5 presents the testing results of the trained agent. Finally, Section 6 discusses conclusions and potential future extensions.

2. Safe reinforcement learning

2.1. Reinforcement learning framework

A standard reinforcement learning (RL) framework consists of an agent that interacts with an environment. Given a state s_t of the environment at time t, these interactions are characterized by the agent taking an action a_t and receiving a reward r_{t+1}. The action a_t results in the environment transitioning to a new state s_{t+1}. The agent's behavior is characterized by the policy $\mu : \mathcal{S} \to \mathcal{A}$, which maps states in the set $\mathcal{S}$ to actions in the set $\mathcal{A}$. In the case of a stochastic policy, μ maps each state to a probability distribution over actions rather than a specific action. The environment, on the other hand, represents a

physical system or a simulator of the physical system that we are trying to control using the policy μ. To formalize the goals of RL, we need a mathematical framework to model the environment. A Markov decision process (MDP) allows this formalization. An MDP model consists of the tuple $(\mathcal{S}, \mathcal{A}, p, r)$, where $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ represents the transition dynamics function, and $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ represents the reward function. The transition dynamics function p gives the probability of the environment evolving to the state s_{t+1} from the state s_t when action a_t is taken. The reward function r gives the reward r_{t+1} obtained for transitioning from state s_t to s_{t+1} when action a_t is taken, i.e., $r_{t+1} = r(s_t, a_t, s_{t+1})$. A more rigorous discussion on MDPs can be found in [23].

The goal of the RL agent is to take actions to maximize the expected value of cumulative rewards. This goal is formalized using the notion of return G_t, defined as

$$G_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-t-1} r_T = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}, \tag{2}$$

where γ is the discount factor and T is the final time step. Notice that the return G_t depends on the initial state s_t and the sequence of actions taken at subsequent time steps as the system evolves. Thus, the goal of the RL agent is to select actions such that the expected value of G_t is maximized.

2.2. Incorporating safety into RL framework

Mathematically, density stability limits in tokamaks can be implemented as inequality safety constraints on the state s_t at time t. These constraints may be static, but most constraints in tokamaks are dynamic in nature, meaning the inequality relations vary with time. An example representation of a dynamic safety constraint is given by

$$c(t, s_t) \geq 0, \tag{3}$$

where c is a nonlinear function. During SRL agent training, safety constraints can be enforced in three ways: (i) hard constraints, which require strict satisfaction of safety limits; (ii) chance constraints, which specify that safety constraints must be satisfied with a predetermined probability level; and (iii) soft constraints, which incorporate safety into the reward function by penalizing unsafe actions, thereby discouraging them without strictly enforcing compliance [15,16].

In this work, we impose density instability limits as soft constraints during agent training. This approach provides a practical mechanism for developing controllers that prioritize safe actions without substantially increasing computational complexity. Moreover, soft constraints preserve the underlying Markov Decision Process (MDP) structure, allowing us to leverage commonly used model-free RL algorithms for training the agent while incorporating stability limits. Furthermore, numerical simulations discussed in Section 5 demonstrate that the agent trained with soft constraints strictly adheres to the density limit in all test cases.

The standard RL formalization, discussed in Section 2.1, does not inherently address safety constraints. In this work, we incorporate soft safety constraints into the RL framework by introducing a penalty term p_t to the reward generated by the environment. This penalty is set to be approximately zero when the state is within the safety limits but takes on negative values as the state approaches the limits, effectively discouraging unsafe actions. Fig. 1 gives an illustration of this RL framework. With the addition of this penalty term, the RL agent's objective becomes maximizing the expected value of the modified cumulative return $\hat{G}_t$, given by

$$\hat{G}_t = \sum_{k=0}^{T-t-1} \gamma^k \hat{r}_{t+k+1}, \tag{4}$$

where $\hat{r}_i = r_i + p_i$ for $i = t, \ldots, T$.

3. Environment model

In this work, a simulator based on a nonlinear zero-dimensional (0D) model of plasma particles is used as the environment. Variants of this model have previously been explored for plasma operation contour (POPCON) analysis and control design in [24–27]. The density response models are given by

$$\dot{n}_\alpha = -\frac{n_\alpha}{\tau_\alpha} + S_\alpha, \tag{5}$$

$$\dot{n}_{DT} = -\frac{n_{DT}}{\tau_{DT}} - S_\alpha + S_{DT} + S^R_{DT}, \tag{6}$$

$$\dot{n}_I = -\frac{n_I}{\tau_I} + S^{sp}_I, \tag{7}$$

$$\dot{n}_s = -\frac{n_s}{\tau_s} + S_s, \tag{8}$$

where n_{DT} is the deuterium–tritium (DT) density, n_α is the alpha particle density, n_I is the (uncontrolled) wall impurity density, and n_s is the (controlled) seeded impurity density. The above model assumes that the deuterium and tritium densities are equal, and the term n_{DT} represents the sum of deuterium and tritium densities. The terms τ_α, τ_{DT}, τ_I and τ_s are alpha particle, DT, impurity and seeded impurity confinement times, respectively. The terms S_{DT} and S_s denote the DT injection rate and seeded impurity injection rate, respectively. These parameters serve as inputs to regulate DT density; in other words, they are the actions prescribed by the controller (actor). Note that the controller in this work is designed to be agnostic to specific physical actuators. The prescribed injection rates can be converted into actuator commands for gas puffing and pellet injection systems, depending on the actuator availability in a given plasma discharge, using an actuator-allocation algorithm [24,26,28,29]. The term S^R_{DT} is the DT recycling rate and is given by

$$S^R_{DT} = \frac{f_{eff}}{1 - f_{ref}(1 - f_{eff})} \left\{ f_{ref} \frac{n_{DT}}{\tau_{DT}} + \left(\frac{n_{DT}}{\tau_{DT}} \right) \times \left[\frac{(1 - f_{ref}(1 - f_{eff}))R^{eff}}{1 - R^{eff}(1 - f_{eff})} - f_{ref} \right] \right\}, \tag{9}$$

where f_{ref} is the fraction of escaping particles that are reflected back into the plasma, f_{eff} is the factor that accounts for the fueling efficiency of the recycled particles, and R^{eff} is the global recycling coefficient.

