Reinforcement Learning Control for Natural Circulation in a Marine Pressurised Water Reactor Cooling System

Felipe Montana[1], Will Jacobs[1], Visakan Kadirkamanathan[1], Gary Brooks[2], and Andy Mills[1]

[1] *University of Sheffield, Sheffield, S1 3JD, UK*
f.montana-gonzalez@sheffield.ac.uk
w.jacobs@sheffield.ac.uk
visakan@sheffield.ac.uk
a.r.mills@sheffield.ac.uk

[2] *Rolls-Royce Plc, London, N1 9FX, UK*
gary.brooks5@rolls-royce.com

ABSTRACT

In safety-critical systems, a system fault response can lead to a system shutdown. While safe at a component level, this poses safety challenges for the system as a whole, requiring an additional system to manage this process. In pressurised water reactor (PWR) submarines a loss of coolant pump can force a shutdown by dropping the control rods, referred to as SCRAM. To avoid this, a possible response is to use natural circulation, a degraded operating mode characterised by strong non-linearities in system dynamics, to provide a limited level of functionality. Under these conditions, conventional model-based control approaches become difficult to apply, as the assumptions underlying nominal system models no longer hold. This paper investigates the feasibility of using reinforcement learning (RL) as a fault-response control strategy for systems operating under degraded and poorly modelled conditions. RL provides a data-driven framework capable of learning control policies directly from a black-box model or simulator, without requiring an explicit analytical model. However, when applied in a safety-critical fault management context, understanding and validating the learnt control policy is essential. We analyse the policy learnt through RL by approximating it with a transparent surrogate model and through visualisation of the policy actions. We further assess the robustness of the policy to modelling errors, providing insight into its sensitivity to discrepancies between the simulated environment and the real system. The proposed approach is evaluated using a simplified submarine reactor cooling loop model that captures key features of fault-induced operation, including changes in system dynamics due to platform pitch and cascading faults. The results demonstrate the potential of reinforcement learning for interpretable control under faulted conditions.

1. INTRODUCTION

Classical control methods are ubiquitous, having reached a level of maturity that enables their widespread application across many domains. Nevertheless, many of these approaches make assumptions, such as linearity, time-invariant dynamics, etc., that may not hold in multi-variable systems with strong coupling and non-linearity. This is especially the case for safety-critical systems operating under faulted conditions, where the system dynamics can change significantly and become highly nonlinear. While iteratively solved fluidic models, e.g., RELAP, are available and can be used for control design, when analytical models are unavailable the design of such control strategies is particularly challenging. On the other hand, control based on machine learning and AI is an emerging area that has shown the potential to be used in a variety of applications (Heess et al., 2017; Tang, Rabault, Kuhnle, Wang, & Wang, 2020; Degrave et al., 2022). Although it has been shown that these types of controllers could be inferior to model-based control methods in many applications where the dynamics can be approximated by a model (Recht, 2019), they have potential to solve many real-life problems where models are not available.

In this paper, we investigate whether reinforcement learning (RL) can serve as a viable fault-response control strategy when system conditions are degraded or poorly characterised. Unlike conventional model-based approaches, RL does not require an explicit analytical model of the system. Instead,

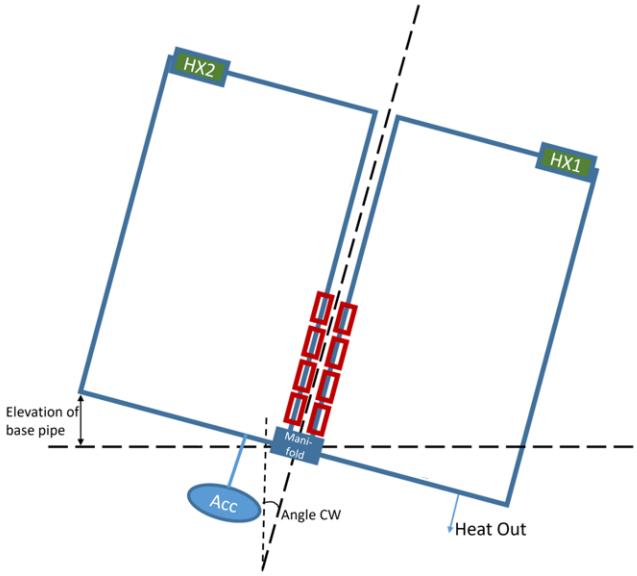


Figure 1. Schematic representation of system used for natural circulation. Each leg contains a heat exchanger (HX) and four heaters (red rectangles). The accumulator (Acc) provides passive coolant injection under low-pressure conditions. The manifold serves as a flow distribution junction connecting the loop branches.

it learns a control policy through interaction with a simulator or black-box model, making it particularly attractive for scenarios where accurate system models are unavailable or unreliable. While not having the guarantees and interpretability of model-based control methods is not necessary for some applications, for safety-critical systems, some features are desirable or necessary:

- Robustness: the performance of the controller or policy should meet the specifications in the presence of a set of changes in the system, external disturbances, etc.
- Safety: when online learning is used, the controller must avoid damaging the system or its surroundings.
- Reliability: Once a controller is obtained, it must always provide a control signal which drives the system towards the goal in a repeatable and specified manner.
- Interpretability: The ability to understand the response of the controller for different inputs is required for the deployment of these controllers in many applications. Understanding the controller is not only required to predict unexpected behaviours but to decrease aversion to these methods, i.e., lack of trust by end users.
- Verifiable: For safety-critical tasks, the capacity to rigorously verify the possible behaviours of the system under control is critical.

