A framework for post-prognosis decision-making utilizing deep reinforcement learning considering imperfect maintenance decisions and Value of Information

Komninos, P.; Zarouchas, D.

DOI
10.1016/j.array.2025.100454

Publication date
2025

Document Version
Final published version

Published in
Array

Citation (APA)
Komninos, P., & Zarouchas, D. (2025). A framework for post-prognosis decision-making utilizing deep reinforcement learning considering imperfect maintenance decisions and Value of Information. *Array*, *27*, Article 100454. https://doi.org/10.1016/j.array.2025.100454

Array 27 (2025) 100454

Contents lists available at ScienceDirect

Array

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

A framework for post-prognosis decision-making utilizing deep reinforcement learning considering imperfect maintenance decisions and Value of Information

P. Komninos *, D. Zarouchas

Center of Excellence in Artificial Intelligence for Structures, Prognostics & Health Management, Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, Delft, 2629 HS, The Netherlands

ARTICLE INFO

Keywords:
Post-prognosis decision-making
Multi-components system
Deep reinforcement learning
Prognostics and Health Management
Imperfect repairs
Value of Information
Uncertainty

ABSTRACT

The digitalization era has introduced an abundance of data that can be harnessed to monitor and predict the health of structures. This paper presents a comprehensive framework for post-prognosis decision-making that utilizes deep reinforcement learning (DRL) to manage maintenance decisions on multi-component systems subject to imperfect repairs. The proposed framework integrates raw sensory data acquisition, feature extraction, prognostics, imperfect repair modeling, and decision-making. This integration considers all these tasks independent, promoting flexibility and paving the way for more advanced and adaptable maintenance solutions in real-world applications. The framework's effectiveness is demonstrated through a case study involving tension-tension fatigue experiments on open-hole aluminum coupons representing multiple dependent components, where the ability to make stochastic RUL estimations and schedule maintenance actions is evaluated. The results demonstrate that the framework can effectively extend the lifecycle of the system while accommodating uncertainties in maintenance actions. This work utilizes the Value of Information to choose the optimal times to acquire new data, resulting in computational efficiency and significant resource savings. Finally, it emphasizes the importance of decomposing uncertainty into epistemic and aleatoric to convert the total uncertainty into decision probabilities over the chosen actions, ensuring reliability and enhancing the interpretability of the DRL model.

1. Introduction

The era of digitalization has offered a vast amount of data that should be processed accordingly to produce insightful information about the engineering assets' current and future condition. Prognostics and Health Management (PHM) plays a pivotal role as being upcoming engineering field that analyzes the health condition of a structure and its components. The general concept of the PHM strategy is depicted in Fig. 1. Given a structure and its components, data are acquired from the placed sensors via one or more Structural Health Monitoring (SHM) techniques, which are stored in a central unit (e.g. computer). Subsequently, the data are processed and features are extracted by a feature extraction model. From the extracted features, one could either perform diagnosis (damage detection, localization) or prognosis, i.e. Remaining Useful Life (RUL) prediction. An intermediate step could also take place between these steps, which is the construction of a Health Indicator (HI). HI represents a unique characteristic derived from SHM data, providing insights into the condition, whether healthy or damaged, of the monitored structure or system [1]. After the prognostics phase, maintenance strategies should be considered and modeled before deciding which maintenance action suits each structure's components, considering structural and operational conditions. Under the concepts of the PHM strategy, this type of decision-making is known as Post-Prognosis Decision-Making (PPDM) [2].

Although substantial research has been conducted on PHM, relatively little attention has been devoted to the critical aspect of the PHM strategy: extending the useful life of structures through effective maintenance scheduling. This objective is the primary driver behind the development of prognostic models. The process of making decisions to prolong the operational life of structures and their components falls under the attention of PPDM. Despite PPDM being introduced differently, the most precise definition was proposed in [3], where PPDM is defined as *the set of actions that should be optimally taken at a given time by satisfying a set of constraints and optimizing a set of objectives*

* Corresponding author.
E-mail addresses: P.Komninos@tudelft.nl (P. Komninos), D.Zarouchas@tudelft.nl (D. Zarouchas).

https://doi.org/10.1016/j.array.2025.100454
Received 25 May 2025; Received in revised form 2 July 2025; Accepted 7 July 2025
Available online 19 July 2025

formulated in an objective (cost) function, to overcome an undesirable upcoming predicted event.

Within the concepts of PPDM, decisions are being made given varying maintenance policies. These policies are rapidly evolving from corrective and preventive maintenance, where maintenance actions are taken after a failure occurrence or in a scheduled-based scenario, to Condition-based maintenance (CBM) that can eliminate the effects of unpredicted failure and optimize operations. CBM offers early warnings of a potential failure, thus proactive decisions about a structure's maintenance plan. Acquiring information from sensory data holds paramount significance in this context. Employing Machine Learning (ML) techniques to analyze this data facilitates the estimation of a structure's present and future condition, particularly in terms of RUL.

The interconnectedness between PPDM and CBM (or Predictive Maintenance[1]) is readily apparent. Although PPDM is still in its infancy, several published works under the CBM umbrella can be found in the literature [4–11]. Nevertheless, these works either assumed a degradation model for creating RUL trajectories or predicted deterministic RUL values which is not representative of a real-case scenario where structures are subject to stochasticity. Related works that considered RUL as the input to the PPDM can be divided into approaches that solve the decision-making problem either numerically or data-driven. The numerical approaches [8,9,12–15] form the cornerstones for constructing more complex PPDM strategies to tackle real-world problems. However, these models are usually tested in numerical examples that are much simpler and less dynamic than a real case. They often consider deterministic variables to describe the inputs, otherwise, it is very difficult to find an optimal solution.

These statements highlight that numerical approaches require domain knowledge and lots of adaptation and preprocessing to design an accurate model [16], thus lacking generalizability and robustness. For example, the authors in [8] followed a model-based PPDM strategy for multi-component systems related to the aircraft industry with a limited stock of spare components. Their novelty lies in maintenance planning for several repairable independent multi-component systems considering that spare parts are not always available. They added the cost of leasing parts from an external supplier when the aircraft shop ran out of spare parts. This strategy managed to schedule long-term horizons and incorporate many different variables. Despite their effectiveness in the aircraft industry, their approach assumes the available days for maintenance, leasing spare parts, and the repair process as deterministic values and not stochastic, which may give overconfidence to the decision-making model, hence increasing the gap between simulation and real-world applications. Similar works related to multi-component systems utilizing the k-out-of-n technique, which is a numerical approach, have been recently published [17–19] and evaluated on numerical examples.

As more data from varying sources are acquired and fused, PPDM solutions are rapidly evolving from numerical approaches to data-driven [6,10,11,20–25], mainly through Reinforcement Learning (RL) after modeling the task as a Markov Decision Process (MDP) or Partially-Observable MDP (POMDP). Data-driven approaches may increase the accuracy and the future horizon on which the decisions take effect. In general, working with stochastic variables and noisy data leads to the need for ML models that are capable of capturing correlations between these variables and accurately approximating their probability distributions. When the demanding input and output data is large, deep learning models usually replace typical ML. In the context of decision-making, RL is replaced by deep RL (DRL), i.e. an ANN is used as a function approximation of the policy (responsible for state–action mapping) that needs optimization. Very recently, DRL emerged on PPDM to offer solutions in tasks with multi-component systems [4,6,8,11,26] where state and action spaces are large, mainly via utilizing the deep Q-Network (DQN) algorithm.

Recent advances in CBM frameworks have emphasized the integration of data-driven approaches with real-world maintenance constraints, enabling more informed and context-aware decision-making in engineering systems. Several works have explored stochastic degradation modeling, uncertainty-aware prognosis, and dynamic maintenance optimization in multi-component environments with imperfect repairs or operational limitations [27–29]. These studies collectively highlight the growing need for maintenance strategies that account not only for RUL predictions but also for the operational feasibility and impact of maintenance actions over time.

When considering maintenance decisions for PPDM, mostly perfect and imperfect maintenance scenarios have been examined. On the first hand, performing perfect maintenance, such as a perfect repair, is usually identical to replacing the structure with a brand-new one. Yet, the significant expenses associated with replacements have prompted a thorough exploration of the viability of repair techniques. On the other hand, imperfect maintenance restores the structure somewhere between the current condition before the repair, i.e. the As-Bad-As-Old (ABAO) condition, and the brand-new condition, i.e. the As-Good-As-New (AGAN) condition. A limited number of studies has considered both perfect and imperfect repairs, with the majority being applied to numerical examples [13,19,30–32] rather than on real-world applications [33,34]. Moreover, all these works have one common limitation; the lack of predicting a component's health state after an imperfect repair, before the repair is actually performed, thus limiting maintenance scheduling optimization. This work marks the first time maintenance actions are scheduled within a horizon even after deciding on an imperfect repair in advance by considering the accumulated uncertainty.

The current work aligns closely with the concept of a selective maintenance policy. A selective maintenance policy refers to the strategic selection of a subset of components to maintain – within each available time slot or maintenance window – based on system condition, component criticality, and limited maintenance capacity. In scenarios where maintenance slots are constrained and not all components can be serviced simultaneously, selective maintenance helps determine which components should be prioritized to maximize system reliability or cost-effectiveness [29]. Despite substantial research into PHM and related CBM strategies, relatively little attention has been devoted to dynamic selective maintenance decision-making under realistic uncertainty. Most prior approaches oversimplify the complexity of real-world applications by assuming deterministic repair effects and constant slot availability.

A typical question in PPDM and generally in sequential decision-making is related to how important the available information is to make a decision without requiring additional data. This is mathematically described by the Value of Information (VoI) [35]. In PPDM literature, VoI is viewed as a gauge of the significance of transitioning to inspection tactics in addition to primary maintenance activities. For instance, the authors in [36] utilized the VoI analysis to facilitate the quantitative assessment of the expected net benefits of collecting new information for non-stationary stochastic or time-dependent decision environments modeled as POMDP with unknown uncertainties. They determined whether to use additional information from inspection actions before taking a maintenance action. Another work suggested improving the quality of maintenance decision-making with the help of three maintenance and three inspection strategies using VoI with an application to a safety–critical marine structure [37]. The objective was to increase the lifetime of the structure by considering both maintenance decisions and inspections, if necessary. Maintenance decisions were derived via integrated crack information through Bayesian updating, thus adding

[1] In the literature related to PPDM, there is often confusion between CBM and Predictive Maintenance. Since PPDM predominantly relies on prognostics, it is frequently associated with Predictive Maintenance. However, some studies classify this under the broader category of CBM. Therefore, in this work, we will use the term CBM to encompass both approaches.

also a measure of the uncertainty. VoI has been additionally considered for making optimal maintenance decisions by taking into account imperfect maintenance scenarios [38–40]. However, these studies once more focused on numerical examples rather than realistic applications simplifying significantly the PPDM step.

To sum up, the existing literature on PHM reveals significant advancements in PPDM. However, despite these strides, several research gaps persist, indicating that PPDM remains in its nascent stages. Firstly, due to the limited number of works related to modeling the RUL behavior after an imperfect repair, existing PPDM frameworks that consider imperfect repairs are scheduling maintenance actions within a horizon until the point of planning that repair. Furthermore, none of the extant works have proposed a comprehensive framework capable of providing actionable decisions and quantifying the associated confidence or probability pertaining to the decision-making process. In other words, interpreting the confidence of the decision-making model in making decisions, especially within the DRL context, is missing from the literature related to PPDM. Additionally, a lack of research exists concerning the correlation between VoI, uncertainty quantification, and decision probabilities, which are pivotal in determining the optimal timing for acquiring new information within the PPDM paradigm. Finally, a thorough understanding of how various sources of uncertainty – epistemic and aleatoric – influence decision-making can significantly enhance our comprehension of the DRL model. This, in turn, facilitates more informed judgments regarding when to trust the model and when to defer to the expertise of a human specialist.

Addressing the aforementioned identified research gaps is imperative for advancing the maturity and efficacy of PPDM methodologies within the broader domain of PHM. In this regard, this study proposes a novel framework related to PPDM based on DRL that works under the concepts of the PHM strategy. Particularly, the scientific contribution and novelty of this research can be summarized as follows:

- This research marks the first PPDM framework that schedules maintenance actions even after an imperfect repair has been planned by estimating the component's health condition after the repair.
- The proposed framework deals with the uncertainty introduced by the stochastic RUL and imperfect repairs. The estimated uncertainty is initially decomposed into epistemic and aleatoric, then is passed through the framework and is converted to probabilistic decisions, thus offering interpretability and a better understanding of the maintenance actions being decided by the DRL model.
- The framework's generalizability based on the user's demands alongside the interpretation of the developed DRL model's confidence over its decisions provides a risk-averse policy. Based on the level of reliability the user demands, decisions taken with relatively low probabilities are transformed into an 'I don't know' output message by the model instead.
- VoI guides the framework in determining the optimal times for acquiring new sensory data to refine its decisions, resulting in computational efficiency and significant resource savings. Simultaneously, it filters out the corresponding unsuccessful runs of the RL agent, guaranteeing reliable scheduling recommendations by the DRL model.

The remainder of this work is organized as follows. Section 2 focuses on constructing the entire framework related to PPDM. Section 3 describes the experimental setup as a case study. The evaluation of the methodology is presented in Section 4 and in Section 5 the main findings and the limitation of the work are discussed.

2. Methodology

The entire process as described in Fig. 1 is particularly depicted in Fig. 2 for this study. After acquiring sensory data from each component of the structure, features are extracted via a unique ANN architecture. Then these features are clustered via a deep clustering model. Subsequently, the clusters are fed to a prognostic model that estimates RUL under uncertainty. The predicted RUL is afterward fed to the imperfect repair model to predict the stochastic recovery of each component after the repair. This model can be extended for additional repairs as well, if the corresponding data are available. After formulating the PPDM task in Section 2.1, details related to developing the feature extraction, prognostic, and imperfect repair models can be found in Section 2.2. Having this information about recovery and predicted RULs of each component, the final step is to implement the PPDM task modeled as an MDP (firstly, as a POMDP and then converted to MDP) as described in Section 2.3, and solved via deep RL (Section 2.4). Fig. 3 depicts the necessary building blocks that form the framework scheme to consider for implementing the PPDM framework. Except for the prognostic and imperfect repair model, putting constraints on the decisions via action masking (Section 2.5) could improve the agent's performance. Additionally, managing the introduced uncertainty and mapping it with the decisions (Section 2.6) is of paramount importance to attach interpretability to the framework. Finally, utilizing VoI assists in choosing the optimal time to acquire new information from sensors to make a new decision (Section 2.7).

2.1. Problem formulation

Before defining the problem, the terminology of several keywords should be defined. 'Slots' represent predefined time intervals in a schedule where maintenance actions can be implemented. Each slot has a 'slot capacity' which defines the maximum number of components that can be put simultaneously for maintenance. A 'task' refers to the problem PPDM should solve. A 'horizon' concerns a time range in which meaningful decisions are made.

The target of PPDM is to make optimal decisions for extending the useful life of each structure's component, hence extending the lifetime of the structure. The current examined PPDM task considers a multi-component system with dependent components. The dependency comes from the slot availability and capacity, meaning that not all components can be maintained on the same day. Given the RUL predictions for each component of a multi-component system, decisions should be made inside a predefined horizon concerning when one or more maintenance actions should be taken. The decisions per component are 'hold', 'imperfect repair', or 'replace' corresponding to 'do nothing', perform an imperfect repair, and replace the component with a brand-new one, respectively. Within the horizon, there are available slots with different capacities per day. The existing slots are not guaranteed to be available every day. An example of the task given two components (with green and red) is illustrated in Fig. 4. In Fig. 4(a), given the RULs of the two components and the corresponding available slots and their capacities (one, three, and two available slots respectively inside the horizon), the policy, i.e. the RL agent, should decide whether and when a maintenance action should be considered. When decisions have been scheduled, the horizon shifts to include the next available slot as shown in Fig. 4(b). Subsequently, the previous decisions are updated accordingly and a new decision is made for the newly available slot.

The difficulty of this task arises when an imperfect repair is decided as the estimated RUL after the repair will have a large uncertainty and the decisions thereafter might be untrustable. This highlights the importance of measuring this uncertainty and converting it to unbiased probabilities over the decisions. In this regard, a safety factor could be added to the framework, namely the probability threshold (p_{thresh}) based on which it is shown whether the RL agent is confident about its decision or not. If the decision has a lower probability than p_{thresh}, then all the decisions from this step till the end of the horizon are transformed into 'N/A', meaning that the agent does not know what decision to make. This increases the reliability of the framework, especially since p_{thresh} can be defined based on the user's demands.

