DOI: 10.13009/EUCASS2025-533
11TH EUROPEAN CONFERENCE FOR AERONAUTICS AND SPACE SCIENCES (EUCASS)

HOW TO GUARANTEE SAFETY FOR NEURAL NETWORK BASED ROCKET ENGINE CONTROLLERS

Jonas Dauer[⋆†], *Kai Dresia*[⋆], *Jan Christian Deeken*[⋆§] *and Günther Waxenegger-Wilfing*[⋆‡]
[⋆] *German Aerospace Center (DLR), Institute of Space Propulsion, 74239 Lampoldshausen, Germany*
[§] *RWTH Aachen University, Institute of Jet Propulsion and Turbomachinery, 52062 Aachen, German*
[‡] *University of Würzburg, Institute of Computer Science, 97074 Würzburg, Germany*
jonas.dauer@dlr.de
[†]Corresponding author

Abstract

This paper introduces an intelligent reinforcement learning controller for the LUMEN rocket engine, trained entirely in simulation to meet complex control requirements such as reusability and throttling. To ensure safe real-world deployment, the focus is on detecting when the agent is operating in situations it hasn't seen during training, which is important to ensure safe use in real-world applications. By introducing unexpected disturbances into the simulation and comparing the agent's actions with real LUMEN engine test data, the method identifies deviations that signal when the controller's performance may be unreliable. This approach represents a crucial first step toward validating the practical applicability and safety of reinforcement learning agents for rocket engine control.

1. Introduction

The control systems for liquid rocket engines have become increasingly sophisticated, especially with the rise of reusable engines [4]. Over time, components degrade - such as through soot accumulation [21, 16, 1], aging seals causing leaks [27], or erosion of turbine blades [10] - demanding more resilient control strategies. To operate reusable launch vehicles cost-effectively, engines must maintain long lifespans without incurring heavy maintenance costs. Moreover, modern mission profiles like orbital maneuvers and controlled landings require engines capable of wide-range throttling and multiple restarts.

Traditionally, rocket engines rely on fixed valve operation sequences to manage startup, steady-state operation, and shutdown. These sequences are usually established through extensive and costly testing. Closed-loop control, when used, typically regulates key parameters like combustion chamber pressure only during steady phases [20]. Various control techniques - ranging from PID to robust methods - have been explored extensively, as reviewed by Pérez-Roca et al. [25]. Precise control reduces the need for excess propellant, thereby boosting payload capacity.

Optimal control of rocket engines, including transient behavior, is crucial for meeting future performance and reliability goals. Reinforcement learning (RL) [30] presents a novel and effective approach for such complex control problems. RL can autonomously discover optimal control policies in simulated environments, requires minimal computational overhead during execution, and naturally handles multi-objective and multi-phase scenarios [19]. Applications of RL span diverse fields, including robotics [17], healthcare [36], flight systems [11, 7], and process control [29, 31]. Notable aerospace applications include RL-based moon landing guidance [2] and spacecraft control [13]. Furthermore, intelligent control systems leveraging AI have been explored for fault management and monitoring in programs like the Space Shuttle [22, 24].

Training reinforcement learning agents directly on real rocket engines is impractical due to the extremely high costs, inherent risks, and limited availability of test hardware. Consequently, these agents must be trained entirely within detailed simulation environments that replicate the engine's complex dynamics, sensor characteristics, and typical disturbances. Simulation allows for extensive trial-and-error learning cycles, enabling the agent to explore a wide range of operating conditions safely and efficiently.

However, a significant challenge arises when transferring a controller trained in simulation to real-world applications. The real engine inevitably differs from the simulation in ways that are difficult to fully capture, such as unmodeled physical phenomena, sensor noise, manufacturing tolerances, environmental variations, or unexpected external disturbances. These differences can cause the agent to encounter situations it has never experienced during

DOI: 10.13009/EUCASS2025-533

training, which may lead to unpredictable or unsafe control actions. In the worst case, this can result in catastrophic failures that compromise the mission or endanger the hardware.

Therefore, ensuring the safe deployment of reinforcement learning controllers in real rocket engines requires rigorous strategies to recognize when the agent is operating outside the domain of its trained experience. Detecting such out-of-distribution scenarios is essential to prevent unreliable or hazardous behavior. This detection capability forms a critical part of bridging the gap between simulation and real-world performance, ultimately enabling the trustworthy deployment of intelligent controllers in practical, safety-critical rocket engine systems.

In this paper, we present a method that assesses whether the agent's current inputs and experiences fall within the range of its training data. By applying this approach, we can detect when the agent is exposed to unfamiliar conditions - both in simulations with introduced disturbances and in real engine test data. This capability serves as a vital safeguard, providing confidence that the agent's decisions remain reliable and supporting the safe deployment of reinforcement learning-based controllers in real-world rocket engine applications.

2. Preliminaries

Below, we introduce the fundamental concepts and definitions that form the basis for the subsequent analysis and methodology.

2.1 Sequential decision making

Sequential decision making [18] is a key concept in reinforcement learning, decision theory, and artificial intelligence. It involves making a series of decisions where each action affects future outcomes. The decision-maker, often referred to as the agent, aims to maximize some cumulative reward or utility over time. To model such problems mathematically, we use decision-making frameworks. One widely used model is the Markov Decision Process (MDP) [26], which assumes that the environment is fully observable and that the system evolves according to probabilistic dynamics. In this work, we focus on the deterministic case, where the outcomes of actions are fully predictable.

2.1.1 Markov Decision Process

An MDP is a mathematical model used to describe environments where an agent makes sequential decisions. Since we are considering only the deterministic case, we assume that the dynamics of the environment are fully computable. This means that the outcome of an action is fully determined by the current state and action taken, with no randomness involved. Formally, an MDP is defined by the tuple (S, A, P, R), where:

- S is the state space, a set of all possible states that the system can be in.
- A is the action space, a set of all possible actions the agent can take.
- P is the deterministic transition function. For a deterministic MDP, the transition function $P(s_{t+1}|s_t, a_t)$ always yields a single next state s_{t+1}, given a state s_t and action a_t. This means that for each state-action pair, the outcome is fully predictable, and there is no randomness in state transitions.
- R is the reward function, which specifies the immediate reward the agent receives when taking action a_t in state s_t, transitioning to state s_{t+1}. The reward function $R(s_t, a_t)$ assigns a fixed reward r_t for each state-action pair.

The objective in an MDP is to find an optimal policy π^*, a mapping from states to actions, that maximizes the expected cumulative reward over time. Figure 1 illustrates how an agent interacts with its environment in a deterministic Markov Decision Process. At each step, the agent chooses an action based on the current state and the policy. The environment then transitions to a new state and returns a reward. Since the process is deterministic, each action in a given state always leads to the same outcome. The agent can use the rewards received to optimize its actions over time, improving the policy to maximize cumulative reward.

2.1.2 Reinforcement Learning

Reinforcement Learning (RL) [30] is the process by which an agent learns an effective policy through interaction with its environment, using feedback in the form of rewards. A policy, denoted by π, is a mapping from states (or observations) to actions, defining the agent's behavior at any given time. The agent is not told which actions to take - instead, it must discover which actions yield the highest rewards through experience. The reward serves as a performance signal, quantitatively evaluating how desirable the outcome of an action is. Over time, the agent uses

DOI: 10.13009/EUCASS2025-533

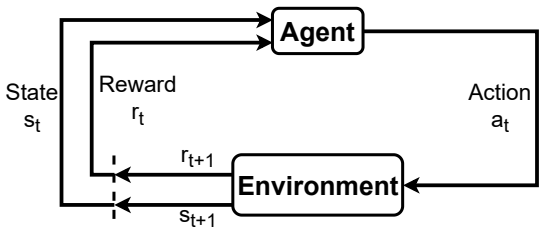

Figure 1: Graphical representation of a Markov Decision Process (MDP). The diagram illustrates states, actions, deterministic transitions, and associated rewards. In a deterministic setting, each action taken in a given state leads to a single, predictable next state.

this signal to distinguish between better and worse behaviors, improving its policy to maximize long-term cumulative reward. The return is defined as the cumulative reward collected over a single episode, serving as an overall measure of the agent's performance during that interaction.