In this paper, we focus on the control of natural circulation in a marine pressurised water reactor (PWR) cooling system. This is an example of a safety-critical system where the transition to natural circulation can occur following a fault that results in the loss of active circulation. The dynamics of natural circulation are highly nonlinear and can be affected by changes in system conditions, such as platform pitch, making this task difficult to control using conventional model-based approaches, where model-based refers to control techniques grounded in mathematical models of the system, as distinct from model-based RL methods discussed in Section 3.1. Our results illustrate the potential of RL for control under faulted conditions. Moreover, they show how the learnt policy can be analysed to understand the response of the agent to different system states.

2. CASE STUDY: NATURAL CIRCULATION IN PRESSURISED WATER REACTORS

Natural circulation is the flow of fluid due to the gravitational forces acting on density changes caused by temperature differences. The system requires one component to increase the temperature of the fluid and another to decrease it. This is a phenomenon that has been considered as an alternative to active systems together with accumulators, condensation and evaporative heat exchanges in reactors (Reyes, 2005). Our work explores the use of natural circulation in the faulted condition of an active PWR.

We consider a simplified submarine reactor cooling loop model that captures key features of marine systems such as changes in system dynamics due to platform pitch. This model, presented in Fig. 1, consists of two legs each one with a heat exchanger that represents a steam generator system used to remove heat from the liquid and 4 heaters that can be turned on/off independently that are used to heat the liquid. These

heaters represent a simplified, discretised approximation of the reactor thermal power output, analogous to control rod positions. Note that in this case, all the actions are discrete. A single temperature measurement between the heaters and heat exchangers is available for each leg.

The natural circulation is achieved by heating the liquid with a controllable set of heaters and cooling it with the heat exchangers. The system can tilt along the axis vertical to the 2-D plane shown in Fig. 1. The angle of the system has an impact on the temperature, the potential energy at the point of density change, and flow in the legs. Without control of the heaters, an angle change can drive the system to unsafe states where the temperature of the liquid exceeds a predefined limit as shown in Fig. 2. The problem addressed in this paper is to learn a control policy with RL to turn on/off the heaters and maintain the temperature of the system below a predefined limit regardless of the angle of the system.

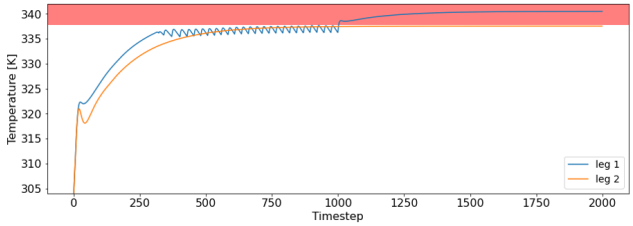


Figure 2. Temperature of both legs of the system. At time step 1000, the system is swayed causing the temperature to increase and exceed the limit (red shaded area). The response in each leg is due to the angle of the system.

Developing analytical models for phase-change processes is a challenging task (Gomez, Bures, & Moure, 2019). Therefore, the system described above was modelled and simulated in Simscape, a toolbox within of Simulink's environment. This tool was selected due to its library that permits to create and simulate two-phase thermal flow systems. To improve the speed of the simulation, the model was exported as a C library and compiled into a dynamic-link library (DLL).

3. REINFORCEMENT LEARNING

Reinforcement learning is a paradigm concerned with learning how to map situations to actions such that a numerical performance measure (called reward) is maximised (Sutton, Barto, et al., 1998). In contrast to conventional model-based control methods, RL methods are capable of learning through the interaction with the environment. In other words, a model of the system is not required to learn a control strategy. This aspect is desirable in situations where designing or maintaining a model of the system is not possible or difficult because the behaviour of the system is expected to change due to degradation, time-varying constraints, system faults, etc.

In RL, an agent or controller learns a policy that maps the state of the system to actions that maximise the expected long-term reward. These policies are learnt by the agent based on interactions with the environment or system and rewards obtained after taking actions, see Fig. 3. To learn a policy, the goal of the agent is to maximise the total amount of reward it receives. This means maximising not immediate reward, but cumulative reward in the long run. Formally, this is the expected discounted return: the sum of future rewards, weighted to prioritise near-term rewards over distant ones. Note that in many cases, the agent cannot directly observe the state of the system, but instead receives an observation that is a function of the state. In the rest of the paper, we will refer to the observation as the state of the system for simplicity. Similarly, we use the term system and environment interchangeably.

3.1. Reinforcement Learning Training and Algorithm Selection

The agent can be trained off-line or on-line. In off-line training, the agent interacts with a simulator of the environment, e.g., a physics simulation engine. On the other hand, in on-line training, the agent interacts directly with the physical system. However, due to the large number of interactions required to learn a policy, on-line training is often impractical for safety-critical or high-cost physical systems, where excessive interactions may risk damage, accelerate wear, or violate operational constraints.

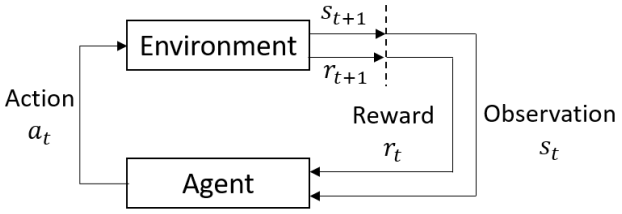


Figure 3. Interaction of agent and environment (system). The controller applies an action a at time t, based on an observation s_t, that drives the system into a new state which generates a new observation. The new observation s_{t+1} and reward r_{t+1} obtained from reaching such a state are used to train the agent.