The source of the wall impurities n_I is the impurity sputtering S^{sp}_I, which is caused by plasma wall interactions, and is modeled using the relation

$$S^{sp}_I = f^{sp}_I \left(\frac{n}{\tau_I} + \dot{n} \right), \tag{10}$$

where $n = n_i + n_e$ is the total plasma density. The total ion density n_i and electron density n_e are given by

$$n_i = n_{DT} + n_\alpha + n_I + n_s, \tag{11}$$

$$n_e = n_{DT} + 2n_\alpha + Z_I n_I + Z_s n_s, \tag{12}$$

where Z_I and Z_s are the average atomic numbers of the wall and seeded impurities.

The term S_α in (5) is the DT fusion reaction rate and is given by

$$S_\alpha = \frac{n^2_{DT}}{4} \langle \sigma v \rangle, \tag{13}$$

where the DT reactivity $\langle \sigma v \rangle$ is computed using the expression

$$\langle \sigma v \rangle = C_1 T_i \omega \sqrt{\frac{\xi}{m_r c^2 T_i^3}} e^{-3\xi} \times 10^{-6}, \tag{14}$$

$$\xi = \left(\frac{B^2_G}{4\omega} \right)^{\frac{1}{3}}, \tag{15}$$

$$\omega = \left[1 - \frac{T_i(C_2 + T_i(C_4 + T_i C_6))}{1 + T_i(C_3 + T_i(C_5 + T_i C_7))} \right]^{-1}. \tag{16}$$

In the above expressions, T_i is the ion temperature in keV. The terms B_G, $m_r c^2$ and C_j for $j = 1, \ldots, 7$ are constants [30].

As evident from (14)–(16), the DT reactivity depends on the ion temperature T_i, which is computed from the relation $E_i = \frac{3}{2} n_i T_i$, where E_i is the ion energy. The electron energy E_e and the ion energy E_i are governed by 0D equations of the form

$$\dot{E}_e = -\frac{E_e}{\tau_{E_e}} + (1-\phi_\alpha)P_\alpha - P_{ei} - P_{br} + P_{oh} + P_{aux,e}, \tag{17}$$

$$\dot{E}_i = -\frac{E_i}{\tau_{E_i}} + \phi_\alpha P_\alpha + P_{ei} + P_{aux,i}, \tag{18}$$

where P_α, P_{ei}, P_{br} and P_{oh} are the alpha particle power from fusion, ion–electron collisional power exchange, bremsstrahlung radiation losses and ohmic heating power, respectively. The term ϕ_α is the fraction of P_α deposited into the plasma ions. The terms τ_{E_e} and τ_{E_i} are the electron and ion energy confinement times. The power deposited into the electrons and ions by external sources are $P_{aux,e}$ and $P_{aux,i}$, respectively.

The alpha particle power from fusion is computed as $P_\alpha = Q_\alpha S_\alpha$, where $Q_\alpha = 3.52$ MeV is the kinetic energy of each alpha particle produced through fusion. The bremsstrahlung radiation losses and Ohmic heating power are modeled by

$$P_{br} = 5.5 \times 10^{-37} Z_{eff} n_e^2 \sqrt{T_{e,keV}}, \tag{19}$$

$$P_{oh} = 2.8 \times 10^{-9} Z_{eff} I_p^2 a^{-4} T_{e,keV}^{-\frac{3}{2}}, \tag{20}$$

where I_p is the plasma current, a is the plasma minor radius, $T_{e,keV}$ is the electron temperature T_e expressed in keV. Note that the electron temperature is related to electron energy and density through the relation $E_e = \frac{3}{2} n_e T_e$. The term Z_{eff} is the effective atomic number given by

$$Z_{eff} = \frac{n_{DT} + 4n_\alpha + Z_I^2 n_I + Z_s^2 n_s}{n_e}. \tag{21}$$

The power exchanged through ion–electron collisions are given by

$$P_{ei} = \frac{3}{2} n_e \frac{T_e - T_i}{\tau_{ei}}, \tag{22}$$

where the relaxation time τ_{ei} can be computed using the equation

$$\tau_{ei} = \frac{3\pi\sqrt{2\pi}\epsilon_0^2 T_e^{\frac{3}{2}}}{e^4 \sqrt{m_e}\Lambda_e} \sum_{ions} \frac{m_i}{n_i Z_i^2} \tag{23}$$

with $e = 1.622 \times 10^{-19}$ C, $m_e = 9.1096 \times 10^{-31}$ kg, and $\epsilon_0 = 8.854 \times 10^{-12}$ F/m, and

$$\Lambda_e = 1.24 \times 10^7 T_{e,K}^{3/2} / (n_e^{1/2} Z_{\text{eff}}^2), \tag{24}$$

where $T_{e,K}$ represents T_e expressed in kelvin.

4. Reinforcement learning agent training

The discussion in the above sections focused on formalizing the elements of reinforcement learning and establishing the environment model. This section reviews the specific steps taken to train the RL agent, including algorithm selection, network architecture, observation and action space definition, reward design, and training procedure.

4.1. Training algorithm selection

Since the inclusion of safety constraints in the RL architecture did not alter the MDP framework, any model-free reinforcement learning algorithm that can handle continuous states and actions can be used to train an agent capable of regulating density without breaching safety limits. In this work, the Deep Deterministic Policy Gradient (DDPG) algorithm was chosen for training the agent [31]. DDPG is particularly well-suited for continuous control tasks, such as the one considered in this work. It combines *deterministic policy* learning with an actor–critic structure, consisting of two neural networks: an actor network that represents the control policy and a critic network that estimates the expected return of taking an action in a given state and subsequently following the current policy. The actor prescribes actions at a given state, while the critic evaluates the quality of these actions based on the estimated expected return. Training involves collecting data through interactions with the environment and updating both networks. While the detailed steps of the DDPG algorithm are beyond the scope of this work, they can be found in [31]. Once training is complete, the actor network is deployed in the plasma control system (PCS) or in a simulator to function as the density controller.