In practice, reinforcement learning is typically carried out in simulated environments, especially when the target application involves safety-critical or high-cost systems such as rocket engines. Training an agent requires large numbers of interactions, including exploration of poor or even dangerous strategies. Attempting this directly on a real rocket engine would be unsafe and economically infeasible, as early-stage policies might cause engine failure or physical damage. Simulation offers a controlled, repeatable, and risk-free environment where agents can explore freely and learn effectively before any real-world deployment.

2.1.3 Sim-to-real gap and domain randomization in rocket engine control

In rocket engine control, reinforcement learning policies are almost always trained in simulation due to the extreme risks and costs involved in real-world experimentation. However, transferring a policy learned in simulation directly to a physical rocket engine is challenging because of the sim-to-real gap [37] - the differences between the simulated training environment and the real engine's operating conditions. Rocket engines operate under complex, highly nonlinear physical processes involving fluid dynamics, combustion, thermal stresses, and hardware variability. Despite advances in simulation fidelity, it remains impossible to perfectly capture all these dynamics, including subtle effects such as slight variations in sensor noise, hardware wear or actuator delays. These discrepancies mean that a control policy that works flawlessly in simulation might lead to instability, inefficient performance, or even catastrophic failure when deployed on a real engine.

To bridge this gap, domain randomization [37] is employed during training. Key engine parameters - such as turbopump efficiencies, temperature variations, sensor noise characteristics, and actuator or sensor delays - are randomized within realistic bounds during simulation. The RL agent learns to control the engine robustly under a wide range of conditions rather than overfitting to a single idealized model. This randomized training encourages the agent to develop control strategies that tolerate uncertainty and adapt to variations it may encounter in practice. As a result, the learned policy is more likely to remain stable and effective when transferred to real rocket hardware.

Still, domain randomization must be carefully designed. Randomizing too narrowly risks insufficient robustness, while excessively broad randomization can make learning inefficient or overly conservative - critical factors given the high stakes of rocket engine operation.

2.2 Neural networks

Neural networks provide a versatile framework for capturing intricate system behaviors, making them well-suited for advanced control applications.

Neural networks Neural networks (NNs) [8] are computational models inspired by the structure and function of the human brain. They consist of interconnected layers of nodes (neurons), where each connection has an associated weight. Neural networks are particularly well suited for learning complex, nonlinear relationships from data, making them valuable in control tasks where analytical models are difficult to derive. In machine learning, neural networks are commonly used for both classification and regression tasks. In classification, the network assigns inputs to discrete categories (e.g., fault detection or operating mode recognition), while in regression, it predicts continuous outputs (e.g., estimating system parameters or control actions). In the context of rocket engine control, neural networks can approximate nonlinear system dynamics or serve as policy approximators in reinforcement learning frameworks. Their

DOI: 10.13009/EUCASS2025-533

ability to generalize from training data enables them to model intricate dependencies between system variables and control actions with relatively low computational cost at runtime.

Recurrent neural networks While standard neural networks process inputs independently, recurrent neural networks (RNNs) [8] are specifically designed to handle sequential data. They incorporate feedback loops that allow information to persist over time, giving them a form of memory. This makes RNNs especially useful for modeling time-dependent behaviors, such as dynamic processes in rocket engines where past states influence current behavior. RNNs are capable of capturing temporal correlations in sensor data, internal engine states, or control histories, which is essential for effective decision-making in reinforcement learning tasks. Variants such as Long Short-Term Memory (LSTM) [12] and Gated Recurrent Units (GRU) [3] address limitations like vanishing gradients, making them more robust for learning long-term dependencies in complex systems.

2.3 Uncertainty

In machine learning, uncertainty refers to the lack of certainty in the predictions made by a model. This uncertainty can arise from different sources, and it is typically categorized into two types: aleatoric uncertainty and epistemic uncertainty [14].

Aleatoric uncertainty Aleatoric uncertainty, also known as statistical uncertainty, comes from inherent randomness or noise in the data. This type of uncertainty cannot be reduced, even with more data, as it represents the natural variability in the system. For example, in a regression problem, noise in the measurements or errors in sensor readings contribute to aleatoric uncertainty. It is important to note that in deterministic environments, such as deterministic Markov decision processes, aleatoric uncertainty does not exist. In such cases, the system's behavior is fully predictable, and there is no inherent randomness in the outcomes, meaning the model can make precise predictions without noise in the process.

Epistemic uncertainty Epistemic uncertainty, or model uncertainty, arises from a lack of knowledge about the true underlying model or system. This type of uncertainty can be reduced with more data or a better model. For example, if a model is underfitting or has a poor understanding of the data distribution, epistemic uncertainty is high. As more data is collected or the model is improved, this uncertainty can decrease.

2.4 Deep ensembles

Deep ensembles are a technique used to estimate uncertainty in machine learning models, particularly epistemic uncertainty [15]. In deep ensembles, multiple independent neural networks are trained, each with different initializations or random seeds. The behavior of the individual models within a deep ensemble differs significantly when handling known versus unknown data. For known data - data encountered during training - all models in the ensemble have been trained on the same dataset and are typically well-optimized to predict these data accurately. Consequently, the models tend to produce very similar or nearly identical predictions, resulting in low uncertainty in their outputs. However, when the ensemble encounters unknown data - data not seen during training - there is generally greater variation in the predictions made by the individual models. This variation arises because each model may have learned slightly different hypotheses about the underlying data distribution, leading to increased disagreement in their predictions. This variability reflects the ensemble's heightened uncertainty regarding the new data, resulting in higher epistemic uncertainty. Thus, while an ensemble generates consistent predictions for known data, it produces more diverse outputs for unknown data, thereby helping to quantify the epistemic uncertainty in its predictions.

3. Out-of-Distribution Detection for Sequential Decision Making Tasks

For applications such as rocket engine control, it is critical that the real-world operating conditions are adequately covered by the simulation environment used for training. If the simulation does not sufficiently represent real-world dynamics, the trained intelligent agent may encounter situations outside its experience during deployment, leading to undefined or unpredictable actions. Such behavior is unacceptable in safety-critical systems, as it can result in potentially dangerous outcomes. In this context, out-of-distribution (ooD) refers to the mismatch between real-world conditions and those represented in the simulation, meaning that the real system behaves differently from what the agent has been trained to handle. In classification and regression tasks, the problem of out-of-distribution detection is well-defined due to fixed input-output mappings [35]. However, in sequential decision making scenarios - such

DOI: 10.13009/EUCASS2025-533

as those encountered in reinforcement learning - predefined labels do not exist, requiring a different formulation of the ooD detection problem. Haider et al. [9] propose defining ooD samples based on deviations in the transition function of the underlying Markov Decision Process (MDP). Building on this perspective, and as shown in Section 2, the performance of an agent within an MDP can be characterized by the cumulative rewards it obtains. Therefore, in this work, we define ooD samples not directly by differences in the transition function P, but instead by changes in the reward signal r_t. The impact of a given perturbation depends on the specific MDP: while a 10% reward loss may be acceptable in some environments, it could have severe consequences in others.