Off-line training has several advantages over on-line training. When trained off-line, the agent learns by exploring the action space. When some of these actions are applied to the system, this can be driven to unsafe states where the system can be damaged. Another advantage is the ability to train the agent to respond to different faults. If a simulator with the capability of injecting faults is available, multiple agents can be trained a priori to respond to different situations. The main difficulty of off-line training is the need of a model or simulator that accurately reflects the behaviour of the real system.

Multiple reinforcement learning algorithms have been proposed (Lillicrap et al., 2015; Haarnoja, Zhou, Abbeel, &

Levine, 2018; Wang et al., 2020). These can be divided into model-free and model-based approaches. The main difference between these approaches is that model-based methods create an internal model of the transitions of the environment or system by interacting with it. This model is then used to predict the next system state and train the agent. Since a model is learnt, model-based approaches are, in general, harder to train.

We use the Proximal Policy Optimisation (PPO) (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) algorithm. PPO is a model-free state-of-the-art reinforcement learning algorithm that has been shown to be effective in multiple applications such as control of physical systems (Degrave et al., 2022) and fine-tuning of large language models (Ouyang et al., 2022).

3.2. Proximal Policy Optimisation

This section presents an overview of the Proximal Policy Optimisation (PPO) algorithm. This is required to understand the results presented in Section 4.

PPO is an actor-critic method. This means that the agent consists of two components, an actor and a critic. The actor learns a policy $\pi : S \to P(A)$ that maps the state $s \in S$ of the system to a probability distribution over actions $a \in A$. The critic learns a value function $V^\pi : S \to \mathbb{R}$ that estimates the expected discounted return from a given state when following the policy π.

In PPO, the actor and the critic are neural networks. During training, the critic is trained to minimise the error between the predicted value and the actual reward obtained from the environment. The actor is trained to maximise the expected reward predicted by the critic. In contrast to other methods, PPO restricts the change in the policy at each iteration by clipping the probability ratio between the new and old policies, achieving better learning stability.

The training of the agent consists of the following steps:

1. The actor and critic are randomly initialised.
2. System is simulated for one episode that consists of n timesteps. At each timestep, the actor receives the current system state information and returns an action which is applied to the system.
3. States, actions and rewards are collected in a buffer during the episode.
4. The collected data is used to update the networks such that the critic accurately estimates the expected discounted return of following the current policy from each visited state, and the actor is updated towards actions whose actual return exceeded the critic's prediction.
5. Steps 2 to 4 are repeated until the total reward obtained during the episodes converges or a predefined number of episodes is reached.

Once the agent is trained, the actor can be used as a controller. Given a state of the system, the actor returns a probability distribution over actions. For a deterministic policy, the action with the highest probability is always selected, i.e., $a_t = \arg\max_a \pi(a|s_t)$. This is called a greedy policy. For a stochastic policy, an action is sampled from the distribution returned by the actor.

4. RESULTS

We trained an agent to maintain the temperature of the system below a predefined limit while maximising the generated power (heaters on), regardless of the tilt angle of the system as described in Section 2. The reward function used to train the agent is defined as follows. Let N be the number of heaters, and let $a_t \in \{0, 1\}^N$ be the action taken by the agent at time t, where $a_{t,i} = 1$ if the heater i is turned on and $a_{t,i} = 0$ otherwise. The reward at time t is defined as:

$$r_t = \begin{cases} n_t^{on}, & \text{if } T_t < T_{limit} \\ n_t^{on} - c, & \text{if } T_t \geq T_{limit} \end{cases} \tag{1}$$

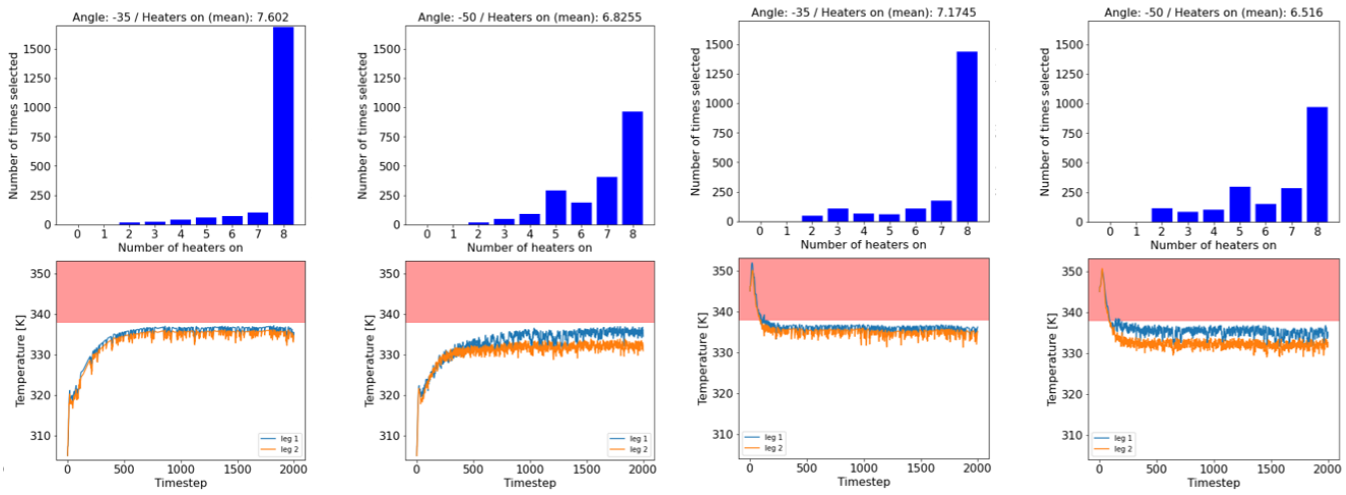
where $n_t^{on} = \sum_{i=1}^{N} a_{t,i}$ is the total number of heaters turned on and c is a constant value that penalises exceeding the temperature limit. This limit is varied across experiments to evaluate the agent's behaviour under different operating constraints.