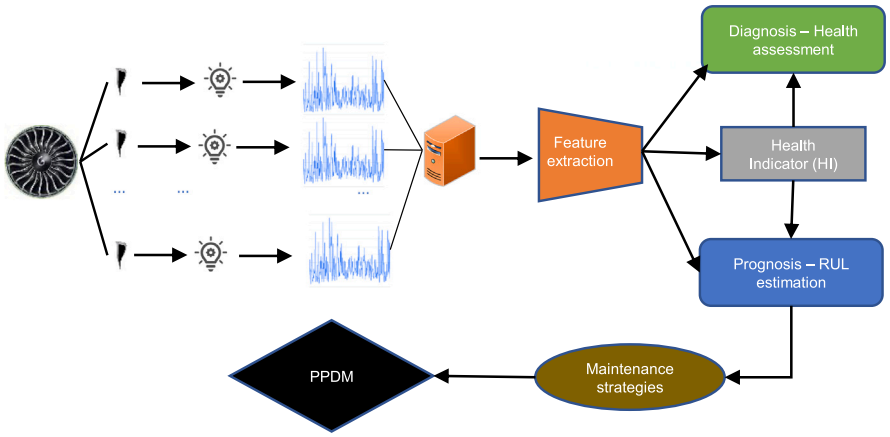


Fig. 1. Illustration of the PHM strategy.

Ideally, an existing scheduling should be updated only when the uncertainty of the agent towards making decisions is high. Reducing the frequency of acquiring new data, extracting features, applying prognostics, and making decisions is of paramount importance for having a computationally efficient framework. Here comes the role of VoI in choosing which day should be the next to run the PPDM framework instead of running it daily.

Before framing PPDM via implementing each building block, it is crucial to highlight all the necessary assumptions:

1. Imperfect repairs restore the state of the structure to a point between ABAO and AGAN conditions, adhering to a truncated normal distribution [13,41,42]. Consequently, RUL following a repair will fall within the range of ABAO and AGAN conditions, neither surpassing the latter nor falling below the former. Additional sequential repairs guarantee a smaller recovery rate of the component than the previous repairs.
2. The prognostic model operates independently of the repair model. The inherent uncertainties are independent.
3. Every specimen, either with or without repair, it exhibits similar sensory data values at the start of each trajectory, as crack growth is not yet detectable. This indicates that the clustering assignments and the RUL estimations are expected to be similar at the beginning of each trajectory.
4. Decisions should be made inside the horizon.
5. The task fails if any of the components' RUL drops below a predefined threshold (RUL_{thresh}).
6. The available slots and the corresponding capacities are static, i.e. if they are defined, they cannot change.
7. To be compatible with the available slots inside the horizon measured in days, the structure's operating cycles per day should be already determined. In this regard, for simplicity, it is assumed that the structure operates for a specific and constant amount of time per day.
8. The user of the PPDM framework defines the maximum number of repairs and replacements. Consequently, the costs are generally considered given a budget constraint.
9. The user additionally decides the horizon length ($L_{horizon}$), RUL_{thresh}, p_{thresh}, and the maximum number of steps (in days) to avoid data acquisition (this corresponds to VoI's contribution).

The choice of p_{thresh} and RUL_{thresh} makes it possible to have risk-averse or risk-prone policies. For example, a risk-averse policy will be considered if p_{thresh} is high or RUL_{thresh} low. Contrarily, if p_{thresh} is small or RUL_{thresh} large, the policy will be risk-prone.

2.2. Feature extraction, prognostics, and imperfect repair modeling

In order to build the PPDM framework, it is crucial to construct the corresponding models of each step presented in the PHM strategy. This includes constructing a model for feature extraction capable of creating health indicators, a prognostic model to predict RUL, and an imperfect repair model to estimate the distribution of recovery after an imperfect repair action. The chosen feature extraction model that performs monotonic clustering representing health indicators is the Deep Soft Monotonic Clustering (DSMC) model, first introduced in [43]. DSMC is an unsupervised deep clustering approach developed for feature extraction in deteriorating systems. The model – based on ANNs – creatively identifies prognostic-related features from raw data through clustering analysis, displaying a gradually rising trend that indicates system deterioration. This trend is not strictly linear but rather flexible to accommodate the possibility of occasional system recovery or the presence of noise in the data, mirroring real-world situations.

After creating the soft monotonic clusters, these trajectories are fed into a prognostic model to perform stochastic RUL predictions utilizing the 95% Confidence Intervals (CI). The chosen prognostic model is the Hidden-Semi Markov Model (HSMM) [44] assuming Gaussian distributions for the observations. Particularly, 7 hidden states were chosen and the model was trained for a maximum of 100 iterations or until the convergence tolerance of 0.5 is met. The DSMC and HSMM models are trained with 5 specimens that reached the end-of-life (EOL), without any repair. Having RUL trajectories before and after imperfect repairs, the corresponding dataset D related to the recovery of the mean RUL is stored. Utilizing D, the imperfect repair model can be trained.

Considering the possibility of imperfect maintenance leads to an improvement in the structure's condition, falling between the ABAO and AGAN conditions. The main aim is to create a model that adequately represents the probabilistic transition from the ABAO state to a better one, influenced by the uncertain nature of subsequent imperfect repairs. This objective is pursued through a gradual process: initially, modeling the first repair and then expanding the model to accommodate multiple repairs. The parameters of the model for a single repair should be trained using Bayesian inference principles to accurately capture the underlying randomness.

Our goal is to estimate the posterior predictive distribution, i.e. the probability of the recovery R given a dataset D ($P(\hat{R}|D)$). To achieve this, following the methodology outlined in [45], the mean recovery (R_{mean}) should be estimated, with the variance of RUL post-repair assumed to be known. Specifically, as the prognostic model remains unaffected by the repair process and mechanical properties remain constant, it is possible, without loss of generality, to compute the corresponding variance by matching the values of equivalent pairs before and after the repair. Essentially, when the prognostic model

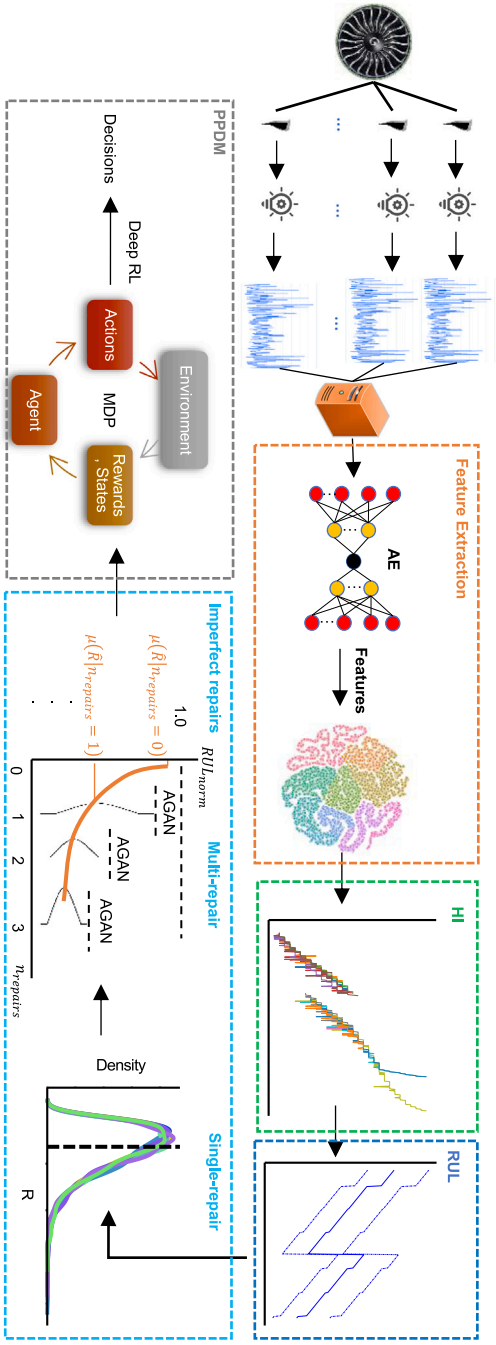


Fig. 2. Description of the entire process as implemented in this work; from acquiring sensor data to PPDM.

operates independently of the repair model, the variance of RUL is influenced solely by external factors, not by behaviors induced by the repairs. This distinction is valuable as it ensures that the variance after a repair is predetermined. Fig. 5(a) visually clarifies this concept. Thereby, only one distribution is needed to estimate R which is R_{mean}, thus $P(R|D) = P(R_{mean}|D)$, where R_{mean} is given by:

$$R_{mean} = \frac{\mu_{new} - \mu_{old}}{1 - \mu_{old}} \tag{1}$$

Here, μ_{old} and μ_{new} are the normalized mean RULs exactly before and after the imperfect repair action. After estimating the distribution of R_{mean}, one could estimate μ_{new} from Eq. (1) as follows:

$$\mu_{new} = (1 - \mu_{old})\, R_{mean} + \mu_{old} \tag{2}$$

It should be noted that μ_{new} and μ_{old} are random variables. To estimate R_{mean}, Bayesian inference is required. Following previous studies [13,42,46], the chosen likelihood is a Truncated Normal distribution, i.e. $P(D|\mu_{mean}, \sigma_{mean}) \sim TruncNorm\ (\mu_{mean}, \sigma^2_{mean}, a_{mean}, b_{mean})$, with prior random variables $\mu_{mean} \sim U(a_1, b_1)$, $\sigma_{mean} \sim U(a_2, b_2)$. To ensure flexibility in choosing different pairs of prior-likelihood distributions depending on the domain knowledge of the imperfect repairs, the Markov Chain Monte Carlo (MCMC) [47] algorithm with No-U-Turn Sampler (NUTS) [48] has been selected to approximate the posterior predictive distribution of R_{mean}.

The same process can be followed for multiple sequential repairs utilizing Eqs. (1) and (2), where μ_{old} and μ_{new} are the normalized mean RULs exactly before and after the nth imperfect repair action. Fig. 5(b) illustrates the behavior of the distribution of R_{mean} for three sequential imperfect repairs. Notice how the distribution shifts towards zero with the increase of number of repairs. As discussed in Section 2, this is attributable to the assumption that each repair results in a reduced percentage of recovery. The red dots are realizations of R_{mean} used to calculate μ_{new}.

Table 1 summarizes the chosen hyperparameters related to Bayesian Inference for estimating R_{mean}. This can be extended to multiple sequential imperfect repairs if trajectories before and after the nth repair are available.

2.3. PPDM task and MDP formulation

The PPDM task as described above naturally fits as a POMDP since the observations that come from the environment are noisy. Nevertheless, as will be further discussed in Section 2.4, POMDP can be converted into a typical MDP by applying recurrent neural networks (RNN) to the ANN architecture to capture the unobserved states.

To establish a task as an MDP, it is crucial to define the environment in which the agent operates. The agent takes actions based on its observations within a finite horizon, $L_{horizon}$, with the goal of prolonging the lifecycle of a multi-component system. This involves extending the lifespan of each individual component or specimen by scheduling imperfect repairs and replacements. Fig. 6 depicts the MDP related to the PPDM task. Particularly, the MDP consists of the following:

- Actions: The agent could take three discrete actions: 'hold', 'imperfect repair', and 'replace', encoded as {0, 1, 2} respectively. The 'hold' action indicates that no action should be taken at the current step, 'imperfect repair' concerns the imperfect maintenance decision, and the 'replace' action is related to replacing the specimen with a brand-new one that starts from its AGAN condition. Depending on the number of specimens that are simultaneously examined, the total number of actions is $3N$, where N is the number of specimens.
- States: The observation (state) space consists of the stochastic RUL, the number of repairs ($n_{repairs}$) and replaces ($n_{replaces}$) that have already been done until step t, the horizon length (days) in which the agent can take actions, the 10 next available slot positions (days) inside the same horizon length, and the capacity of slots available at each slot position. The number of next available slot positions should be large enough to give the agent more exploration space available. All these variables are normalized in the $[0, 1]$ range according to their upper bounds. Particularly, the RUL is normalized based on the largest predicted RUL for each specimen separately, the repairs and replaces are normalized based on their maximum possible sequential repairs and replaces inside the predefined horizon, the normalized capacity is calculated by dividing with the maximum possible available slots given a slot position, and the next available slot positions are divided by the horizon length to calculate the normalized slot positions.

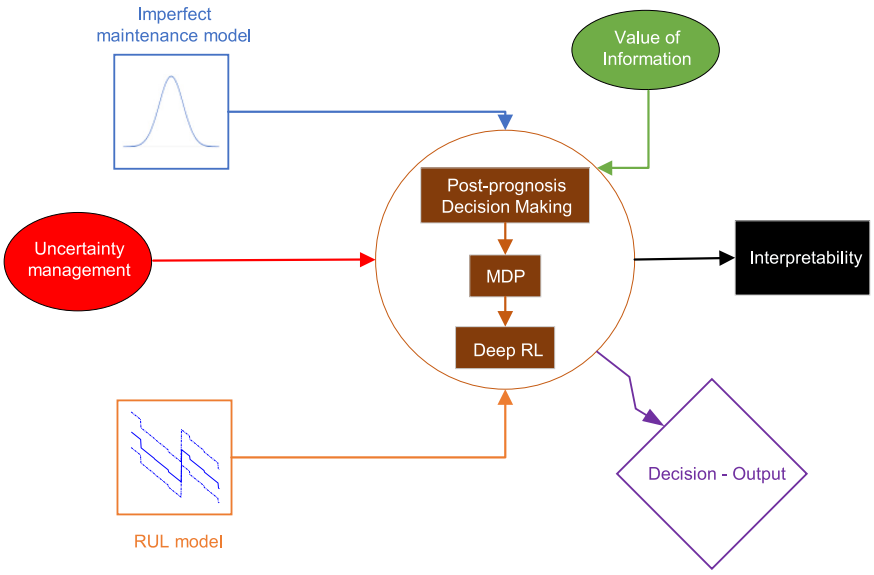


Fig. 3. Basic components for implementing the PPDM task.

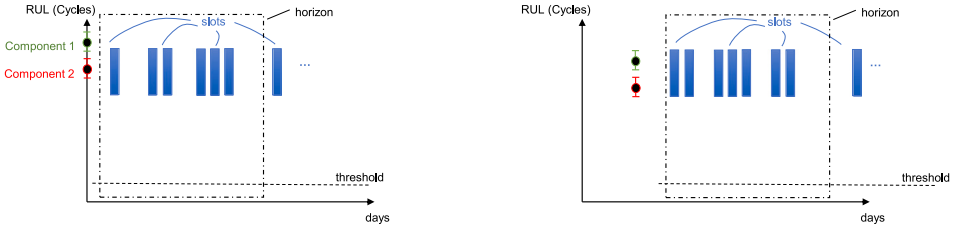


Fig. 4. Illustration of the PPDM task when the first (Fig. 4(a)) and second (Fig. 4(b)) RUL data points are estimated given the real-time acquired data.

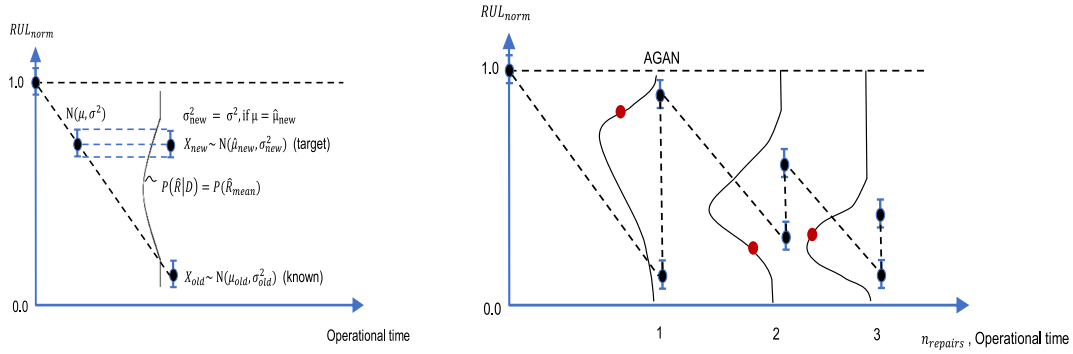


Fig. 5. Recovery distribution under the assumption that the RUL model is independent of the repair process (Fig. 5(a)). Recovery distribution subject to multiple repairs (Fig. 5(b)).

Table 1
Hyperparameters related to Bayesian Inference.