4.2. State abstraction, plasma parameters, and reward function

The environment model presented in Section 3 is a continuous-time dynamical system governed by ordinary differential equations (ODEs). To enable the use of RL algorithms, the ODEs are discretized using the forward Euler method. The resulting discrete-time model consists of six plasma states $s_t = [n_{\alpha,t}, n_{DT,t}, n_{I,t}, n_{S,t}, E_{I,t}, E_{e,t}]^T$, where the subscript t denotes the discrete time-step index. To ensure the agent has complete information about the environment, s_t must be provided to the agent along with the target state $\bar{n}_{DT}$ and the Greenwald limit $n_{gw,t}$ specified in (1). However, the objective of this work is to regulate the DT density $n_{DT,t}$ to track $\bar{n}_{DT}$ while ensuring that $n_{e,t}$ remains below $n_{gw,t}$. Thus, the training process can be simplified by aggregating the states and reducing the effective state space through a mapping function $\phi : [s_t^T, \bar{n}_{DT,t}, n_{gw,t}]^T \mapsto \phi(s_t) = [n_{DT,t} \times 10^{-19}, \bar{n}_{DT} \times 10^{-19}, s_{e,t}]^T$, where $s_{e,t} = n_{e,t} \times 10^{-20} - c_s n_{gw,t}$ is a state that tracks safety limit violations. Here, c_s is a safety margin coefficient. Note that the states $n_{DT,t}, \bar{n}_{DT}$ are scaled by 10^{-19} for normalization.

In the simulations, the safety margin coefficient was set to $c_s = 0.9$. Targets $\bar{n}_{DT}$ were randomly selected from a uniform distribution over the range $[6.0 \times 10^{19}, 12 \times 10^{19}]\,\text{m}^{-3}$ and remained constant throughout each episode. The initial $n_{DT,0}$ value was uniformly selected from the discrete set $\{5.75 \times 10^{19}, 8.75 \times 10^{19}\}\,\text{m}^{-3}$. The other initial plasma states were set to the following in all the episodes: $n_{\alpha,0} = 2.0 \times 10^{18}\,\text{m}^{-3}$, $n_{I,0} = 1 \times 10^{18}\,\text{m}^{-3}$, $n_{S,0} = 1 \times 10^{18}\,\text{m}^{-3}$, $E_{e,0} = 1.8 \times 10^5$, and $E_{i,0} = 1.5 \times 10^5$. Note that the initial conditions were selected to ensure that the Greenwald limit was not exceeded at $t = 0$. The deuterium–tritium injection rate $S_{DT,t}$ and seeded impurity injection rate $S_{s,t}$were used as inputs. The actuators $S_{DT,t}$ and $S_{s,t}$ were constrained by saturation limits of 2.0×10^{19} and 2.0×10^{17}, respectively, and both were normalized to vary between 0 and 1.

The plasma parameters used in the simulations correspond to the flattop phase of an ITER scenario. For each episode, the plasma current I_P and minor radius a were randomly selected from uniform distributions over [14, 16] MA and [1.9, 2.1] m, respectively, and remained constant throughout the episode. Additionally, the wall impurities and seeded impurities were modeled as Beryllium ($Z_I = 4$) and Helium-3 ($Z_s = 2$) ions, respectively. The fraction of $P_{\alpha,t}$ deposited into the plasma ions was assumed to be $\phi_\alpha = 0.15$. The energy confinement time $\tau_{E,t}$ was calculated using the IPB98(y,2) scaling law [32], given by

$$\tau_{E,t} = H_H \times 0.05621 \left(\frac{I_p}{10^6}\right)^{0.93} B_T^{0.15} M^{0.19} \times R^{1.97} \epsilon^{0.58} \kappa^{0.78} P_t^{-0.69} V^{-0.69} n_{e_{19}}^{0.41}, \tag{25}$$

where $H_H = 1$ is the H-factor, $\epsilon = a/R$, $R = 6.2$ m is the major radius, $B_T = 5.3\ T$ is the toroidal magnetic field, $V = 837\,\text{m}^3$ is the plasma volume, and κ is the vertical elongation at the 95% flux surface. The constant M is given by $M = 3\gamma + 2(1-\gamma)$, where $\gamma = 0.5$ represents the tritium fraction. The term P_t is the total plasma power in MW and is computed as $P_t = P_{aux,t} - P_{br,t} + P_{\alpha,t} + P_{oh,t}$. During simulations, both $P_{aux,e,t}$ and $P_{aux,i,t}$ were set to 50 MW across all episodes. The