In our problem, the action space is defined by all possible combinations of heater on/off states. For the observations, the agent receives the system angle and the temperatures for each leg at the current and two previous timesteps, allowing it to infer the temperature rate of change. During training, the initial temperature and angle are randomly selected at each episode.

4.1. Agent Performance

Once trained, we started testing the agent from different initial conditions, i.e., different temperatures and angles. The results are shown in Fig. 4. The agent is able to maintain the temperature below the limit for any condition (only two platform pitch angles are shown for brevity). The top plots in Fig. 4 show the histogram of the values of n^{on} during the simulation. It can be seen that the number of heaters turned on is reduced when the system starts with a higher temperature.

We then analyse the control of the system under different conditions and policies. Figure 5 shows the temperature of the system and the number of heaters turned on over time. Initially, an agent without training is used to control the system. As shown in the figure, the number of heaters turned on oscillates with an average of 5 heaters on. Then, a partially trained agent is used to control the system. As expected, the



(a) Cold start

(b) Over temperature start

Figure 4. Temperature of the system controlled by a RL agent from different initial conditions. The top row shows the histogram of the values of n^{on} during the simulation, i.e., the number of heaters turned on at different time steps. The red shaded area indicates an undesirable temperature. The left and right plots, in (a) and (b), show the system at different angles, $-35°$ and $-50°$, respectively.

mean number of heaters turned on increases. At timestep 250, a fully trained agent is used to control the system. Note that the average number of heaters increases again. At this point of the simulation, only non-greedy actions have been used. This means that the action is sampled from the distribution returned by the actor instead of selecting the action with the highest probability. This also explains why a steady state is never reached. At timestep 390, the greedy policy is used. This reduces the variability of the number of heaters turned on. The system is rotated counterclockwise from an angle of 0 degrees to an angle of 35 degrees at timestep 450. The change increases the temperature of leg 2, forcing the agent to reduce the number of heaters turned on. Finally, at timestep 580, a heat exchanger fault is injected. This reduces the cooling capacity of the system. The agent responds by further reducing the number of heaters turned on.

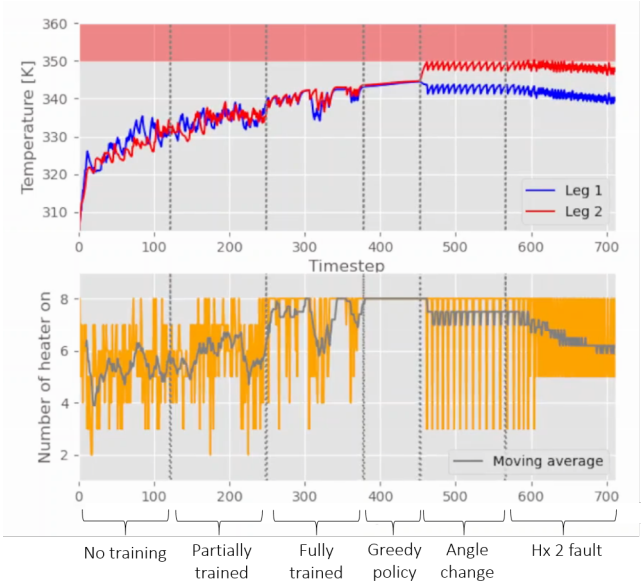


Figure 5. Temperature of the system and number of heaters turned on over time under different conditions.

4.2. Robustness Analysis

As explained in Section 3, the agent is trained off-line using a simulator of the system. This means that there could be discrepancies between the simulated environment and the real system. This difference, called the reality gap, often results in poor performance if the policy is not robust to modelling errors (Christiano et al., 2016).

A robust control or policy can be defined as a policy with the ability to cope with variations and uncertainties in the environment (Moos et al., 2022). Robustness has been studied extensively in optimal control. Nevertheless, many of these techniques cannot be applied to RL because the agent learns a policy by interacting with the environment instead of using a mathematical model.

