Extended Abstract

Motivation Tokamaks are promising machines for energy-efficient fusion, but research on machine configurations is hindered by long experimental cycles and our limited understanding of how fusion works internally. The problem is a natural fit to apply simulation-based reinforcement learning: we will build on prior work to explore the possibility of optimizing fusion energy production using an RL-trained agent. Solving our objective could mean that we could produce efficient fusion, a form of clean, safe, and renewable energy.

Method We use a coupled simulation of TORAX and OpenFUSIONToolkit, which combines the unique plasma physics and tokamak simulation functionality of both. This coupled simulation is novel, published only a couple months ago. We chose two input parameters related to reactor heating to optimize, and generated a large offline dataset, and then trained an RL agent using this dataset to generate a more optimized heating schedule.

Implementation The dataset was generated by simulating 10,000 trajectories with a variety of randomly sampled heating schedules. A custom reward function was devised that uses scalar outputs from the simulated trajectory to reward high energy efficiency while penalizing unsafe plasma configurations, and this was used to assign a numerical reward to each simulated trajectory. Then, IQL was used to train the agent on this dataset, and the agent was evaluated in a closed-loop setting, where its heating decisions were iteratively applied to the simulation and the resulting states were fed back into the agent for future decisions.

Results The trained agent was able to achieve a peak energy efficiency over twice that of the baseline heating schedule, while maintaining very similar levels of safety to the baseline. Hyperparameter tuning yielded an agent with an even higher reward, but its energy production was not nearly as stable as the base agent.

Discussion Several parameters had to be tuned and optimized during the process of dataset collection and reward calculation. While changing some of these parameters had a measurable effect on our model's stability and outcomes, others were based primarily on heuristics and not quantitatively verified. However, our final setup yielded an agent that had a high reward while yielding stable outputs and realistic plasma configurations.

Conclusion We demonstrate that RL can be used to optimize the heating schedule for a tokamak-based fusion reactor. However, there is still much room for further improvement, including creating a more comprehensive reward function, collecting a more robust dataset, and changing our IQL algorithm to be probabilistic rather than deterministic.

Controlling a Fusion Reactor with Reinforcement Learning

Siddharth Bhatia
Department of Computer Science
Stanford University
smbhatia@stanford.edu

Deniz Yilmaz
Department of Physics
Stanford University
dyilmaz@stanford.edu

Sameer Agrawal
Department of Computer Science
Stanford University
sameer06@stanford.edu

Abstract

Tokamaks, experimental reactors which manipulate magnetic fields to produce nuclear fusion, are a potential source of abundant clean energy but the solution space for controlling these reactors to maximize net energy output is underexplored. We focus on the issue of modulating the overall pulse shape of a fusion reaction as it runs by controlling the neutral beam injection (NBI) and electron cyclotron resonance heating (ECRH) actuators. To achieve real-time control, we train an IQL agent using offline RL on a novel, coupled physical simulation of ITER, a massive fusion project currently under construction in France. Evaluating our agent on this model, we achieve a peak Q_f of 10, which nearly doubled our baseline.

1 Introduction

Nuclear fusion is one of the most promising sources of green energy today. It produces virtually no waste (unlike its counterpart, the nuclear fission reactor), and primarily uses isotopes of hydrogen as fuel, which are virtually unlimited in nature Sadik-Zada et al. (2024). Arguably one of the most promising concepts for future fusion power plants is the tokamak, a machine that confines high-temperature, ionized gas (plasma) using extremely strong magnetic fields.

In order to achieve efficiency at the level of commercial energy production, the plasma inside a tokamak must be heated to around 100 million degree Celsius and confined long enough for fusion reactions to occur, which necessitates careful control of a large number of actuators and control inputs within the tokamak. Generally, this is done via expert-driven manual tuning or inefficient searches of the parameter space, but a promising new approach to this problem involves using reinforcement learning (RL) to learn control schemes.

While RL has been applied to various tokamak control problems (see Section 2), one key parameter that has not been thoroughly investigated is *input heating*. In this project, we make use of novel tokamak simulation software to train an agent to control two key input-heating parameters for the reactor. We demonstrate that the trained agent is able to achieve great improvements in the reactor's efficiency, while still retaining a stable, safe plasma configuration.

Stanford CS224R 2025 Final Report

2 Related Work

Efforts toward improving tokamak control largely stem from the goal of generating more energy than put in (quantified by the parameter Q_f = energy out (fusion) / energy in (plasma)), and mainly follow two slightly different approaches: improving active stabilization with feedback control, or, tuning system operating points in a way that improves passive stability (Walker et al. (2020)).

Traditionally, both these methods of reactor control have been done using complex, handcrafted algorithms, but recent developments in optimization techniques have enabled the automatic discovery of *new* algorithms for these tasks. For example, Graber and Schuster used an automated controller that uses certain physics-based heuristics to converge decide on optimal inputs for heating and fueling (Graber and Schuster (2019)). They demonstrated that this technique can achieve a desired plasma temperature and density while minimizing impurities (Graber and Schuster (2019)). Dubbioso, et al. achieved similar results with even less expert input by using an optimization method called "extremum-seeking" to achieve accurate vertical plasma stabilization in a tokamak by precisely tuning the voltage applied to a set of coils within the reactor (Dubbioso et al. (2022)). However, these methods are limited, as they still rely somewhat on hand-tuned models to "start" the optimization, and are often not generalizable to new, unseen reactor states.