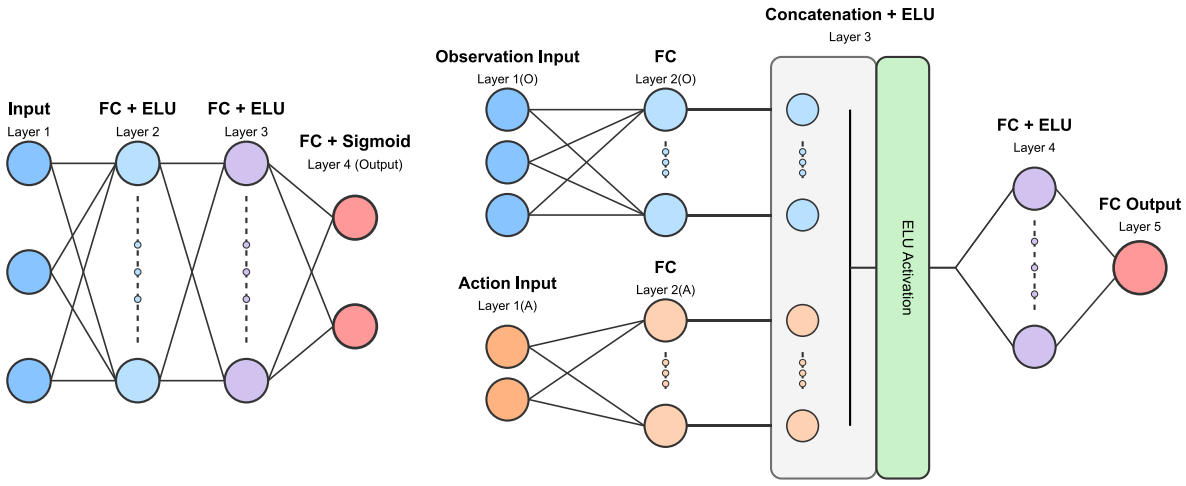


Fig. 2. Actor (left) and critic (right) network architectures (FC: fully connected layer; ELU: exponential linear unit).

other confinement times were scaled versions of $\tau_{E,t}$, with the ion and electron confinement times given by $\tau_{E,i,t} = \tau_{E,t} \times \zeta_i$ and $\tau_{E,e,t} = \tau_{E,t} \times \zeta_e$, where $\zeta_i = 1.1$ and $\zeta_e = 0.9$. Confinement times for deuterium–tritium particles, wall impurities, seeded impurities, and alpha particles were calculated as $\tau_{DT,t} = \tau_{E,t} \times k_{DT}$, $\tau_{I,t} = \tau_{E,t} \times k_I$, $\tau_{s,t} = \tau_{E,t} \times k_s$, and $\tau_{\alpha,t} = \tau_{E,t} \times k_\alpha$, with $k_{DT} = 2.5$, $k_I = 6$, $k_s = 6$, and $k_\alpha = 4$.

The reward function was selected as

$$r_{t+1} = \frac{1}{3}\left(\exp\left(-\frac{\hat{n}_{t+1}^2}{2\sigma_1^2}\right) + \exp\left(-\frac{\hat{n}_{t+1}^2}{2\sigma_2^2}\right) + \exp\left(-\frac{\hat{n}_{t+1}^2}{2\sigma_3^2}\right)\right) - q_s\left(\frac{{S_{DT,t}^*}^2 + {S_{s,t}^*}^2}{{\bar{S}_{DT}^*}^2 + {\bar{S}_s^*}^2}\right), \tag{26}$$

where $\hat{n}_{t+1} = (n_{DT,t+1} - \bar{n}_{DT})/\bar{n}_{DT}$, $\sigma_1 = 0.25$, $\sigma_2 = 0.05$, $\sigma_3 = 0.005$, and $q_s = 0.25$. In the above equation, the terms $S_{DT,t}^*$ and $S_{s,t}^*$ represent the normalized inputs and $\bar{S}_{DT}^* = 1$ and $\bar{S}_s^* = 1$ represent their upper limits, respectively. The exponential terms in the reward function penalize deviation from the target. Note that three exponential functions with decreasing variances allow the reward function to remain sensitive to both large and small deviations from the target. On the other hand, the final term in the reward function penalizes large normalized input values relative to their saturation limits. To penalize crossing of the Greenwald limit, an inverted sigmoid function with negative output was used as the penalty function, defined as

$$p_{t+1} = -\frac{1}{1 + \exp(-\alpha s_{e,t+1})}, \tag{27}$$

where $\alpha = 100$ is a smoothness parameter that controls the steepness of the penalty.

4.3. Agent architecture and training parameters

The architectures of the actor and critic neural networks used in the simulations are shown in Fig. 2. In both the networks, each hidden layer contains 256 neurons and uses the exponential linear unit (ELU) activation function. The actor network's final output layer includes a sigmoid activation function to account for input saturation. Both networks were trained using the Adam optimizer [33] with a learning rate of 1×10^{-3}.

The DDPG algorithm settings used during training are summarized in Table 1. As DDPG employs a deterministic policy, Ornstein–Uhlenbeck (OU) noise [31,34] with a mean of 0, a standard deviation of 0.1 and a mean reversion rate of 0.15 was added to the actions to encourage exploration of the state space by the agent.

To compare the effectiveness of the proposed method, two different agents were trained over 2000 episodes. For the first agent (Agent 1), safety violations were not penalized during training. In other words, the value of p_{t+1} was set to 0 irrespective of the value of $s_{e,t}$. On the other hand, the second agent (Agent 2) was penalized for safety violations as per (27). In both the cases, each episode lasted 20 s with a step size of 0.1 s. The moving average of training episodic rewards, computed with a window size of 15 episodes, is shown in Fig. 3. To evaluate performance without exploration noise, the agents were tested in a noise-free environment every 50 episodes during training, with each test spanning 10 episodes. In the first and second training sessions, the agents from the 1200th and 750th episodes, respectively, achieved the highest average cumulative returns during noise-free evaluations. Hence, they were selected for final testing.

Table 1
DDPG agent options.

Parameter	Value	Description
Discount factor	0.99	Factor to balance immediate and future rewards
Experience buffer length	1×10^6	Size of the replay buffer for storing experiences
Mini-batch size	64	Number of experiences sampled per training step
No. of warm start steps	1024	Number of samples to be collected before learning
Target smooth factor	1×10^{-3}	Factor to control target networks updates
Target update frequency	1	Controls how often target networks are updated

5. Reinforcement learning agent testing

The trained agents were tested extensively in closed-loop control simulations across 100 different test cases, each with a randomly selected $\bar{n}_{DT}$, $n_{DT,0}$, plasma current I_p and minor radius a. Figs. 4, 5, 6, and 7 present the simulation results. The subscript t, representing the time-step index, has been omitted in the figures and the following analysis for notational brevity. Fig. 4 shows the range of state trajectories obtained from these 100 test cases, along with sample trajectories and corresponding targets from two representative cases. As evident from the figure, the controller corresponding to Agent 1 closely tracks the target in both sample test cases. However, the safe controller corresponding to Agent 2 fails to track the target in one of the two cases. Furthermore, Agent 2 results in a lower maximum n_{DT} value than Agent 1. Subsequent analysis shows that this closed-loop evolution arises from the controller prioritizing safety limit avoidance over target tracking.