Hyperparameter	Description	Value
α_{mean}	Lower bound of truncated normal distribution	0.4
b_{mean}	Upper bound of truncated normal distribution	1.0
α_1	Lower bound of Uniform distribution for μ_{mean}	0.4
b_1	Upper bound of Uniform distribution for μ_{mean}	0.9
α_2	Lower bound of Uniform distribution for σ_{mean}	0.01
b_2	Upper bound of Uniform distribution for σ_{mean}	0.2
Acceptance rate	Proportion of accepted samples to be added to the chain	0.8
Warmup samples	Initial simulation samples (burn-in samples)	200
Iterations	Simulation samples	50 000
No. chains	Simulated Markov chains	1

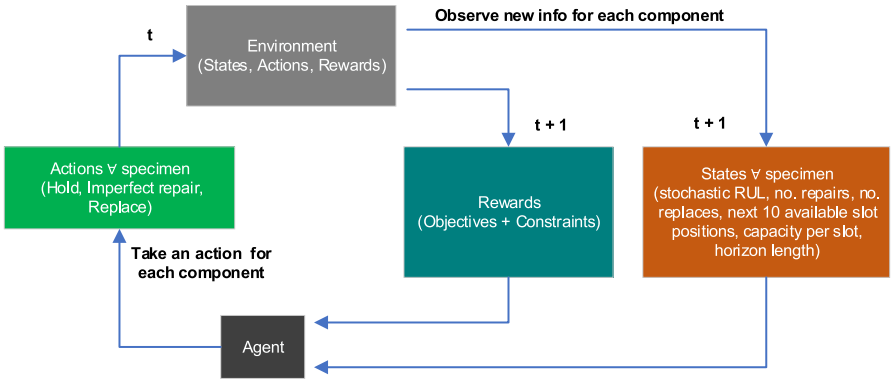


Fig. 6. Illustration of the MDP (states, actions, rewards).

- Rewards: A reward function should be as simple as possible to help the agent reach the optimal policy. In this case, the agent must follow a policy that minimizes maintenance actions (and thus the costs) and extends the structure's health (by extending each specimen's EOL condition). An additional negative reward should be added to the agent when it reaches the terminal state. This state is reached if any specimen's RUL drops below a predefined threshold. After some trial and error, the simplest discovered reward function and its sub-components are given below:

$$r = r_{act} + r_{cost} + r_{maint} + r_{penalty} + r_{success} \tag{3}$$

$$r_{act} = \frac{5t}{L_{horizon}} \tag{4}$$

$$r_{penalty} = -50, \text{ if any } \left[RUL^i < threshold\right], \text{ i} = 1, 2, \ldots, N \tag{5}$$

$$r_{cost} = -2, \text{ if any}(\alpha_t = 1 \text{ or } \alpha_t = 2) \tag{6}$$

$$r_{maint} = \frac{3.5 \cdot \sum_{i=1}^{N} 1 \cdot \left[\alpha_t^i \neq 0\right]}{N} \tag{7}$$

$$r_{success} = 100, \text{ if } t = L_{horizon} \tag{8}$$

Each part of the equation plays a crucial role in optimizing the RL agent. The agent tries to extend the usage of the system via Eq. (4) without dropping below the predefined threshold that activates the penalty (Eq. (5)). In this study, $threshold = 0$, but different values could be given depending on the desired system's safety. For instance, for an aircraft's engine, having $threshold = 0.8$, i.e. the components' RUL should always be above 80% of the perfect healthy condition. Furthermore, the agent incurs a penalty for each scheduled maintenance action (Eq. (6)) to encourage waiting before scheduling any maintenance. The agent also receives a positive reward for consolidating as many maintenance actions as possible within the same time slot via Eq. (7), where N is the slot capacity. With this reward, the agent prompts the scheduling of maintenance actions at the same available slot, promoting a strategy with fewer schedules overall. Finally, the agent is rewarded with a huge value (Eq. (8)) when an episode, i.e. when reaching a terminal state, has been successfully finished.
- Environment: This MDP formulation is episodic, i.e. there are one or more conditions that terminate the episode before resetting the states. The episode terminates successfully if the agent accurately schedules maintenance actions within the horizon without any specimen's RUL dropping below $threshold$. Otherwise, the episode reaches an unsuccessful early termination. When an episode is finished, a new one starts with random initialization of the state space.

2.3.1. Random initialization of environment

The advantage of our approach is that the training is offline using synthetic data, thus having a theoretically infinite amount of training samples. The idea is to train the agent with enough data to always guaranteeing in-distribution real data that will be observed at the evaluation step. Indeed, because sensor data is converted into RUL and the rest of the data in PPDM are related to logistics (slot position, capacity, etc.), achieving a generalized training subject to the user's demands is possible. This is accomplished by randomly distributing the initial RUL values of the environment and the other state variables per episode accordingly following a specified distribution, such as Gaussian, Uniform, and Poisson. Considering this, each state variable is distributed given a random distribution with the following characteristics[2]:

- Initial mean RUL [days]. Varying values of mean RUL are drawn from a Uniform distribution: $U(7, 40)$.
- RUL noise [float]. RUL noise is drawn from a Uniform distribution: $U(1, 5)$. This noise is added to create stochastic RUL trajectories.
- Maximum number of repairs [integer]. At each episode, a different number of maximum repairs is initialized, which follows a Uniform distribution: $U(0, 4)$. The upper bound reflects our case study (four imperfect repairs are allowed before replacement).
- Maximum number of replacements. The maximum number of replacements follows a Uniform distribution: $U(0, 2)$. The upper bound reflects our case study (two replacements are allowed within the horizon).
- Available slot position [days]. Only on specific days of the week, there are available slots. These days are determined by a Poisson distribution $Poisson(\lambda_{Poisson} = 4)$. Thus, there will be available slots to schedule a maintenance action 4 days per week on average.
- Capacity per slot position [integer]. The capacity of each slot position is drawn from a Uniform distribution: $U(1, 4)$. This implies that at each available slot position, 1–4 slots may be available to put specimens for imperfect repair or replacement. One slot corresponds to one component/specimen. The upper bound reflects our case study (four components).
- Distribution of recovery [float]. As estimated by the imperfect repair model. Since this random variable is very sensitive and significantly affects the entire process, it is decided to consider its real estimation based on the model's training utilizing the corresponding training specimens.

[2] It should be marked that each state variable is an array of N values representing each of the N specimens. Each value is randomly distributed, hence different values are drawn for each component which enlarges the observation space and increases the complexity of the environment.

Since there is no prior information related to the PPDM task, Uniform distributions reflect the most uncertain condition.

2.4. The DRL model

The action space at each step in a system with N components should ideally be represented as an N-dimensional array, where each dimension corresponds to a specimen. However, due to the discrete nature of outputs from algorithms like DQN, which are commonly used in PPDM literature, representing such multidimensional arrays directly is challenging. To address this, a workaround involves mapping each combination of actions to an integer, typically using a ternary numerical system. For example, consider a multi-components system with four components and three possible actions each, and at step t the agent chooses the action array $[2, 1, 0, 0]$ corresponding to ['replace', 'imperfect repair', 'hold', 'hold'] for each specimen respectively. Since the DQN algorithm outputs a single integer, the agent should choose between a possible 3^4 discrete actions ranging between $[0, 3^4)$. This integer should then be converted to the ternary system representing the desired array. Here, $\alpha_t = 63$ since $63_3 = 2100 \equiv [2, 1, 0, 0]$. As such, a unique expression for each integer can be assigned. Note that if $\alpha_t < 25$, say, $\alpha = 4$, then $4_3 = 11$, hence $\alpha_t = [1, 1]$ and left zero-padding should be applied until reaching the desired N-dimensional array; here, $\alpha_t = [0, 0, 1, 1]$. Eventually, the action space for the DQN algorithm is $Discrete(3^N)$, but the desired actions should ideally form an N-dimensional array instead. If N is large, even DRL suffers from the exponentially increasing dimensionality of the action space. This can rapidly render the application of discrete-action RL algorithms intractable to domains with multi-dimensional action spaces [49]. This inspires motivation to choose an actor-critic method, such as Proximal Policy Optimization (PPO) [50] that considers N-independent Softmax functions to assign probabilities for each action.

Under the concepts of DRL, the PPO algorithm requires two ANN architectures; one that models the policy (policy network) and another that models the value function (value network). Fig. 7 depicts this architecture. Observations are fed in the shared layers including two Long-Short Term Memory (LSTM) layers and a Fully Connected (FC) layer. Subsequently, the extracted hidden features are fed into the policy and value network. These networks have similar architectures, with the only difference on the final activation function, which is a Softmax for the policy network to produce probabilities[3] for each action, whilst the value network consists of a Rectified Linear Unit (ReLU) activation function that predicts a single value. Between each hidden layer, there are additional layers, namely a Dropout and a batch-normalization layer. Adjacent to each layer shown in Fig. 7 there is a ReLU activation function, except between and after the LSTM layers where a hyperbolic tangent function (Tanh) is considered. In the same figure, the input dimensions of each layer are shown.

Although at each step inside an episode N_{obs} are stored, this architecture demands a length L of previous observations to be used, hence a two-dimensional array $[N_{obs}, L]$. The hidden states extracted by the LSTM layers reflect the unobserved states of the environment, hence silently converting the POMDP into an MDP inside the ANN architecture [11,22]. All hyperparameters related to DRL implemented with the PPO algorithm are stored in Table 2. A linear learning rate scheduler has been implemented to improve the model's learning capabilities. Hence, the learning starts at a faster pace with $lr_{max} = 10^{-2}$ and it linearly decreases until $lr_{min} = 10^{-6}$. The number of observations depends on the number of specimens N and is $N_{obs} = 4N + 21$, where 21 stands for the next 10 available slot positions and the corresponding capacity of each, plus the horizon length). Finally, to have a robust training process, 8 parallel environments are initialized with different randomness ($n_{workers} = 8$). The PPO agent is trained in parallel in these environments.

[3] The output of the Softmax activation function gives an overestimation of probabilities making the ANN overconfident about its predictions. By applying uncertainty quantification techniques to these predictions, it is possible to have an unbiased estimate of the model's beliefs about its decisions [51,52].

2.5. Constraints & action masking

In RL, action masking is a technique used to restrict the set of actions that an agent can take in a given state. This restriction is based on the environment's rules or constraints, preventing the agent from selecting actions that are not permitted or valid in that particular state. The theory behind action masking lies in the idea of creating a more realistic and efficient learning environment for the RL agent. By limiting the available actions, action masking reduces the complexity of the learning problem by focusing the agent's attention on only the relevant actions. This can be particularly useful in environments where certain actions are not feasible or allowed in specific states, or where the action space is large and needs to be pruned to improve learning efficiency.

Action masking is applied to the action space based on the current state of the environment. It involves determining which actions are permissible or valid in the current state and filtering out the rest. This is typically done by defining a mask vector that indicates the availability of each action. The agent's policy and value estimation are then computed based on the masked action space. By excluding invalid actions, the agent can focus its learning efforts on the subset of actions that are relevant to the current state. During the learning process, the agent explores the environment by selecting actions and observing the resulting rewards and next states. Action masking ensures that the agent only considers actions that are permissible in each state, thereby guiding exploration towards more promising areas of the state–action space.

In DRL, action masking is mainly applied in the last layer of the ANN. For the PPO algorithm in discrete action spaces, such as the one examined in this study, the ANN's last layer used to approximate the policy typically contains a Softmax activation function to produce a measure of probabilities for taking each action. Consider the outcome of the ANN's last hidden layer to be z_{α_t}. This represents the logits of each action α_t. Then the probability of each action α_t given state s_t at step t is calculated by:

$$\pi_\theta(\alpha_t|s_t) = \frac{exp(z_{\alpha_t})}{\sum_{\alpha_t'} exp(z_{\alpha_t'})} \tag{9}$$

The invalid action masking technique [54–56] for discrete domains underlines that each invalid action can be masked (its probability is set to zero) by assigning a huge negative number to the corresponding logits z_{α_t}. Hence, after passing this logit through the Softmax activation function, the corresponding probability of taking this action will be very close to zero.

In this work, actions related to 'imperfect repair' or 'replace' are invalid when the agent has reached the maximum number of repairs or replacements, respectively, within an episode. Action masking is applied to the corresponding maintenance actions that ought to surpass those limits by assigning a large negative number before applying the Softmax activation function to assign a zero probability to the corresponding action. Additionally, the number of maintenance actions within an available slot should never exceed the capacity of that slot. A similar action masking approach is applied to the corresponding maintenance actions with the lowest probability scores until the capacity limit is satisfied.

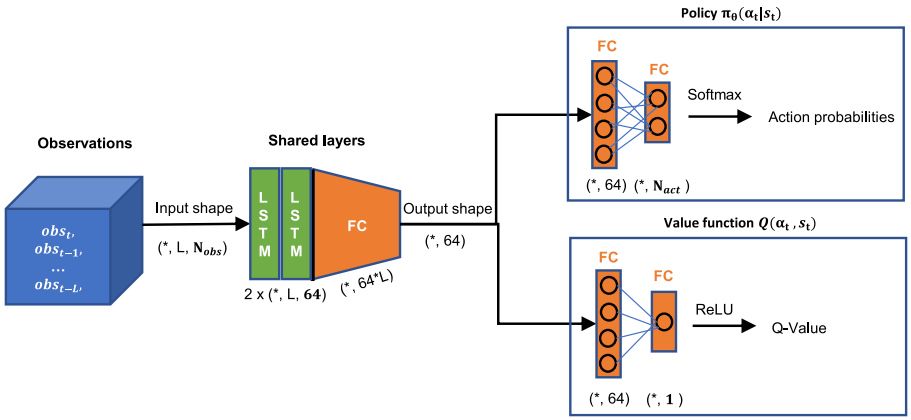


Fig. 7. ANN architecture for both policy network and value function. N_{obs} and N_{act} reflect the number of observations and actions respectively. The '*' dimension reflects the batch dimension.

Table 2
Hyperparameters related to DRL utilizing PPO algorithm.

Hyperparameter	Description	Value
lr_{min}	Minimum learning rate	10^{-6}
lr_{max}	Maximum learning rate	10^{-2}
n_{steps}	Number of steps per update of weights	64
$batch$	Batch size	128
$buffer$	Replay buffer limit to use for retraining the model with SL	$8.4 \cdot 10^6$
$epochs$	Number of epochs when optimizing the surrogate loss	5
n_{obs}	Number of observations	$4N + 21$
L	Length of past observations to be used by the LSTM layers	8
d_{rate}	Dropout rate	0.5
γ	Discount factor	1.0
λ_{GAE}	Factor for trade-off of bias vs variance for GAE	0.95
ϵ	Clipping parameter	0.2
$C_{entropy}$	Entropy coefficient for the loss calculation (see [53])	0.01
$steps_{train}$	Total training steps	10^6
$n_{workers}$	Environments to run in parallel with different randomness	8

2.6. Mapping uncertainty with decision probabilities

The policy network is responsible for assigning probabilities to each available action at each step. However, due to the tendency of ANNs to be overconfident, these probabilities can be easily either close to 1.0 or 0.0. In order to have unbiased estimates of these probabilities, the Mutual Information (MI) theory will be considered by decomposing the total uncertainty into aleatoric and epistemic uncertainties [57].

Given a dataset $D = \{X, Y\}$ within the context of a discriminative classification task, the aleatoric uncertainty can be determined using Shannon's entropy [58], computed as the expected conditional entropy:

$$\mathbb{E}_{p(x)}\left[H[p(y \mid x)]\right] = \mathbb{E}_{p(x)}\left[-\sum_{c=1}^{K} p(y=c \mid x) \ln p(y=c \mid x)\right] \tag{10}$$

where c is the corresponding class (in a classification task) or action (in an RL task), K is the number of classes/actions, and $H[p(.)]$ is the entropy. The entropy of a discrete probability distribution is an information-theoretic measure of uncertainty and is calculated by MI [59]:

$$I(y,x) = KL[p(x,y) \| p(x)\, p(y)] = H[p(y)] - \mathbb{E}_{p(x)}[H[p(y|x)]] \tag{11}$$

where KL is the Kullback–Leibler (KL) divergence [60]. It has been proven in [61] that the decomposition of the uncertainty into epistemic and aleatoric in deep learning via the MI theory can be performed utilizing M number of ANNs (deep ensembles) as follows:

$$I(y,\ M|,x,\ X,\ Y) = H\left[\mathbb{E}_{p(M|X,Y)}[p(y|x,M)]\right] - \mathbb{E}_{p(M|X,Y)}[H[p(y|x,M)]] \tag{12}$$

In this equation, the term to the left reflects the epistemic uncertainty, the first term on the right side is the total uncertainty, and the second on the right is the aleatoric uncertainty. From Shannon's entropy, the total uncertainty is easily calculated. To estimate the aleatoric uncertainty, it is crucial to approximate the epistemic first. However, estimating epistemic uncertainty is difficult since we need to calculate the posterior distribution over the parameters θ. To capture the epistemic uncertainty in an ANN, one approach is to use deep ensembles [62,63], i.e. utilizing M models that converge differently due to varying random initializations. For DRL tasks this is computationally prohibitive. A better approach is to approximate these M models with the Monte Carlo (MC) dropout technique. MC dropout is acknowledged as a Bayesian approximation, thereby enabling standard point predictions alongside meaningful uncertainty assessments [64].

