

<https://doi.org/10.1038/s42005-025-02146-6>

Active ramp-down control and trajectory design for tokamaks with neural differential equations and reinforcement learning



Allen M. Wang^{1,2}✉, Cristina Rea¹, Oswin So², Charles Dawson², Darren T. Garnier¹ & Chuchu Fan²

The tokamak offers a promising path to fusion energy, but disruptions pose a major economic risk, motivating solutions to manage their consequence. This work develops a reinforcement learning approach to this problem by training a policy to ramp-down the plasma current while avoiding limits on a number of quantities correlated with disruptions. The policy training environment is a hybrid physics and machine learning model trained on simulations of the SPARC primary reference discharge (PRD) ramp-down, an upcoming burning plasma scenario which we use as a testbed. To address physics uncertainty and model inaccuracies, the simulation is massively parallelized on GPU with randomized physics parameters during policy training. The trained policy is then run in feedback on a transport simulator as a demonstration. We also directly address the crucial issue of control validation by demonstrating that a constraint-conditioned policy can be a trajectory design assistant that designs a library of feed-forward trajectories to handle different physics conditions and user constraint settings, a promising approach for the sensitive context of burning plasma tokamaks. Finally, we demonstrate that the training environment can be a useful platform for feed-forward optimization approaches by optimizing feed-forward trajectories that are robust to physics uncertainty.

Plasma disruptions, events where control of the plasma is lost, stands as a major barrier to realizing economical fusion energy through tokamak devices. Unmitigated disruptions in existing tokamak pilot plant designs, such as ARC, may disrupt machine operations for months or longer¹. Even at the energies of upcoming burning plasma devices, such as SPARC and ITER, disruptions represent a major challenge, driving costly design requirements and posing risks of major delays to tokamak operations^{2,3}. To overcome the challenge of disruptions, considerable advances are needed in avoidance, mitigation, resilience, and recovery (AMRR) techniques. Mitigation involves suddenly terminating the plasma discharge with techniques such as massive gas injection (MGI)⁴. Given its crucial role in machine protection, considerable amounts of effort has been allocated towards developing mitigation actuators and disruption prediction algorithms to trigger said mitigation actuators^{5–7}. While robust, reliable mitigation is crucial for investment protection, it is still expected to induce considerable stress on tokamak components and should be considered a method of last

resort⁸. Thus, avoidance techniques to terminate the plasma safely would be preferable.

To safely terminate the plasma, the plasma current and stored energy must be reduced to a benign level; this is often referred to as a ramp-down. The primary figure of merit for a successful ramp-down is often the plasma current I_p at which the plasma terminates as both structural loading and the risk posed by runaway electrons scale non-linearly with I_p ^{9,10}. In recent years, works have developed both Sequential Quadratic Programming (SQP) and Bayesian Optimization approaches for designing feed-forward ramp-down trajectories that are programmed into the machine^{11–13}. However, many challenges remain unaddressed such as demonstrating scalability to allow for the generation of many trajectories to handle different scenarios, and the optimization of trajectories that are robust to sources of uncertainty. In addition, active adjustment of the plasma trajectory through real-time feedback remains an underdeveloped area with calls for further advancement¹⁴.

¹Plasma Science and Fusion Center, Massachusetts Institute of Technology, Cambridge, MA, USA. ²Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA. ✉e-mail: allenw@mit.edu

The problem setting of ramp-downs faces the considerable challenge of physics uncertainty. While recent work successfully demonstrated the application of reinforcement learning (RL) to the problem of shape control¹⁵, the physics relevant to the problem, ideal Magnetohydrodynamics (MHD), is a relatively well-understood and well-simulated aspect of tokamak plasma physics. Disruption avoidance, however, is more challenging as it involves complex physical phenomena such as turbulent transport for which accurate simulation from physics-based principles remains an open research problem. The highest fidelity simulations of core turbulence performed today with nonlinear gyrokinetics take millions of CPU-hours to simulate a steady-state condition and, even then, require boundary condition assumptions on quantities such as the pedestal pressure and impurity densities¹⁶. The extreme computational requirements of physics-based simulations stands in stark contrast to tokamak operations requirements. When unexpected physics or technical issues arise during tokamak operations, dynamics models, trajectories, and controllers need to be updated as soon as possible to ensure subsequent discharges are successful. As a consequence, scenario and control design is often performed with highly simplified dynamics models through empirical power-law scalings and linear systems obtained with techniques from classical system identification¹⁷. In fact, both the ITER and SPARC nominal operating scenarios were designed using empirical plasma operational contour (POPCON) scalings with scenario validation performed with higher fidelity simulations^{18–20}.

This challenge has motivated the development of data-driven dynamics models of existing devices for the purposes of reinforcement learning, but standard sequence-to-sequence models in the deep learning toolbox, such as recurrent neural networks (RNNs) and transformers, are generally not sample efficient. This is a major challenge as burning plasma tokamaks must work reliably with as few discharges of training data as possible, given the high cost of disruptions; existing works often use more than 10^4 discharges of data^{21,22}. Sample-efficient dynamics modeling methods are needed.

This paper aims to advance the state-of-the-art on three fronts: sample-efficient dynamics model learning, active ramp-down control, and optimizing feed-forward trajectories with robustness to dynamics uncertainty. To address the model learning problem, we train a simple hybrid dynamics model that contains both physics and machine learning components on a relatively small dataset of simulations of SPARC PRD ramp-downs with the control-oriented simulator RAPTOR^{23,24}. This dynamics model is fully implemented in the machine learning framework JAX²⁵, and is thus fully differentiable and GPU-capable, enabling massive parallelization across physics assumptions. We then wrap this dynamics model in an OpenAI Gym²⁶ environment to create what we call “PopDownGym”, which is depicted in Fig. 1a. We use PopDownGym to address the matters of feed-forward trajectory optimization and feed-back control in a unified reinforcement learning (RL) approach. As a first step towards experimental demonstration of active ramp-down control, we train a policy on PopDownGym with Proximal Policy Optimization (PPO) and demonstrate its transfer to RAPTOR where it successfully ramps down a simulated plasma while avoiding user specified limits. We address the issue of validation by demonstrating how a constraint-conditioned policy can be used to design a library of trajectories to handle different conditions by performing policy rollout on simulators under different physics settings. Finally, we bridge the gap between reinforcement learning and more conventional trajectory optimization approaches by using the RL training environment to perform robust optimization of a feed-forward trajectory with an evolutionary algorithm.

Results

We use the simulator RAPTOR configured to the SPARC primary reference discharge (PRD)¹⁸, an 8.7 megaampere (MA) H-mode scenario projected to reach an energy gain, Q , of approximately 11, as our testbed. We first generate a dataset of 481 ramp-down simulations with varying plasma current and auxiliary heating ramp-rates. 336 of these ramp-down simulations are then used to train the PopDownGym dynamics model while the rest are used for validation.

As shown in Table 1, the policy receives an observation from an eight dimensional observation space to select an action vector from a four dimensional action space. It has the goal of ramping down the plasma current to below two MA, while avoiding user-specified constraints on eight quantities which the device operator will want to limit. While information on the risk profile of SPARC as a function of plasma current is not available, this goal current was chosen as ITER disruptions below 3MA are expected to be benign²⁷. As discussed further in the Methods section, we train the control policy to accept user-specified constraint limits as inputs. This allows the constraints to be adjusted at inference time without retraining.

This section highlights two uses cases for the trained policy: 1) as a trajectory design assistant to design feed-forward trajectories for a wide range of physics conditions, and 2) as a real-time controller for actively avoiding disruptive limits. As a step towards demonstrating 2) experimentally, we transfer the control policy to RAPTOR. Finally, we demonstrate that the same gym environment can be leveraged to perform robust optimization of trajectories (i.e. optimization of feedforward trajectories that perform well under different conditions).

Application 1: policies as trajectory design assistants

It is desirable for burning plasma devices such as SPARC and ITER to achieve their core mission with as few discharges and disruptions as possible. In this context, the application of a learned control policy, which generally takes a non-trivial amount of trial and error to transfer to reality, is not desirable without extensive validation and operational history on existing devices. However, we demonstrate that the learned control policy can still be used as a trajectory design assistant to design a library of feedforward trajectories to handle a variety of different scenarios. These feedforward trajectories can then be validated against higher fidelity simulations before being programmed into the machine. Figure 1c depicts this workflow.

Constraint-conditioned policies. Due to uncertainty about disruptive and machine limits, a key feature in a useful trajectory design assistant is the ability for the user to adjust the constraints at inference time. For example, there is uncertainty about what values of β_p may be correlated with dangerous neoclassical tearing modes (NTMs) and what values of l_i or the Shafranov coefficient may be correlated with a vertical displacement event (VDE). After new information about such limits is acquired through tokamak operations, it is desirable to redesign trajectories given this new information as quickly as possible to enable operational continuity.

To address this challenge, we train a constraint-conditioned policy that allows the user to specify the constraint boundaries at inference time. Figure 2 shows a library of trajectories generated with two sets of constraint settings; one more relaxed, the other more stringent. The new trajectories are generated without any policy re-training, thus, they can be rapidly generated with a single policy rollout. In our demonstration, this is done within seconds. In Fig. 2, the policy mostly does a good job of trying to satisfy the constraints, with a small amount of violation in l_i at the end of the trajectory in the more stringent constraint case. We find that the policy struggles to satisfy the more stringent $\frac{dW_{th}}{dt}$ requirement, even in experiments where we tried increasing the penalty of constraint violation. We hypothesize this is due to it being physically infeasible to avoid the large stored energy change caused by the confinement regimes transition in certain cases.