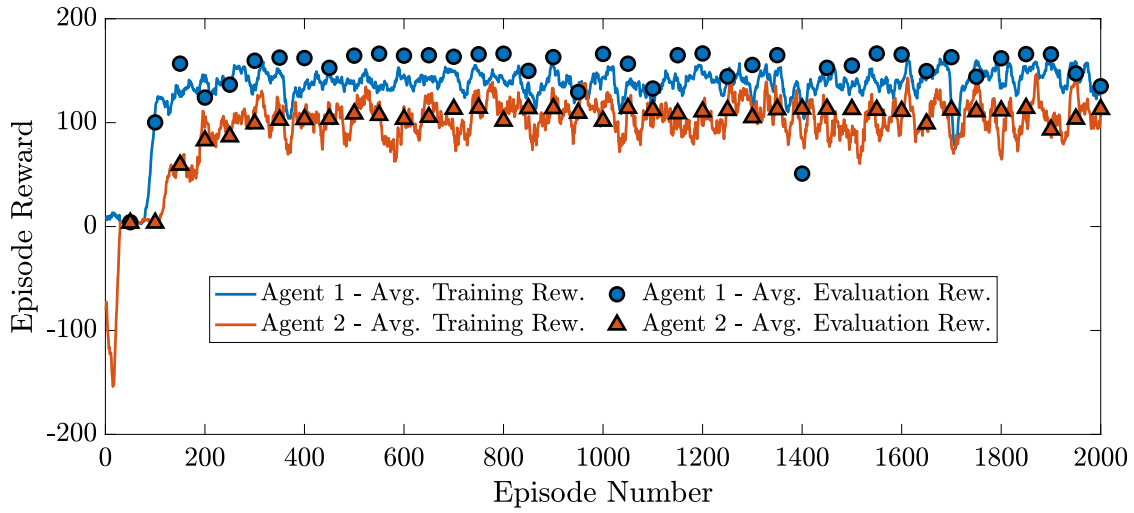


Fig. 3. Moving average of rewards obtained per episode during training, with average evaluation rewards shown every 50 episodes.

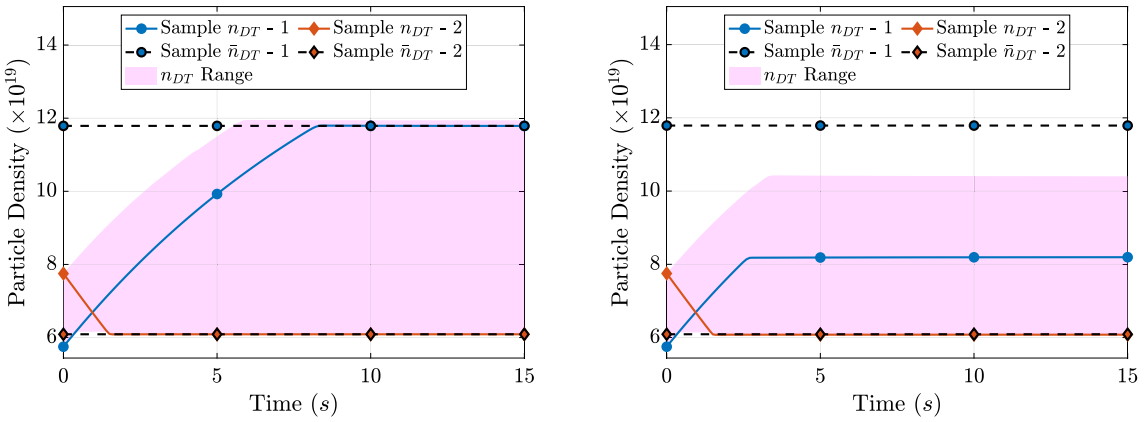


Fig. 4. Variation of deuterium–tritium density n_{DT} across 100 test cases, with two representative n_{DT} trajectories and their corresponding target trajectories $\bar{n}_{DT}$. The left subfigure shows results for Agent 1, and the right for Agent 2. Agent 1 achieves higher n_{DT} values overall, as seen from the range of trajectories. In the sample cases, Agent 1 successfully tracks the target in both instances, whereas Agent 2 fails to do so in one case.

The controllers' performances across all test cases can be better evaluated using Fig. 5, which displays both the error trajectory for the sample cases and the range of tracking errors $\tilde{n}_{DT} = n_{DT} - \bar{n}_{DT}$ for the 100 test cases. The figure indicates that the error converges to the origin regardless of the initial error when Agent 1 is used. In contrast, Agent 2's controller exhibits non-zero errors in some cases due to its prioritization of safety constraints.

Fig. 6 shows the range of actions prescribed by the neural network controller. As expected, both controllers do not rely on the seeded impurity source S_s^* to regulate n_{DT}. It is worth noting that a high seeded impurity injection could indirectly increase the DT density. This occurs because impurity injection reduces the plasma temperature, thereby lowering DT reactivity and decreasing S_α in (6). Consequently, the DT density may increase at the expense of fusion power, indicating that the seeded impurity injection rate indirectly influences the controllability of n_{DT}. However, this effect is minimal due to the disparity in actuator saturation limits—the seeded impurity source has a saturation limit two orders of magnitude lower than that of S_{DT}^*. Additionally, the reward function in (26) penalizes high values of the normalized inputs. As a result, both controllers avoid using S_s^* to regulate n_{DT}.