Recent advancements in reinforcement learning (RL) somewhat address this issue by enabling the use of RL methods for tokamak control. These agents do not require any prior knowledge of plasma dynamics, and are frequently able to discover new states outside of their training data. For example, in 2022, researchers at DeepMind applied maximum a posteriori Policy optimization (MPO), an actor-critic algorithm, to develop control systems for the magnetic coils within the tokamak vessel(Degrave et al. (2022)). They demonstrated that their RL actor is able to achieve better plasma confinement than baseline control mechanisms while also discovering new plasma configurations. Other works such as the research published by Char et al. (2023) use offline methods with experimental data to train an RL agent to control neutral beam injectors, which are essentially particle accelerators attached to the tokamak, used for heating the plasma (Char et al. (2023)).

However, while promising, these works focus mainly on fine-grained real-time control mechanisms for optimizing plasma configuration, ignoring potentially coarser-grained ways of modulating the overall pulse shape. Work in this second category is far more limited. For example, Di Grazia, et al. sought to optimize only simulation start-up parameters, such as initial voltage and current trajectories, to achieve optimal plasma currents within the tokamak (Di Grazia et al. (2024)). However, their method involved *no* real-time control, and also made use of a simpler linear plasma model (Di Grazia et al. (2024)). Additionally, their work did not make use of RL techniques, framing the issue as a constrained optimization problem instead (Di Grazia et al. (2024)).

Therefore, there is a key research gap in using RL methods for optimizing *coarse-grained* input variables (such as reactor heating) for overall pulse design.

3 Method

3.1 Tokamak Pulse Simulation

We used a novel, coupled simulation of the TORAX Citrin et al. (2024) and OpenFUSIONToolkit Char et al. (2023) packages. Tokamak simulation involves two separate calculations: solving the Grad-Shafranov equations gives the equilibrium state of the plasma, and transport equations compute how the plasma profiles in time. Solving the transport equations requires the magnetohydrodynamic equilibrium (the solutions to the Grad-Shafranov equations) as input, and solving the Grad-Shafranov equations requires the current and pressure profiles that the transport solutions provide. OpenFUSIONToolkit provides a package that computes the equilibrium, and TORAX provides the machinery necessary for evolving the plasma state in time. Typically, these two simulations operate separately, but this novel coupled simulation combines the two. By passing solutions back and forth between OpenFUSIONToolkit and TORAX, this coupled simulation models the entire tokamak pulse with unprecedented physics accuracy. To our knowledge, there has been no prior study researching how to optimize operational parameters for a tokamak using this coupled simulation.

Running this simulation at high granularity to preserve physics accuracy was quite computationally expensive; collecting 600 pulse trajectories on an M5 Macbook Pro took around 13 hours. Due to

these limitations, we decided to generate an offline dataset instead of creating an RL environment for an agent to interact with directly. To generate our dataset, we simulated 10,000 pulse trajectories with randomly sampled fueling schedules, collecting reward, state, and action triplets at each fueling decision time step (see 4.1 for more information).

3.2 Reinforcement Learning Method

For our RL method, we used an implicit Q-learning (IQL) algorithm, which learns a $Q^\pi(s, a)$ function that approximates total expected reward given an action. In particular, we used double Q-learning, which learns two separate Q-functions to mitigate overestimation bias.

This IQL algorithm is particularly well-suited to optimizing pulse trajectories in a fusion reactor because

- Because out-of-distribution actions are never queried during training, the policy is less likely to decide on a fueling command that could cause disruptions in the plasma.
- IQL is compatible with real or simulated static data, which is ideal since an RL agent cannot feasibly be trained on a physical fusion reactor.

Our network architecture during training consisted of two Q networks, $Q_{\phi_1}(s, a)$ and $Q_{\phi_2}(s, a)$, $V_\psi(s)$, and an actor $\pi_\theta(a|s)$. All of our networks are three-layer MLPs with a hidden dimension of 256 and ReLU activation functions. In order to scale actions to be within a specified range, the actor uses a sigmoid activation, the output of which is then rescaled to the necessary range for each actuator.

The Q-function is fitted with Bellman targets:

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r + \gamma V_\psi(s') - Q_\phi(s, a)\right)^2\right] \tag{1}$$

with discount factor $\gamma = 0.99$. Our target Q networks, which are used to fit the value function, are updated more slowly, using an exponential average of network weights for stability. The target network update rate is $\rho = 0.005$.