To illustrate, consider a 10% deviation in the combustion chamber pressure. While this represents a quantifiable divergence in the system dynamics, such a deviation is typically non-critical from a safety and operational standpoint. Conversely, a 10% deviation in the mixture ratio - a parameter crucial for combustion stability and engine health - can have severe and potentially catastrophic consequences. Transition function-based ooD detection methods [9, 23], which primarily monitor changes in state transitions without assessing their operational relevance, are generally unable to distinguish between these two cases and treat both deviations as equally significant anomalies. In contrast, reward-based ooD detection [28] leverages the reward function's sensitivity to deviations that materially impact the agent's objectives. By incorporating domain knowledge into the reward design, it becomes possible to weigh deviations according to their actual effect on performance and safety. Thus, changes in the system that cause substantial degradation in expected rewards - such as deviations in critical parameters like mixture ratio - can be identified more reliably and prioritized over less consequential variations, like moderate offsets in combustion chamber pressure. This nuanced differentiation is vital for practical applications where maintaining safety and mission success is paramount, highlighting a fundamental advantage of reward-based ooD detection over transition function-based approaches.

To investigate this, we analyze the relationship between reward degradation and the computed ooD scores. We evaluate two distinct approaches: (1) The Probabilistic Ensemble Dynamics Model (PEDM) by Haider et al. [9] explicitly predicts the next state of the environment based on the current state and the action taken. By using an ensemble of models, it captures uncertainty and provides probabilistic predictions over possible future states. This makes PEDM robust and well-suited for modeling complex transition dynamics. (2) A reward prediction approach using recurrent neural networks that encode sequences of past observations and actions to implicitly capture temporal dependencies. Instead of predicting next states, this method forecasts expected rewards, making the reward itself the central signal for ooD detection. By comparing these approaches, our goal is to assess the effectiveness of reward-based prediction as a reliable indicator of out-of-distribution behavior in sequential decision making tasks.

4. Out-of-Capability Detection

To distinguish our approach from existing methods, we introduce the term out-of-capability detection. As previously discussed in Section 3, out-of-capability detection is grounded in the prediction of the reward signal. Given that the transition function and the reward function may depend on different subsets of state variables, we argue that anomaly detection in sequential decision making processes should be guided primarily by the reward. Based on this rationale, we propose the following algorithm, which is specifically designed to support out-of-capability detection by leveraging reward-based evaluation.

4.1 General approach

The algorithm is designed to evaluate whether a policy π, which was originally trained on an MDP $M = (S, A, P, r, s_0)$, can still perform effectively when applied to a different MDP $M' = (S, A, P', r', s_0)$ that may have altered transition dynamics and reward functions. To achieve this, the method first collects experience by running the policy π on the original MDP M, generating a dataset of state-action-reward transitions. A recurrent neural network (RNN)-based model is then trained on this data to predict the expected reward given the current state and action, while also capturing temporal dependencies and latent environment dynamics through its hidden state representation.

When deploying the policy on the new MDP M', the algorithm continuously compares the reward predicted by the model to the actual reward observed in the environment. The discrepancy between these two - expressed as a prediction error score - serves as an indicator of how well the current environment aligns with the conditions the policy was trained for. A low prediction error score suggests that the policy is operating within familiar and expected conditions, implying it remains capable in this environment. Conversely, a high prediction error score signals a significant mismatch between predicted and observed rewards, highlighting the presence of anomalies or out-of-distribution situations. Such anomalies indicate that the policy is encountering states or transitions that deviate from its training experience, suggesting that its ability to perform optimally may be compromised.

By focusing on reward prediction errors rather than solely on state distributions, the algorithm provides a practical and task-relevant measure of capability. This reward-centric score effectively detects when the policy is likely operating

DOI: 10.13009/EUCASS2025-533

beyond its reliable range, enabling early identification of performance degradation due to changes in the environment.

4.2 Estimating uncertainty

While reward prediction error is useful for detecting out-of-capability behavior, it may not fully capture the model's confidence, especially in poorly represented regions of the state-action space. To address this, the algorithm uses an ensemble of neural networks to estimate epistemic uncertainty via the standard deviation of their predictions. High uncertainty signals low confidence in the reward prediction, flagging situations where the policy may be unreliable even if the reward error is low.

This uncertainty-aware approach enhances detection by identifying uncertain states and enables operation in environments where the true reward is not directly observable. The final classification combines both the reward prediction error and the ensemble uncertainty: a state-action pair is considered within the policy's capability only if both the prediction error and uncertainty are below respective thresholds. This combined score provides a more robust measure of when a policy may be operating beyond its trained capabilities.

5. LUMEN

This research is conducted as part of the Liquid Upper Stage Demonstrator Engine (LUMEN) project [32] that is implemented by the Institute of Space Propulsion at the German Aerospace Center (DLR). LUMEN, as shown in Figure 2, is a modular expander-bleed engine that serves as a versatile testbed for advanced propulsion research. In 2024, it entered operational use, marking the first real-world application of neural network-based control in a rocket engine [6]. Its modular architecture, extensive instrumentation, electrically actuated valves, and complex expander-bleed cycle make it particularly well-suited for investigating advanced control strategies and dynamic engine behavior. With up to six controllable valves, LUMEN enables the simultaneous regulation of multiple engine parameters, exceeding the complexity of traditional thrust and mixture ratio control. This high level of control authority provides a valuable platform for deepening system-level understanding and evaluating technologies relevant to future upper stage developments.

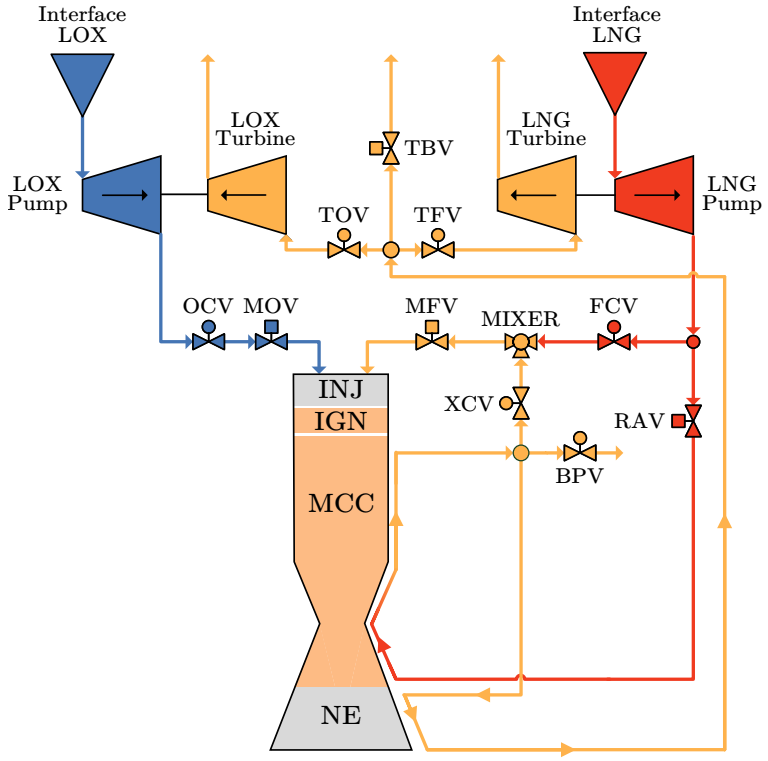

Figure 2: Flow scheme of LUMEN [32]

EcosimPro [33] is capable of accurately simulating the behavior of the LUMEN engine, making it a suitable environment for developing and testing advanced control strategies. It is a modeling and simulation tool designed for the analysis of multidisciplinary systems in 0D and 1D domains, supporting both continuous and discrete dynamics

DOI: 10.13009/EUCASS2025-533

through differential-algebraic equations and event-based logic. The tool provides a graphical interface that enables users to build complex system models by combining predefined components from various libraries. Particularly relevant are the European Space Propulsion System Simulation (ESPSS) libraries, developed under the supervision of the European Space Agency (ESA), which are tailored for the simulation of liquid rocket propulsion systems. These libraries have been continuously refined and extended in recent years.