Fig. 6 also illustrates the range of S_{DT}^* values prescribed by both controllers. Notably, S_{DT}^* remains saturated for a longer duration when Agent 1 is used compared to Agent 2. This behavior can be attributed to the fact that Agent 1 does not account for the safety limit during training. As a result, when high target densities are selected, the controller corresponding to Agent 1 continues injecting DT particles to meet the target. In contrast, the safe controller corresponding to Agent 2 reduces the injection rate to avoid violation of the safety limit, which in some cases leads to degraded tracking performance.

The prioritization of Greenwald limit avoidance by Agent 2 is evident from Fig. 7. The figure shows the range of s_e trajectories and corresponding Greenwald limit values across the 100 test cases. Note that, since the plasma current and minor radius are randomly selected in each episode, the Greenwald limit varies and is not constant. When Agent 1 is used, the s_e trajectories breach the Greenwald limit in some cases, whereas they remain below the safety margin in all cases when Agent 2 is used. Note that the safety margin $s_e = 0$ corresponds to the center of the inverted sigmoid penalty function defined in (27). Since the penalty increases as s_e approaches zero, Agent 2 learned to avoid breaching the safety margin to maximize the episodic reward.

Fig. 7 also shows the s_e trajectories for the two sample test cases considered in the previous figures. In Sample Trajectory 1, the Greenwald limit is clearly violated when Agent 1 is used. In contrast, when Agent 2 is used, the state s_e increases toward the safety margin but

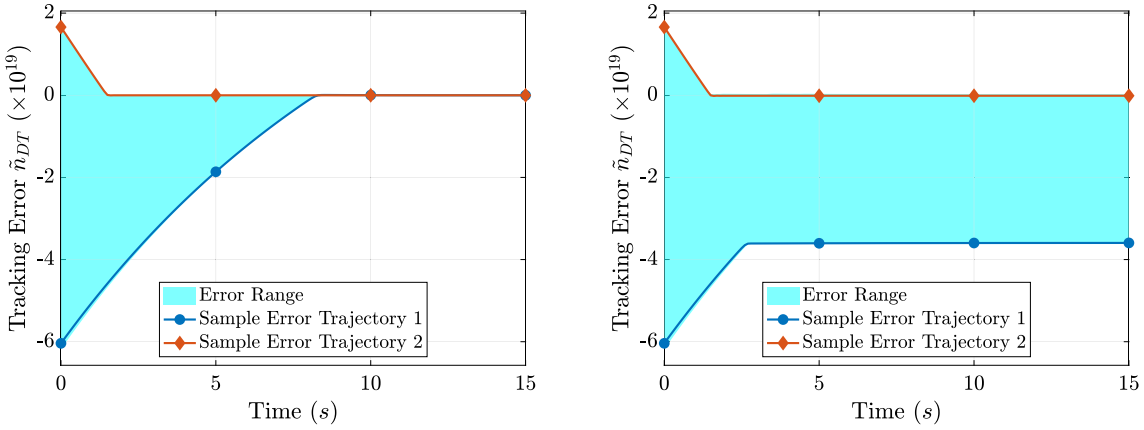


Fig. 5. Variation of the tracking error $\tilde{n}_{DT}$ across 100 test cases, with two representative error trajectories shown for illustration. The left subfigure shows results for Agent 1, and the right for Agent 2. Agent 1 achieves zero tracking error in all test cases, while Agent 2 exhibits nonzero error in some cases. Sample Error Trajectory 1 in the right subfigure highlights a case where Agent 2 fails to achieve zero tracking error.

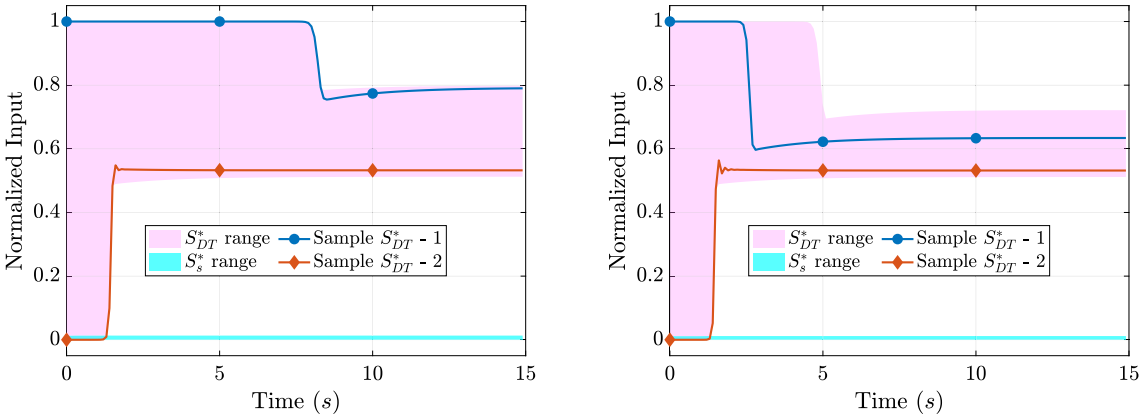


Fig. 6. Variation of normalized actions S^*_{DT} and S^*_s across 100 test cases, with two sample S^*_{DT} trajectories shown for illustration. The left subfigure shows results for Agent 1, and the right for Agent 2. As expected, neither agent relies on S^*_s for regulating n_{DT}. The extended saturation of S^*_{DT} observed in the left subfigure is consistent with the higher n_{DT} values achieved by Agent 1 in Fig. 4.