To approximate the maximum value function of the data distribution, we fit the value function using expectile regression:

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^\zeta\left(Q_{\bar{\phi}}(s, a) - V_\psi(s)\right)\right] \tag{2}$$

where the asymmetric expectile loss is:

$$L_2^\zeta(x) = |\zeta - \mathbf{1}(x < 0)| \cdot x^2. \tag{3}$$

Our policy is fitted using advantage-weighted regression, which multiplies the mean squared error of the policy's action with respect to the action in the dataset with exponentiated advantage, allowing the policy to essentially learn to take the optimal actions in the dataset.

$$\mathcal{L}_\pi(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\exp(\beta \cdot A(s, a)) \cdot \|\pi_\theta(s) - a\|^2\right] \tag{4}$$

4 Experimental Setup

For our experiment, we chose to optimize for the ITER Hybrid scenario, which is one of the main planned operation modes for the ITER tokamak. Specifically, we decided to optimize two key actuators related to the energy input for the reactor: neutral beam injection (NBI) and electron cyclotron resonance heating (ECRH). Default values for the actuators during this pulse are described in Budny et al. (2008), and we roughly followed these parameters using a Python notebook sourced from Walker et al. (2020) studying this same scenario.

The pulse lasts 600 seconds, which includes a ramp-up phase, a flat-top phase, and a ramp-down phase. We focused our optimization only on the "flat-top" phase of the pulse, which corresponds to time t = 80s to 480s. This is because the ramp-up and ramp-down phases are constrained by the physical requirements of the reactor, and most of the reward (which largely comes from the parameter Q_f, is collected during the flat-top phase.

Category	Variables	Description
Energy & Confinement	Q_f, H_{98}, τ_E	Fusion gain, confinement quality factor, energy confinement time
	W_{th}, dW_{th}/dt	Thermal stored energy and its rate of change
	P_{aux}, P_{α}, P_{ohmic}	Auxiliary, alpha, and ohmic heating power
	$P_{\text{LH margin}}$	Ratio of total heating to L-H threshold power
Safety & Stability	q_{95}, q_0, q_{min}, $\rho_{q_{\text{min}}}$	Edge and core safety factors
	β_N, l_{i3}	Normalized beta, internal inductance
	f_{GW}, f_{NI}, f_{BS}	Greenwald fraction, non-inductive and bootstrap current fractions
Plasma Profiles	T_e, T_i, n_e at $\rho = 0.2, 0.5, 0.8$	Electron/ion temperature and density at three radial locations
	q, s at $\rho = 0.2, 0.5, 0.8$	Safety factor and magnetic shear profiles
Derived Quantities	$T_{e,\text{pk}}$, $T_{i,\text{pk}}$, $n_{e,\text{pk}}$	Core peaking factors (core value / volume average)
	$T_{e,\text{avg}}$, $T_{i,\text{avg}}$, $n_{e,\text{avg}}$	Volume-averaged temperature and density
Control Inputs	P_{ECRH}, P_{NBI}	Previous-step actuator values (MW)

Table 1: State vector components grouped by category. Scalars are evaluated at the start of each decision interval; profile variables are sampled at three normalized radial positions $\rho = 0.2, 0.5, 0.8$. The previous action is included to encourage temporal smoothness in the actor's output.

4.1 Data Collection

Using a Latin hypercube sampling (LHS) method, we generated a dataset of 10,000 fueling trajectories, each with modified ECRH and NBI schedules. Specifically, at time intervals of 20 seconds from t=80 to t=480 (resulting in 21 total actions per trajectory, and 210,000 total transitions in the dataset), a "delta" value is sampled (between 0 and 1) for both ECRH and NBI using LHS. Then, this delta value is scaled to be between -8 MW and 8 MW. Then, the sampled delta value is applied to the value for ECRH and NBI from the *previous* timestep. ECRH was started with a default value of 20 MW, and NBI was started with a default value of 16.5 MW. Additionally, ECRH was bounded to be within 0 to 40 MW, and NBI was bounded to be within 0 to 33 MW. See Figure 1 for an example schedule generated with this method.

Note that while actions are sampled at times t=80 to t=480, the specified ECRH and NBI values are applied to the simulation with a 20 seconds delay (i.e. at times t=100 to t=500). The simulation software then linearly interpolates between adjacent scheduled ECRH and NBI values. For example, if a value of 25 MW is chosen for ECRH at time t=80, this is applied to the simulation at t=100 (and the default value of 20 MW is applied to the simulation at t=80). Then, from times t=80 to t=100, the simulation linearly interpolates between these two values (i.e. ECRH would be 22.5 MW at t=90). Therefore, the action chosen at t=80 starts to affect the simulation at the very next simulation timestep, which is the expected behavior.

Then, the coupled Tokamaker-TORAX simulation described in Section 3.1 was used to simulate the generated pulse for each of these fueling trajectories. At each timestep (from t=80 to t=480), 33 scalar parameters were extracted from the simulation. The chosen parameters capture behavior regarding generated energy, energy consumed, fueling rates, how well the plasma stayed confined, and several safety factors. Some parameters were further expanded to be calculated at multiple radii within the "donut"-shaped tokamak. The state also included the *previous action*, i.e. the values chosen for ECRH and NBI at the prior timestep. Eventually, this yielded a 43-dimensional state vector (see Table 1 for a grouped summary). The associated reward given this state was calculated using the method described in Section 4.2.