Application 2: demonstration of ramp-down feedback control in RAPTOR

One use case for this learned policy is to serve as a controller that uses real-time observations of the plasma state to make decisions about how to ramp-down the plasma. As a step towards experimentally demonstrating such a capability, we demonstrate the successful transfer of the control policy to RAPTOR as a feedback controller. Figure 1 shows the setup for our sim2sim demonstrations. The control policy receives observations from RAPTOR at each time and decides on an action. To smooth out the effects of numerical spikes from the simulation, we employ an action buffer such that the average

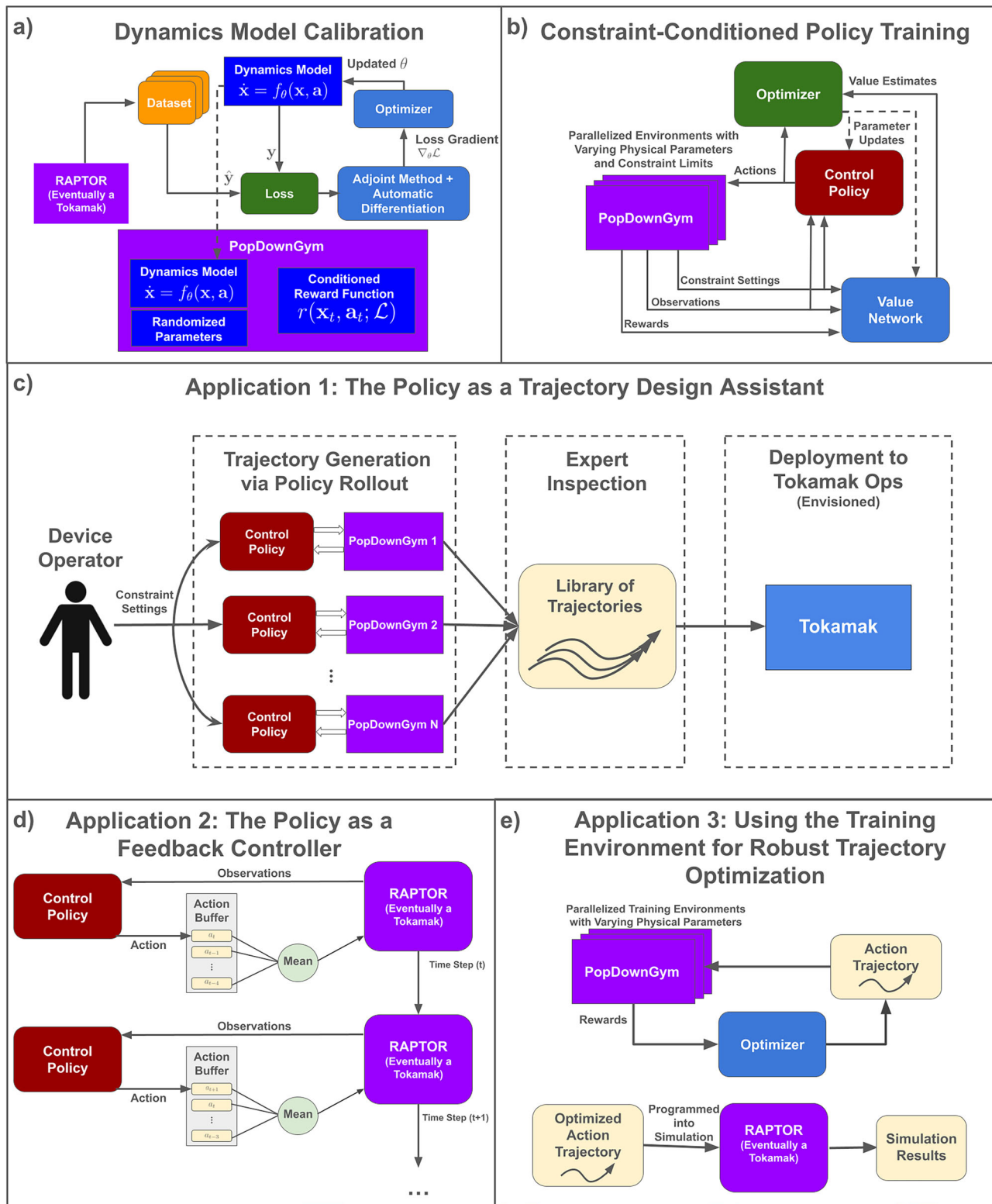


Fig. 1 | An overview of the methods developed in this paper. **a** Depicts the developed methodology for calibrating a dynamics model using the techniques over-viewed in the Methods subsection on “Neural Differential Equations and Differentiable Simulation”. The dynamics model calibrated in this work is detailed in the Methods subsection titled “Dynamics Model”. This dynamics model forms the core of the training environment “PopDownGym”, which also contains a reward function and random variables. **b** Depicts the training workflow for the constraint-

conditioned policy using Proximal Policy Optimization (PPO). **c** Depicts the workflow for applying the policy to generate a library of trajectories, which is demonstrated in Fig. 2. **d** Depicts the demonstration of running the policy as a controller in closed-loop against RAPTOR, with results shown in Fig. 3. **e** Depicts how the training environment can be used for robust trajectory optimization, which is demonstrated in Fig. 4.

Table 1 | Observation and action spaces with bounds

Parameter	Description	Min	Max
Observations o_t			
I_i	Normalized internal inductance	0.5	6.0
I_p	Plasma current [MA]	1.0	9.0
V_R	A summary statistic of the current profile defined by Eq. 42 in Romero 2010 ⁴⁸ [V]	−5.0	5.0
W_{th}	Plasma stored energy [J]	10^5	3×10^7
$\langle n_i \rangle$	Volume-averaged fuel density [10^{19}m^{-3}]	1.0	30.0
P_{aux}	Auxiliary heating power [MW]	0.0	25.0
g_s	Geometry parameter	0.0	1.0
Actions a_t			
dl_p/dt	Change in plasma current [MA/s]	−3.0	−0.5
dP_{aux}/dt	Change in aux. heating [MW/s]	−5.0	5.0
$u_{fuel,19}$	Fueling rate [$10^{19}/\text{s}$]	0.0	10.0
dg_s/dt	Geometry change rate [1/s]	0.0	1.0
Constraint Limits $\mathcal{L}$			
I_i	Normalized inductance	2	3
$n_{g,frac}$	Greenwald fraction	0.5	0.8
β_N	Normalized beta	0.015	0.028
β_p	Poloidal beta	0.25	0.4
dB_v/dt	Vertical field change rate [T/s]	0.2	0.4
dW_{th}/dt	Energy change rate [W]	2×10^7	7×10^7
Γ	Shafranov coefficient	3.4	3.6
l_{95}	Inverse safety factor	0.35	0.45

Observations out of bounds terminate the episode during training. Action bounds are enforced with hyperbolic tangent functions applied to the outputs of the policy. Constraint limit ranges are the range of values the constraint-conditioned policy is trained to handle. The observation space was chosen as the minimal set of parameters needed to avoid the specified constraint limits. Note that P_{aux} is specified as an observation variable to enable constraints on its rate of change by using $\frac{dP_{aux}}{dt}$ as an action variable. Note that I_i , I_p , W_{th} , $\langle n_i \rangle$ are quantities that are routinely estimated in real-time in existing plasma control systems (PCS), P_{aux} , g_s can be straight-forwardly implemented as state variables in a PCS, and Romero 2010⁴⁸ provides formulas to estimate V_R from quantities available from equilibrium reconstruction.

of the five latest actions is input into RAPTOR. Similar techniques are employed in other reinforcement learning applications where noise and outlier measurements need to be handled²⁸. We do not directly account for experimental measurement noise and partial observability, but note that the policy can be deployed in conjunction with a previously developed real-time state observer to mitigate this issue²⁹.

Figure 3 shows results of one such demonstration compared against a naive baseline feedforward trajectory, where many of the constraints are violated. The policy overall does a good job of avoiding the constraint boundaries, but there are imperfections. Like in the stringent constraint case in Fig. 2, we observe a small amount of I_i constraint violation at the end of the ramp-down. In the action space, this can be attributed to the policy deciding to extremely quickly ramp the plasma current once it is close to its goal, which is not necessarily undesirable. We also observe spikes in $\frac{d}{dt} W_{th}$ and $\frac{d}{dt} B_v$ at the H to L mode transition, which we attribute to the numerics of boundary condition switching at the H to L mode transition. After the spike, the values quickly settle to values within the specified constraints. Similar spikes in $\frac{d}{dt} B_v$ also occur towards the end of the simulation.

The policy exhibits some notable qualitative behaviors. For one, it begins with a large plasma current ramp-rate, but decreases the ramp-rate as the plasma state enters the regime where the H to L mode transition is expected. This is an intuitively correct behavior for two reasons: 1) the H to L mode transition can lead to a large $\frac{d}{dt} B_v$, and one way to decrease its value is

to decrease the ramp-rate, and 2) β_p starts approaching the user-specified limit, which happens when the current decreases faster than the stored thermal energy.