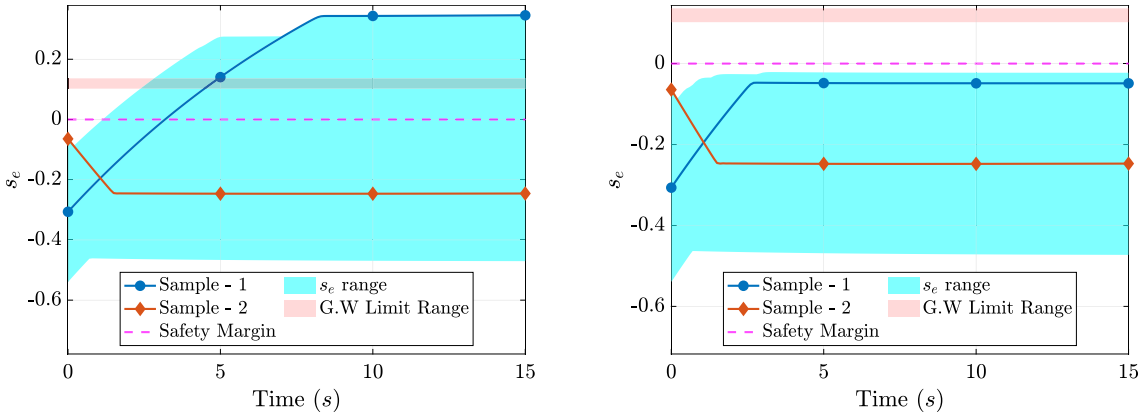


Fig. 7. Variation of the safety limit violation state s_e and the Greenwald limit across 100 test cases, with two sample s_e trajectories shown for illustration. The left subfigure shows results for Agent 1, and the right for Agent 2. Agent 1 clearly violates the Greenwald limit in some cases, as seen in the left subfigure. In contrast, Agent 2 ensures that s_e remains below the Greenwald limit in all test cases, reflecting its prioritization of density limit avoidance over tracking which is consistent with observations in Figs. 4 and 5.

stabilizes once the margin is reached. The transition point in the trajectory corresponds to the time when n_{DT} reaches a steady value in Fig. 4 and the input begins to drop below the saturation level in Fig. 6.

6. Conclusion

This work presents a density controller designed through reinforcement learning to inherently avoid violations of stability limits. While the controller in this study focused on avoiding the Greenwald limit during target DT density tracking, the proposed framework could incorporate other plasma density limits. Safety limits were integrated into the RL algorithm through a penalty term, preserving the MDP structure and enabling the use of model-free RL algorithms such as DDPG. The DDPG algorithm was employed to train the agent, which was then extensively tested in closed-loop simulations. The simulation results indicate that this approach is promising for developing safe feedback controllers for density regulation. Notably, soft safety constraints introduced through reward shaping were sufficient to train an agent that avoided Greenwald limit violations. The preliminary findings of this work could lay the groundwork for developing advanced plasma density and burn control strategies. Future research directions include extending this RL framework to (i) incorporate multiple safety limits, (ii) independently regulate deuterium and tritium densities, (iii) integrate particle density regulation with energy control, and (iv) ensure safe density regulation during the ramp-up and ramp-down phases of the discharge. In parallel, efforts could also focus on testing the proposed controller in existing tokamaks operating with deuterium plasmas. For validation, the agent will be retrained using an environment model that governs the evolution of deuterium density instead of DT density, and its actor network will eventually be deployed in the plasma control system (PCS). Measurement noise may degrade the performance of the actor network during closed-loop empirical testing. To address this, the agent will be evaluated for robustness in the presence of measurement noise, and, if necessary, additional filtering blocks will be incorporated into the closed-loop feedback architecture to mitigate its effects.

CRediT authorship contribution statement

Sai Tej Paruchuri: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Ian Ward:** Writing – review & editing, Visualization, Software, Methodology. **Nicholas Rist:** Writing – review & editing, Visualization, Software, Methodology. **Vincent Graber:** Writing – review & editing, Methodology, Formal analysis. **Hassan Al Khawaldeh:** Software, Methodology, Formal analysis. **Zibo Wang:** Software, Methodology. **Eugenio Schuster:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition. **Andres Pajares:** Formal analysis. **June-Woo Juhn:** Formal analysis.

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.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Fusion Energy Sciences, under Award Numbers DE-SC0010661 and DE-SC0010537.

Data availability

No data was used for the research described in the article.