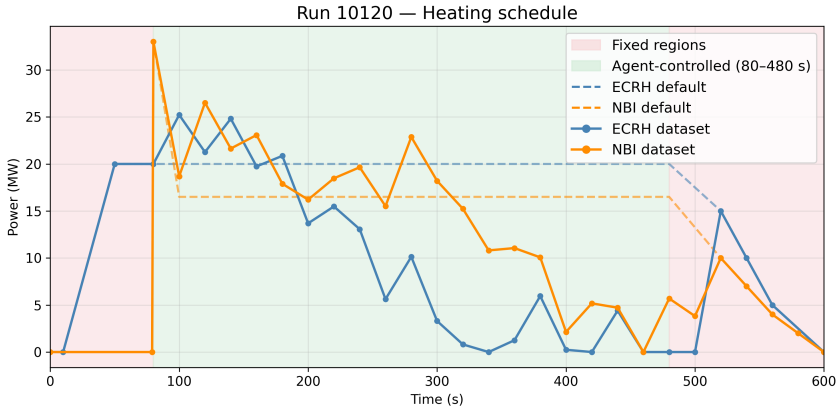

Figure 1: Example randomly sampled fueling schedule from offline dataset.

4.2 Reward Formulation

A "successful" pulse is characterized by a high Q_f, corresponding to high energy efficiency, safe operation, and minimal flux consumed. There are several operational limits of the ITER tokamak, including a minimum value for q_95 and a maximum value for f_{GW}. The q_95 value is a safety factor which is a proxy for plasma stability. f_{GW} is the Greenwald fraction, which represents the density limit of the plasma. If the plasma exceeds this limit, it could potentially lose confinement. If plasma instabilities develop or if plasma confinement is lost, disruptions that stress or damage the machine may occur. In addition to these safety requirements, it is also important to minimize the total flux consumed over the course of a pulse. When current is driven through poloidal coils wrapping around the tokamak, magnetic flux is induced. This process is essential to achieving fusion, and crucially, there is a finite amount of flux available. Therefore, we penalize the agent for using it excessively.

The reward at each decision step is composed of a step reward, safety penalties, and a terminal bonus at the final timestep:

$$\begin{aligned} r_t = w_r \cdot \ln\big(\bar{Q}_{f,[t,t+\Delta t]} + 1\big) \\ - w_{q_{95}} \sum_{t'=t}^{t+\Delta t} \max\big(q_{95}^{\min} - q_{95}(t'),\, 0\big) \\ - w_{\text{fgw}} \sum_{t'=t}^{t+\Delta t} \max(f_{\text{GW}}(t') - f_{\text{GW}}^{\max},\, 0) \end{aligned} \tag{5}$$

where $\bar{Q}_{f,[t,t+\Delta t]}$ is the mean fusion gain over the interval, and the penalty terms accumulate violations at each simulation timestep within the interval. At the final step, an additional terminal bonus is added:

$$r_{\text{terminal}} \;+\!=\; w_Q \cdot \bar{Q}_{\text{f, flat-top}} - w_\Phi \cdot \Phi_{\text{consumed}} \tag{6}$$

where $\bar{Q}_{\text{flat-top}}$ is the time-averaged fusion gain over the entire flat-top phase and Φ_{consumed} is the total flux consumed during the pulse.

Distributions of each of the weighted components in reward are included in Figure 2, and the specific weights used in all experiments are included in Table 2. See Section 6 for more information on how weight values were chosen.

4.3 Model Evaluation

The model was evaluated using a closed-loop evaluation scheme. Specifically, the simulation was run in increments of 20 seconds, and after each increment, a state was extracted from the simulation

Parameter	Symbol	Value
Step reward weight	w_r	1.0
q_{95} penalty weight	$w_{q_{95}}$	1.2
f_{GW} penalty weight	$w_{f_{GW}}$	2.0
Flat-top Q bonus weight	w_Q	1.0
Flux penalty weight	w_Φ	0.012

Table 2: Reward function weights used in all experiments.

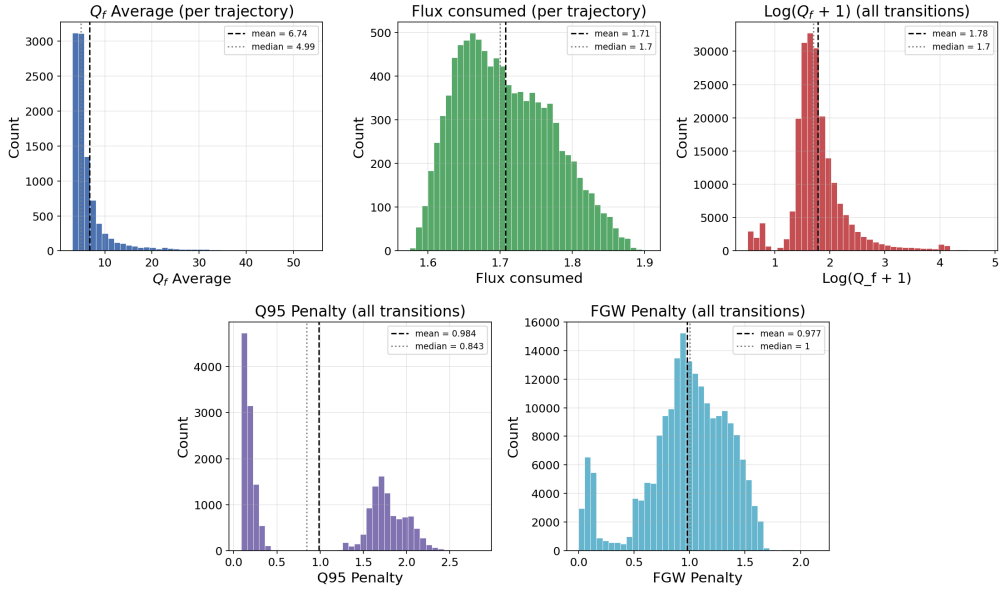

Figure 2: Histograms of reward components after re-weighting.