6. Test Case

Based on our EcosimPro simulation, we employed reinforcement learning to train a controller capable of regulating the LUMEN engine. The trajectory depicted in Figure 3 was initially defined as the reference. The target combustion chamber pressure is represented in orange on the left-hand side of the figure, while the target mixture ratio is shown in orange on the right-hand side. The trajectories achieved by the trained controller are shown in blue; these values were obtained from the simulation results. Further details regarding the training of the agent are provided by [5] and [34]. The results demonstrate that the agent is capable of successfully tracking the specified trajectories within the simulation.

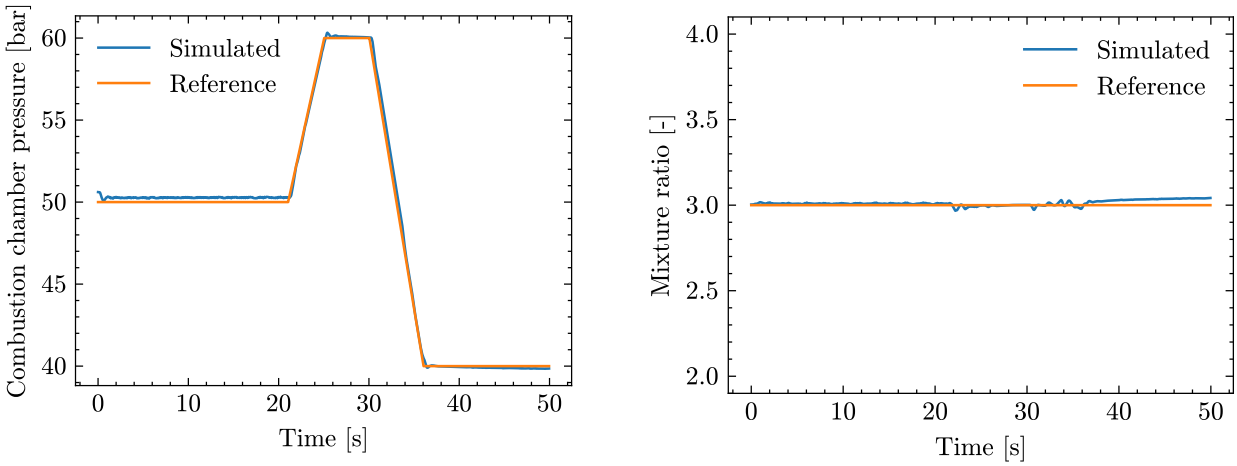

Figure 3: Target and simulated trajectories for combustion chamber pressure (left) and mixture ratio (right) during simulation of the LUMEN engine. The reference trajectory for the combustion chamber pressure starts at 50 bar, ramps up to 60 bar where it remains constant for 5 seconds, and then ramps down to 40 bar. The reference trajectory tor the mixture ratio remains constant at 3 throughout the simulation. The trajectories achieved by the trained controller (blue) are also shown.

After demonstrating that the agent was capable of controlling the LUMEN engine in simulation, the trained controller was deployed in a real-world test. This test, conducted on the DLR test bench P8.3, confirmed the fundamental feasibility of utilizing neural network-based controllers for such applications [5]. However, it also revealed one of the main limitations of the current approach. As illustrated in Figure 4 (left), the controller induced oscillations in the system, which were reflected in the combustion chamber pressure. These real-world oscillations, which were absent in the simulation, are likely attributable to an inaccurate model of the valve dynamics. The discrepancy between simulated and actual system behavior, combined with the controller's overfitting to the training environment, prevented a successful transfer of the controller from simulation to reality.

After incorporating improved valve dynamics into the simulation and successfully training a second agent, a follow-up test was conducted. The results are shown on the right-hand side of Figure 4. In contrast to the first test, the agent gained control of the valves only after 18 seconds. This delayed activation, along with a modified open-loop sequence, accounts for the deviations in combustion chamber pressure observed between 15 and 20 seconds. Nevertheless, Figure 4 clearly demonstrates that, following a brief transient phase, the target combustion chamber pressure was reliably achieved and maintained until the test was prematurely terminated.

7. Results

In this section, the out-of-capability score developed in this work is first compared with the PEDM algorithm in simulation. Subsequently, the data obtained from the experimental tests are evaluated.

DOI: 10.13009/EUCASS2025-533

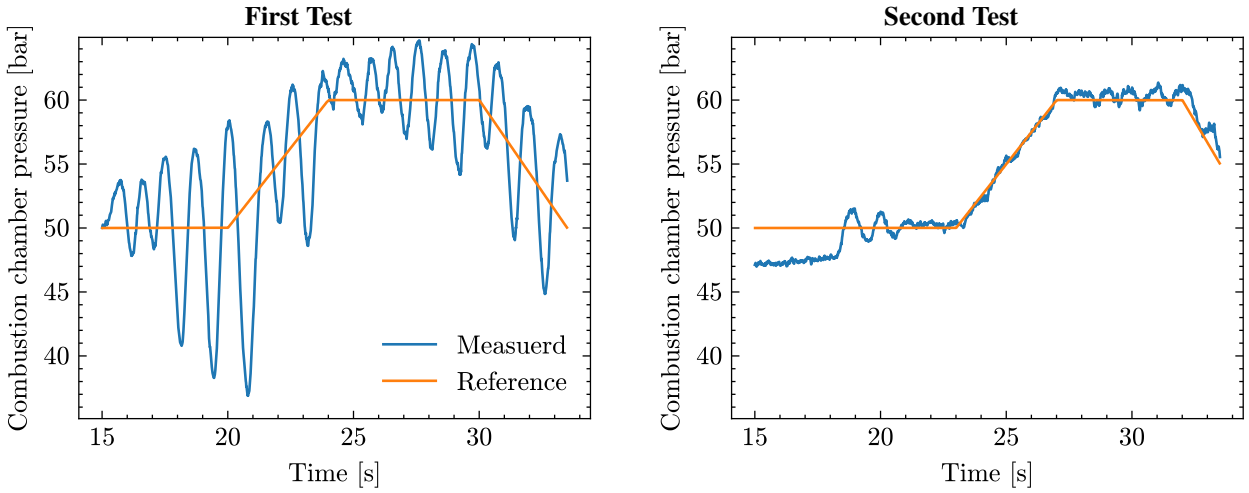

Figure 4: Combustion chamber pressure from two real-world test runs [5]. The left plot presents data from the first test, while the right plot corresponds to the second test. In the first test, the initial 15 seconds were conducted in open-loop mode; in the second test, this phase lasted 18 seconds. Differences in the open-loop sequences explains the divergence observed in the initial part of the trajectories. Both tests were terminated prematurely due to different limiting factors.

7.1 Simulated results

For the simplest test case, multiplicative noise was applied to the measured values of combustion chamber pressure and mixture ratio. Figure 5 illustrates the relationship between the achieved return and the calculated out-of-capability (ooC) and PEDM scores. The figure presents results from multiple episodes, each corresponding to a different noise level. The noise is defined as multiplicative, such that the noisy signal is generated according to:

$$\tilde{x} = x \cdot \eta \quad \text{with} \quad \eta \sim \mathcal{N}(1, \sigma^2). \tag{1}$$

Here, $\tilde{x}$ denotes the noisy measurement, x the original signal, and η a Gaussian random variable with mean 1 and variance σ^2, ensuring that the noise scales the signal proportionally. This noise model was applied to both the combustion chamber pressure and the mixture ratio. The respective standard deviations are shown in the legend. The dots in the figure represent the mean value of the corresponding score in each episode.