Application 3: feed-forward optimization with robustness to physics uncertainty

A natural question that arises when using a trained control policy to generate feedforward trajectories is whether we might instead directly optimize feedforward trajectories, skipping the policy representation step entirely. This approach has a number of potential benefits, particularly given the risk involved in transferring an untested learned control policy to hardware. The main difficulty that arises here is that while a feedback policy is able to observe and adjust to disturbances and uncertainty in the dynamics, purely feedforward trajectories must be designed to be robust to those disturbances without feedback.

To address this challenge, we implement a robust feedforward trajectory optimization pipeline that uses a large number of parallel simulations, each with different physics assumptions (i.e. random choices for the parameters in Table 2). We optimize a single feedforward trajectory to minimize the average constraint violation across all of these simulations. The optimized trajectory and its performance across a range of random parameter values are shown in Fig. 4, showing that the optimized feedforward trajectory achieves similar hitting time and constraint satisfaction rates to the feedback policy learned using RL. The trajectory transferred to RAPTOR is also shown in Fig. 4, where we see it does a similarly good job of avoiding the constraint limits as the RL trained policy, with the exception of the I_i limit, which it violates towards the end of the trajectory, likely due to a sim2sim gap between the training and test environments. Given that the feedforward trajectory experiences this issue, but the learned policy manages to mostly avoid this limit, this result highlights the advantages of real-time feedback for making course corrections.

Sensitivity analysis

Our reinforcement learning system also enables sensitivity analysis with respect to constraint limits and physics uncertainty. The former is enabled by the constraint-conditioned policy which allows the exploration of the optimal ramp-down for different constraint settings, and the latter is enabled by the gym environments' massive parallelization. Such information is critical for both device design and operations as it helps designers, operators, and physics modelers make decisions on how to allocate scarce resources. For example, our results in Fig. 5 show that the optimal time to ramp-down is highly sensitive to the I_i constraint, thus indicating that improvements to the vertical stability (VS) system is critical for enabling fast, safe ramp-downs. On the physics uncertainty front, Fig. 5 indicates, for example, strong sensitivity to the H-factor, but relatively low sensitivity to the ratio of electron temperature to ion temperature. While such results require validation, they provide information on where to allocate the efforts of extremely computationally expensive simulations and device experiments.

Robustness analysis

Similar to the sensitivity analysis carried out for the constraint-conditioned policy learned via PPO, we can examine the robustness of the static feedforward trajectory to varying physical parameters. Figure 6 shows a sensitivity analysis indicating which parameters have the strongest effect on the constraint satisfaction of the feedforward trajectory; we see that H-factor has the strongest effect, with low H-factor making constraint violation more likely.

Discussion

This paper develops a reinforcement learning approach to designing control policies and robust trajectories for the ramp-down phase. We place a particular emphasis on meeting several key needs of burning plasma tokamaks such as SPARC and ITER. Namely, we develop a methodology that: 1) requires relatively few discharges of training data, 2) is adaptable between

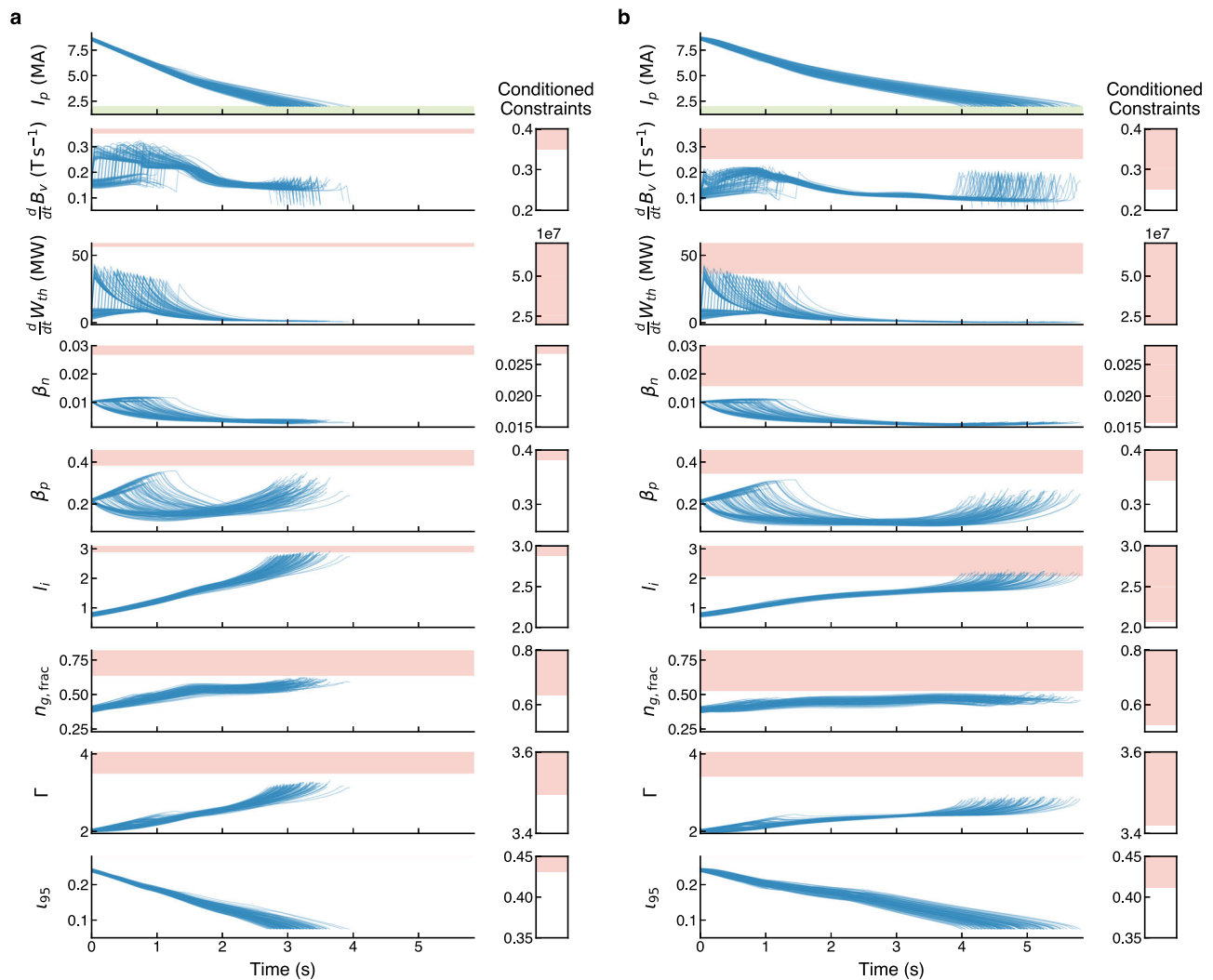


Fig. 2 | Demonstration of application 1: policies as trajectory design assistant. Figures above show an example of libraries of action trajectories generated under flexible (a) and more stringent (b) constraints. Each trajectory corresponds to one possible sample from the distribution of initial conditions and physics parameters, specified in Table 2. The green shaded region in the I_p plots denotes the target plasma

current of below 2 MA, while the red shaded regions denote the constraint values. The constraint values conditioned on for each scenario in relation to the minimum and maximum values used for training are visualized as red bars on the right of each plot.

discharges, and 3) enables control validation by generating a library of trajectories offline.

To train the policy, we propose a new sample-efficient approach to build hybrid physics and machine learning models using the frameworks of neural differential equations and differentiable simulation. The sample efficiency gained from the physics structure of the model enabled a successful demonstration of sim2sim policy transfer while training the dynamics model on only ≈ 300 simulations, in contrast to prior works which used order 10^4 simulations. The model is fully implemented in JAX, enabling massive GPU parallelism across physics uncertainty.

We also address the issue of control validation for near-term burning plasma tokamaks such as SPARC and ITER and demonstrate two ways in which fusion experiments may leverage the benefits of reinforcement learning in a way more compatible with the sensitive context. First, we further leverage GPU parallelism during policy training to parallelize across constraint settings to train a constraint-conditioned policy. This constraint-conditioned policy enables the user to adjust constraint settings at inference time and generate new trajectories in seconds with a single policy rollout. We propose that the constraint-conditioned policy may be a useful trajectory design assistant for tokamak operators, as unexpected developments during tokamak operations often demands the adjustment of settings. Since

the constraint-conditioned policy is being used to design trajectories offline, its results can be checked and validated by human operators and against validation simulations. It may also be integrated with other, more classical, techniques such as model predictive control (MPC)³⁰. Second, we demonstrate that the gym environment used in conjunction with other methods for feed-forward optimization. As a demonstration, we use an evolutionary algorithm to perform robust optimization of a feed-forward trajectory on parallel simulators with different physics settings. The resulting feed-forward trajectory is transferred to RAPTOR and avoids most constraint limits, but does violate the I_i constraint towards the end of the trajectory by a significant amount, an issue which the policy run in closed-loop with RAPTOR largely avoids. This result highlights one benefit of having a policy make corrections in real-time, a benefit which can be evaluated against the risks of running a learned policy on an expensive machine.