parameters and passed into the actor. The actor would then output a two-dimensional action with values for ECRH and NBI, which were then stored in global ECRH and NBI schedules. After updating the schedule, the entire simulation was run again from $t = 0$ to the *next* timestep using the accumulated sequence of previously selected actions. This process was repeated for all decision timesteps, from $t = 80$ to $t = 480$.

Then, a final simulation using the entire actor-determined ECRH and NBI schedule was run from $t = 0$ to $t = 600$, and the output of this simulation was used to compute rewards for evaluation.

5 Results

A tabular summary of all of these results is in Table 3.

5.1 Quantitative Evaluation

All actor ECRH and NBI schedules were compared against the baseline described in Budny et al. (2008), which quickly increases ECRH to 20 MW at time t=50, and NBI to 33 MW at time t=80, before then gradually decreasing these values back to 0 at time t=520. A graph of this schedule is in Figure 3.

This default configuration achieved a peak Q_f of 4.8 (see Figure 4), and a final reward of 8.66 using our reward function.

An initial IQL training run on our dataset of 10,000 trajectories with the following hyperparameter values: $\zeta = 0.8$, $\beta = 3.0$, and lr=1×10^{-4} yielded a far higher reward of around 35.27. The trained agent's ECRH and NBI schedules when evaluated are in Figure 5. The agent was also able to achieve a peak Q_f around twice that of the baseline, at around 10 (see Figure 4).

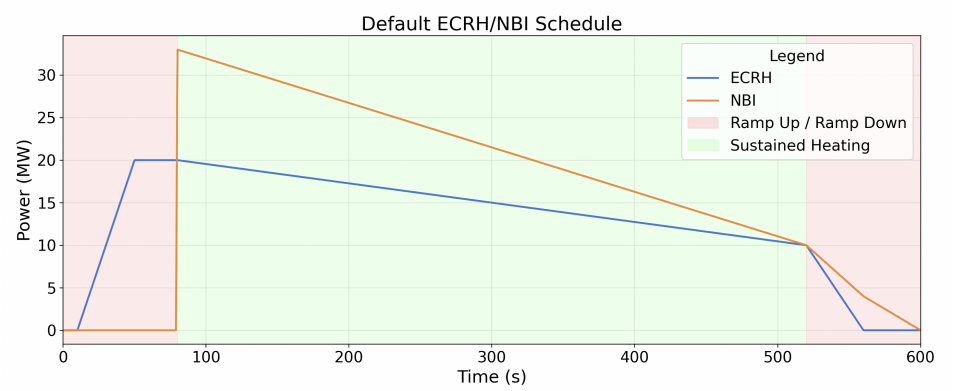

Figure 3: The default ECRH and NBI heating schedule.

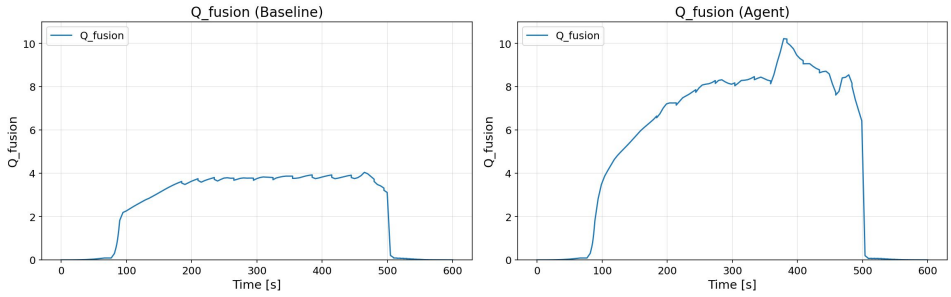

Figure 4: Comparison of Q_f for agent versus baseline. Note that peak Q_f nearly doubles.

5.1.1 Hyperparameter Grid Search

Additionally, after using these baseline values, hyperparameter grid search was run on both ζ and β to further optimize the model. Because of the high computational cost associated with closed-loop evaluation, a very coarse-grained grid was used, testing the following values for ζ: $[0.7, 0.8]$ and the following values for β: $[1, 3, 10]$, yielding 6 total hyperparameter configurations. The best hyperparameter combination found was $\zeta = 0.7$ and $\beta = 1$, which yielded a total reward of 46.69, and a peak Q_f of 22.0, but this was not sustained throughout the pulse, and is therefore not a viable policy.