References

[1] N. Ohyabu, Density limit in tokamaks, Nucl. Fusion 19 (11) (1979) 1491, http://dx.doi.org/10.1088/0029-5515/19/11/008.
[2] M. Greenwald, Density limits in toroidal plasmas, Plasma Phys. Control. Fusion 44 (8) (2002) R27, http://dx.doi.org/10.1088/0741-3335/44/8/201.
[3] E. Strait, et al., Progress in disruption prevention for ITER, Nucl. Fusion 59 (11) (2019) 112012, http://dx.doi.org/10.1088/1741-4326/ab15de.
[4] N.W. Eidietis, Prospects for disruption handling in a tokamak-based fusion reactor, Fusion Sci. Technol. 77 (7–8) (2021) 738–744, http://dx.doi.org/10.1080/15361055.2021.1889919.
[5] M. Lehnen, et al., Disruptions in ITER and strategies for their control and mitigation, PLASMA- SURFACE INTERACTIONS 21 463 (2015) 39–48, http://dx.doi.org/10.1016/j.jnucmat.2014.10.075.
[6] B. Sieglin, M. Maraschek, A. Gude, F. Klossek, F. Felici, M. Bernert, O. Kudlacek, A. Pau, W. Treutterer, the ASDEX Upgrade Team, the EUROfusion WPTE Team, Disruption avoidance and investigation of the H-Mode density limit in ASDEX Upgrade, Plasma Phys. Control. Fusion 66 (2) (2023) 025004, http://dx.doi.org/10.1088/1361-6587/ad163a.
[7] B. Sieglin, M. Maraschek, A. Gude, F. Felici, F. Klossek, O. Kudlacek, P. Lang, A. Pau, B. Ploeckl, W. Treutterer, Dynamic pulse scheduling in ASDEX Upgrade: Disruption avoidance and investigation of the H-Mode density limit, Fusion Eng. Des. 191 (2023) 113546, http://dx.doi.org/10.1016/j.fusengdes.2023.113546.
[8] B. Sieglin, M. Maraschek, O. Kudlacek, A. Gude, W. Treutterer, M. Kölbl, A. Lenz, Rapid prototyping of advanced control schemes in ASDEX Upgrade, Fusion Eng. Des. 161 (2020) 111958, http://dx.doi.org/10.1016/j.fusengdes.2020.111958.
[9] A. Pajares, et al., A reference governor for plasma-shape control in tokamaks, in: 2023 IEEE Conference on Control Technology and Applications, CCTA, 2023, pp. 804–809, http://dx.doi.org/10.1109/CCTA54093.2023.10253265.
[10] J. Barr, et al., Development and experimental qualification of novel disruption prevention techniques on DIII-D, Nucl. Fusion 61 (12) (2021) 126019, http://dx.doi.org/10.1088/1741-4326/ac2d56.
[11] Y. Ou, et al., Receding-horizon optimal control of the current profile evolution during the ramp-up phase of a tokamak discharge, Control Eng. Pract. 19 (1) (2011) 22–31, http://dx.doi.org/10.1016/j.conengprac.2010.08.006.
[12] 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, http://dx.doi.org/10.1088/0029-5515/55/2/023001.
[13] A.D. Ames, et al., Control barrier functions: Theory and applications, in: 2019 18th European Control Conference, ECC, 2019, pp. 3420–3431, http://dx.doi.org/10.23919/ECC.2019.8796030.
[14] B. Li, et al., A survey on the control Lyapunov function and control barrier function for nonlinear-affine control systems, IEEE/ CAA J. Autom. Sin. 10 (3) (2023) 584–602, http://dx.doi.org/10.1109/JAS.2023.123075.
[15] S. Gu, et al., A review of safe reinforcement learning: Methods, theories, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 46 (12) (2024) 11216–11235, http://dx.doi.org/10.1109/TPAMI.2024.3457538.
[16] L. Brunke, et al., Safe learning in robotics: From learning-based control to safe reinforcement learning, Annu. Rev. Control. Robot. Auton. Syst. 5 (1) (2022) 411–444, http://dx.doi.org/10.1146/annurev-control-042920-020211.
[17] M. Greenwald, et al., A new look at density limits in tokamaks, Nucl. Fusion 28 (12) (1988) 2199, http://dx.doi.org/10.1088/0029-5515/28/12/009.
[18] M. Giacomin, et al., First-principles density limit scaling in tokamaks based on edge turbulent transport and implications for ITER, Phys. Rev. Lett. 128 (18) (2022) 185003, http://dx.doi.org/10.1103/PhysRevLett.128.185003.
[19] M. Bernert, et al., The H-mode density limit in the full tungsten ASDEX Upgrade tokamak, Plasma Phys. Control. Fusion 57 (1) (2014) 014038, http://dx.doi.org/10.1088/0741-3335/57/1/014038.
[20] A.D. Maris, et al., Correlation of the L-mode density limit with edge collisionality, Nucl. Fusion 65 (1) (2024) 016051, http://dx.doi.org/10.1088/1741-4326/ad90f0.
[21] J.W. Berkery, et al., Density limits as disruption forecasters for spherical tokamaks, Plasma Phys. Control. Fusion 65 (9) (2023) 095003, http://dx.doi.org/10.1088/1361-6587/ace476.
[22] T. Ravensbergen, et al., Density control in ITER: An iterative learning control and robust control approach, Nucl. Fusion 58 (1) (2017) 016048, http://dx.doi.org/10.1088/1741-4326/aa95ce.
[23] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[24] V. Graber, E. Schuster, Nonlinear burn control in ITER using adaptive allocation of actuators with uncertain dynamics, Nucl. Fusion 62 (2) (2022) 026016, http://dx.doi.org/10.1088/1741-4326/ac3cd8.
[25] V. Graber, E. Schuster, Nonlinear adaptive burn control of two-temperature tokamak plasmas, in: 2019 IEEE 58th Conference on Decision and Control, CDC, 2019, pp. 3239–3244, http://dx.doi.org/10.1109/CDC40024.2019.9029943.

[26] V. Graber, E. Schuster, Nonlinear adaptive burn control and optimal control allocation of over-actuated two-temperature plasmas, in: 2020 American Control Conference, ACC, 2020, pp. 1411–1416, http://dx.doi.org/10.23919/ACC45564.2020.9147885.

[27] J.J. Martinell, J.E. Vitela, An optimal burn regime in a controlled tokamak fusion power plant, IEEE Trans. Plasma Sci. 44 (3) (2016) 296–305, http://dx.doi.org/10.1109/TPS.2016.2521884.

[28] S.T. Paruchuri, V. Graber, A. Pajares, E. Schuster, Static actuator-sharing algorithm for concurrent control of multiple plasma properties, Plasma Phys. Control. Fusion 67 (1) (2024) 015005, http://dx.doi.org/10.1088/1361-6587/ad8ef2.

[29] S.T. Paruchuri, V. Graber, H.A. Khawaldeh, E. Schuster, Dynamic actuator allocation via reinforcement learning for concurrent plasma control objectives, IEEE Trans. Plasma Sci. 52 (9) (2024) 4140–4146, http://dx.doi.org/10.1109/TPS.2024.3377137.

[30] H.-S. Bosch, G. Hale, Improved formulas for fusion cross-sections and thermal reactivities, Nucl. Fusion 32 (4) (1992) 611, http://dx.doi.org/10.1088/0029-5515/32/4/I07.

[31] T.P. Lillicrap, et al., Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.

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

[33] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017, arXiv:1412.6980.

[34] G.E. Uhlenbeck, L.S. Ornstein, On the theory of the Brownian motion, Phys. Rev. 36 (1930) 823–841, http://dx.doi.org/10.1103/PhysRev.36.823.