The biggest limitation of this work is the use of a relatively low fidelity plasma simulator as a testbed, the choice of which was motivated by the rapid iteration cycles enabled by RAPTOR, which proved critical for advancing and iterating the large number of new techniques presented in this work. Ongoing work is progressing to replace the “RAPTOR” box used in this paper with an existing tokamak. In that setting, data from the tokamak will be used to train a dynamics model on which the control policy

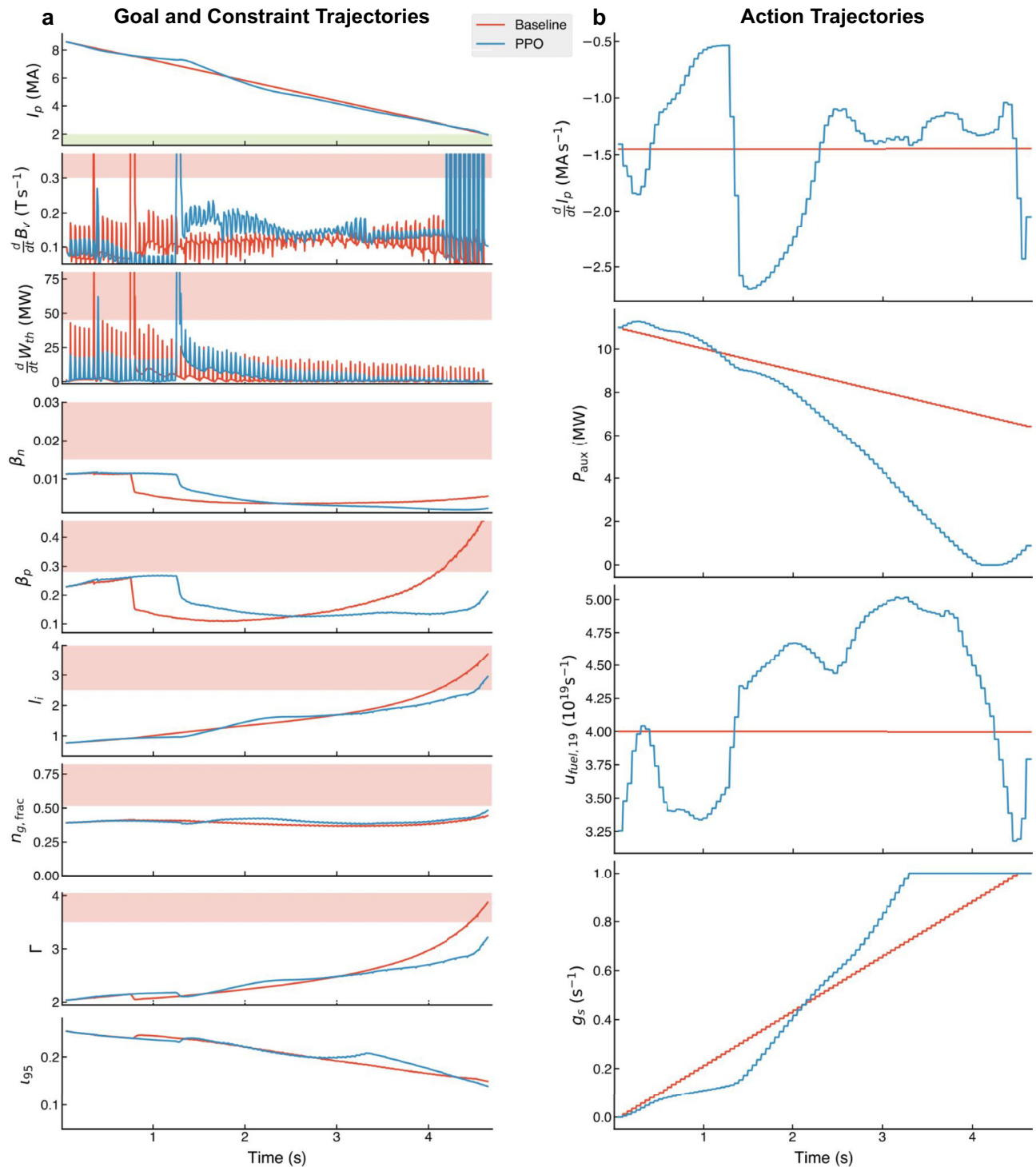


Fig. 3 | Demonstration of application 2: ramp-down feedback control. A comparison of RAPTOR simulation results from a naive baseline feed-forward trajectory (red) against the PPO trained policy running in closed loop (blue). The goal and constraint trajectories are shown in (a), while the action trajectories are shown in (b). The PPO policy clearly yields a considerable reduction in constraint violation for β_p , I_i , L . While the nominal SPARC PRD ramp-down has a constant ramp-rate of 1 MA/s, our baseline was selected to have the same average ramp-rate as the PPO policy to

better highlight how more complex time traces can yield a reduction in constraint violation while keeping the same ramp-down time. Note that while the nominal action space is the rates of change of P_{aux} and g_s , the action trajectories plot (right) shows the time-integrated values for interperitability. Simulation of the stored energy and vertical field rate of change involve spikes at sawtooth and confinement regime transition events which are due to RAPTORS approach to handling boundary conditions at these events.

is then trained. Future work should also apply this approach to the highest fidelity simulations possible of upcoming devices such as SPARC and ITER. It is the hope of the authors that such a research programme will enable the practical application of the powerful tools of machine learning to make progress on the challenging and critical problem of disruptions.

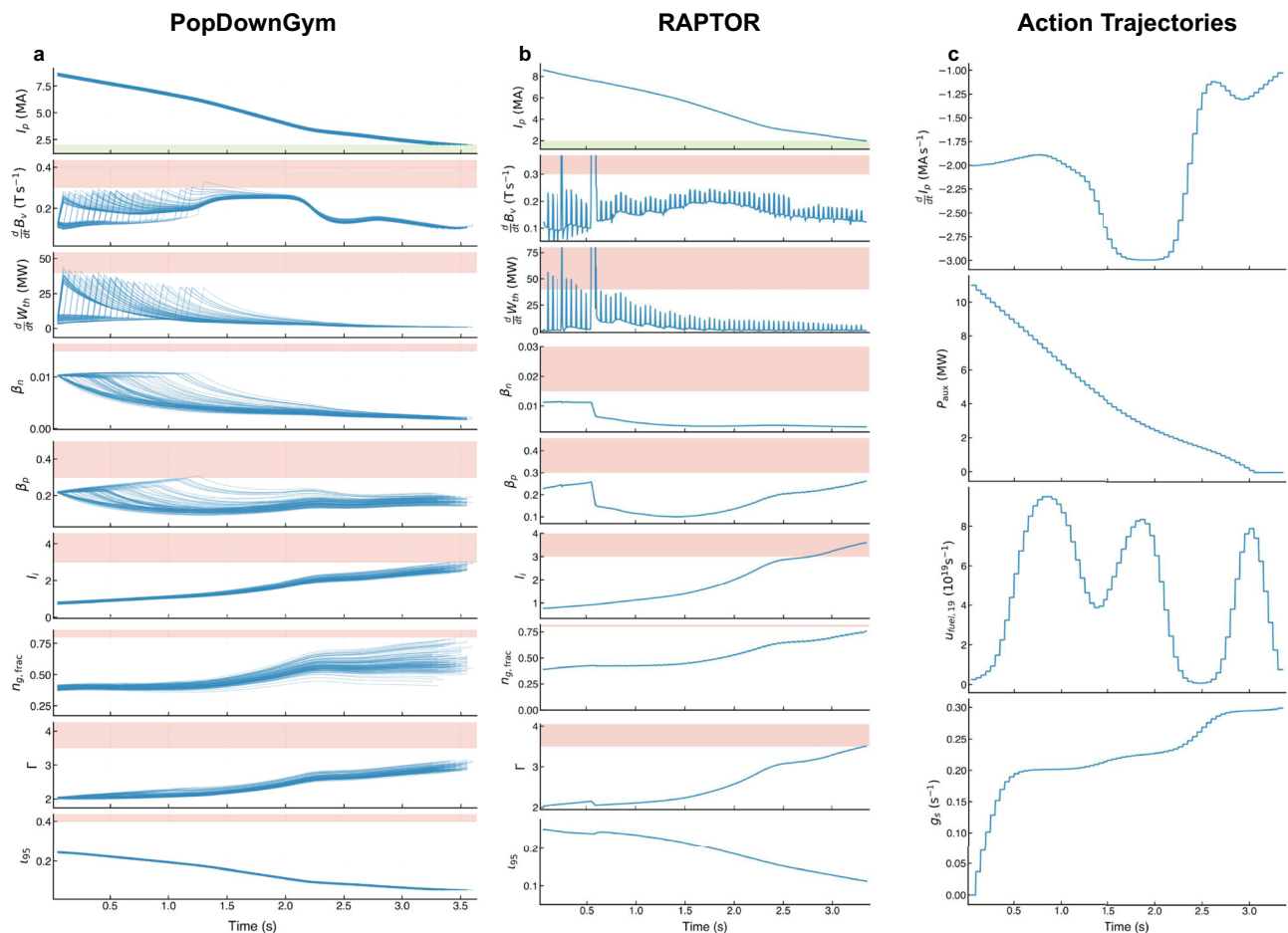
Methods

The first three subsections aim to provide the reader with an overview of the techniques this paper draws from, with subsequent subsections providing further technical detail. The first subsection aims to define the framework and capabilities of RL, and show how it can be applied to the

Table 2 | Randomized parameter ranges used for policy training

Parameter	Description	Minimum value	Maximum value
k_{dil}	Main ion dilution	0.8	0.9
k_{HL}	Fraction of the LH threshold at which the HL transition occurs	0.55	0.75
H	H factor used in the IPB89 and IPB98 scalings	0.8	1.0
Z_{eff}	Effective ion charge	1.2	1.8
k_{te_ti}	Ratio of volume-averaged electron to ion temperature	1.0	1.2
k_N	Ratio of particle confinement time to energy confinement time	7.0	9.0
k_{rad}	Ratio of total core radiated power to bremsstrahlung	2.0	3.0

Values were sampled from the ranges according to a uniform distribution.