Given a set of parameters Θ representing the weights of an ANN, the posterior predictive distribution of y^* given x^* and D equals:

$$p\left(y^* \mid x^*, D\right) = \frac{1}{T}\sum_{i=1}^{T} p\left(y_i^* \mid x_i^*, \Theta\right) \tag{13}$$

where T is the number of forward passes performed to predict a particular data point y given a single input sample x. In practice, T stochastic forward passes are conducted through the ANN with dropout applied at each layer for every sample, followed by extracting the mean and variance from the outcomes.

Setting Eq. (13) to Eq. (12) gives the following:

$$\begin{aligned}\text{Model uncertainty} &= \text{Total uncertainty - Aleatoric uncertainty} \\ &= \frac{1}{T} H\left[\sum_{i=1}^{T} p\left(y_i^* \mid x_i^*, \Theta\right)\right] - \frac{1}{T}\sum_{i=1}^{T} H\left[p\left(y^* \mid x^*, \Theta\right)\right]\end{aligned} \tag{14}$$

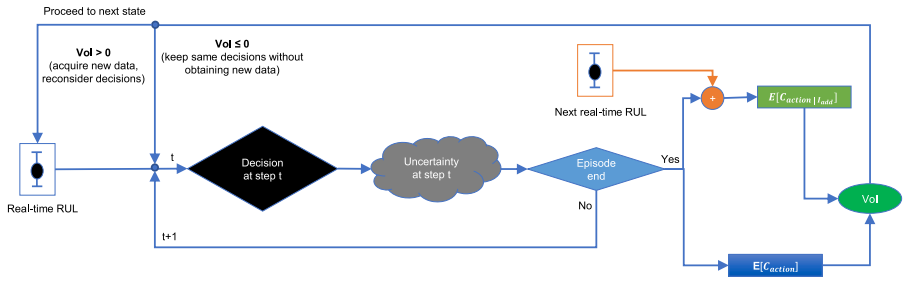


Fig. 8. Loop for deciding whether to acquire new information based on estimated VoI.

In the case of three discrete actions that are passed through the Softmax activation function $p(y^* \mid x^*, \Theta)$ corresponds to the array $[p_{i1}, p_{i2}, p_{i3}]$ where p_{ij} is the probability of taking action $j = \{1, 2, 3\}$ at stochastic pass $i \in [1, T]$. In this study and according to a previous work [64], $T = 30$ has been chosen. Taking the average over T stochastic passes makes an unbiased estimate of the probabilities of taking each action a_t at step t feasible. In this regard, the policy that the agent follows can give real probabilities of decisions. These decisions are interconnected with the measured uncertainty, which is crucial for bounding its upper values for normalization purposes. This is justified by Lemma 1 discussed in Appendix A.2. This means that the uncertainty can be normalized by dividing it by $log3$. This guarantees that the uncertainty will always be in the range of $[0, 1]$. This is useful for training the VoI model and utilization of p_{thresh} concerning reliability.

Lemma 2 presented again in Appendix A.2 analyzes the relationship between p_{thresh} and the combined epistemic and aleatoric uncertainty. In this regard, we can ensure the physical meaning of the measured uncertainty, thus adding interpretability to our DRL model. Depending on the defined by the user p_{thresh}, if the maximum of the probabilities is less than this threshold, the agent's decision is converted to the *'N/A'* action. This is a hidden action that is not seen inside the MDP but is added after estimating the uncertainty during inference. In this context, when the agent is uncertain about its final decision, it is advisable to not rely solely on DRL and instead wait for human feedback. This approach renders the task risk-averse.

2.7. Deciding optimal time to acquire new information

Acquiring new data in real-time every day is cost-prohibitive and sometimes unnecessary. For instance, when all specimens are in their AGAN condition, it is already known that the agent should wait for the initial steps before taking any maintenance action. Consequently, there is no need to acquire new information in the subsequent steps, thus conserving resources. Avoiding acquiring unnecessary information can be achieved by utilizing VoI. The VoI theory is a concept used in decision theory to quantify the benefit or utility gained from acquiring new information. At its core, it explores how obtaining additional information can enhance decision-making processes, particularly in situations where uncertainty exists. VoI represents the value gained from acquiring new information at the expense of costs.

In the RL field, VoI is often analyzed within the framework of expected value. The agent calculates the expected value of different actions based on their probabilities of occurrence and associated payoffs. By incorporating the potential value of acquiring additional information, the agent can adjust their actions to maximize expected utility. As introduced in varying prior works [36,40,65], VoI is defined by the following equation:

$$VoI = \mathbb{E}\left[C_{action}\right] - \mathbb{E}\left[C_{action}|_{I_{add}}\right] \tag{15}$$

where C_{action} is the cost of taking a specific action by the agent and $C_{action}|_{I_{add}}$ is the cost of taking an action given additional information. In this study, the expected action costs represent the total uncertainty of the agent's decision. When $VoI < 0$, there is no need to acquire additional information and reconsider the chosen action.

Fig. 8 depicts the loop based on which the algorithm automatically decides whether to acquire new information. Recalling that each episode in the formulated MDP finishes successfully when the agent reaches the end of the horizon or fails when any of the specimens' RULs drops below the threshold. This means that given the current RUL information in real-time, operational conditions, and other logistic information, the agent creates a schedule for future maintenance actions within the horizon. During each episode, uncertainty can be estimated for each step. Recalling Eq. (15), the expected costs in this study are related to the estimated uncertainty. The greater the degree of uncertainty, the higher the expected costs. VoI is applied throughout the entire episode as the average of the estimated uncertainty measured at each step. Therefore, the expected cost given a series of actions for each episode is as follows:

$$\mathbb{E}\left[C_{action}\right] = \frac{\sum_{i=0}^{steps_{episodic}} U^{(i)}}{steps_{episodic}} \tag{16}$$

where $steps_{episodic}$ represents the total steps until the episode has finished, either successfully or not, and U is the total uncertainty. If $\mathbb{E}\left[C_{action}|_{I_{add}}\right]$ is estimated, VoI can be measured. If VoI is non-positive, the uncertainty estimated at the current episode is less than the expected uncertainty that will be observed in the next episode. In this case, acquiring information about the next episode is unnecessary. This can be extended to ignoring multi-steps ahead by counting the times that VoI is sequentially non-positive. Contrarily, when VoI becomes once positive, the expected uncertainty included in the next episode is less than the current one, hence the upcoming data are expected informative for the agent to update the decisions.

The formulation of VoI, though, demands the estimation of $\mathbb{E}\left[C_{action}|_{I_{add}}\right]$. Unfortunately, this information is not available beforehand. The typical solution is to apply pre-posterior analysis in VoI. However, given the large number of state space, this is computationally expensive (since Bayesian updates are being performed). In this regard, the approach proposed in this work is to create a parameterized surrogate model f_θ based on ANN with parameters θ that outputs $\mathbb{E}\left[C_{action}|_{I_{add}}\right]$ s.t.:

$$\begin{aligned}
\mathbb{E}\left[C_{action}|_{I_{add}}\right] &= f_\theta(\mathbb{E}\left[C^{tot}_{action}\right]_{n_{ep}}, \left[S_{0,n_{ep}}\right]) \\
\mathbb{E}\left[C^{tot}_{action}\right]_{n_{ep}} &= \{\mathbb{E}\left[C_{action}\right]_{n_{ep}-m}, \mathbb{E}\left[C_{action}\right]_{n_{ep}-m+1}, \dots, \mathbb{E}\left[C_{action}\right]_{n_{ep}}\} \\
\left[S_{0,n_{ep}}\right] &= \{s_{0,n_{ep}-m}, s_{0,n_{ep}-m+1}, \dots, s_{0,n_{ep}}\}
\end{aligned} \tag{17}$$

Here, a state $s_{0,n_{ep}}$ corresponds to the first available observations of the episode n_{ep}, m is a hyperparameter representing the number of previous episodes to consider as input for the surrogate model to make n-episodic predictions ahead, and $\mathbb{E}\left[C_{action}\right]_{n_{ep}}$ is the expected cost at episode n_{ep} as calculated by Eq. (16). Utilizing an ANN to make such predictions given sequential data leads to the choice of an

LSTM-based architecture illustrated in Fig. 9. Three non-bidirectional LSTM layers are structured. Between each layer, a Tanh activation function has been applied. Then, the dimensions are flattened into a one-dimensional array (the batch dimension remains untouched). This array is passed through three FC layers. Adjacent to the first two FC layers, a ReLU activation function is applied, whilst the last FC layer has a Tanh activation function. After each layer, batch normalization and Dropout are additionally considered. The hyperparameters of the surrogate model are stored in Table 3. A linear learning rate scheduler has been implemented to improve the model's learning capabilities. The optimization is performed with the Adam optimizer.

The key concept is to generate data for training the VoI model by evaluating the trained agent at each episode. For each episode n_{ep}, the corresponding $\mathbb{E}\left[C_{action}\right]_{n_{ep}}$ is calculated while simultaneously the initial observations $s_{0,n_{ep}}$ of that episode are being stored. Generating episodes sequentially based on a real case study means that each sequential episode contains the next available information regarding input data. For example, being at $n_{ep} = 2$ means that the expected costs $\mathbb{E}\left[C_{action}\right]_{n_{ep}=1}$, $\mathbb{E}\left[C_{action}\right]_{n_{ep}=2}$ and $\mathbb{E}\left[C_{action}|_{I_{add}}\right]_{1\to 2}$ are known. By specifically defining the case study, the RUL trajectories and the distribution of recovery for multiple repairs are available. Hence, it is possible to initially evaluate the RL agent for generating data related to VoI in an offline manner utilizing the training trajectories considered for estimating the distribution of recovery, before rendering the scheduling in real-time with the testing trajectories.

Generating data for training the VoI model requires an adaptation of the environment used for training the RL agent. By initializing the first episode to reflect a random day based on the RUL trajectories, each sequential episode should reflect the exact next day. Then, scheduling is applied for each episode and the uncertainty can be estimated. This time the episodes have a strong time dependency, which justifies the utilization of the LSTM module. After trial and error, generating 50 sequential episodes before randomly initializing the first episode for a new sequence was enough to train the VoI model. Having 50 generated sequential episodes corresponds to the hyperparameter m, hence $m = 50$. Consequently, in this setup, the evaluation of the agent can be applied in an offline manner and is possible to generate as much episodic data as desired. Subsequently, labeled data can be generated to be fed into the surrogate modeling for training. In total, 10000 episodes were generated, hence $10000/50 = 200$ random initializations were performed regarding the first episode of the sequence. From these samples, 8000 is considered to train and 2000 to validate the VoI model.

Finally, it is important to note that both successful and unsuccessful episodes are included as training data. An unsuccessful episode suggests that the (considerably well-trained) DRL model failed to make the correct decisions, resulting in higher uncertainty. In this context, the trained VoI model learns to filter out these unsuccessful episodes by suggesting the avoidance of data acquisition on these days. This indirectly enhances the DRL agent's performance, at the expense of fewer schedules within a given horizon. Although this approach recommends fewer schedule updates by the agent, it is more cost-efficient because it reduces the number of days considered to acquire new data and run the framework.

3. Case study

Consider a multi-component system consisting of a structure with four components/specimens. These components correspond to open-hole aluminum coupons. Each working day the structure functions for a specific and constant number of cycles. Maintenance actions should be made for the structure's components to extend the useful life of the entire structure. This extension concerns approximately doubling the average EOL of the components. It is assumed that this average EOL is 25–30 days, thus extending the structure's life up to 60 days ($L_{horizon} = 60$) suffices. In order to increase the useful life by this amount within the horizon, four sequential repairs and two replacements are allowed. This means that the maximum possible number of maintenance actions is 10 if four repairs are planned between two replacements. Nevertheless, scheduling multiple maintenance actions is both cost-prohibitive and pointless, especially in case when additional imperfect repairs do not offer a significant recovery of the component.

Table 3
Hyperparameters related to surrogate model for the estimation of $C_{action}|_{I_{add}}$.

Hyperparameter	Description	Value
lr_{min}	Minimum learning rate	10^{-6}
lr_{max}	Maximum learning rate	10^{-2}
$batch$	Batch size	128
$epochs$	Number of epochs when optimizing the surrogate loss	400
m	Length of past data to be used by the LSTM layers	30
n	The number of multi-episode predictions	7
$samples_{train}$	The number of samples considered for training	6400
$samples_{val}$	The number of samples considered for validation	1600
$samples_{test}$	The number of samples considered for test	2000

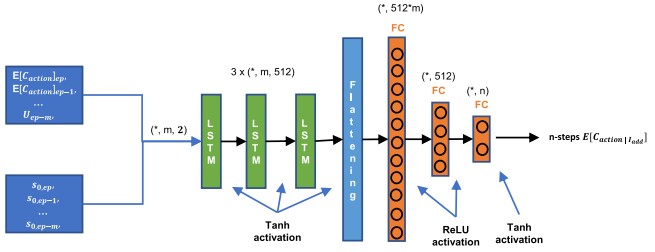
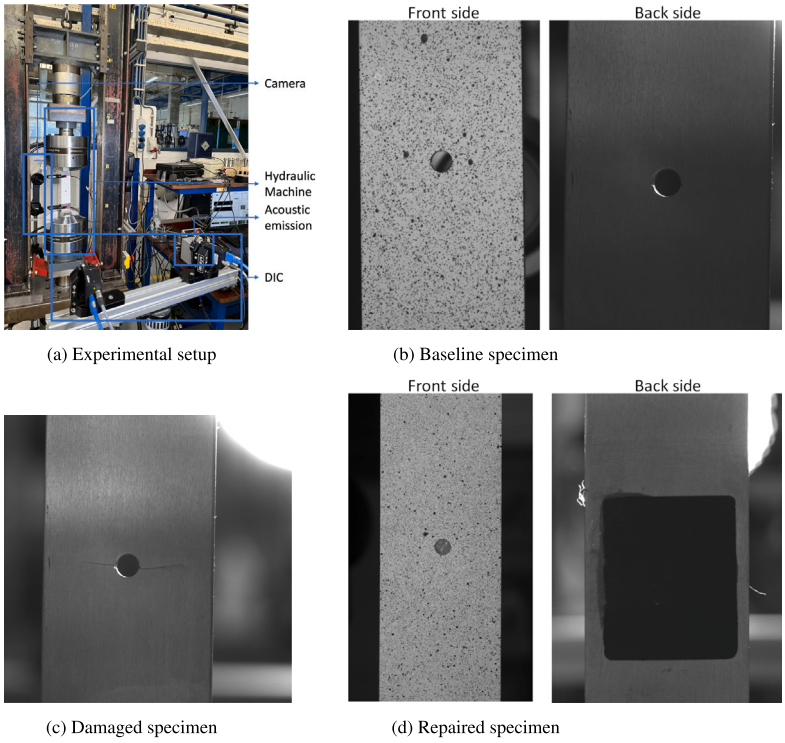


Fig. 9. ANN architecture of the surrogate model.

To test the capabilities of the methodology, feature extraction, prognostics, and imperfect repair modeling have already been implemented in a previous work [45] based on an experimental campaign as shown in Fig. 10 involving tension-tension fatigue experiments on open-hole aluminum coupons subject to imperfect repairs. Fig. 10(a) illustrates the experimental setup and the different specimen conditions can be found in Figs. 10(b)–10(d), for the baseline, damaged and repaired conditions respectively. The same campaign is considered for the PPDM task. The raw signals corresponding to acoustic emission data are recorded by an AMSY-6 Vallen Systeme GmbH and two VS900-M wideband sensors. From the recorded data, the considered low-level features include amplitude, duration, energy, counts, hit time, rise time, root mean square error, and signal strength and are presented in Table B.1. The initial five coupons are tested until failure to gather information about their fatigue life before repair and to determine the consistent lifetime percentage at which the repair is conducted. The remaining coupons undergo testing for 14,000 fatigue cycles, equivalent to 60% of the average fatigue life. Half of these are then tested until failure, while the other half are stopped at 11,000 cycles, which is about 60% of the average repaired coupon lifetime, demonstrating that it is not necessary to reach failure to develop the repair model. More detailed information about the specimens can be found in Table B.2. Since the trajectories of specimens with and without repairs are different, their names will be defined with two numbers for clarity; the first number indicates the number of repairs performed, and the second refers to the specific specimen. For example, the specimen labeled '02' without any repair is named specimen '0_2'. Similarly, specimen number '10' is split into two names: '0_10' for the trajectory without repair and '1_10' for the trajectory with repair.