Common techniques used in other machine learning problems include ensemble methods (Hans & Udluft, 2010; Rajeswaran, Ghotra, Ravindran, & Levine, 2016) where multiple agents or environments are considered during training, adversarial training (Pinto, Davidson, Sukthankar, & Gupta, 2017; H. Zhang, Chen, Boning, & Hsieh, 2021) where an adversary is trained to learn optimal perturbations that im-

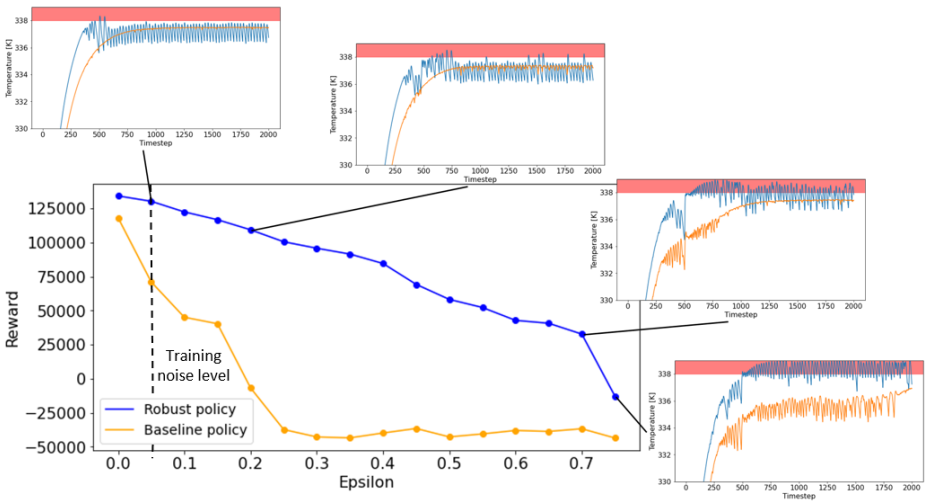


Figure 6. Reward collected during the simulation for different noise magnitudes ϵ for the robust (blue) and the baseline (orange) policy. The vertical dashed line marks the training noise level ($\epsilon = 0.05$, corresponding to 1 K). The four small plots show episodes at selected noise levels (marked on the x-axis), each using the robust policy. In these plots, the temperature of each leg is shown. As noise increases, temperatures begin to go over the maximum allowed temperature indicated by the red band, which results in a reduction of reward.

proves the robustness of the policy learnt by the agent. Other methods penalise large differences in actions for similar states (H. Zhang et al., 2020).

Another common technique is the fast gradient sign method (FGSM). Here, a perturbation is selected and FGSM finds the sign that modifies the input to maximise the loss based on the backpropagated gradients. In other words, the method uses the gradient of the loss with respect to the input data to adjust the input such that the loss is maximised. In (Mandlekar, Zhu, Garg, Fei-Fei, & Savarese, 2017), the idea of FGSM is used to generate different types of perturbations that can simulate uncertainty in the system's dynamics and perturbations in the observed state. Specifically, a loss function is used to create perturbations that move the state of the system such that the policy is encouraged to apply actions with larger norms, which can lead to instability.

Due to the simplicity of FGSM, we use it to generate input noise. The loss function was selected to generate input noise to make the agent believe that the system is in a state where the optimal action is uncertain, i.e., states with high action entropy. During training, the magnitude ϵ of perturbation is set to 0.05, which corresponds to noise amplitude of 1 K (the input temperature received by the agent can be 1 K different from the real value). During testing, we increase ϵ beyond the level used during training to compare the response of the robust agent to the baseline agent. Moreover, to demonstrate the effects of training with noise generated by the FGSM, an agent was also trained with random noise. These two policies are compared in Section 4.3.

Figure 6 shows the reward obtained by the agent, trained with the FGSM, for different levels of input noise. This is compared to a baseline agent trained without robustness considerations. Note that the policy is tested with noise levels higher than those used during training. For those cases, the agent is not able to maintain the temperature below the limit. Nevertheless, the robust policy performs better compared to the baseline policy as described below. This plot also shows the system temperatures for different noise levels when the robust policy is used. The results show that while the reward collected decreases (negative rewards are collected when the temperature goes above the limit) as the magnitude of the noise increases for both agents, the rate at which the reward decreases with the robust agent is slower. This result also shows that the learnt policy is robust to some noise magnitudes not observed during training. Specifically, after the transient, the temperature of the system is maintained below the limit when noise of magnitude 4 K ($\epsilon = 0.2$) is added to the input.

4.3. Policy Interpretation

Besides the policy robustness, understanding the response of the agent to different inputs is important for its deployment, especially for safety-critical systems. This is because the response of the agent may be unsafe. In terms of machine learning, interpretability is the ability to provide explanations, e.g., logical decision rules, in understandable terms (from the domain knowledge related to the task) to a human (Y. Zhang, Tiňo, Leonardis, & Tang, 2021).

To understand how FGSM affects the learning of a policy, we plot the greedy policy and value function V^{π} across the state space. Recall that the observation received by the agent is a 7-

dimensional vector with the current temperature, two lagged terms for each leg, and the system angle. For the purpose of visualisation, we project onto the current temperatures of each leg, as the angle remains constant across the scenarios considered. Actions and expected discounted returns are obtained by passing the full state to the trained actor and critic. Some colour scatter at similar temperature values is expected and reflects the policy variation due to differing lag contexts. We compare policies learnt under FGSM and random noise perturbations. These results show the effectiveness of selecting noise that maximises the effect on the agent.

In the ideal case, the plot of value function V^{π} is expected to show a clear distinction of regions where high and low rewards are anticipated. In other words, as the temperature of the system approaches the maximum allowed temperature, the value function should return a lower reward. This is because the number of heaters turned on must be reduced to avoid exceeding the maximum temperature. For the policy plot, we expect to see regions where all the heaters are turned on (heating phase), a region where the temperature is maintained by oscillating the number of heater (steady state) and a protection region where heaters are turned off to avoid exceeding the maximum temperature in the legs.