**Fig. 4 | Demonstration of application 3: robust feed-forward optimization.**

a Shows results from parallel PopDownGym instances running the same feed-forward trajectory found via robust optimization. **b** Shows results from RAPTOR

simulation running the feed-forward trajectory. **c** Shows the feed-forward trajectory used in both cases.

problem of trajectory optimization, which is typically approached from the perspective of optimal control. We also provide a brief explanation of how we train a constraint-conditioned policy by drawing upon techniques from the goal-conditioned RL literature³¹. The key necessary ingredient for the RL workflow is a time-dependent dynamics model, so in the second subsection we discuss the challenges involved in plasma dynamics modeling. In the third subsection, we propose the framework of neural differential equations and differentiable simulation to address the challenges of time-dependent plasma dynamics modeling by synthesizing physics-based principles with data and provide a brief introduction to that paradigm. Following these higher level subsections, the fourth subsection

specifies the reward function used in this work, the fifth subsection details the dynamics model used as the training environment, the sixth subsection details the RL algorithms used, and the final subsection details RAPTOR simulation settings.

Reinforcement learning and optimal control

Reinforcement learning is typically concerned with solving Partially Observable Markov Decision Processes (POMDPs). A key component of a POMDP is the dynamics model, f , which may be stochastic. This model maps a system state vector $\mathbf{x}_t$ (e.g., stored energy and density) and an action vector $\mathbf{a}_t$ (e.g., auxiliary heating and fueling) to the system state vector

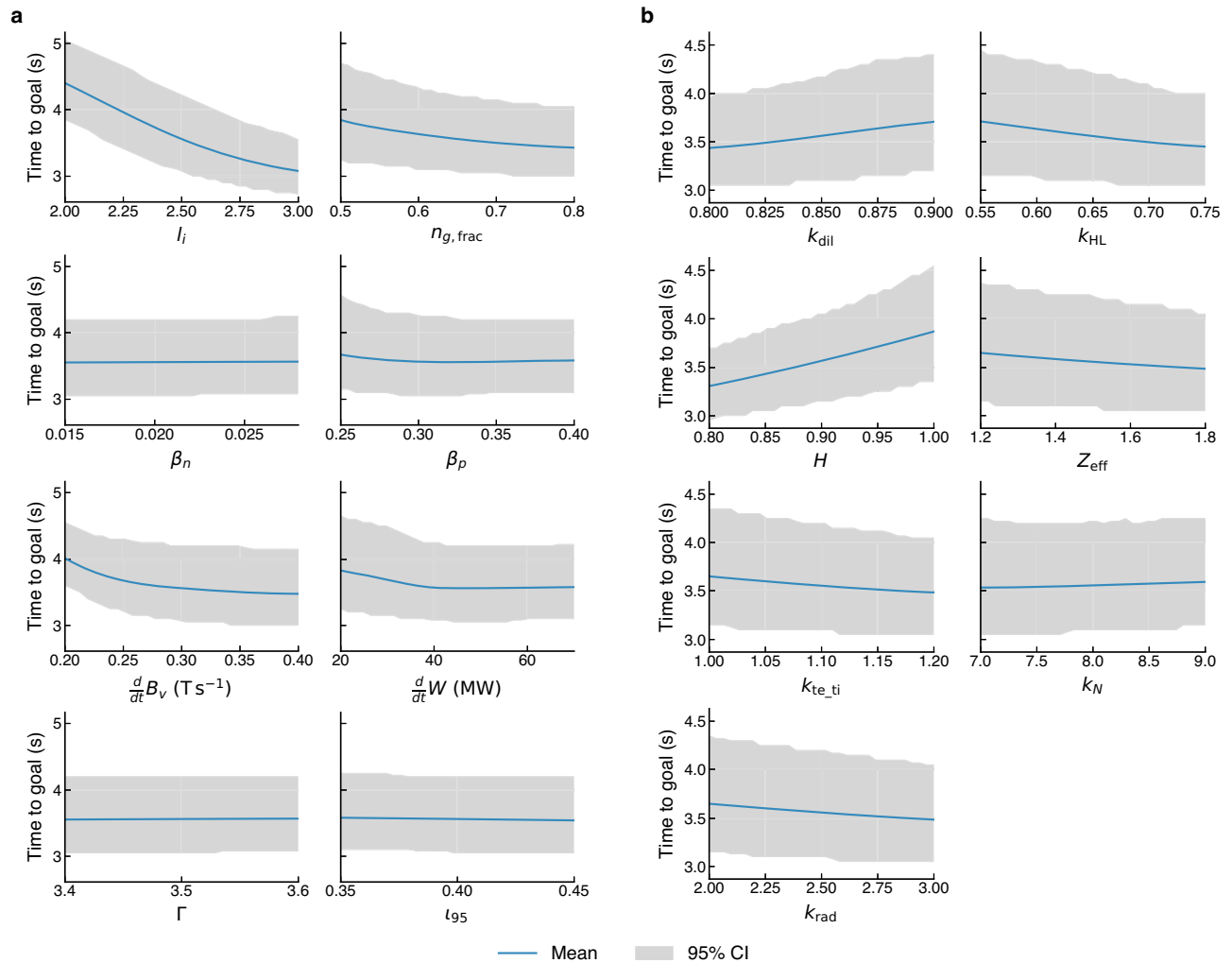


Fig. 5 | Sensitivity of time to goal. This plot shows the sensitivity of the time to goal with respect to constraint limit settings (a) and physics uncertainty (b), with the 95% confidence intervals corresponding to the physics uncertainty. In (b), the given

physics parameter is fixed while the other physics uncertainty parameters are sampled randomly.

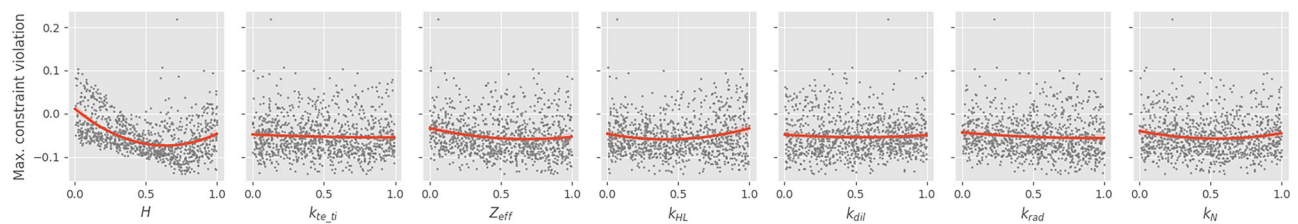


Fig. 6 | Sensitivity of constraint satisfaction. This plot shows the effect of varying physical parameters on constraint satisfaction for the feedforward trajectory. Raw data points are shown in gray, and a quadratic best fit and 95% confidence interval are shown in orange.

at the next time step:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{a}_t) \quad (1)$$

To account for the fact that certain quantities are derived from the system state (e.g. the average pressure can be derived by energy and volume), and to account for the real world challenges of imperfect sensing, POMDPs also involve an observation function, O , that generates an observation $\mathbf{o}_t$ from the state and action: $\mathbf{o}_t = O(\mathbf{x}_t, \mathbf{a}_t)$. Finally, a POMDP involves a reward function $r(\mathbf{x}_t, \mathbf{a}_t)$ that should be maximized, and which is usually designed by a human to specify the desired outcome. For example, in the context of video games, the reward function is designed to express wins and losses. In the context of

the ramp-down problem addressed by this paper, we design the reward function to reward de-energizing the plasma while avoiding disruptive and machine limits, the details of which are provided in the Methods.

There are two typical POMDP solution types. The first is a control policy π , which determines the optimal action to take given an observation: $\mathbf{a}_t = \pi(\mathbf{o}_t)$. In the modern deep reinforcement learning paradigm, the control policy is typically implemented as a deep neural network. Given that it can decide the action given an observation, a control policy can serve as a real-time feed-back controller.

The other solution type is an action trajectory, which is a sequence of actions across a T step time horizon: $[\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_{T-1}]$. This action trajectory can then be programmed into a machine. The computational problem of

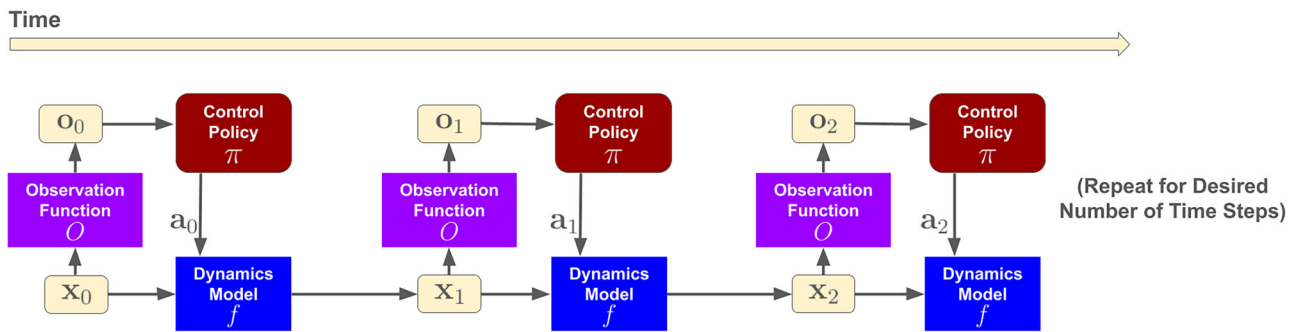


Fig. 7 | Graphic depicting policy rollout. Here, the process of a control policy π generating trajectories of action $\mathbf{a}_t$, state $\mathbf{x}_t$, and observation $\mathbf{o}_t$ is visualized.

finding optimal trajectories is often referred to as trajectory optimization and has largely been advanced by the field of optimal control. In fact, the POMDP formulation is directly analogous to standard formulations of stochastic optimal control³².