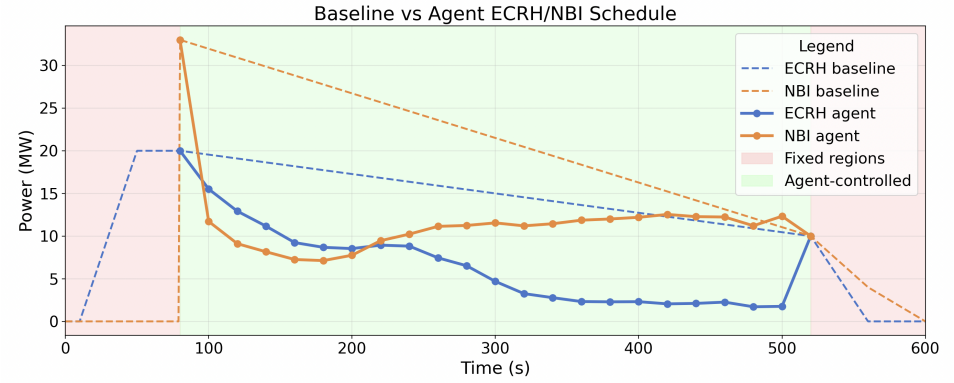

Figure 5: The agent's ECRH and NBI heating schedule.

Method	Reward	Peak Q_f
Baseline Heating Schedule	8.66	4.8
IQL Agent w/ Default Hyperparameters	35.27	10.0
Hyperparameter Grid Search	46.69	22.0
Other explored architectures (Section 5.3)		
BC	34.5	18.8
CQL	12.9	4.6
TD3+BC (results indicate reward hacking)	40.1	44.7

Table 3: Summary of results.

5.2 Qualitative Analysis

In addition to achieving better Q_f than the baseline, the model's generated heating schedule also looks quite smooth, with fairly small changes from timestep to timestep compared to the original 8 MW jumps from the original training dataset (see Figure 1). This implies that including the previous action enabled the actor to learn this additional behavior.

Additionally, while it is difficult to confirm whether this discovered schedule is truly physically plausible, or simply exploiting inaccuracies in the simulation software, the results look quite promising. Specifically, comparing the graphs of several other simulation parameters (such as various safety factors and flux consumption) between the default heating schedule (see Figure 3) and the actor's show that they are quite similar. As an example, a side-by-side comparison of the graphs of two safety factors (see Figure 7): q_{95} and q_0 show that they are near-identical between the two schedules, with q_{95} staying above its prescribed minimum of 3.0 for most of the 600 second pulse, and q_0 remaining near its prescribed value of 1.

5.3 Model Architecture Comparison

After training our IQL model, trained other models on the collected trajectories to evaluate if they performed better on the task: a Behavioral Cloning model (BC), which tries to imitate the actions in the dataset; a Conservative Q-Learning (CQL) model, which adds a regularization term to standard Q learning to penalize out-of-distribution actions; and a TD3 model with behavior cloning (TD3BC), another offline RL algorithm described in Fujimoto and Gu (2021) which has twin critics updated more frequently than the policy and is adapted for offline RL through behavior cloning. The performance of these agents is shown in Table 3.

The variation in produced heating schedules, shown in Figure 6, shows the diversity of methods which can be applied to the same goal. The policy of the BC agent is smooth, which is expected as it would be trying to reproduce the mean behavior from the dataset. The CQL's added regularization term likely over-penalized ECRH actions, which are rarer in the dataset, thinking that any action above zero would be out-of-distribution; this contributed to its poor performance. The TD3BC model's large volatility likely indicates that it discovered a reward hack, where rapidly cutting heating energy use as energy production peaks could increase the Q_f net energy measurement. This means that further modifications to the reward function, or to the simulation software itself (which is not infallible), would be required.

6 Discussion

A key challenge early in this project was ensuring that the reward function was properly tuned such that the agent balanced achieving a high Q_f without compromising on safety. For example, early iterations of the reward function directly scaled Q_f without applying an $\ln$ transformation, but this would have likely made the agent forcefully try to increase energy efficiency, potentially creating unsafe or even unphysical plasma configurations in the process. Simply increasing the weighting of the safety penalty would not have solved this, as the safety penalty (which is accumulated throughout each 20 s time step within the 600 s pulse) would have then overwhelmed the reward computation, removing all useful signal from Q_f. An elegant solution was introducing an $\ln$ transformation to Q_f, as the reward would still increase with increasing Q_f, but increase would slow as Q_f increased further. To determine the numerical weight values themselves, we noted that $\ln(5+1) \sim 1.8$, where $Q_f = 5$