Recall that in PPO, the policy π is updated to favour actions whose actual return exceeded the prediction of the value function V^{π}. Because the policy learnt by the agent depends on this value, learning an accurate and consistent value function is important. We plot the expected reward across the system states (temperature), see Fig. 7. In this plot, blue points are states where the agent expects to collect a large reward. At lower temperatures, more heaters may be turned on resulting in a high reward. As the system temperature rises, the expected return gets lower as the capacity for more heating is lowered. Once the maximum allowed temperature is exceeded, a negative reward is received. The figure shows that the agent trained with FGSM perturbation has well-defined regions of the state space with different expected rewards. On the other hand, the value function of the agent trained with random noise expects negative rewards when the system temperature is not near the limit, illustrating a more conservative policy.

An explanation for the difference described above is the exploration during training. Note that the states plotted (tested) for the three policies are the same. However, this does not mean that these states were visited during the policy training. During training, the state space exploration is guided by the actions selected by the agent. Because the agent trained with FGSM (artificially) sees states where the optimal action is uncertain, the actions applied to the system have a large variability. As a result, the region covered during training is larger compared to policies with low variability where the same actions are always applied. This means that the value

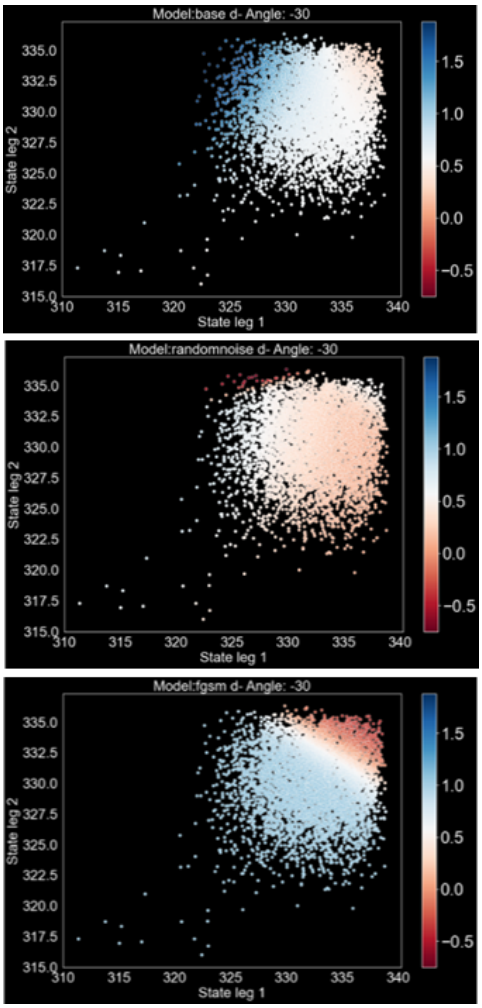


Figure 7. Expected reward (normalised) at different states for different models: baseline (top), model trained with random noise (centre) and model trained with FGSM attacks (bottom). Exceeding the maximum allowed temperature results in a penalty (negative reward).

function of agents with poor exploration has to extrapolate, a task that in general is difficult for neural networks.

Figure 8 shows the action with the highest probability, i.e., greedy policy, at different system temperatures. Each colour represents a different action, i.e., number of heaters turned on. Note that this is the policy applied when the system is tilted by 30 degrees. Therefore, the dynamics of each of the legs are different. When the temperature is close to the limit (338K), the FGSM model selects more heaters on (darker blue colour) compared to the policy trained with random noise. As a result, the temperature is closer to the limit without exceeding it, as shown in the results in the previous section. The results show that training an agent with random noise creates a more conservative policy, i.e., the number of heaters turned on is lower. Near the maximum temperature in both legs, the number of heaters turned on is reduced to 5 (white points),

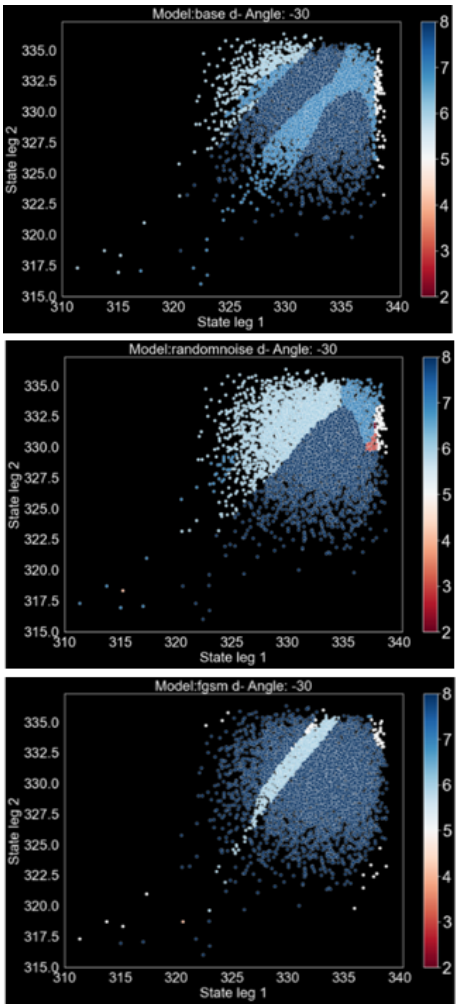


Figure 8. Policy learnt by agent across system states (temperature). The colour represents the action (number of heaters on) with maximum probability at different states for different models: baseline (top), model trained with random noise (centre) and model trained with FGSM attacks (bottom).

preventing overheating during the initial transient. Reduced heating also occurs close to the peak temperature of the less efficient leg 2, while the pitch angle means that leg 1 is not at the thermal limit. For the FGSM policy, this is a smaller region compared to the other two policies (less conservative). A narrow light blue band can also be observed for the FGSM policy, where some heaters are turned off. This represents the region of steady natural circulation for the system, where the thermally coupled legs operate within a small temperature range of each other. Investigation into the control action used in this region leads to the use of mimic learning, as described below, to understand the control actions used on each leg. We have observed that the controller has learnt to turn off heaters in the less efficient leg to prevent thermal overshoot.