While control policies trained through RL are often thought of as products for real-time control, a process known as policy rollout can also be used to generate trajectories by running the control policy against the dynamics model f and observation function O . This process is shown in Fig. 7. We would like to note the other key distinction between the RL and optimal control frameworks, which is that the latter explicitly supports constraints. However, as shown in the Methods section, constraints can be implemented in reward functions via penalty terms. In fact, many nonlinear optimization methods used in optimal control implement constraints as cost terms under the hood with methods such as augmented lagrangians and barrier functions^{33,34}.

In this work, we also leverage techniques from goal-conditioned reinforcement learning³¹, which enables the training of policies that can be instructed at inference time to switch its task. We leverage this capability to train a policy that knows how to handle conditions with different constraint limits; we call the resulting policy a constraint-conditioned policy. This can be accomplished by simply conditioning the reward function on the constraint limits $\mathcal{L}$, adding $\mathcal{L}$ as an additional input to the policy, and parallelizing training environments across different samples of $\mathcal{L}$. This methodology is depicted in Fig. 1b. In this setting, the reward function and policy are modified to be:

$$r(\mathbf{x}_t, \mathbf{a}_t; \mathcal{L}) \quad \mathbf{a}_t = \pi(\mathbf{o}_t; \mathcal{L}) \quad (2)$$

Challenges of plasma dynamics modeling

In the context of plasma control, defining the dynamics function f and observation function O are considerable challenges. While accurate full-discharge simulation of plasmas from first principles is itself an open research problem, application of RL techniques generally requires stable, massively parallelized simulation with randomized physics parameters and thus imposes even more stringent software requirements. These challenging requirements have motivated work on developing neural network dynamics models trained on machine data^{21,22}. However, the relative paucity of available fusion data makes sample-efficiency and robustness of these models a considerable challenge.

At the same time, reactor and control design are often performed using relatively simple 0-D models which are used to find design points that are then validated using higher fidelity transport simulations. In the context of reactor design, such models are often referred to as Plasma Operating Contour (POPCON) models³⁵ and include well-established physics, such as reactivity curves, with empirical scalings, such as τ_E scalings. Despite their simplicity, such models have produced predictions of key performance metrics such as energy gain and τ_E that are in line with nonlinear gyrokinetic simulations. In the context of control design, classical system identification techniques are often employed to fit linear dynamical systems to empirical

data; controllers are then designed on these linear models³⁶. Despite the complexity of plasma dynamics, such techniques have proven effective for control design in certain cases^{24,37}. In recent years, advances in control-oriented simulation research has gradually improved the fidelity of models that can be used for control design, including efforts such as RAPTOR²⁴, COTSIM³⁸, and METIS³⁹. However, these simulators typically still require assumptions on quantities such as the edge boundary condition, and are not yet deployed for routine use in tokamak control.

Neural differential equations and differentiable simulation

While classical system identification techniques have yielded successes^{24,37}, further advances are needed. Fortunately, recent advances made by the scientific machine learning community in differentiable simulation and neural differential equations offers a powerful new toolbox for complex nonlinear system identification^{40–42}. This work applies such advances to train POPCON-like dynamics models that combine physics structure with machine learning to predict dynamics from relatively few samples. In contrast to standard POPCON models, the model is time-dependent and its free parameters can be updated to new empirical data, and, in contrast to classical system identification techniques, the model can be highly nonlinear.

Consider the following nonlinear dynamical system where f_θ is a neural network with free parameters θ :

$$\frac{d\mathbf{x}}{dt} = f_\theta(\mathbf{x}) \quad (3)$$

Such an equation is known as a neural differential equation (NDE). Both training and inference of NDEs involves numerically integrating the neural network from its initial condition $\mathbf{x}(0)$ across some time horizon to some terminal value. While standard feed-forward neural networks with parameters θ , NN_θ , have the following input-output relationship:

$$\mathbf{y} = \text{NN}_\theta(\mathbf{x}) \quad (4)$$

NDEs have the following input-output relationship:

$$\mathbf{y} = \text{diffeqsolve}(f_\theta, \mathbf{x}_0) \quad (5)$$

where `diffeqsolve` is a numerical integration scheme such as explicit or implicit Runge-Kutta methods. Then, given data $\hat{\mathbf{y}}$, and a loss function defining error between simulation and data $L(\mathbf{y}, \hat{\mathbf{y}})$, adjoint methods^{40,41} can be used to efficiently compute the gradient of this loss function with respect to the model parameters $\nabla_\theta L(\mathbf{y}, \hat{\mathbf{y}})$. While we have thus centered this discussion around the case where f_θ is a neural network, the same methodology applies to all functions f_θ that are differentiable with respect to θ . Historically, the differentiability requirements have limited the complexity of f_θ . However, modern automatic differentiation frameworks allow for highly complex choices of f_θ that can also include physics. This more general setting is often referred to as differentiable simulation and enables the

development of f_θ models that contain both physics and neural network components. The particular f_θ model employed in this work is detailed in the “Dynamics Model” subsection in the Methods, and the application of this methodology for tuning the dynamics model is depicted in Fig. 1a.

Reward function

The constraint-conditioned reward function is defined as:

$$r(\mathbf{x}_t, \mathbf{a}_t; \mathcal{L}) = \underbrace{\mathbb{1}_{I_{p,t} \leq 2MA}(\mathbf{x}_t)}_{\text{Reward for reaching goal}} + \underbrace{I(\mathbf{x}_t)}_{\text{Guide term for current}} + \underbrace{\sum_{i=1}^{n_{\text{cons}}} b_i(\mathbf{x}_t; \mathcal{L}_i)}_{\text{Penalty terms for constraints}} \quad (6)$$

where the first term provides a fixed reward upon reaching the goal of two MA of current, the second term $I(\mathbf{x}_t)$ rewards decreases in current to help guide the policy towards the goal and is given by:

$$I(\mathbf{x}_t) = -1 + \frac{4}{\exp(0.5I_{p,t}) + 2.0 + \exp(-0.5I_{p,t})} \quad (7)$$

This reward term was inspired by one used for tracking error in quadruped locomotion²⁸ with coefficients chosen through manual tuning. Finally, the summation term consists of penalty terms for violating user-specified constraints on undesirable limits such as disruptive limits. In this work, all constraints in state are implemented in the following form:

$$g_i(\mathbf{x}_t) \leq \mathcal{L}_i \quad (8)$$

where $g_i(\mathbf{x}_t)$ is the i_{th} quantity to be constrained, expressed as a function of state, and $\mathcal{L}_i$ is the user-specified limit for the constraint. For example, the Greenwald Limit would be implemented with the following g and $\mathcal{L}_i$:

$$g_i(\mathbf{x}_t) = \bar{n}_e \frac{\pi a^2}{I_p} \quad \mathcal{L}_i = 1 \quad (9)$$

There are a number of different approaches for implementing constraints in POMDPs^{43,44}. In our formulation, each constraint is replaced with the following “reward barrier function”, which we find to solve our problem well in practice:

$$b_i(\mathbf{x}_t; \mathcal{L}_i) = \log \left(1 - \sigma \left(\text{clip} \left(\frac{g_i(\mathbf{x}_t)}{\mathcal{L}_i} \right) \right) \right) \quad (10)$$

where $\sigma(\cdot)$ is a sigmoid and $\text{clip}(\cdot)$ is a clip function. This reward term is largely inspired by the logarithmic barrier functions from classical interior point methods³³ with the sigmoid and clip functions ensuring a lower bound on reward to prevent numerical divergence. The net effect of this function is to take on a value close to zero when $g_i(\mathbf{x}_t)$ is away from $\mathcal{L}_i$, but then rapidly fall off to a negative value when $g_i(\mathbf{x}_t)$ approaches $\mathcal{L}_i$. Supplementary Fig. S2 in the Supplementary Information visualizes this function.

Dynamics model

Randomized parameters. One of the goals of this dynamics model was to explore the efficacy of training on highly simplified models, but with massively parallelized, randomized physics simulation. The randomized parameters and their domains of randomization are shown in Table 2. In many cases, accurate simulations of these parameters would require supercomputers. Instead, we take the approach of designing policies and trajectories with robustness to variations in these parameters.