Since there have not been any practical experiments in the literature related to multiple imperfect repairs, additional trajectories have been generated based on the existing ones according to the following two steps:



(a) Experimental setup (b) Baseline specimen

(c) Damaged specimen (d) Repaired specimen

Fig. 10. The experimental campaign as implemented in [45]. Experimental setup (a) and specimen states (baseline (b), damaged (c), repaired (d)).

i. The first RUL data point after a repair corresponding to μ_{new} and, consequently, the mean of the distribution of recovery $\mu(R_{mean})$ is exponentially reduced according to the following equation:

$$\mu\left(R_{mean}|n_{repairs}\right) = \kappa \cdot exp(-\lambda n_{repairs}),\ n_{repairs} \in 0,\ 1,\ 2,\ \ldots \quad (18)$$

where λ, κ are the parameters that need to be determined, with λ being the rate parameter, and κ a constant. By setting $n_{repairs}$ to zero, we have:

$$\kappa = \mu(\hat{R}_{mean}) \quad (19)$$

The above equation can be used after training the imperfect repair model for one repair. The variable λ can be any float number in the range of $[0, 1]$. To increase the complexity of the task, a sharp exponential decrease has been chosen with $\lambda = 0.7$. Consequently, after the second repair, it is expected the distribution of recovery to be close to zero, thus the next repairs will not offer any significant recovery. It is expected that the RL agent should capture this phenomenon during training.

ii. Starting with the first RUL data point after the nth repair, the corresponding RUL data point from the $(n-1)$th repair with the same value is identified. The entire trajectory is then generated by using the mean and variance of the trajectory from the previous repair. Additional noise, modeled by a Uniform distribution $U(-3, 3)$ measured in days, is added to this generated trajectory.

The DSMC model and the HSMM are trained on specimens '01'-'05', and, then, both models are utilized to predict RUL on the rest of the specimens. From the specimens subject to imperfect repairs, specimens '09'-'11' are taken into account for training the imperfect repair model based on which the RL agent is trained as well. Additionally, the RL agent was evaluated considering these specimens to generate the required data related to VoI. Specimens '06'-'08' correspond to ones used for evaluating the imperfect repair model and the outcome of the PPDM framework. During the evaluation phase of the RL agent, another RUL trajectory is generated, namely specimen '12', by taking the average values of specimens '06'-'08', in order to include four components in the multi-components system. Trajectories for the second, third, and fourth sequential repairs are generated utilizing the aforementioned process.

Table 4
Chosen hyperparameters related to PPDM for this case study.

Hyperparameter	Description	Value
$n_{components}$	No. components/specimens	4
$L_{horizon}$ [days]	Horizon length	60
$n^{max}_{repairs}$	Maximum no. sequential repairs	4
$n^{max}_{replaces}$	Maximum no. replaces	2
p_{thresh}	Probability threshold	0.6
$skip^{max}_{episodes}$ [days]	Maximum no. episodes to skip based on VoI	10
$slot_{pos}$ [days]	Weekly slot availability as a random variable	$\lambda_{Poisson} = 4$
$capacity$	Capacity of available slots per available day	$U(1, 4)$

Table 5
Characteristics of test specimens/components for this case study.

Test specimen No.	Component No.	EOL (cycles)	EOL (days)
06	0	24 565	29.49
07	1	17 445	20.94
08	2	17 250	20.71
12	3	19 753	23.71

Concerning the PPDM task, the hyperparameters for this case study are stored in Table 4. These hyperparameters are defined by the user of the framework each time before training the RL agent. In this setup, we seek to extend the EOL condition by doubling the average total number of working cycles. Particularly, since the average EOL before repair is approximately 25000 cycles, the structure should be extended up to 50000 cycles by performing maintenance actions to the components. This number of cycles is chosen to reflect a horizon of 60 working days. The structure containing these components is assumed to be working for a fixed and constant amount of time every day. In this regard, each day corresponds to $50000/60 = 833$ working cycles. In Table 5, the relation between working cycles and days for the four test specimens is shown.

4. Results

In this section, the results of the entire framework are presented and discussed. Since the data of the chosen case study comes from the work presented in [45], the feature extraction (clustering assignments) and the distribution of recovery for a single repair are illustrated in Appendix C, for comprehensiveness, and will not be discussed again. However, the behavior of the distribution of recovery for multiple repairs will be analyzed in Section 4.1 and the PPDM outcome in Section 4.2.

4.1. Single and multiple repairs

Three specimens are considered to train the imperfect repair model and another three to evaluate its performance. Based on the clustering assignments provided by the DSMC model, as shown in Fig. C.1, prognostics are implemented via the HSMM algorithm and illustrated in Fig. C.2. From these RUL trajectories before and after the first repair, the distribution of recovery can be estimated using the MCMC technique. This distribution represents the mean of recovery, while the variance can be estimated by the corresponding variance of RUL before repairing the specimen, under the assumption that the RUL model is independent of the repair process, as mentioned in Assumption 2. This can be extended for multiple repairs by assuming an exponential decrease in the mean of recovery. Having the distribution of recovery after the first repair, κ can be calculated by Eq. (19), hence $\kappa = 0.67$. Determining the values of λ and κ, three additional distributions are estimated for representing two, three, and four sequential repairs utilizing Eq. (18). These distributions alongside the distribution representing the first repair can be seen in Fig. 11.

As expected, because the exponential reduction was chosen on purpose to be sharp, from the second repair and thereafter the distribution of recovery drastically tends to zero. Consequently, in this case study, performing more than one or two repairs sequentially may not assist in extending the useful life of the examined component. Therefore, a well-trained agent is expected to make an 'imperfect maintenance' decision once or twice before considering a replacement of the component with a brand-new one.

4.2. Performance of the RL agent

The RL agent was trained on a single GPU (NVIDIA GeForce RTX 2080). The entire training process for one million steps was approximately 40 min. The performance of the RL agent concerning the case study considering four components is depicted in Fig. 12. Particularly, Fig. 12(a) shows the agent's reward gains as training progresses and Fig. 12(b) the ratio of successful episodes, i.e. the episodes where the agent has successfully reached the end of the horizon. Based on these graphs, training for 1000 episodes is enough to achieve high performance. These plots reflect the mean values after taking a rolling horizon with a rolling window of 100 for all plots.

Decomposing the uncertainty into epistemic and aleatoric provides us with insightful information related to whether the RL agent struggles to take a decision due to difficulties in training (epistemic) or to noisy data (aleatoric). Recalling that the numbers 0, 1, and 2 represent the maintenance actions 'hold', 'imperfect repair', and 'replace' respectively, in Fig. 13, the left part illustrates a part of an episode where the agent makes decisions with probabilities based on the red bars. Additionally, the corresponding uncertainties are depicted. During inference, each of these steps represents the average probability after running the step T times as discussed in Section 2.6. The process of constructing two of these steps is illustrated in the right part of the figure where the probabilities are shown with blue bars. The upper graph shows the step with large aleatoric uncertainty since each of the MC samples gives almost equal probabilities in choosing either 'hold' or 'imperfect maintenance' decision. The lower graph has zero uncertainties since the agent chooses the 'hold' decision with maximum confidence for all the MC samples. Although not shown here and because of the satisfying training of the RL agent, it is noteworthy that the epistemic uncertainty is always approximately or exactly zero. Nevertheless, if the agent is poorly trained, having large epistemic uncertainty would be easily observable since each MC sample representing a specific step would give quite different probabilities over the actions.

4.3. Scheduling maintenance actions

During the evaluation phase, the environment was run for 60 sequential episodes equal to the horizon length. In Fig. 14 the scheduling representing the first episode is visualized. In this schedule, the actual EOL of each component is shown, only before taking any maintenance action. Here, some key findings are observed that validate the satisfactory training of the RL agent based on the constructed reward function.

Firstly, the agent prioritizes scheduling maintenance actions for multiple components on the same day when the slots' capacity allows it. This leads to two consequences. On the first hand, there is a preference for maintaining the components earlier but on the same day rather than waiting for more days, even though RULs may allow it. On the other hand, the agent prioritizes scheduling earlier to avoid any RUL dropping below the threshold which signifies an unsuccessful episode.

Secondly, only one imperfect repair action is considered before replacement, which aligns with the sharp exponential decrease in recovery. The agent has learned that performing more than one sequential repair does not significantly increase the RUL and is cost-prohibitive. Simultaneously, a repair always precedes a replacement to extend the useful life as much as possible.

Thirdly, on day 57 and thereafter the agent considers deciding with a probability less than p_{thresh}. Thus, the scheduling from this step and so on becomes 'N/A', meaning that the agent does not know what decision to make. This points to the reliability of this framework in letting the user choose the amount of trust that should be given to the framework (based on the chosen by the user hyperparameter p_{thresh}) or when human feedback is preferred. As will be better shown in Fig. 17, only a couple of days at the end of the horizon are becoming 'N/A'. This occurs primarily because, after performing imperfect repairs, the RUL uncertainty increases significantly. Consequently, the agent is uncertain whether another repair close to the horizon's end will be beneficial.

To further show the behavior of each component in terms of RUL during the scheduling of the first episode, Fig. 15 depicts how the predicted RUL is affected based on the agent's maintenance actions. Since this refers to the first episode, i.e. when all components are in the brand-new condition, the corresponding shown RULs represent the first data point of each of the trajectories as illustrated in Fig. C.2. This figure additionally presents the probabilities of taking each maintenance action. All other decisions made by the agent that are not shown correspond to the 'hold' decision and do not affect RUL behavior. As expected, after scheduling an 'imperfect repair', the variance of the RUL drastically increases until a 'replace' decision is made. Then, RUL contains only the variance related to its stochastic behavior as predicted by the prognostic model. From this figure, it is clear our previous statement related to scheduling components for maintenance on the same day rather than waiting for more days. In other words, although the agent could take additional 'hold' decisions before performing a maintenance action since RUL allows it, the overall cost is lower if more components are maintained on the same day than maintained later but on different dates.

Since those predicted RULs refer to the brand-new condition of each component, according to assumption 3, the plots have rationally similar trends, even after an imperfect repair maintenance action. In Fig. C.4, an example of another RUL behavior is illustrated representing day 36, where the corresponding RUL data points have different values. For component 1, this day corresponds to the last row of the scheduling related to Fig. 17.

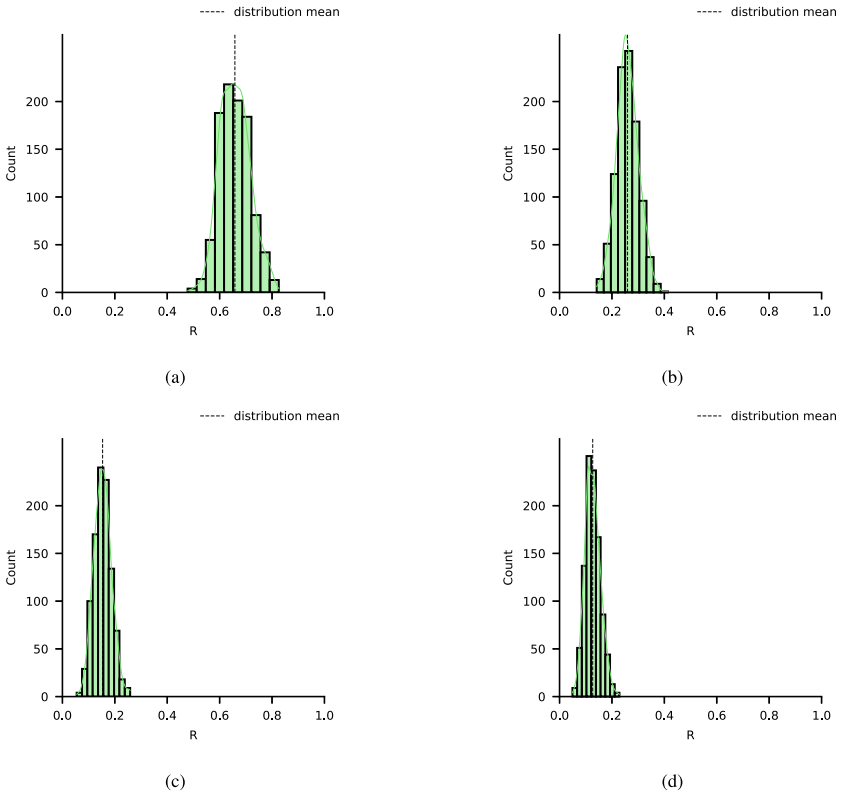


Fig. 11. Posterior predictive distributions of the mean of recovery for one (Fig. 11(a)), two (Fig. 11(b)), three (Fig. 11(c)), and four (Fig. 11(d)) sequential repairs.

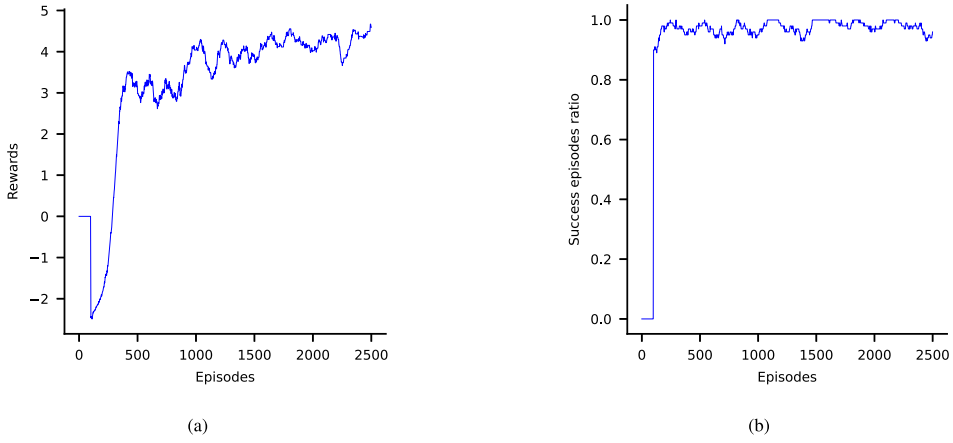


Fig. 12. Training performance of the RL agent with four components. Fig. 12(a) shows the rewards per episode and (Fig. 12(b)) the ratio of successful episodes as the training progresses.

4.4. The role of VoI in choosing the optimal time to acquire new information

Similarly to the RL agent, the VoI model was trained on a single GPU (NVIDIA GeForce RTX 2080). The entire training process for 400 epochs was approximately 40 min. However, generating data to train the VoI model required a huge amount of time up to 23 h. This prolonged duration was primarily due to the uncertainty quantification and decomposition processes involved in generating data related to VoI. Each step within an episode was executed in a loop of 30 samples to compute MI, which contributed significantly to the overall time required.

The performance of the VoI model (training and validation losses) is illustrated in Fig. 16(a) where both losses converge satisfactorily. Subsequently, after running the first episode related to the case study, the VoI model outputs the next 10 predicted uncertainties and the sequential number of future episodes that $VoI < 0$ is measured by Eq. (15). Following this process, the episodes where data are not required are depicted in Fig. 16(b). When the current episode in real-time is far from an 'imperfect repair' decision, the uncertainty remains mainly low, thus more episodes up to a maximum of 10 could be ignored. When such a maintenance action is approaching in real-time, the uncertainty increases, hence ignoring the next episodes should be avoided as shown in the last episodes of the same figure. Interestingly, for this case study consisting of a 60-day horizon, only 7 days were needed to acquire data, saving a vast amount of resources up to 88.3%.