The previous results allow us to understand how the different training strategies affect the learnt policy. However, it is difficult to identify what action is going to be applied for a particular state. Another possible approach for policy interpretability is the extraction of information from these complex models. Mimic learning can be used to train structured and interpretable models such as decision trees which can be verified easily (Bastani, Pu, & Solar-Lezama, 2018). We use this approach to obtain the 'rules' that the agent follows to apply an action.

Although the agent receives a 7-dimensional state, only the current leg temperatures are used as inputs to the mimic tree. While this simplification makes the tree unusable to control the system, it makes interpretation easier by considering a feature set with direct physical meaning, as rules expressed in terms of lag temperatures would be difficult to interpret. However, if necessary, this approach can be extended to the full 7-dimensional state.

For training and testing of the decision tree, 6000 pairs of data were collected. Accuracy of up to 97% can be achieved by training a tree with a depth of 11. This accuracy achieved with the reduced input demonstrates that current temperatures capture the dominant structure of the learnt policy. Figure 9 shows a segment of the decision tree. Each node, except the leaves, has an 'if' condition of the form $X[i] \leq y$ for $i \in [0, 1]$, where $X[0]$ and $X[1]$ are the temperatures of the leg 1 and leg 2, respectively. The list of values in each node and leaf shows the number of times an action a was taken by the agent in the state defined by the conditions. The length of the list of values is determined by the number of actions found in the training data. This means that, for this case, the agent applied 6 different actions during the interaction with the environment. The list of actions is shown in Table 1. The actions show a preference for leg 1 heating for this system (along the diagonal reachable thermal states of the system). The physics of the system prevent large differences in the temperature seen in the two legs, and because they are never reached, the controller has maximised its reward by having a policy to turn on all heaters, i.e., top-left and bottom-right of the partition map. This undesirable behaviour is arguably not a concern, as it is prevented by the (modelled) system's physics, but revealing such controller behaviour illustrates the value in our tools for exploring the RL strategy and the opportunity to adjust the controller manually or through a revised training scheme.

Figure 10 shows the partition of the state space created during the training of the decision tree. The black lines show the partitions that correspond to the red nodes of Fig. 9. The regions with different colours represent different actions. As expected, these regions look similar to the regions shown in the visualisation of the state space, see Fig. 8 (centre).

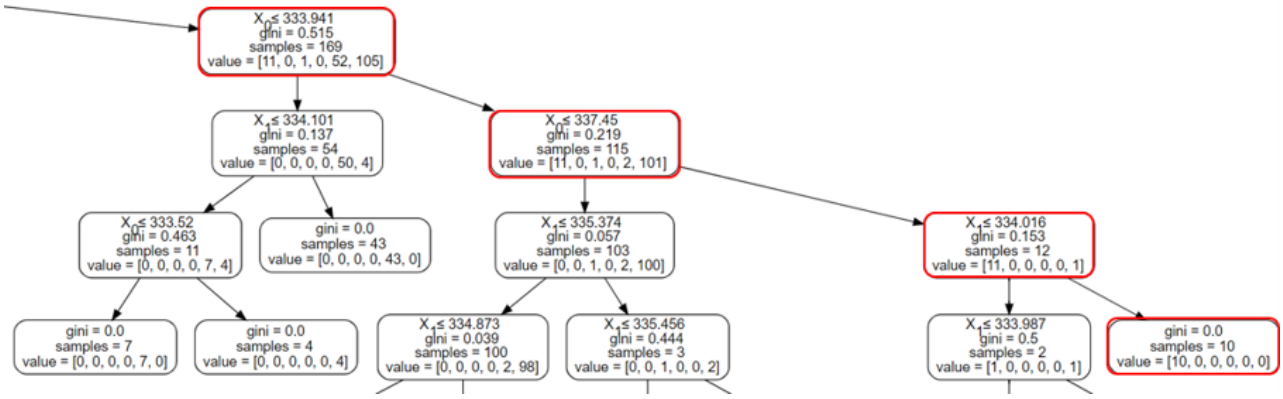


Figure 9. Segment of tree showing the decision (red nodes) of turning off heaters in leg 1 (action 1) when the temperature is close to the limit, i.e., greater than 337.45K.

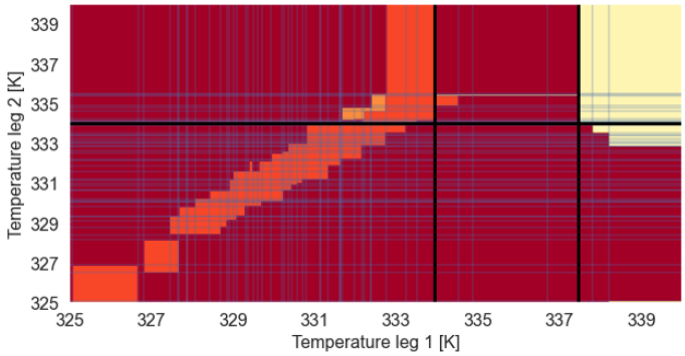


Figure 10. Partition and boundaries of the decision tree trained with the FGSM model. Black lines show the partitions indicated by the red nodes in Fig 9. The different colours present different actions.