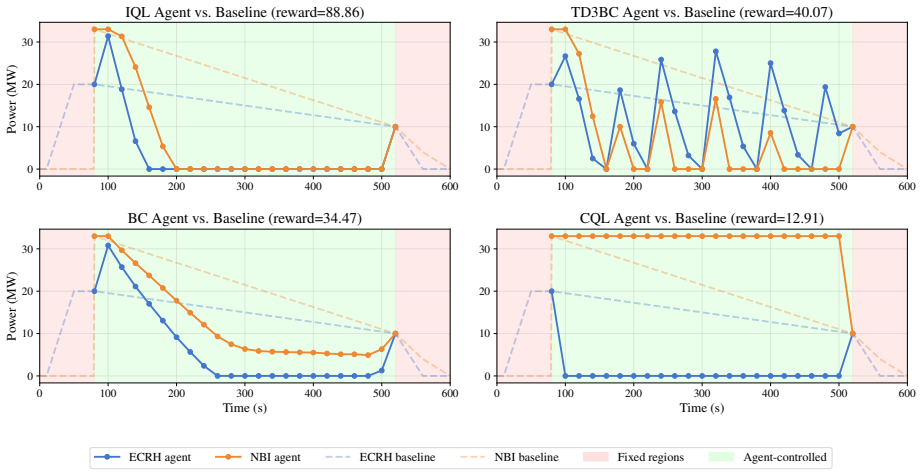

Figure 6: Comparison of the heating schedules for different architectures.

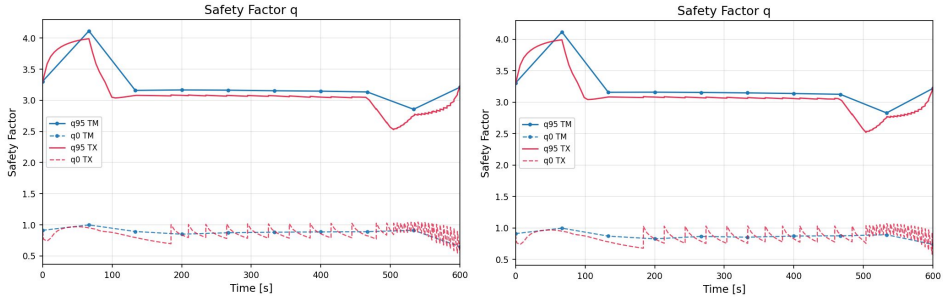

Figure 7: Comparison of safety factors for baseline (left) and agent (right) heating schedules. Note that these figures look very similar (although they are still different!), indicating that the agent achieved higher energy efficiency without compromising safety.

is the baseline peak Q_f. We then aimed to weight all safety penalties and the flux consumption such that their magnitudes were around 1.8, ensuring that a high Q_f achieved via unsafe means or by consuming excess fuel would be no better than the baseline schedule (see Table 2 for our final weight values). This method of weighting seems to have been effective, given that our model's output ECRH and NBI schedule yielded very similar safety outcomes to the baseline (see Figure 7).

Another challenge was ensuring that the dataset included sufficient coverage of the action space without introducing unstable or unphysical simulations into the data. Initial attempts at trajectory collection randomly sampled ECRH and NBI values within a specified range at each timestep, with no regard for the value from the previous timestep. However, this yielded unstable trajectories that were liable to fail before completing, which is why we made use of the "delta-sampling" method described in Section 4.1. Additionally, the original delta was scaled between -2 MW and 2 MW, but the agent trained on this dataset only obtained a reward of 14.59, which is nominally higher than the baseline reward of 8.66. This demonstrates how IQL benefits from greater variation in the dataset.

Finally, deciding on a reasonable configuration for the state vector also took significant effort. The original "data_tree" outputted by the simulation software included 102 scalar parameters describing various aspects of the simulation. However, most of these parameters could easily be derived from others, meaning that creating a state vector out of all 102 values would yield large amounts of redundant information. Therefore, we sought to select a linearly independent set of state parameters such that each new parameter added new information. Interestingly, almost all the state parameters had a negative correlation with reward (see Figure 8), highlighting that more state parameters could potentially be pruned without losing much information.

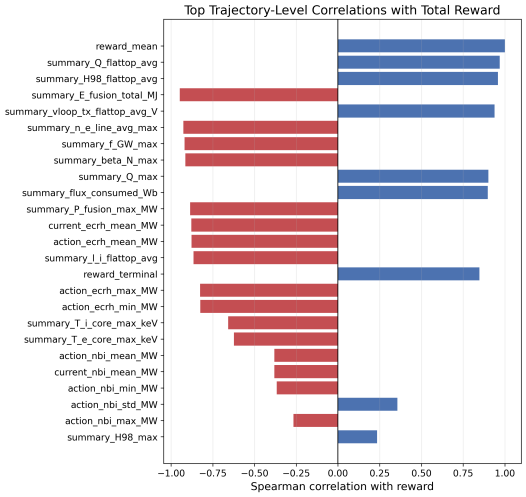

Figure 8: Correlation between trajectory-level state parameters and total reward.

We noted that fusion rate is most sensitive to peaked profiles, which is why we included several for temperature and density peaking factors, calculated using the core value divided by the average. We also included magnetic shear, since this gives information about current profile shape, which drives plasma stability. Additionally, as mentioned previously, including the prior action seemed to help "stabilize" the agent heating schedules, ensuring that they do not oscillate drastically between timesteps. For example, while the input dataset had jumps of up to 8 MW between adjacent timesteps, our final generated heating schedule did not have a jump exceeding 2.5 MW. This is further supported by the fact that removing this from the state led to a far lower reward (around 13.37, compared to our original reward of 35.27).