The expected relationship between reward and noise level is clearly observable. As the standard deviation of the noise increases, the agent's ability to accurately perceive the true internal state of the system deteriorates. This increasing discrepancy between the observed state and the environment's actual internal state causes the agent's actions to become progressively less optimal with respect to the true system dynamics. Consequently, there is a pronounced inverse relationship between the noise standard deviation and the reward achieved by the agent.

An examination of the two algorithms reveals one key observation: both algorithms effectively identify out-of-distribution trajectories. In both cases, a clear inverse relationship can be observed between the achieved return and the calculated score. Specifically, as the return decreases, the corresponding out-of-capability or PEDM score increases, indicating that the current environment deviates progressively from the conditions seen during training. This behavior is consistent with the intended design of both algorithms, where higher scores are expected in response to unfamiliar or perturbed system dynamics. The relationship also reflects the definition of return, which, as described in Section 2, quantifies the overall performance of an episode as the cumulative reward. In this case, we assume that a lower return results from changes in the underlying system dynamics, caused by the applied sensor noise. As the noise level increases, the agent's perception of the true system state becomes increasingly distorted, leading to suboptimal decisions and, consequently, reduced performance as reflected in the return.

Another noteworthy observation is the increased prediction error of the out-of-capability (ooC) algorithm at low noise levels, specifically in the range from no noise up to a standard deviation of 0.5%. This behavior can be attributed to the design of the training process: while the reinforcement learning agent was trained using domain randomization and thus exposed to noise levels uniformly sampled between 0% and 1%, the ensemble underlying the ooC algorithm was trained exclusively on data with a noise standard deviation of 1%. As a result, the ooC model achieves its highest predictive accuracy when the noise level matches its training conditions, and the prediction error increases slightly at lower noise levels. In contrast, the standard deviation measure produced by the ooC model consistently decreases as the noise level decreases, independent of the prediction error. This reflects that the standard deviation quantifies the

DOI: 10.13009/EUCASS2025-533

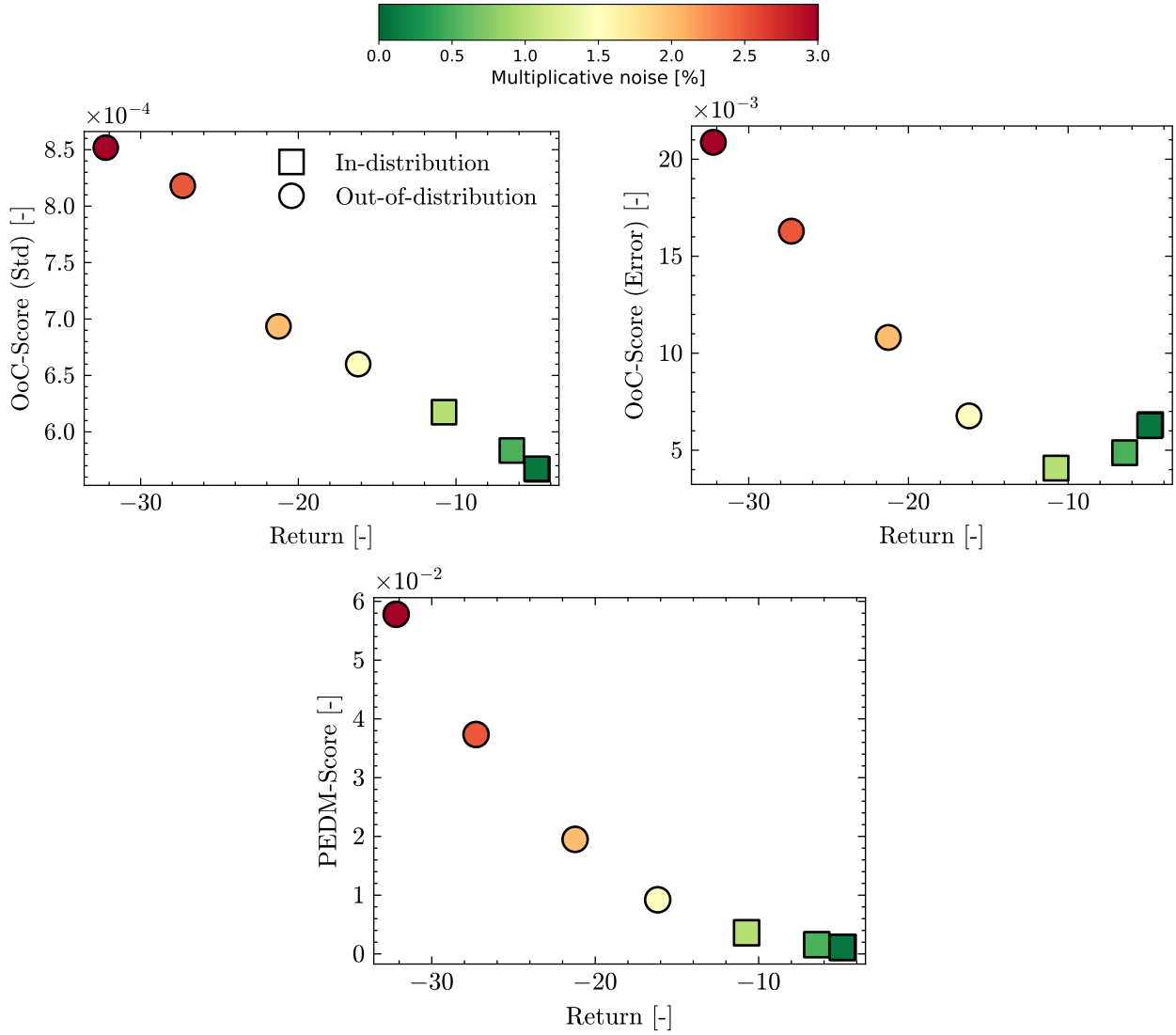

Figure 5: Relationship between achieved return and mean ooC and PEDM scores under varying levels of multiplicative noise on combustion chamber pressure and mixture ratio. Each point shows the average score per simulation episode at the noise levels indicated in the legend. Noise levels included in training are marked with squares, while those outside the training range are shown as circles. Top-left: OoC standard deviation; top-right: OoC error; bottom: PEDM results.

DOI: 10.13009/EUCASS2025-533

model's uncertainty, which correctly reduces as the input data becomes closer to the training distribution of the agent, even when the prediction error does not.

In the following experiment, the behavior of the anomaly scores is analyzed in the event of a component fault occurring during an episode. The trajectory previously presented in Figure 3 is reused for this purpose. To simulate the fault, the efficiency of the oxygen turbopump (OTP) is reduced to 80% starting at second 27. The objective of the detection algorithms is to recognize the loss in efficiency and the associated change in system dynamics, and to respond by producing elevated anomaly scores. Prior to analyzing the individual scores for the respective results, it is essential to establish thresholds that delineate when a sample is considered out-of-distribution. To determine these thresholds, we compute the 99th percentile of each score observed within the training dataset. This approach is motivated by the desire to avoid the influence of extreme outliers in the training data, which could lead to overly conservative thresholds and reduced sensitivity to anomalies. Table 1 summarizes these percentile-based thresholds obtained within the training data.

Table 1: Scores at the 99th percentile within the training data, used to define robust out-of-distribution detection thresholds.