Geometry evolution parameter. To enable the full training environment to run on GPU in the absence of a GPU-capable Grad-Shafranov

solver, we use a pre-computed set of ideal MHD equilibrium shapes and restrict the policy to only decide on how quickly to evolve through said shapes. To do so, we define a “progress variable” $g_s \in [0, 1]$ where 0 corresponds to the first shape and 1 corresponds to the last. The policy then decides on the rate of change of g_s at each time step. Key quantities used in the model are interpolated between the shapes and thus become functions of g_s . The quantities used are: $V(g_s)$, the plasma volume, $\frac{\partial V}{\partial \rho}(\rho, g_s)$, the derivative of volume with respect to the toroidal flux coordinate, $\kappa_a(g_s)$, the area elongation, $a(g_s)$, the minor radius, $\delta(g_s)$, the triangularity. From here on out, the g_s argument is suppressed for notational simplicity.

Power and particle balance. We make use of simple power and particle balance equations¹⁷:

$$\frac{dW_{th}}{dt} = \frac{-W_{th}}{\tau_E} + P_{Ohm} + P_\alpha - k_{rad}P_{brems} + P_{aux} \quad (11)$$

$$\frac{dN_i}{dt} = \frac{-N_i}{k_N \tau_E} + fuel \quad (12)$$

The Ohmic power, P_{Ohm} , alpha heating power, P_α , and Bremsstrahlung radiation power P_{brems} terms require profile shapes to be accurately calculated. We take the approach of mapping volume averaged quantities to profiles shapes learned via principal components analysis (PCA) on the RAPTOR dataset. First, volume averaged electron and ion temperatures and densities are obtained from state variables as such:

$$\frac{2W_{th}}{3V} = \langle p \rangle \approx \langle T_e \rangle \langle n_e \rangle + \langle T_i \rangle \langle n_i \rangle = \langle T_e \rangle \frac{\langle n_i \rangle}{k_{dilation}} + \frac{\langle T_e \rangle}{k_{te_ti}} \langle n_i \rangle \quad (13)$$

$$\langle n_i \rangle = \frac{N_i}{V}, \quad \langle T_e \rangle = \frac{2W_{th}}{3V} \left(\frac{\langle n_i \rangle}{k_{dilation}} + \frac{\langle n_i \rangle}{k_{te_ti}} \right)^{-1}, \quad \langle n_e \rangle = \frac{\langle n_i \rangle}{k_{dilation}}, \quad \langle T_i \rangle = \frac{\langle T_e \rangle}{k_{te_ti}} \quad (14)$$

Observe that the only unknown in the set of equations above is the volume-averaged electron temperature $\langle T_e \rangle$. We represent each of the four profiles with two single-component PCA basis: one for H-mode and one for L-mode. Below, the T_e profile is used as an example, but the same methodology applies for the other profiles:

$$T_e(\rho) = cv(\rho) + b(\rho) \quad (15)$$

where $v(\rho)$ and $b(\rho)$ are learned via PCA on the RAPTOR simulation dataset. The volume integral and its approximation with Legendre-Gauss quadrature can then be computed as:

$$\begin{aligned} \langle T_e \rangle &= \frac{1}{V} \int_0^1 [cv(\rho) + b(\rho)] \frac{\partial V}{\partial \rho}(\rho) d\rho \\ &\approx \frac{1}{\sum_{j=1}^{n_{\text{gauss}}} w_j \frac{\partial V}{\partial \rho}(\rho_j)} \left[c \sum_{j=1}^{n_{\text{gauss}}} w_j v(\rho_j) \frac{\partial V}{\partial \rho}(\rho_j) + \sum_{j=1}^{n_{\text{gauss}}} w_j b(\rho_j) \frac{\partial V}{\partial \rho}(\rho_j) \right] \end{aligned} \quad (16)$$

where n_{gauss} is the number of Legendre–Gauss points, w_j are the weights, and ρ_j are the Legendre–Gauss points. Rearranging the above equation for c we get:

$$c \approx \frac{\langle T_e \rangle \sum_{j=1}^{n_{\text{gauss}}} w_j \frac{\partial V}{\partial \rho}(\rho_j) - \sum_{j=1}^{n_{\text{gauss}}} w_j b(\rho_j) \frac{\partial V}{\partial \rho}(\rho_j)}{\sum_{j=1}^{n_{\text{gauss}}} w_j v(\rho_j) \frac{\partial V}{\partial \rho}(\rho_j)} \quad (17)$$

Now everything on the right hand-side is a function of state or a pre-computed quantity. Since $v(\rho)$ and $b(\rho)$ are constant when running the simulation, c fully parameterizes the $T_e(\rho)$ profile.

H-mode and L-mode. We use the IPB98 scaling for H-mode⁴⁵ and IPB89⁴⁶ for L-mode. The H to L mode transition threshold is subject to considerable uncertainty, making our approach of randomizing across physics uncertainty particularly relevant. In this model, we treat the HL back-transition threshold as a constant multiple of the established the LH threshold from⁴⁷, with the constant randomized across simulations:

$$P_{HL} = k_{HL} 2.15 e^{0.107} n_{e,20}^{0.782} B_T^{0.772} a^{0.975} R^{0.999} \quad (18)$$

Internal inductance evolution. Romero et al.⁴⁸ derived a simple three-state ODE system for the coupled internal inductance and plasma current evolution. We make use of their equations for the internal inductance and plasma current evolution, which are exact, and replace their voltage dynamics, which are ad-hoc, with a neural network, resulting in the following ODE system:

$$\frac{dL_i}{dt} = \frac{2}{I_p} (-V_{ind} - V_R) \quad (19)$$

$$\frac{dI_p}{dt} = \frac{1}{L_i} (2V_{ind} + V_R) \quad (20)$$

$$\frac{dV_R}{dt} = NN(I_p, L_i, V_R, V_{ind}, \langle T_e \rangle) \quad (21)$$

Algorithms

Optimizing constraint-conditioned policies with proximal policy optimization. We apply PPO⁴⁹, a state of the art on-policy reinforcement learning algorithm to solve for the constraint-conditioned policy that maximizes the reward function (6). A fully connected network with three hidden layers of 256 units each is used for both the policy and value networks with hyperbolic tangent activation functions. The environment observations are concatenated with the desired constraint values to form the input vector for the policy network. The value network additionally takes the current simulation parameters as input to improve performance^{50,51}. All input observations and output actions are normalized to the range $[-1.0, +1.0]$.

We exploit the capabilities of JAX and collect transitions from 2048 environments in parallel. During each training step, it took $0.652 (\pm 0.024)$ seconds to collect experience and $0.048 (\pm 0.002)$ seconds to update the policy and value networks. Training converged in under two hours on a single Nvidia RTX 4090 GPU. Inference of just the control policy took an average of $38 \mu s$, and a full roll-out of the policy along with the training environment took $15.4 ms$. All compute times were determined after Just-In-Time (JIT) compilation.

Robust feedforward trajectory optimization with evolutionary strategies. We apply a highly parallelizable black-box optimization algorithm known as evolutionary strategies (ES) to optimize robust feedforward trajectories⁵². In contrast to policy gradient methods like PPO, which consider the expected discounted reward, ES requires an end-to-end objective function that assigns a reward to a given fixed-horizon trajectory; we maximize:

$$U_{dense}(\mathbf{a}_{0:T}) = \mathbb{E}_{\theta} \left[\frac{1}{T} \sum_{t=0}^T r(\mathbf{x}_t, \mathbf{a}) \right] \quad (22)$$

where the expectation is with respect to the random parameters θ sampled uniformly over the ranges given in Table 2.

The feedforward trajectory is represented as a cubic interpolation between 10 equally spaced control points for each action. Trajectories are

rolled out for 100 timesteps with $0.05 s$ timestep. We use a population size of 256, approximate the expectation in Eq. (22) using 10^3 random samples, and run the evolutionary strategy for 250 generations. The learning rate is initialized at 10^{-1} and decays with rate 0.995 to a minimum of 10^{-3} . We use the ES defined by Salimans et al.⁵² as implemented in JAX in the evosax library⁵³.

RAPTOR simulation settings

RAPTOR is configured to evolve the T_e , T_i , and current profiles by solving the corresponding transport equations. Following prior work, the particle transport equations are not solved¹¹, but rather a separate instance of the simple particle balance model (12) is implemented RAPTOR side and receives the fueling commands. This separate model is configured with the maximum k_N value of 9.0 used in the training model to challenge the policy as gas injection can be used to directly slow down the density decrease, but actuators that can directly speed up the density decrease do not exist. This approach is motivated by the challenges of modeling plasma density dynamics¹⁷; the goal instead is to provide reasonable density set-points for real-time density controllers. The H-L back-transition is triggered when the RAPTOR simulated power conducted to scrape-off layer falls below the HL back-transition threshold.

Data availability

Data relevant to this study is included in the code repository github.com/MIT-PSFC/PopDownGym.

Code availability

The gym environment, trained policies, trajectories, and scripts to generate figures will be available at the time of publication at github.com/MIT-PSFC/PopDownGym. Code used for RAPTOR simulations, and data generated by RAPTOR simulations will not be available due to intellectual property limitations and the closed-source status of RAPTOR at the time of this work.