7 Conclusion

We demonstrate that an RL agent trained using IQL can discover tokamak-heating schedules that significantly outperform baseline heating schedules constructed using human-designed heuristics. Our actor doubles the energy efficiency of the base heating schedule while maintaining safe plasma configurations.

However, there are many areas for further improvement. For one, our reward function focused on a small subset of the total simulation output, and could be further tuned to include more aspects of the generated pulse. Additionally, our IQL algorithm made use of a deterministic actor, which may be better suited for highly sensitive applications like a nuclear reactor. It may be interesting to see how introducing non-determinism via an action-distribution would affect the agent's performance. Additionally, given IQL's sensitivity to the variance in the dataset, we could observe even greater performance with a larger dataset that more thoroughly samples the action space.

Overall, our work demonstrates that RL can be effectively used to optimize input heating schedules for a tokamak to greatly improve efficiency, furthering efforts towards commercially viable nuclear fusion.

8 Team Contributions

- **Deniz Yilmaz:** Dataset collection, RL state and reward design, help with closed-loop evaluation, paper writing.
- **Sameer Agrawal:** IQL implementation, hyperparameter grid-search, closed-loop evaluation.
- **Siddharth Bhatia:** IQL optimization, model architecture exploration, grid search, paper writing.

Changes from Proposal Overall, our work split remained quite similar to the proposal. Due to Deniz's greater familiarity with nuclear fusion, she ended up working on the domain-specific aspects of the project, like designing the state vector and reward function. Sameer had access to some of Stanford's HPC clusters and a private, high-performance 64-core machine through his research that he used to train IQL and run various computationally-intensive experiments.

9 Acknowledgements

We would like to thank the CS224R teaching team, particularly Prof. Chelsea Finn and Marcel Torne for their guidance and support throughout the quarter.

We would also like to thank Dr. Oak Nelson and Freddie Sheehan, two members of the Columbia Fusion Research Center who provided us with the simulation software and were great sources of fusion-related knowledge during this project.

References

R.V. Budny, R. Andre, G. Bateman, F. Halpern, C.E. Kessel, A. Kritz, and D. McCune. 2008. Predictions of H-mode performance in ITER. *Nuclear Fusion* 48, 7 (may 2008), 075005. doi:10.1088/0029-5515/48/7/075005

Ian Char, Joseph Abbate, Laszlo Bardoczi, Mark Boyer, Youngseog Chung, Rory Conlin, Keith Erickson, Viraj Mehta, Nathan Richner, Egemen Kolemen, and Jeff Schneider. 2023. Offline Model-Based Reinforcement Learning for Tokamak Control. In *Proceedings of The 5th Annual Learning for Dynamics and Control Conference (Proceedings of Machine Learning Research, Vol. 211)*, Nikolai Matni, Manfred Morari, and George J. Pappas (Eds.). PMLR, 1357–1372. https://proceedings.mlr.press/v211/char23a.html

Jonathan Citrin, Ian Goodfellow, Akhil Raju, Jeremy Chen, Jonas Degrave, Craig Donner, Federico Felici, Philippe Hamel, Andrea Huber, Dmitry Nikulin, David Pfau, Brendan Tracey, Martin Riedmiller, and Pushmeet Kohli. 2024. *TORAX: A Fast and Differentiable Tokamak Transport Simulator in JAX*. arXiv:2406.06718 [physics] doi:10.48550/arXiv.2406.06718

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. 2022. Magnetic Control of Tokamak Plasmas through Deep Reinforcement Learning. *Nature* 602, 7897 (Feb. 2022), 414–419. doi:10.1038/s41586-021-04301-9

Luigi Emanuel Di Grazia, Federico Felici, Massimiliano Mattei, Antoine Merle, Pedro Molina, Cristian Galperti, Stefano Coda, Basil Duval, Antoine Maier, Adriano Mele, et al. 2024. Automated shot-to-shot optimization of the plasma start-up scenario in the TCV tokamak. *Nuclear Fusion* 64, 9 (2024), 096032.

S. Dubbioso, L. E. Di Grazia, G. De Tommasi, M. Mattei, A. Mele, and A. Pironti. 2022. Vertical stabilization of tokamak plasmas via extremum seeking. *IFAC Journal of Systems and Control* 21 (2022), 100203. doi:10.1016/j.ifacsc.2022.100203

Scott Fujimoto and Shixiang Shane Gu. 2021. A Minimalist Approach to Offline Reinforcement Learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Vincent Graber and Eugenio Schuster. 2019. Nonlinear Adaptive Burn Control of Two-Temperature Tokamak Plasmas. In *2019 IEEE 58th Conference on Decision and Control (CDC)*. 3239–3244. doi:10.1109/CDC40024.2019.9029943

Elkhan Richard Sadik-Zada, Andrea Gatto, and Yannic Weißnicht. 2024. Back to the future: Revisiting the perspectives on nuclear fusion and juxtaposition to existing energy sources. *Energy* 290 (2024), 129150. doi:10.1016/j.energy.2023.129150

Michael Walker, Peter De Vries, Federico Felici, and Eugenio Schuster. 2020. Introduction to Tokamak Plasma Control. In *Proceedings of the 2020 American Control Conference (ACC)*. 2901–2918. doi:10.23919/ACC45564.2020.9147561