Score type	99th percentile
OoC (Std)	1.28×10^{-3}
OoC (Error)	1.41×10^{-2}
PEDM	1.37×10^{-1}

Figure 6 illustrates the resulting trajectory along with the corresponding scores. The first observable effect of the OTP fault is a noticeable drop in combustion chamber pressure from the moment the efficiency reduction is introduced. The agent is unable to maintain the desired pressure level, as such a fault scenario was not included in the domain randomization during training. Even after the load point change to a target combustion chamber pressure of 40 bar, the simulated value remains significantly below the reference.

Figure 6 provides further insight into the behavior of the two evaluated detection methods. Both the standard deviation-based ooC score and the PEDM score exhibit a distinct and immediate increase at the onset of the OTP fault, which occurs at 27 seconds. This sudden rise reflects the system's deviation from nominal dynamics due to the abrupt reduction in turbopump efficiency. Notably, both scores remain elevated throughout the remainder of the episode, indicating a sustained mismatch between the actual system behavior and the conditions encountered during training. It is worth noting, however, that the ooC standard deviation score temporarily falls below the threshold. This short-lived dip is likely caused by a few individual samples during the setpoint transition, where the system dynamics may momentarily resemble patterns seen during training, particularly in terms of the reward signal. However, it is very difficult to clearly explain what is happening at this point - especially considering that the PEDM score does not exhibit a corresponding dip. It would certainly be worthwhile to investigate this discrepancy further, but such an analysis lies well beyond the scope of this work. Despite this ambiguity, the ooC score rises again shortly thereafter, correctly indicating that a change in system dynamics persists beyond the operating point transition. Overall, both scores are able to distinguish between in-distribution and out-of-distribution behavior, supporting their suitability for continuous anomaly monitoring and detection of persistent changes in system dynamics. However, the PEDM score appears to be slightly better suited for this task, as it remains more stable and does not exhibit the unexplained dip observed in the ooC score.

In contrast, the reward prediction error of the ooC method exhibits a distinct temporal profile. Although it also peaks as the fault is introduced, the error decreases significantly following the transition to the new operating point at 40 bar and for the majority of the subsequent samples remains below the threshold. However, there are intermittent samples where the error rises slightly above the threshold, though these surpasses are marginal. This behavior can be attributed to the structure of the reward function, which primarily depends on the differences between actual and reference values of the combustion chamber pressure and the mixture ratio. Despite the magnitude of these differences, their evolution remains predictable under the altered dynamics, allowing the model to produce accurate reward estimates even under fault conditions. As a result, while the uncertainty remains high due to the lack of comparable training data, the prediction error alone tends to underestimate the degree of deviation from nominal behavior.

In summary, both the out-of-capability score and the Probabilistic Ensemble Dynamics Model score demonstrate strong performance in simulation-based evaluations. Notably, within the ooC framework, the uncertainty-based metric exhibits greater precision and robustness compared to the reward prediction error. Given the practical challenges associated with accurately measuring reward errors in real-world scenarios, the uncertainty score provides a more

DOI: 10.13009/EUCASS2025-533

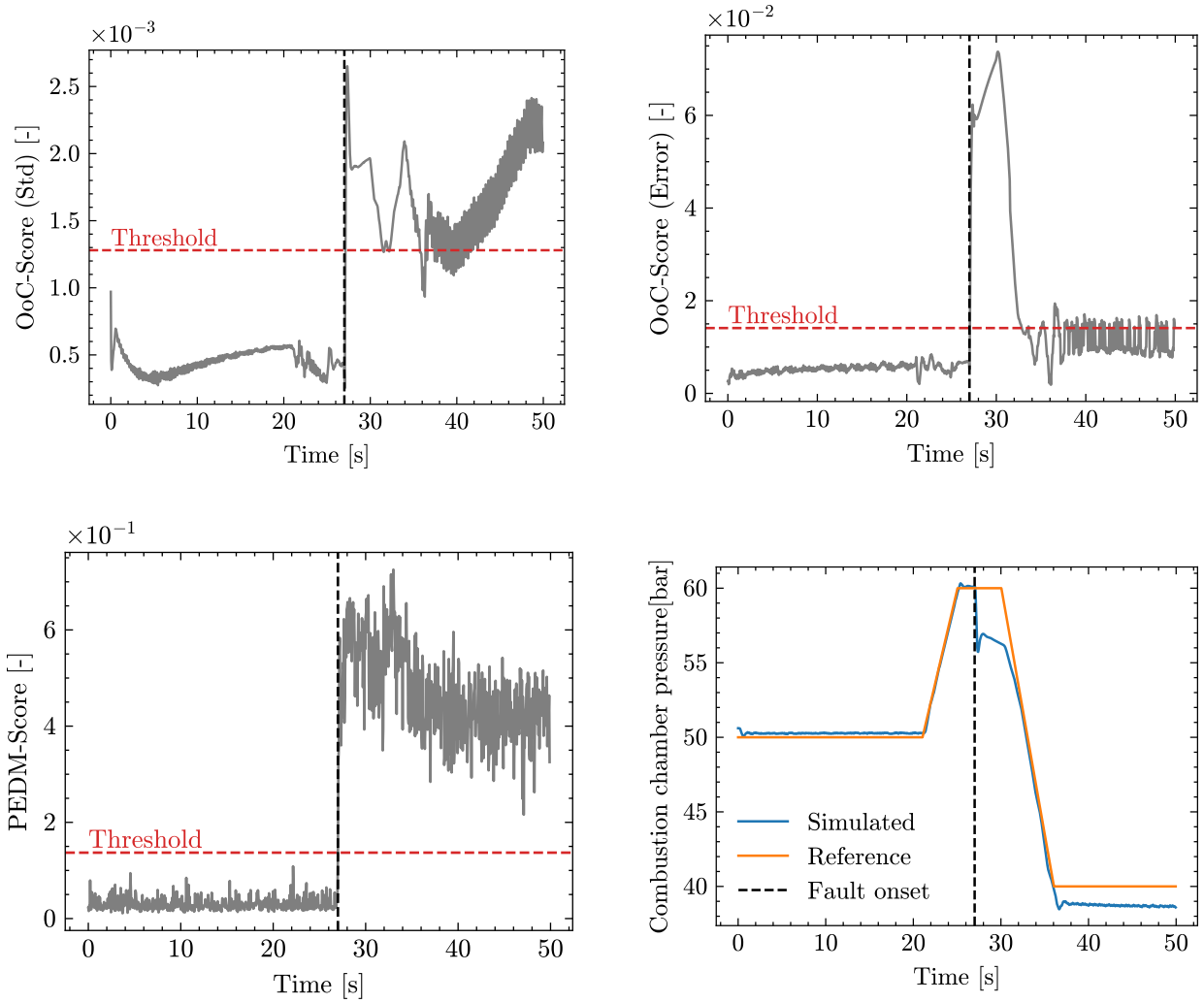

Figure 6: Reference (orange) and simulated (blue) combustion chamber pressures, along with the anomaly detection scores (gray), for a trajectory where the OTP efficiency is reduced to 80%. Initially, the system operates under nominal conditions. The dashed black vertical line marks the time point at which the efficiency is decreased, resulting in detectable deviations in the anomaly score. The red dashed horizontal line represents the threshold distinguishing in-distribution from out-of-distribution samples.

reliable and informative indicator of out-of-distribution states. Consequently, the subsequent analysis of the real test data is centered exclusively on the uncertainty measure derived from the ooC approach.