Based on the VoI model and the episodes to ignore, it is important to visualize how the scheduling continues in real-time after having all the models trained. In Fig. 17, the first five schedules of the first component are shown considering VoI. In this regard, these schedules correspond to episodes 0, 11, 14, 25, and 36, as shown in Fig. 16(b). Importantly, it

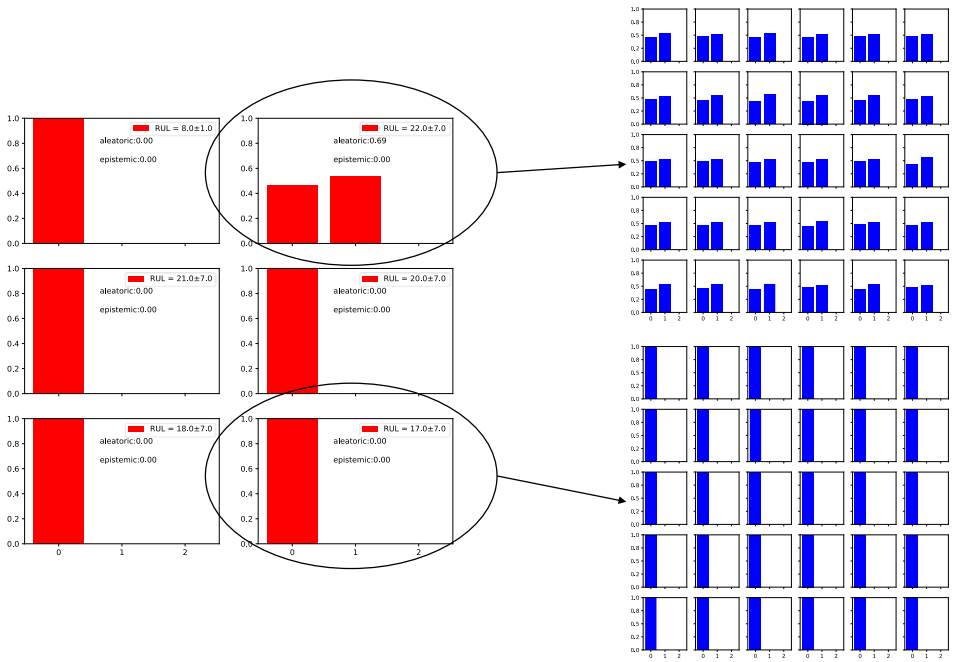


Fig. 13. Probabilities of taking each decision over a part of an episode and the corresponding uncertainty estimates are presented with red color in the graph. During inference, each of these steps is run T times. These samples and their corresponding probabilities are illustrated for two specific steps with blue color. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

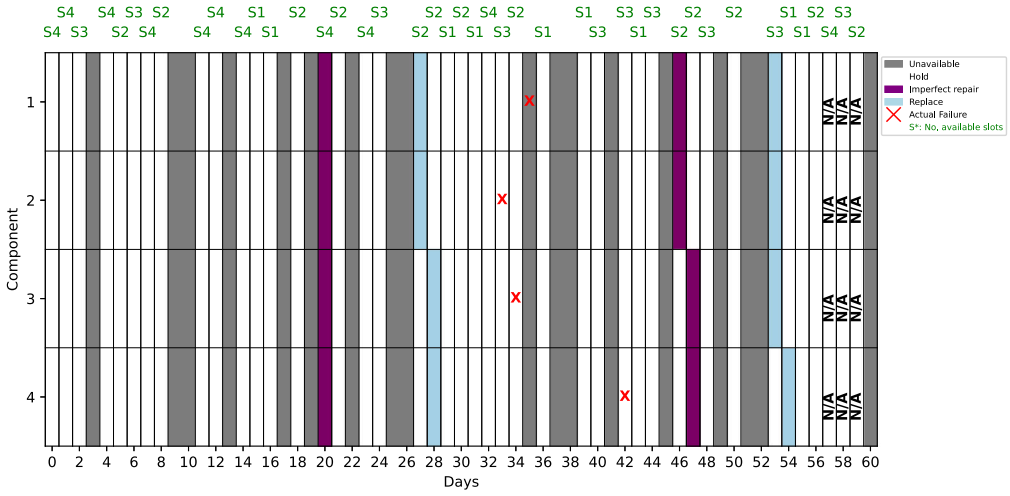


Fig. 14. Scheduling for the first episode. The decisions of the RL agent for the first episode are shown inside the horizon.

is observed that until the moment an 'imperfect repair' decision is made in real-time, the schedule stays almost the same. This shows the agent's confidence in its decisions, aligning with the low uncertainty and the multiple possible disregarded episodes. However, when an imperfect repair action is close while one or more of the components are close to the actual EOL condition, the uncertainty is increased and the schedule must be adjusted before proceeding with a new one. Thus, VoI suggests acquiring new information and updating the decisions. This is akin to how human experts update their schedules following a significant maintenance action.

4.5. Sensitivity analysis based on the number of specimens

To demonstrate the RL agent's performance limits, a sensitivity analysis was conducted for varying numbers of components. Similarly to Fig. 12, the rewards and ratio of successful episodes for each number of components are depicted in Figs. 18(a) and 18(b) respectively, after taking the mean values of a rolling horizon with a rolling window of 100. From these plots, it is obvious that the performance drops as the number of components increases. Based on the plot of the ratio of successful episodes, one could agree that the performance of the agent is questionable for a multi-component system with more than 70 components. Further improvements to the framework should be made to tackle systems with a larger number of dependable components.

5. Discussion and conclusion

The proposed PPDM framework integrates several sophisticated methodologies to address the challenges of maintenance scheduling in multi-component systems subject to imperfect repairs. This study introduces a novel approach that combines stochastic RUL estimations, imperfect repairs modeling, DRL, uncertainty quantification and decomposition, and VoI to decide the optimal time to acquire new information. The integration of stochastic RUL predictions and imperfect

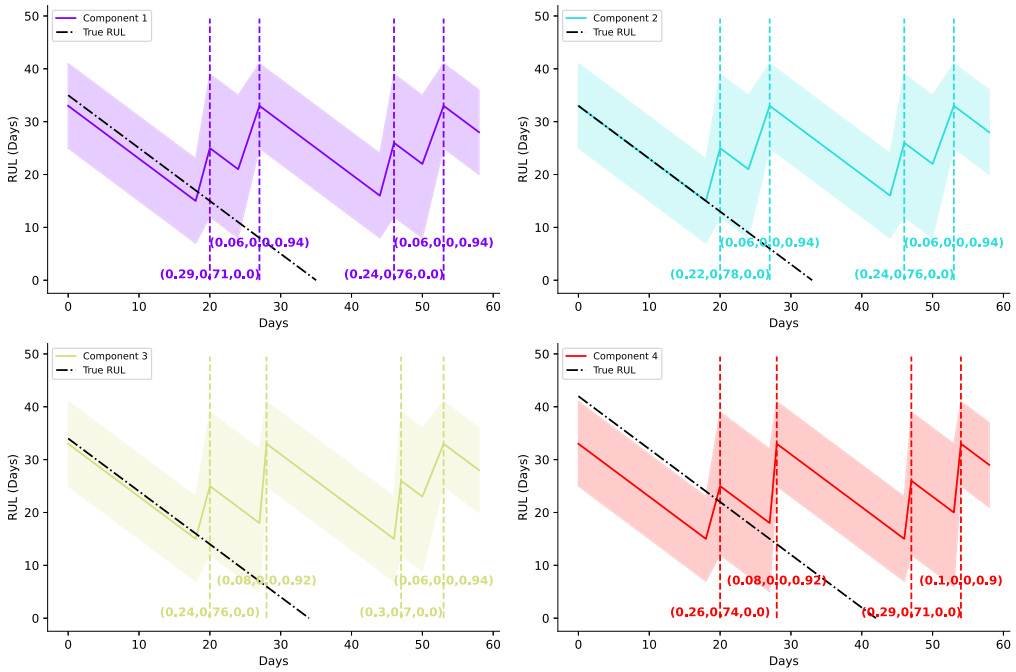


Fig. 15. RUL behavior of each component during the scheduling of the first episode within the horizon. For clarity, only the decisions related to maintenance actions. Additionally, the probabilities of making a decision by the agent are shown, corresponding to 'hold', 'imperfect repair', and 'replace' respectively.

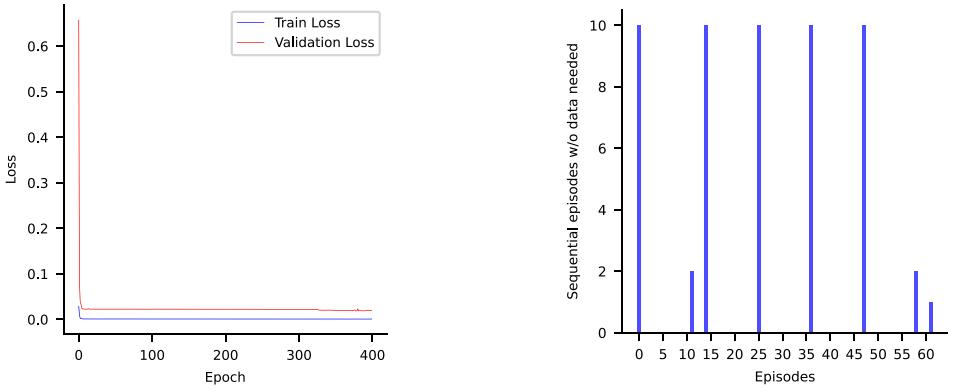


Fig. 16. Results related to VoI model. Fig. 12(a) shows VoI model's training and validation loss, and Fig. 12(b) the number of next sequential episodes to ignore after running the VoI model on the current episode.

repairs allows for a more realistic and dynamic assessment of components' health over time. Improving the health state of each component can extend the useful life of the entire structure consisting of multiple dependent components. The primary advantage of this framework is its ability to integrate all steps of the PHM strategy while keeping each of them independent. This design provides flexibility, allowing for the use of various feature extraction, prognostic, and imperfect repair modeling techniques to input data into the PPDM framework. Consequently, one could choose to utilize all, a part, or none of the implemented in this work steps necessary to feed the PPDM framework with information.

By incorporating the uncertainty associated with RULs and repairs, the framework can more accurately predict the future condition of the components, which is crucial for making optimal maintenance decisions. The utilization of DRL enables the agent to learn optimal maintenance strategies through interaction with the environment, even with large observation and action space. A key feature of the framework is its ability to quantify and decompose uncertainty into epistemic and aleatoric through techniques such as MC dropout and MI. This allows the framework to transform the total uncertainty into decision probabilities, improving reliability and interpretability, something that is lacking particularly in DRL and scenarios where data is incomplete or noisy. Given measures of uncertainty, the VoI model is considered to determine the optimal timing for acquiring new information, thereby reducing unnecessary data collection and focusing resources on the most impactful days within the horizon.

The experimental evaluation demonstrated the framework's effectiveness in a real-world case study involving tension-tension fatigue experiments on open-hole aluminum coupons. The results showed that the framework could successfully schedule maintenance actions to extend the structure's lifecycle while satisfying operational and logistic constraints. The potential to perform scheduling given a horizon even under multiple repairs highlights the superiority of the proposed approach. Moreover, estimating the decision probabilities of the agent provides a framework that identifies its limits. Knowing when it is confident in its decisions and when it is uncertain facilitates a hybrid approach that combines the power of AI with human feedback, enhancing reliability and promoting feasibility and risk-averse policies. Finally, a sensitivity analysis has been conducted for varying numbers

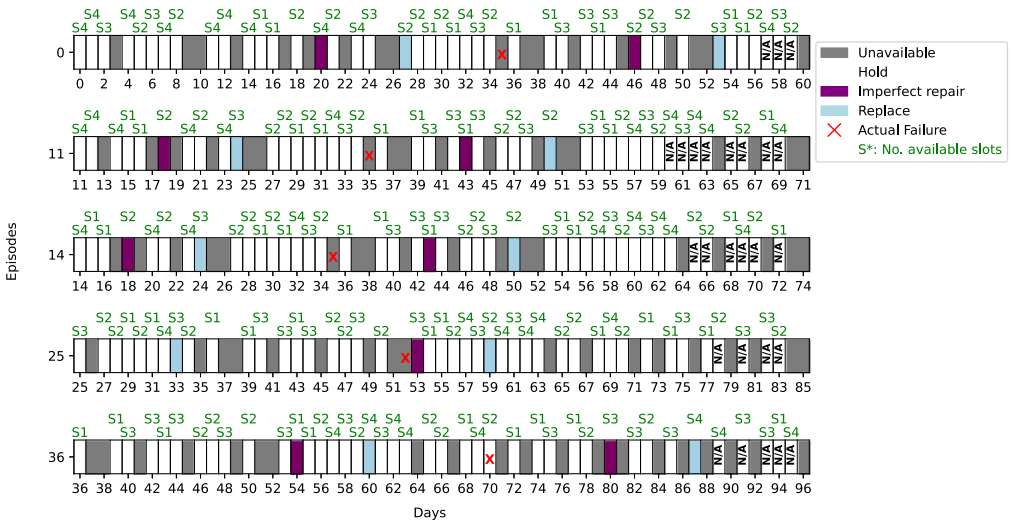
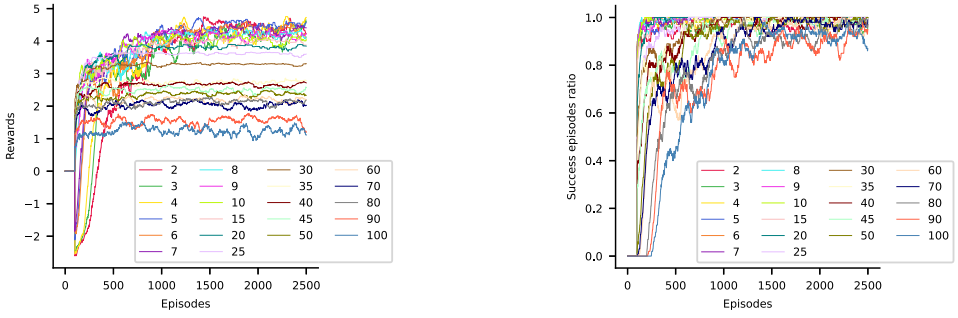


Fig. 17. Number of next sequential episodes to ignore after running the VoI model on the current episode.



(a) (b)

Fig. 18. Training performance of the RL agent with different number of components ranging from 2 to 100. Fig. 18(a)) shows the rewards per episode and (Fig. 18(b)) the ratio of successful episodes as the training progresses.

of components, demonstrating the performance limits of the designed framework.

The 'N/A' output in the proposed framework serves as an explicit indicator of high uncertainty, derived from the combination of epistemic and aleatoric components. While it is primarily designed to support risk-averse decision-making by deferring low-confidence actions, its practical role in mission-critical systems—where continuous autonomous operation is often required—warrants discussion. Crucially, the interpretation of 'N/A' depends on the dominant source of uncertainty. If the decision is driven by epistemic uncertainty, it reflects a lack of model knowledge, often due to under-represented scenarios in the training data. In such cases, retraining the agent with additional or more diverse data can reduce the frequency of uncertain outputs over time. Conversely, if the uncertainty is aleatoric, it signals that the input data are inherently ambiguous, noisy, or insufficient to support a reliable decision, regardless of model quality. In this case, the 'N/A' output functions as a critical safeguard, preventing potentially misleading or overconfident actions in uncertain conditions. As such, even in continuous autonomous operations, frequent 'N/A' outputs, when they come from aleatoric uncertainty, do not undermine autonomy, but rather enhance system safety by transparently acknowledging the limits of the available information.

Despite these key contributions, important limitations exist that are worth discussing to enhance the practical applicability of the framework. Firstly, due to the large number of hyperparameters, it is impossible with the current hardware to perform a parametric study (except for the necessary and already implemented sensitivity analysis) or to apply an automatic hyperparameter tuning, such as Bayesian Optimization. This limits the framework in a time-consuming trial-and-error approach. Secondly, the sensitivity analysis revealed that the performance of the DRL agent diminishes with an increasing number of components. This suggests that while the current model performs well for a moderate number of components, enhancements are needed to maintain efficacy in more complex scenarios. Future work could explore Hierarchical RL architectures, where decisions are made at multiple abstraction levels. For example, a high-level policy could first select subsets of components to consider, and low-level policies could then determine specific maintenance actions within each group. Distributed multi-agent frameworks, where individual agents are responsible for subsets of components and coordinate through a shared global reward or communication protocol, are another promising future direction. These frameworks may scale more efficiently and allow for parallelization of learning. Thirdly, while VoI offers runtime efficiency, generating data for training the VoI model is computationally expensive due to uncertainty quantification and decomposition. Potential solutions to this challenge would focus on parallelizing the process (as done with RL environment workers), particularly on multi-GPU or distributed systems. This would significantly reduce wall-clock time. Finally, rather than generating VoI data from all episodes, a representative subset of episodes with diverse uncertainty patterns could be selected using clustering or active learning methods, reducing redundant computations.

Table B.1
The low-level features that are considered and extracted by the AMSY-6 Vallen Systeme GmbH as presented in [45].

Feature name	Unit	Description
Threshold	Decibel [dB]	Values below this threshold are discarded.
Amplitude	Volts [V]	The amplitude of the corresponding signal.
Duration	Seconds [s]	The duration that a signal constantly remains above the threshold.
Energy	$10^{-14}V^2s$ [eu].	Energy is the integral of the squared acoustic emission-signal over time
Counts	–	The number of positive threshold crossings of a hit.
Hit time	Seconds [s]	The absolute time when a hit is above the threshold.
Rise time	Seconds [s]	The time between the first threshold crossing and the maximum amplitude.
RMS	–	Root mean square (RMS) error.
Signal strength	10^{-9}Vs [nVs]	The integral of the rectified AE signal over time.

Table B.2
Technical details. The specimens that were under repair contain two trajectories corresponding to one before and one after the repair.