Received: 29 July 2024; Accepted: 19 May 2025;

Published online: 04 June 2025

References

1. Maris, A. D., Wang, A., Rea, C., Granetz, R. & Marmor, E. The impact of disruptions on the economics of a tokamak power plant. *Fusion Sci. Technol.* **80**, 636–652 (2023).
2. De Vries, P. et al. Requirements for triggering the iter disruption mitigation system. *Fusion Sci. Technol.* **69**, 471–484 (2016).
3. Sweeney, R. et al. Mhd stability and disruptions in the sparc tokamak. *J. Plasma Phys.* **86**, 865860507 (2020).
4. Whyte, D. et al. Mitigation of tokamak disruptions using high-pressure gas injection. *Phys. Rev. Lett.* **89**, 055001 (2002).
5. Rea, C., Montes, K., Erickson, K., Granetz, R. & Tinguely, R. A real-time machine learning-based disruption predictor in diii-d. *Nucl. Fusion* **59**, 096016 (2019).
6. Montes, K. J. et al. Machine learning for disruption warnings on alcator c-mod, diii-d, and east. *Nucl. Fusion* **59**, 096015 (2019).
7. Pau, A. et al. A machine learning approach based on generative topographic mapping for disruption prevention and avoidance at jet. *Nucl. Fusion* **59**, 106017 (2019).
8. Eidietis, N. et al. Implementing a finite-state off-normal and fault response system for disruption avoidance in tokamaks. *Nucl. Fusion* **58**, 056023 (2018).
9. Hender, T. et al. Mhd stability, operational limits and disruptions. *Nucl. Fusion* **47**, S128 (2007).
10. Izzo, V. et al. Runaway electron confinement modelling for rapid shutdown scenarios in diii-d, alcator c-mod and iter. *Nucl. Fusion* **51**, 063032 (2011).
11. Teplukhina, A. et al. Simulation of profile evolution from ramp-up to ramp-down and optimization of tokamak plasma termination with the raptor code. *Plasma Phys. Controlled Fusion* **59**, 124004 (2017).

12. Van Mulders, S. et al. Scenario optimization for the tokamak ramp-down phase in raptor. part a: analysis and model validation on asdex upgrade. *Plasma Phys. Controlled Fusion* **66**, 025006 (2023).
13. Van Mulders, S. et al. Scenario optimization for the tokamak ramp-down phase in raptor. part b: Safe termination of demo plasmas. *Plasma Phys. Controlled Fusion* **66**, 025007 (2023).
14. Boozer, A. H. Plasma steering to avoid disruptions in iter and tokamak power plants. *Nucl. Fusion* **61**, 054004 (2021).
15. Degrave, J. et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* **602**, 414–419 (2022).
16. Rodriguez-Fernandez, P., Howard, N. & Candy, J. Nonlinear gyrokinetic predictions of sparc burning plasma profiles enabled by surrogate modeling. *Nucl. Fusion* **62**, 076036 (2022).
17. Walker, M. L., De Vries, P., Felici, F. & Schuster, E. Introduction to tokamak plasma control. In: *2020 American Control Conference (ACC)*, 2901–2918 (IEEE, 2020).
18. Creely, A. et al. Overview of the sparc tokamak. *J. Plasma Phys.* **86**, 865860502 (2020).
19. Rodriguez-Fernandez, P. et al. Predictions of core plasma performance for the sparc tokamak. *J. Plasma Phys.* **86**, 865860503 (2020).
20. Mukhovatov, V. et al. Overview of physics basis for iter. *Plasma Phys. Controlled Fusion* **45**, A235 (2003).
21. Abbate, J., Conlin, R. & Kolemen, E. Data-driven profile prediction for diii-d. *Nucl. Fusion* **61**, 046027 (2021).
22. Char, I. et al. Offline model-based reinforcement learning for tokamak control. In: *Learning for Dynamics and Control Conference*, 1357–1372 (PMLR, 2023).
23. Felici, F. *Real-time control of tokamak plasmas: from control of physics to physics-based control*. Tech. Rep. (EPFL, 2011).
24. Felici, F. et al. Real-time physics-model-based simulation of the current density profile in tokamak plasmas. *Nucl. Fusion* **51**, 083052 (2011).
25. Bradbury, J. et al. JAX: composable transformations of Python + NumPy programs <http://github.com/google/jax> (2018).
26. Brockman, G. et al. OpenAI gym. Preprint at <https://arxiv.org/abs/1606.01540> (2016).
27. de Vries, P. C. et al. Multi-machine analysis of termination scenarios with comparison to simulations of controlled shutdown of iter discharges. *Nucl. Fusion* **58**, 026019 (2017).
28. Hwangbo, J. et al. Learning agile and dynamic motor skills for legged robots. *Sci. Robot.* **4**, eaau5872 (2019).
29. Felici, F., De Baar, M. & Steinbuch, M. A dynamic state observer for real-time reconstruction of the tokamak plasma profile state and disturbances. In: *2014 American Control Conference*, 4816–4823 (IEEE, 2014).
30. Ravensbergen, T. et al. Density control in iter: an iterative learning control and robust control approach. *Nucl. Fusion* **58**, 016048 (2017).
31. Chane-Sane, E., Schmid, C. & Laptev, I. Goal-conditioned reinforcement learning with imagined subgoals. In: *International conference on machine learning*, 1430–1440 (PMLR, 2021).
32. Bertsekas, D. *Reinforcement learning and optimal control* (Athena Scientific, 2019).
33. Nesterov, Y. et al. *Lectures on convex optimization*, vol. 137 (Springer, 2018).
34. Bertsekas, D. P. *Constrained optimization and Lagrange multiplier methods* (Academic Press, 2014).
35. Houlberg, W., Attenberger, S. & Hively, L. Contour analysis of fusion reactor plasma performance. *Nucl. Fusion* **22**, 935 (1982).
36. Wang, Y., Xiao, B., Liu, L. & Guo, Y. System identification for east plasma shape and position control. *Fusion Eng. Des.* **129**, 140–146 (2018).
37. Moreau, D. et al. Plasma models for real-time control of advanced tokamak scenarios. *Nucl. Fusion* **51**, 063009 (2011).
38. Pajares, A. & Schuster, E. Integrated robust control of individual scalar variables in tokamaks. In: *2019 IEEE 58th conference on decision and control (CDC)*, 3233–3238 (IEEE, 2019).
39. Artaud, J.-F. et al. Metis: a fast integrated tokamak modelling tool for scenario design. *Nucl. Fusion* **58**, 105001 (2018).
40. Kidger, P. *On neural differential equations*. Ph.D. thesis (University of Oxford, 2021).
41. Chen, R. T., Rubanova, Y., Bettencourt, J. & Duvenaud, D. K. Neural ordinary differential equations. In: *Advances in Neural Information Processing Systems* 31 (NIPS, 2018).
42. Rackauckas, C. et al. Universal differential equations for scientific machine learning. Preprint at <https://arxiv.org/abs/2001.04385> (2020).
43. So, O. & Fan, C. Solving stabilize-avoid optimal control via epigraph form and deep reinforcement learning. In *Proceedings of robotics: science and systems* (2023).
44. Altman, E. *Constrained Markov decision processes* (Routledge, 2021).
45. Editors, I. P. B., Chairs, I. P. E. G., Co-Chairs, Team, I. J. C. & Unit, P. I. Chapter 1: overview and summary. *Nuclear Fusion* **39**, 2137 (1999).
46. Kaye, S. et al. Iter I mode confinement database. *Nucl. Fusion* **37**, 1303 (1997).
47. Martin, Y. et al. Power requirement for accessing the h-mode in iter. *J. Phys. Conf. Ser.* **123**, 012033 (2008).
48. Romero, J. et al. Plasma internal inductance dynamics in a tokamak. *Nucl. Fusion* **50**, 115002 (2010).
49. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. Preprint at <https://arxiv.org/abs/1707.06347> (2017).
50. Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W. & Abbeel, P. Asymmetric actor critic for image-based robot learning. In *Proceedings of robotics: science and systems* (2018).
51. Zhu, Y. et al. Reinforcement and imitation learning for diverse visuomotor skills. In *Proceedings of robotics: science and systems* (2018).
52. Salimans, T., Ho, J., Chen, X., Sidor, S. & Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. Preprint at <https://arxiv.org/abs/1703.03864> (2017).
53. Lange, R. T. evosax: Jax-based evolution strategies. Preprint at <https://arxiv.org/abs/2212.04180> (2022).

Acknowledgements

This work was funded in part by Commonwealth Fusion Systems. The authors would like to thank Federico Felici and Simon Van Mulders for help with RAPTOR and technical discussions. The authors would also like to thank colleagues at the MIT Plasma Science and Fusion Center (PSFC) and Commonwealth Fusion Systems (CFS) for helpful technical discussions.

Author contributions

Allen M. Wang led the project, developed the Gym environment, the initial PPO implementation, RAPTOR transfer, and wrote most of the paper. Oswin So developed the high performance final PPO implementation, trained the constraint conditioned policies, and led visualizations. Charles Dawson developed the robust optimization algorithm and benchmarked the Gym environment. Darren Garnier conceptualized the project and helped Allen M. Wang get up to speed on plasma physics and fusion. Cristina Rea advised the project on the fusion and disruptions front, and revised the paper. Chuchu Fan advised the project on the controls and reinforcement learning front.

Competing interests

A.M.W., D.G., and C.R. received funding from Commonwealth Fusion Systems (CFS) and have worked directly with CFS on the SPARC project. All other authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at <https://doi.org/10.1038/s42005-025-02146-6>.

Correspondence and requests for materials should be addressed to Allen M. Wang.

Peer review information *Communications Physics* thanks Gergely Papp, Timo Ravensbergen, Simon van Mulders and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. A peer review file is available.

Reprints and permissions information is available at <http://www.nature.com/reprints>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025