7.2 Real test data

Figure 7 presents the evaluation results of the two algorithms - out-of-Capability and PEDM - applied to the real test data illustrated in Figure 4. It is evident in both methods that the score in the first experiment is significantly higher than in the second experiment. This discrepancy can be attributed to the training environment of the first agent, in which the valves were inaccurately modeled, causing a substantial deviation between the simulated environment and real-world conditions. Furthermore, the plots from the first test reveal that the anomaly detection score does not remain consistently elevated but instead exhibits noticeable fluctuations over time. Contrary to the continuous score increase suggested by Figure 6, the detection score exhibits fluctuations that correlate with variations in the combustion chamber pressure. This behavior suggests that, along the trajectory, there exist regions within the state space where the simulation and the real system exhibit close similarity, while in other regions, the divergence between them becomes more pronounced. Notably, these fluctuations cause the ooC score to occasionally fall below the threshold, whereas the PEDM score remains consistently above it throughout. For us, however, the clearly visible upward spikes are more

DOI: 10.13009/EUCASS2025-533

decisive, as they provide a definitive indication that the agent is operating outside of its training MDP.

In the second test, the initial phase is particularly prominent. This phenomenon arises because the first three seconds of the trajectory are executed in open-loop mode, during which the agent does not actively select any actions. Consequently, this specific sequence was not included in the training dataset. Since the neural networks underlying both algorithms were not trained on data from this interval, the system behavior during this phase differs from the scenarios the models have learned to represent, resulting in a slightly higher detection score, especially for the out-of-capability detection method, compared to the remainder of the test. An examination of the PEDM scores reveals a similar pattern. Throughout the remainder of the test, the out-of-capability scores predominantly remain below the predefined threshold, whereas the PEDM scores exhibit fluctuations around the threshold, intermittently exceeding it. At present, it is not possible to conclusively determine which behavior is preferable; both patterns are justifiable when interpreted in the context of the underlying system dynamics and trajectory.

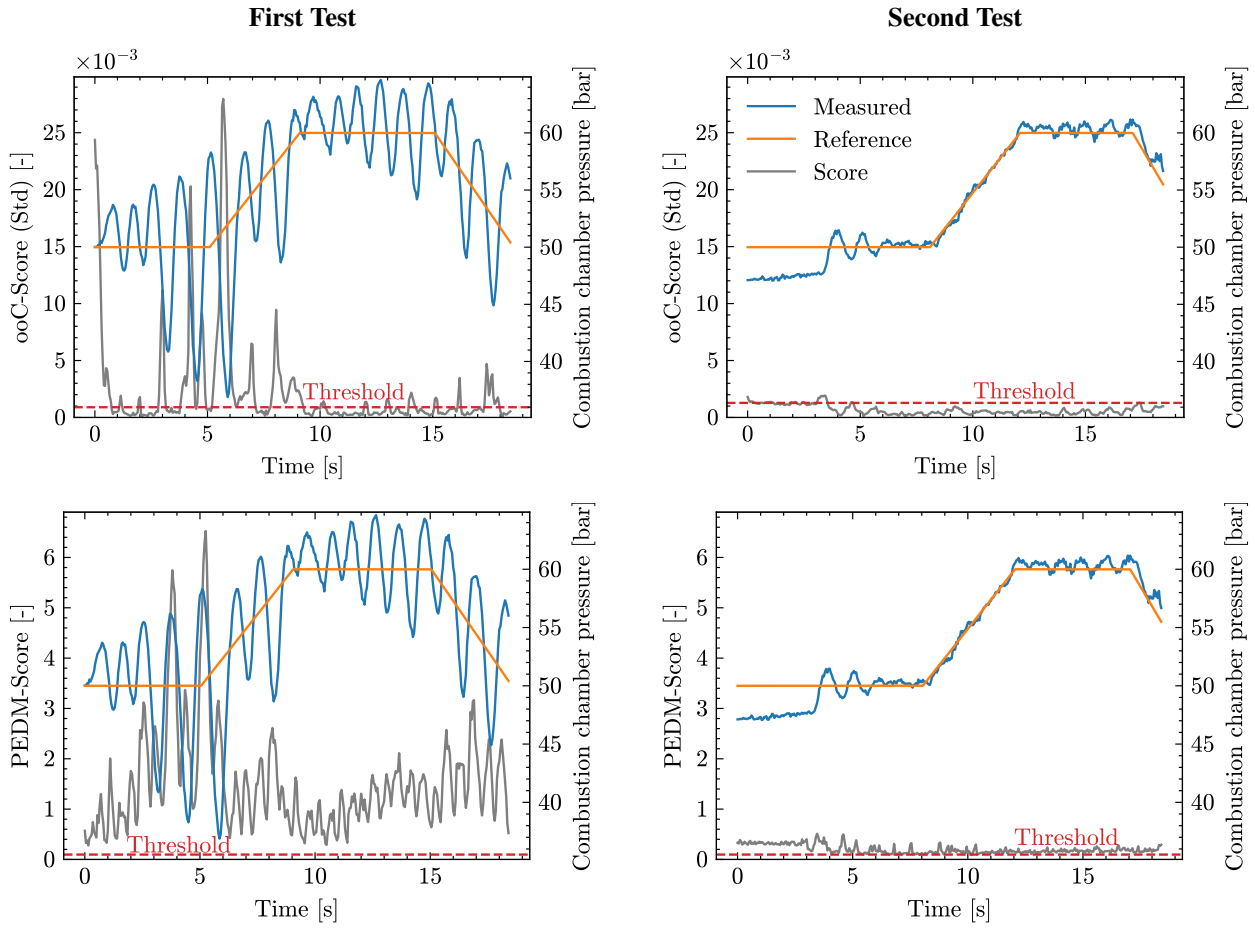

Figure 7: Measured (blue) and target (orange) combustion chamber pressures alongside the anomaly detection score (red) for both tests. The scores highlight deviations between simulation and reality, with higher values in the first test and reduced deviations after simulation adjustments in the second.

Overall, both scoring methods demonstrated strong performance on the real data. As anticipated, the results from both algorithms clearly indicate a significant mismatch between the simulation and reality in the first test. In contrast, the substantially lower scores observed in the second test suggest that, following adjustments to the simulation, it more closely approximates real-world behavior in this case.

8. Conclusion and Outlook

Significant discrepancies between simulation environments and real-world conditions can compromise the performance of intelligent controllers trained exclusively in simulation, rendering them ineffective when deployed on physical systems. This challenge is especially pronounced in safety-critical applications, such as rocket engine control, where unpredictable or unstable behavior is unacceptable. In this study, we examined two methodologies designed to evalu-

DOI: 10.13009/EUCASS2025-533

ate whether a trained agent can be reliably transferred from simulation to real-world operation. Initially, we introduced controlled variations in the simulation dynamics to assess the sensitivity and robustness of these approaches under systematic deviations. Encouraging results from these controlled experiments motivated subsequent validation on real-world test data. Both methods effectively detected discrepancies between simulation and reality, highlighting their promise as reliable tools for monitoring the operational validity of learned policies during deployment.

Building on these findings, future work will focus on embedding explicit safety mechanisms directly within reinforcement learning algorithms, with the ultimate goal of developing intelligent controllers capable of reliably and safely managing rocket engines. The aim is to train neural networks that not only optimize control performance but also rigorously adhere to safety constraints throughout the learning process. To support safe deployment in uncertain real-world environments, we plan to incorporate uncertainty estimation techniques that provide reliable measures of the model's confidence in its predictions. This combination of safety-aware learning and uncertainty quantification is intended to enhance the robustness and trustworthiness of controllers when transferred from simulation to reality.