Specimen No.	Specimen name(s)	Repair 1 time (T1)	Crack size [mm] @T1	Fatigue life (cycles)	Fatigue life (s)
Baseline					
01	0_1	–	1.5	26 478	7349
02	0_2	–	3	22 563	6850
03	0_3	–	2.5	23 342	7079
04	0_4	–	2.5	23 750	8737
05	0_5	–	6	19 250	4814
Average	–	–	3.1	23 076	6966
1 Repair, reached EOL					
06	0_6, 1_6	14 000	1	24 565	6470
07	0_7, 1_7	14 000	5	17 445	4551
08	0_8, 1_8	14 000	6	17 250	4547
Average	–	14 000	4	19 753	5189
1 Repair, did not reach EOL					
09	0_9, 1_9	14 000	1	–	–
10	0_10, 1_10	14 000	4	–	–
11	0_11, 1_11	14 000	0	–	–
Average	–	140 00	1.6	–	–

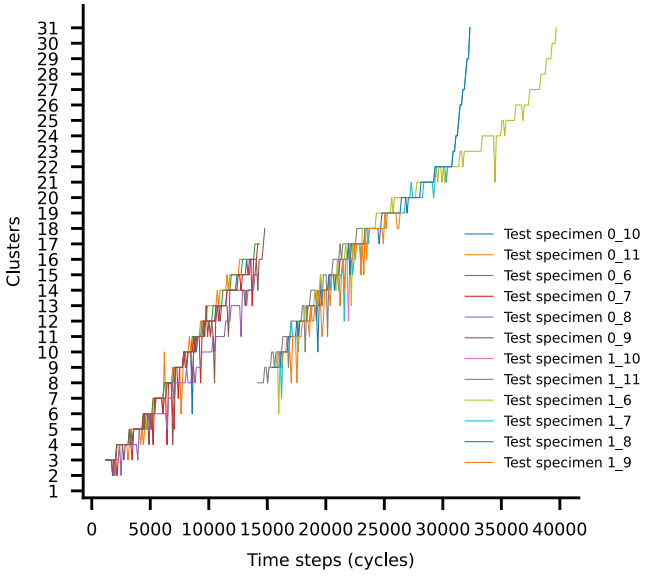


Fig. C.1. The predicted by the DSMC model clustering assignments concerning the test specimens.

Overall, this study lays a strong foundation for advancing the field of PPDM. By addressing the identified limitations and building on the proposed framework, future research can further enhance its practical applicability. This framework could be a pillar for researchers to improve and promote PPDM into more advanced tasks that many industrial processes demand. Real-world validation of the model in various industrial settings will be crucial for establishing its effectiveness and ensuring its adoption in CBM practices across different sectors. Having a framework that integrates the steps of the PHM strategy independently could offer endless opportunities for constructing unique frameworks, applicable to wide or specific domains.

Declaration of Generative AI in scientific writing

During the preparation of this work, the authors used ChatGPT based on GPT3.5 in order to improve the readability and language of some parts of the paper. The tool was in no way used to analyze and draw insights from the data, perform literature research, or extract any information other than feedback on the writing style based on the provided inputs. The tool was only used to perform minimal changes and provide feedback based on the provided input text, where the scientific content of the input sentences remains unchanged. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the content of the publication.

Fundings

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

CRediT authorship contribution statement

P. Komninos: Writing – review & editing, Writing – original draft, Software, Methodology, Formal analysis, Data curation, Conceptualization. **D. Zarouchas:** Writing – review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

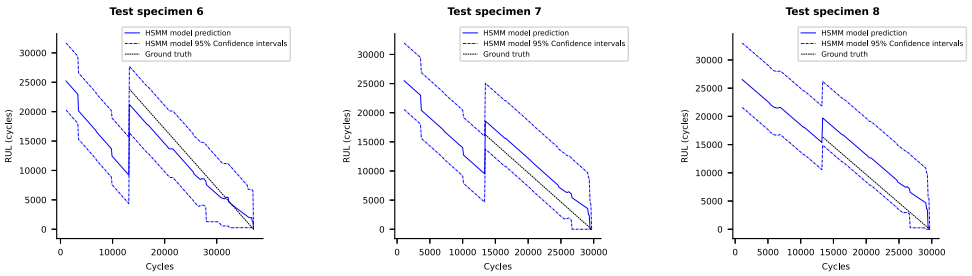


Fig. C.2. Stochastic RUL predictions of the three testing specimens. These specimens correspond to the ones with repair that reached the EOL. The true RUL corresponds to the trajectory part that comes after the repair.

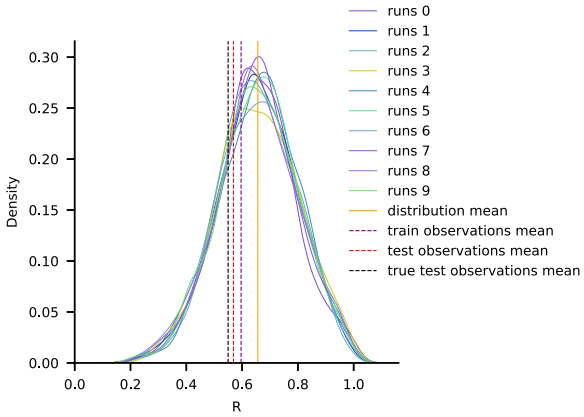


Fig. C.3. Posterior predictive distribution of the mean of recovery.

Appendix A. Extended theoretical background

A.1. Proximal policy optimization

Consider a stochastic policy $\pi_\theta : S \times A \to [0, 1]$, parameterized by a parameter vector θ, which assigns probabilities to each of the available actions given a state. For a finite horizon, the goal is to maximize the expected discounted return of the policy:

$$J = \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} \gamma^t r_t \left(s_t, s_{t+1} \right) \right] \tag{A.1}$$

where $\tau = \left(s_0, \alpha_0, r_0, \ldots, s_{T-1}, \alpha_{T-1}, r_{T-1} \right)$ is a trajectory related to the environment's variables. Policy gradient algorithms aim to derive the gradient of the expected discounted return $\bigtriangledown_\theta J$ concerning the policy parameter θ as their fundamental concept. According to Sutton & Barto [66] and the introduced advantage function $A(s_t, \alpha_t)$ [67], the proposed policy gradient estimate to the objective J is as follows:

$$\nabla_\theta J = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(\alpha_t | s_t) A(s_t, \alpha_t) \right] \tag{A.2}$$

where:

$$A(s_t, \alpha_t) = Q(s_t, \alpha_t) - V(s_t, \alpha_t) \tag{A.3}$$

This is the advantage function, representing the advantage of taking action α_t in state s_t compared to the average action value. Then the parameters θ are updated:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J \left(\pi_{\theta_k} \right) \tag{A.4}$$

In PPO, the advantage function is often estimated using the generalized advantage estimation (GAE) technique [68], which combines rewards and value estimates:

$$A^{GAE}(s_t, \alpha_t) = \sum_{\tau=0}^{\infty} (\gamma \lambda_{GAE})^\tau \left(r_{t+\tau} + \gamma V(s_{t+\tau+1}) - V(s_{t+\tau}) \right) \tag{A.5}$$

PPO parameterizes the policy using an ANN with parameters θ. This network outputs probabilities for selecting each action given a state, denoted as $\pi_\theta(\alpha|s)$. Moreover, PPO introduces a surrogate objective function that constrains the policy update to be close to the current policy. This helps prevent large policy updates that can lead to instability. The surrogate objective is defined as:

$$L(\theta, s_t, \alpha_t) = \frac{\pi_\theta(\alpha_t \mid s_t)}{\pi_{\text{old}}(\alpha_t \mid s_t)} A^{GAE}(s_t, \alpha_t) \tag{A.6}$$

where π_{old} is the probability of selecting action α_t given state s_t under the old policy at step t. To further stabilize training and prevent large policy updates, PPO introduces a clipped surrogate objective. Instead of directly maximizing the surrogate objective, PPO maximizes a clipped version of the surrogate objective, which is bounded by a specified clipping parameter ϵ :

$$L_{clip}(\theta, s_t, \alpha_t) = \left(\frac{\pi_\theta(\alpha_t \mid s_t)}{\pi_{\text{old}}(\alpha_t \mid s_t)} A^{GAE}(s_t, \alpha_t), \right.$$
$$\left. \text{clip} \left(\frac{\pi_\theta(\alpha_t \mid s_t)}{\pi_{\text{old}}(\alpha_t \mid s_t)}, 1 - \epsilon, 1 + \epsilon \right) A^{GAE}(s_t, \alpha_t) \right) \tag{A.7}$$

PPO updates the policy parameters θ by maximizing the clipped surrogate objective typically using stochastic gradient ascent with Adam optimizer. This involves computing the gradient of the clipped surrogate objective with respect to θ and updating θ in the direction that increases the objective as follows:

$$\theta_{k+1} = \arg\max_\theta \mathbb{E}_{s_t, \alpha_t \sim \pi_{\text{old}}} \left[L_{clip}(\theta, s_t, \alpha_t) \right] \tag{A.8}$$

To further stabilize training, PPO typically performs multiple optimization epochs on the collected data before updating the policy. This helps to reduce the variance in gradient estimates and improve sample efficiency. Finally, PPO can also update the parameters of the value function network to improve value estimation by regression on MSE given the equation below:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|D_k| T} \sum_{\tau \in D_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - R_t \right)^2 \tag{A.9}$$

where D_k is a dataset containing older steps stored in the buffer, $V_\phi(s_t)$ is the value function approximated by an ANN with parameters ϕ. The reward R_t at step t comes from these collections stored in the buffer.

PPO is designed to provide stable and reliable training by constraining policy updates. It typically requires fewer samples to achieve good performance compared to some other policy gradient methods and performs well across a wide range of tasks and environments, making it a popular choice in the RL community.

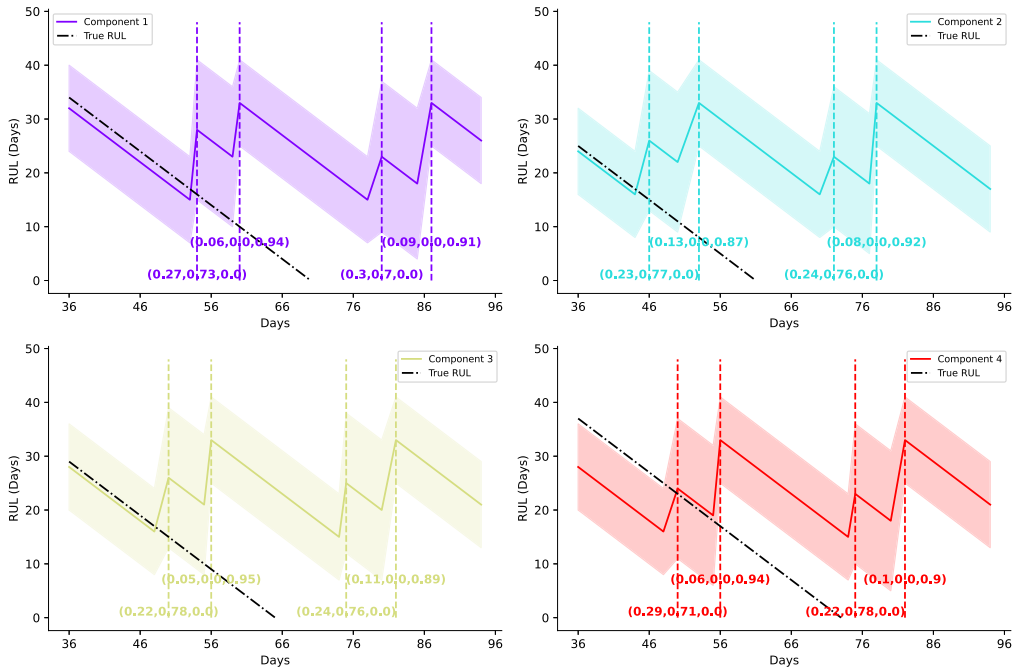


Fig. C.4. RUL behavior of each component during the scheduling of day 36 within the horizon. For clarity, only the decisions related to maintenance actions. Additionally, the probabilities of making a decision by the agent are shown, corresponding to 'hold', 'imperfect repair', and 'replace' respectively.

A.2. Proofs

Lemma 1. *Given an agent that chooses over three possible decisions, the measured total uncertainty produced by Shannon's entropy and MC dropout is upper bounded to* $log3$*, s.t.:*

$$U = \frac{1}{T} H\left[\sum_{i=1}^{T} p(y_i)\right] \leq log3 \tag{A.10}$$

where U *is the total uncertainty.*

Proof. According to Shannon's entropy and Jensen's inequality [69], for k decisions we have:

$$-\sum_{y=1}^{k}\left[p_y log(p_y)\right] = \sum_{y=1}^{k}\left[log(\frac{1}{p_y})\right] \leq logk \tag{A.11}$$

The given total uncertainty from Eq. (14) for 3 decisions is:

$$U = \frac{1}{T} H\left[\sum_{i=1}^{T} p(y_i)\right] = -\sum_{y=1}^{3}\left[\frac{1}{T}\sum_{i=1}^{T} p(y_i) log\left(\frac{1}{T}\sum_{i=1}^{T} p(y_i)\right)\right] \tag{A.12}$$

Setting $P_y = \frac{1}{T}\sum_{i=1}^{T} p(y_i)$ then Eq. (A.12) becomes:

$$U = \frac{1}{T} H\left[\sum_{i=1}^{T} p(y_i)\right] = -\sum_{y=1}^{3}\left[P_y log(P_y)\right] \leq log3 \tag{A.13}$$

Lemma 2. *Given an agent that chooses over three possible decisions, the probability threshold* p_{thresh} *based on which a decision is considered acceptable has a connection with the total uncertainty, s.t.:*

$$U \leq -p_{thresh} log(p_{thresh}) - (1 - p_{thresh}) log\left(\frac{1 - p_{thresh}}{2}\right) \tag{A.14}$$

Proof. Since only one decision should be made for each component, we care only about the maximum probability $p(y_i^{max}) = max(p(y_i)), i = \{1,2,3\} \geq p_{thresh}$. The worst case scenario is $p(y_i^{max}) = p_{thresh}$. Let us assume that the first decision is the maximum, thus $p(y_1) = p_{thresh}$. Then, based on Eq. (A.13), the maximum corresponding acceptable uncertainty is:

$$\begin{aligned} U &= -\sum_{y=1}^{3}\left[\frac{1}{T}\sum_{i=1}^{T} p(y_i) log\left(\frac{1}{T}\sum_{i=1}^{T} p(y_i)\right)\right] \\ &= -\frac{1}{T}\sum_{i=1}^{T} p(y^i_{thresh}) log\left(\frac{1}{T}\sum_{i=1}^{T} p(y^i_{thresh})\right) \\ &\quad -\sum_{y=2}^{3}\left[\frac{1}{T}\sum_{i=1}^{T} p(y_i) log\left(\frac{1}{T}\sum_{i=1}^{T} p(y_i)\right)\right] \\ &= -\frac{1}{T} T p_{thresh} log\left(\frac{1}{T} T\right) p_{thresh} - \sum_{y=2}^{3}\left[\frac{1}{T}\sum_{i=1}^{T} p(y_i) log\left(\frac{1}{T}\sum_{i=1}^{T} p(y_i)\right)\right] \\ &= -p_{thresh} log(p_{thresh}) - \sum_{y=2}^{3}\left[\frac{1}{T}\sum_{i=1}^{T} p(y_i) log\left(\frac{1}{T}\sum_{i=1}^{T} p(y_i)\right)\right] \end{aligned} \tag{A.15}$$

Setting again $P_y = \frac{1}{T}\sum_{i=1}^{T} p(y_i)$ we have:

$$U = -p_{thresh} log(p_{thresh}) - \sum_{y=2}^{3} P_y log(P_y) \tag{A.16}$$

If the first decision corresponds to p_{thresh} then $P_1 = p_{thresh}$ and $\sum_{y=2}^{3} P_y = 1 - p_{thresh}$. According to Jensen's inequality the entropy of the terms P_2 and P_3 is maximized when $P_2 = P_3$, hence:

$$P_2 + P_3 = 1 - p_{thresh} \Leftrightarrow P_2 = P_3 = \frac{1 - p_{thresh}}{2} \tag{A.17}$$

Setting Eq. (A.17) to the second term ofEq. (A.16) we have:

$$\begin{aligned} U &= -p_{thresh} log(p_{thresh}) - -\sum_{y=2}^{3} \frac{1 - p_{thresh}}{2} log(\frac{1 - p_{thresh}}{2}) \\ &= -p_{thresh} log(p_{thresh}) - (1 - p_{thresh}) log\left(\frac{1 - p_{thresh}}{2}\right) \end{aligned} \tag{A.18}$$

The same equation will be given assuming any of the probabilities P_1, P_2, P_3 equals to p_{thresh}. If any of the maximum P_i is less than p_{thresh}, then the total uncertainty in Eq. (A.18) is always smaller than the above expression.

Appendix B. Additional details of the experimental campaign

See Tables B.1 and B.2.

Appendix C. Supplementary results

In this section, we present additional results already implemented in previous works concerning feature extraction (clustering assignments) (see Fig. C.1), prognosis (see Fig. C.2), and distribution of recovery for a single repair (see Fig. C.3).

Data availability

Data will be made available on request.

References

[1] Moradi M, Broer A, Chiachío J, Benedictus R, Loutas TH, Zarouchas D. Intelligent health indicator construction for prognostics of composite structures utilizing a semi-supervised deep neural network and SHM data. Eng Appl Artif Intell 2023;117:105502. http://dx.doi.org/10.1016/j.engappai.2022.105502, URL https://www.sciencedirect.com/science/article/pii/S0952197622004924.