Table 1. List of actions applied by the agent during data collection.

Action ID	Heaters on - leg 1	Heaters on - leg 2
1	1	4
2	3	1
3	3	4
4	4	1
5	4	2
6	4	4

5. CONCLUSION

In this paper, we demonstrate that RL provides a viable framework for synthesising fault-response control policies in safety-critical thermal systems. In such cases, available models of the system might not reflect the system's dynamics. The presented results show that by interacting with a simulator of the controlled system, the agent learns control actions that maintain operation within safety constraints. We also show that adversarial training using FGSM leads to robust policies. Finally, we demonstrate that a structured and interpretable surrogate model can be trained to approximate the behaviour of the reinforcement learning agent with high fidelity. This surrogate enables policy verification and analysis.

ACKNOWLEDGMENT

This work was partly funded by the Aerospace Technology Institute under the REPLENISH project.

REFERENCES

Bastani, O., Pu, Y., & Solar-Lezama, A. (2018). Verifiable Reinforcement Learning via Policy Extraction. *Advances in Neural Information Processing Systems*, *31*.

Christiano, P., Shah, Z., Mordatch, I., Schneider, J., Blackwell, T., Tobin, J., ... Zaremba, W. (2016). Transfer from Simulation to Real World Through Learning Deep Inverse Dynamics Model. *arXiv Preprint arXiv:1610.03518*.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., ... others (2022). Magnetic Control of Tokamak Plasmas Through Deep Reinforcement

Learning. *Nature*, *602*(7897), 414–419.

Gomez, H., Bures, M., & Moure, A. (2019). A Review on Computational Modelling of Phase-transition Problems. *Philosophical Transactions of the Royal Society A*, *377*(2143), 20180203.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft Actor-critic: Off-policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning* (pp. 1861–1870).

Hans, A., & Udluft, S. (2010). Ensembles of Neural Networks for Robust Reinforcement Learning. In *2010 Ninth International Conference on Machine Learning and Applications* (pp. 401–406).

Heess, N., Tb, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., … others (2017). Emergence of Locomotion Behaviours in Rich Environments. *arXiv Preprint arXiv:1707.02286*.

Lillicrap, T., Hunt, J., Pritzel, A., Hess, N., Erez, T., Tassa, Y., … Wierstra, D. (2015). Continuous Control with Deep Reinforcement Learning. *arXiv Preprint arXiv:1509.02971*.

Mandlekar, A., Zhu, Y., Garg, A., Fei-Fei, L., & Savarese, S. (2017). Adversarially Robust Policy Learning: Active Construction of Physically-plausible Perturbations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 3932–3939).

Moos, J., Hansel, K., Abdulsamad, H., Stark, S., Clever, D., & Peters, J. (2022). Robust Reinforcement Learning: A Review of Foundations and Recent Advances. *Machine Learning and Knowledge Extraction*, *4*(1), 276–315.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., … others (2022). Training Language Models to Follow Instructions with Human Feedback. *Advances in Neural Information Processing Systems*, *35*, 27730–27744.

Pinto, L., Davidson, J., Sukthankar, R., & Gupta, A. (2017). Robust Adversarial Reinforcement Learning. In *International Conference on Machine Learning* (pp. 2817–2826).

Rajeswaran, A., Ghotra, S., Ravindran, B., & Levine, S. (2016). Epopt: Learning Robust Neural Network Policies Using Model Ensembles. *arXiv Preprint arXiv:1610.01283*.

Recht, B. (2019). A Tour of Reinforcement Learning: The View from Continuous Control. *Annual Review of Control, Robotics, and Autonomous Systems*, *2*(1), 253–279.

Reyes, J. (2005). *Natural Circulation in Water Cooled Nuclear Power Plants Phenomena, Models, and Methodology for System Reliability Assessments* (Tech. Rep.). Dr. Jose Reyes (US).

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv Preprint arXiv:1707.06347*.

Sutton, R. S., Barto, A. G., et al. (1998). *Reinforcement Learning: An Introduction* (Vol. 1) (No. 1). MIT press Cambridge.

Tang, H., Rabault, J., Kuhnle, A., Wang, Y., & Wang, T. (2020). Robust Active Flow Control over a Range of Reynolds Numbers Using an Artificial Neural Network Trained Through Deep Reinforcement Learning. *Physics of Fluids*, *32*(5).

Wang, H.-n., Liu, N., Zhang, Y.-y., Feng, D.-w., Huang, F., Li, D.-s., & Zhang, Y.-m. (2020). Deep Reinforcement Learning: A Survey. *Frontiers of Information Technology & Electronic Engineering*, *21*(12), 1726–1744.

Zhang, H., Chen, H., Boning, D., & Hsieh, C.-J. (2021). Robust Reinforcement Learning on State Observations with Learned Optimal Adversary. *arXiv Preprint arXiv:2101.08452*.

Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D., & Hsieh, C.-J. (2020). Robust Deep Reinforcement Learning Against Adversarial Perturbations on State Observations. *Advances in Neural Information Processing Systems*, *33*, 21024–21037.

Zhang, Y., Tiňo, P., Leonardis, A., & Tang, K. (2021). A Survey on Neural Network Interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *5*(5), 726–742.