References

[1] J. A. B. Bossard, W. M. Burkhardt, K. Y. Niiya, and F. Bram. Effect of propellant flowrate and purity on carbon deposition in LO_2/methane gas generators. In *The 1989 JANNAF Propulsion Meeting*, volume 1, 1989.

[2] L. Cheng, Z. Wang, and F. Jiang. Real-time control for fuel-optimal moon landing based on an interactive deep reinforcement learning algorithm. *Astrodynamics*, 3(4):375–386, Dec. 2019.

[3] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. URL `https://arxiv.org/abs/1412.3555`.

[4] S. Colas, S. L. Gonidec, P. Saunois, M. Ganet, A. Remy, and V. Leboeuf. A point of view about the control of a reusable engine cluster. In *Proceedings of the 8th European Conference for Aeronautics and Space Sciences (EUCASS)*, Madrid, Spain, 2019.

[5] K. Dresia. *Rocket Engine Control with Deep Reinforcement Learning*. Ph.d. thesis, RWTH Aachen, 2025.

[6] K. Dresia, T. Traudt, M. Börner, D. Suslov, W. Armbruster, R. dos Santos Hahn, E. Kurudzija, C. Groll, M. A. Müller, S. Klein, J. Hämisch, J. Deeken, J. Hardi, and S. Schlechtriem. Hot-fire testing and system analysis of the lumen liquid upper stage demonstrator engine. In *3rd International Conference on Flight Vehicles, Aerothermodynamics and Re-entry (FAR)*, Arcachon, France, 2025.

[7] B. Gaudet, R. Linares, and R. Furfaro. Deep reinforcement learning for six degree-of-freedom planetary landing. *Adv. Space Res.*, 65(7):1723–1741, Apr. 2020.

[8] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL `http://www.deeplearningbook.org`. Book in preparation for MIT Press.

[9] T. Haider, K. Roscher, F. S. da Roza, and S. Günnemann. Out-of-distribution detection for reinforcement learning agents with probabilistic dynamics models. In *AAMAS*, pages 851–859, 2023. URL `https://dl.acm.org/doi/10.5555/3545946.3598721`.

[10] M. Hampson. Reusable rocket engine turbopump condition monitoring. Technical Report 841619, Space Systems Technology, 1984. Presented at the Proceedings of Space Systems Technology.

[11] S. Heyer, D. Kroezen, and E.-J. V. Kampen. Online adaptive incremental reinforcement learning flight control for a cs-25 class aircraft. In *Amer. Inst. Aeronaut. Astronaut. Sci. Tech. Forum*, Jan. 2020. Art. no. 1844.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735â1780, Nov. 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

[13] K. Hovell and S. Ulrich. On deep reinforcement learning for spacecraft guidance. In *AIAA Scitech Forum*, Orlando, FL, USA, Jan. 2020. American Institute of Aeronautics and Astronautics. Art. no. 1600.

[14] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

[15] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles, 2017. URL `https://arxiv.org/abs/1612.01474`.

DOI: 10.13009/EUCASS2025-533

[16] M. Lausten, D. Rousar, and S. Buccella. Carbon deposition with LOX/RP-1 propellants. In *Proceedings of the 21st Joint Propulsion Conference*, Monterey, CA, USA, July 1985. American Institute of Aeronautics and Astronautics. Art. no. 1164.

[17] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Robust and versatile bipedal jumping control through reinforcement learning, 2023. URL https://arxiv.org/abs/2302.09450.

[18] M. L. Littman. *Algorithms for sequential decision-making*. PhD thesis, USA, 1996. AAI9709069.

[19] V. G. Lopez and F. L. Lewis. Dynamic multiobjective control for continuous-time systems using reinforcement learning. *IEEE Transactions on Automatic Control*, 64(7):2869–2874, July 2019. doi: 10.1109/TAC.2018.2876967.

[20] C. F. Lorenzo and J. L. Musgrave. Overview of rocket engine control. In *Proceedings of the AIP Conference*, pages 446–455, Albuquerque, NM, USA, 1992.

[21] L. Meland and F. Thompson. History of the titan liquid rocket engines. In *Proceedings of the 25th Joint Propulsion Conference*, Sacramento, CA, USA, 1989. American Institute of Aeronautics and Astronautics. Art. no. 2389.

[22] J. L. Musgrave and D. E. Paxson. A demonstration of an intelligent control system for a reusable rocket engine. Tech. Memorandum 105794, NASA, June 1992.

[23] L. Nasvytis, K. Sandbrink, J. Foerster, T. Franzmeyer, and C. S. de Witt. Rethinking out-of-distribution detection for reinforcement learning: Advancing methods for evaluation and detection, 2024. URL https://arxiv.org/abs/2404.07099.

[24] E. Nemeth. Reusable rocket engine intelligent control system framework design, phase 2. Contractor Report 187213, NASA, Sept. 1991.

[25] S. Pérez-Roca et al. A survey of automatic control methods for liquid-propellant rocket engines. *Progress in Aerospace Sciences*, 107:63–84, May 2019.

[26] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[27] R. J. Roelke. Miscellaneous losses. In *Turbine Design Applications*, volume 2. Jan. 1973.

[28] A. Sedlmeier, T. Gabor, T. Phan, L. Belzner, and C. Linnhoff-Popien. Uncertainty-based out-of-distribution classification in deep reinforcement learning. In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence*. SCITEPRESS - Science and Technology Publications, 2020. doi: 10.5220/0008949905220529. URL http://dx.doi.org/10.5220/0008949905220529.

[29] S. Spielberg, R. Gopaluni, and P. Loewen. Deep reinforcement learning approaches for process control. In *Proc. 6th Int. Symp. Adv. Control Ind. Process*, pages 201–206, Taipei, Taiwan, May 2017. IEEE.

[30] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

[31] B. D. Tracey, A. Michi, Y. Chervonyi, I. Davies, C. Paduraru, N. Lazic, F. Felici, T. Ewalds, C. Donner, C. Galperti, J. Buchli, M. Neunert, A. Huber, J. Evens, P. Kurylowicz, D. J. Mankowitz, M. Riedmiller, and T. T. Team. Towards practical reinforcement learning for tokamak magnetic control, 2023. URL https://arxiv.org/abs/2307.11546.

[32] T. Traudt, W. Armbruster, C. Groll, R. H. Dos Santos Hahn, K. Dresia, M. Börner, S. Klein, D. Suslov, E. Kurudzija, J. Haemisch, M. A. Müller, J. C. Deeken, J. Hardi, and S. Schlechtriem. Lumen, the test bed for rocket engine components: Results of the acceptance tests and overview on the engine test preparation. In *9th Edition of the Space Propulsion Conference*, Glasgow, Scotland, 2024.

[33] J. Vilá, J. Moral, V. Fernández-Villacé, and J. Steelant. An overview of the espss libraries: Latest developments and future. In *Proc. Space Propulsion Conf.*, Seville, Spain, 2018.

[34] G. Waxenegger-Wilfing, K. Dresia, J. Deeken, and M. Oschwald. A reinforcement learning approach for transient control of liquid rocket engines. *IEEE Transactions on Aerospace and Electronic Systems*, 57(5):2938–2952, 2021. doi: 10.1109/TAES.2021.3074134.

DOI: 10.13009/EUCASS2025-533

[35] J. Yang, K. Zhou, Y. Li, and Z. Liu. Generalized out-of-distribution detection: A survey. *CoRR*, abs/2110.11334, 2021. URL https://arxiv.org/abs/2110.11334.

[36] C. Yu, J. Liu, and S. Nemati. Reinforcement learning in healthcare: A survey, 2020. URL https://arxiv.org/abs/1908.08796.

[37] W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, 2020. doi: 10.1109/SSCI47803.2020.9308468.