[2] Bousdekis A, Magoutas B, Apostolou D, Mentzas G. Review, analysis and synthesis of prognostic-based decision support methods for condition based maintenance. J Intell Manuf 2018;29. http://dx.doi.org/10.1007/s10845-015-1179-5.

[3] Goebel K, Celaya J, Sankararaman S, Roychoudhury I, Daigle M, Saxena A. Prognostics: The Science of Making Predictions. CreateSpace Independent Publishing Platform (April 3, 2017): 1539074838 ISBN-13: 978-1539074830; 2017, p. 396.

[4] Lepenioti K, Pertselakis M, Bousdekis A, Louca A, Lampathaki F, Apostolou D, Mentzas G, Anastasiou S. Machine learning for predictive and prescriptive analytics of operational data in smart manufacturing. In: Advanced information systems engineering workshops: cAiSE 2020 international workshops, grenoble, France, June 8–12, 2020, proceedings 32. Springer; 2020, p. 5–16.

[5] Liu Q, Dong M, Chen F. Single-machine-based joint optimization of predictive maintenance planning and production scheduling. Robot Comput-Integr Manuf 2018;51:238–47.

[6] Ong KSH, Niyato D, Yuen C. Predictive maintenance for edge-based sensor networks: A deep reinforcement learning approach. In: 2020 IEEE 6th world forum on internet of things (WF-ioT). IEEE; 2020, p. 1–6.

[7] Skima H, Varnier C, Dedu E, Medjaher K, Bourgeois J. Post-prognostics decision making in distributed MEMS-based systems. J Intell Manuf 2019;30:1125–36.

[8] de Pater I, Mitici M. Predictive maintenance for multi-component systems of repairables with remaining-useful-life prognostics and a limited stock of spare components. Reliab Eng Syst Saf 2021;214:107761. http://dx.doi.org/10.1016/j.ress.2021.107761, URL https://www.sciencedirect.com/science/article/pii/S095183202100288X.

[9] Bousdekis A, Mentzas G. A proactive model for joint maintenance and logistics optimization in the frame of industrial internet of things. In: Sifaleras A, Petridis K, editors. Operational research in the digital era – ICT challenges. Cham: Springer International Publishing; 2019, p. 23–45.

[10] Hu Y, Miao X, Zhang J, Liu J, Pan E. Reinforcement learning-driven maintenance strategy: A novel solution for long-term aircraft maintenance decision optimization. Comput Ind Eng 2021;153:107056.

[11] Andriotis C, Papakonstantinou K. Managing engineering systems with large state and action spaces through deep reinforcement learning. Reliab Eng Syst Saf 2019;191:106483. http://dx.doi.org/10.1016/j.ress.2019.04.036, URL https://www.sciencedirect.com/science/article/pii/S0951832018313309.

[12] Nguyen KT, Do P, Huynh KT, Bérenguer C, Grall A. Joint optimization of monitoring quality and replacement decisions in condition-based maintenance. Reliab Eng Syst Saf 2019;189:177–95. http://dx.doi.org/10.1016/j.ress.2019.04.034, URL https://www.sciencedirect.com/science/article/pii/S0951832018313097.

[13] Do P, Voisin A, Levrat E, Iung B. A proactive condition-based maintenance strategy with both perfect and imperfect maintenance actions. Reliab Eng Syst Saf 2015;133:22–32. http://dx.doi.org/10.1016/j.ress.2014.08.011, URL https://www.sciencedirect.com/science/article/pii/S095183201400204X.

[14] Meissner R, Rahn A, Wicke K. Developing prescriptive maintenance strategies in the aviation industry based on a discrete-event simulation framework for post-prognostics decision making. Reliab Eng Syst Saf 2021;214:107812. http://dx.doi.org/10.1016/j.ress.2021.107812, URL https://www.sciencedirect.com/science/article/pii/S0951832021003331.

[15] Deng Q, Santos BF. Lookahead approximate dynamic programming for stochastic aircraft maintenance check scheduling optimization. European J Oper Res 2022;299(3):814–33. http://dx.doi.org/10.1016/j.ejor.2021.09.019, URL https://www.sciencedirect.com/science/article/pii/S0377221721007943.

[16] Kim S-J, Lee C-W. Diagnosis of sensor faults in active magnetic bearing system equipped with built-in force transducers. IEEE/ASME Trans Mechatronics 1999;4(2):180–6. http://dx.doi.org/10.1109/3516.769544.

[17] Lyu D, Si S. Importance measure for K-out-of-n: G systems under dynamic random load considering strength degradation. Reliab Eng Syst Saf 2021;216:107892. http://dx.doi.org/10.1016/j.ress.2021.107892, URL https://www.sciencedirect.com/science/article/pii/S0951832021004105.

[18] Wang J, Zhu X. Joint optimization of condition-based maintenance and inventory control for a k-out-of-n:F system of multi-state degrading components. European J Oper Res 2020;290. http://dx.doi.org/10.1016/j.ejor.2020.08.016.

[19] Olde Keizer MC, Teunter RH, Veldman J. Clustering condition-based maintenance for systems with redundancy and economic dependencies. European J Oper Res 2016;251(2):531–40. http://dx.doi.org/10.1016/j.ejor.2015.11.008, URL https://www.sciencedirect.com/science/article/pii/S0377221715010218.

[20] Zhang P, Zhu X, Xie M. A model-based reinforcement learning approach for maintenance optimization of degrading systems in a large state space. Comput Ind Eng 2021;161:107622. http://dx.doi.org/10.1016/j.cie.2021.107622, URL https://www.sciencedirect.com/science/article/pii/S036083522100526X.

[21] Rocchetta R, Bellani L, Compare M, Zio E, Patelli E. A reinforcement learning framework for optimal operation and maintenance of power grids. Appl Energy 2019;241:291–301.

[22] Wang H, Gu M, Yu Q, Tao Y, Li J, Fei H, Yan J, Zhao W, Hong T. Adaptive and large-scale service composition based on deep reinforcement learning. Knowl-Based Syst 2019;180:75–90. http://dx.doi.org/10.1016/j.knosys.2019.05.020, URL https://www.sciencedirect.com/science/article/pii/S0950705119302266.

[23] Liu Y, Chen Y, Jiang T. Dynamic selective maintenance optimization for multi-state systems over a finite horizon: A deep reinforcement learning approach. European J Oper Res 2020;283(1):166–81. http://dx.doi.org/10.1016/j.ejor.2019.10.049, URL https://www.sciencedirect.com/science/article/pii/S0377221719309014.

[24] Skordilis E, Moghaddass R. A deep reinforcement learning approach for real-time sensor-driven decision making and predictive analytics. Comput Ind Eng 2020;147:106600. http://dx.doi.org/10.1016/j.cie.2020.106600, URL https://www.sciencedirect.com/science/article/pii/S036083522030334X.

[25] Andrade P, Silva C, Ribeiro B, Santos BF. Aircraft maintenance check scheduling using reinforcement learning. Aerospace 2021;8(4). http://dx.doi.org/10.3390/aerospace8040113, URL https://www.mdpi.com/2226-4310/8/4/113.

[26] Zhou Y, Li B, Lin TR. Maintenance optimisation of multicomponent systems using hierarchical coordinated reinforcement learning. Reliab Eng Syst Saf 2022;217:108078. http://dx.doi.org/10.1016/j.ress.2021.108078, URL https://www.sciencedirect.com/science/article/pii/S0951832021005767.

[27] Li X, Fan D, Liu X, Xu S, Huang B. State of health estimation for lithium-ion batteries based on improved bat algorithm optimization kernel extreme learning machine. J Energy Storage 2024;101:113756. http://dx.doi.org/10.1016/j.est.2024.113756, URL https://www.sciencedirect.com/science/article/pii/S2352152X24033425.

[28] Qiang Y, Wang X, Liu X, Wang Y, Zhang W. Edge-enhanced graph attention network for driving decision-making of autonomous vehicles via deep reinforcement learning. Proc Inst Mech Eng Part D: J Automob Eng 2025;239(4):1168–80. http://dx.doi.org/10.1177/09544070231217762, arXiv:https://doi.org/10.1177/09544070231217762.

[29] Jin K, Liu X, Liu M, and KQ. Modified quality loss for the analysis of product quality characteristics considering maintenance cost. Qual Technol Quant Manag 2022;19(3):341–61. http://dx.doi.org/10.1080/16843703.2022.2044120, arXiv:https://doi.org/10.1080/16843703.2022.2044120.

[30] Cheng W, Zhao X. Maintenance optimization for dependent two-component degrading systems subject to imperfect repair. Reliab Eng Syst Saf 2023;240:109581. http://dx.doi.org/10.1016/j.ress.2023.109581, URL https://www.sciencedirect.com/science/article/pii/S0951832023004957.

[31] Mosayebi Omshi E, Grall A. Replacement and imperfect repair of deteriorating system: Study of a CBM policy and impact of repair efficiency. Reliab Eng Syst Saf 2021;215:107905. http://dx.doi.org/10.1016/j.ress.2021.107905, URL https://www.sciencedirect.com/science/article/pii/S095183202100421X.

[32] Zhang F, Shen J, Ma Y. Optimal maintenance policy considering imperfect repairs and non-constant probabilities of inspection errors. Reliab Eng Syst Saf 2020;193:106615. http://dx.doi.org/10.1016/j.ress.2019.106615, URL https://www.sciencedirect.com/science/article/pii/S0951832019300274.

[33] Zhao X, He S, Xie M. Utilizing experimental degradation data for warranty cost optimization under imperfect repair. Reliab Eng Syst Saf 2018;177:108–19. http://dx.doi.org/10.1016/j.ress.2018.05.002, URL https://www.sciencedirect.com/science/article/pii/S0951832017312206.

[34] Pedersen TI, Liu X, Vatn J. Maintenance optimization of a system subject to two-stage degradation, hard failure, and imperfect repair. Reliab Eng Syst Saf 2023;237:109313. http://dx.doi.org/10.1016/j.ress.2023.109313, URL https://www.sciencedirect.com/science/article/pii/S0951832023002272.

[35] Howard RA. Information value theory. IEEE Trans Syst Sci Cybern 1966;2(1):22–6. http://dx.doi.org/10.1109/TSSC.1966.300074.

[36] Song C, Zhang C, Shafieezadeh A, Xiao R. Value of information analysis in non-stationary stochastic decision environments: A reliability-assisted POMDP approach. Reliab Eng Syst Saf 2022;217:108034. http://dx.doi.org/10.1016/j.ress.2021.108034, URL https://www.sciencedirect.com/science/article/pii/S095183202100541X.

[37] Zou G, González A, Banisoleiman K. A widely-applicable structural maintenance decision-analytic modelling approach assisted by information value computation. Ocean Eng 2021;237:109596. http://dx.doi.org/10.1016/j.oceaneng.2021.109596, URL https://www.sciencedirect.com/science/article/pii/S002980182100980X.

[38] Straub D. Value of information analysis with structural reliability methods. Struct Saf 2014;49:75–85.

[39] Zhang W-H, Qin J, Lu D-G, Thöns S, Faber MH. Voi-informed decision-making for SHM system arrangement. Struct Heal Monit 2022;21(1):37–58.

[40] Kamariotis A, Chatzi E, Straub D. Value of information from vibration-based structural health monitoring extracted via Bayesian model updating. Mech Syst Signal Process 2022;166:108465. http://dx.doi.org/10.1016/j.ymssp.2021.108465, URL https://www.sciencedirect.com/science/article/pii/S0888327021008104.

[41] HU C, PEI H, WANG Z, SI X, ZHANG Z. A new remaining useful life estimation method for equipment subjected to intervention of imperfect maintenance activities. Chin J Aeronaut 2018;31(3):514–28. http://dx.doi.org/10.1016/j.cja.2018.01.009, URL https://www.sciencedirect.com/science/article/pii/S1000936118300256.

[42] Wang Z-Q, Hu C-H, Si X-S, Zio E. Remaining useful life prediction of degrading systems subjected to imperfect maintenance: Application to draught fans. Mech Syst Signal Process 2018;100:802–13. http://dx.doi.org/10.1016/j.ymssp.2017.08.016, URL https://www.sciencedirect.com/science/article/pii/S0888327017304429.

[43] Komninos P, Kontogiannis T, Eleftheroglou N, Zarouchas D. A robust generalized deep monotonic feature extraction model for label-free prediction of degenerative phenomena. 2024, [Accessed 10 October 2024] https://pure.tudelft.nl/admin/files/222348032/DSMC_PK_TK_NE_DZ.pdf.

[44] Dong M, He D, Banerjee P, Keller J. Equipment health diagnosis and prognosis using hidden semi-Markov models. Int J Adv Manuf Technol 2006;30(7):738–49. http://dx.doi.org/10.1007/s00170-005-0111-0.

[45] Komninos P, Galanopoulos G, Kontogiannis T, Eleftheroglou N, Zarouchas D. A Bayesian inference-based framework for modeling imperfect post-repair behavior of remaining useful life under uncertainty. Expert Syst Appl 2025;288:127723. http://dx.doi.org/10.1016/j.eswa.2025.127723, URL https://www.sciencedirect.com/science/article/pii/S0957417425013454.

[46] Van PD, Bérenguer C. Condition-based maintenance with imperfect preventive repairs for a deteriorating production system. Qual Reliab Eng Int 2012;28(6):624–33. http://dx.doi.org/10.1002/qre.1431, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/qre.1431 URL https://onlinelibrary.wiley.com/doi/abs/10.1002/qre.1431.

[47] van Ravenzwaaij D, Cassey P, Brown SD. A simple introduction to Markov chain Monte–Carlo sampling. Psychon Bull Rev 2018;25(1):143–54. http://dx.doi.org/10.3758/s13423-016-1015-8.

[48] Hoffman MD, Gelman A, et al. The no-u-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo.. J Mach Learn Res 2014;15(1):1593–623.

[49] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. Continuous control with deep reinforcement learning. 2015, arXiv preprint arXiv:1509.02971.

[50] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. 2017, arXiv:1707.06347.

[51] Zadrozny B, Elkan C. Transforming classifier scores into accurate multiclass probability estimates. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining. KDD '02, New York, NY, USA: Association for Computing Machinery; 2002, p. 694–9. http://dx.doi.org/10.1145/775047.775151.

[52] Kull M, Filho TMS, Flach P. Beyond sigmoids: How to obtain well-calibrated probabilities from binary classifiers with beta calibration. Electron J Stat 2017;11(2):5052–80. http://dx.doi.org/10.1214/17-EJS1338SI.

[53] Ahmed Z, Le Roux N, Norouzi M, Schuurmans D. Understanding the impact of entropy on policy optimization. In: International conference on machine learning. PMLR; 2019, p. 151–60.

[54] Vinyals O, Ewalds T, Bartunov S, Georgiev P, Vezhnevets AS, Yeo M, Makhzani A, Küttler H, Agapiou J, Schrittwieser J, Quan J, Gaffney S, Petersen S, Simonyan K, Schaul T, van Hasselt H, Silver D, Lillicrap T, Calderone K, Keet P, Brunasso A, Lawrence D, Ekermo A, Repp J, Tsing R. StarCraft II: A new challenge for reinforcement learning. 2017, arXiv:1708.04782.

[55] Berner C, Brockman G, Chan B, Cheung V, Debiak P, Dennison C, Farhi D, Fischer Q, Hashme S, Hesse C, et al. Dota 2 with large scale deep reinforcement learning. 2019, arXiv preprint arXiv:1912.06680.

[56] Huang S, Ontañón S. A closer look at invalid action masking in policy gradient algorithms. 2020, arXiv preprint arXiv:2006.14171.

[57] Der Kiureghian A, Ditlevsen O. Aleatory or epistemic? Does it matter? Struct Saf 2009;31(2):105–12.

[58] Shannon CE. A mathematical theory of communication. Bell Syst Tech J 1948;27(3):379–423.

[59] Cover TM, Thomas JA, et al. Entropy, relative entropy and mutual information. Elements Inf Theory 1991;2(1):12–3.

[60] Kullback S, Leibler RA. On information and sufficiency. Ann Math Stat 1951;22(1):79–86.

[61] Malinin A. Uncertainty Estimation in Deep Learning with Application to Spoken Language Assessment (Ph.D. thesis), University of Cambridge; 2019.

[62] Lakshminarayanan B, Pritzel A, Blundell C. Simple and scalable predictive uncertainty estimation using deep ensembles. Adv Neural Inf Process Syst 2017;30.

[63] Jain S, Liu G, Mueller J, Gifford D. Maximizing overall diversity for improved uncertainty estimates in deep ensembles. In: Proceedings of the AAAI conference on artificial intelligence. Vol. 34, 2020, p. 4264–71.

[64] Gal Y, Ghahramani Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: Balcan MF, Weinberger KQ, editors. Proceedings of the 33rd international conference on machine learning. Proceedings of machine learning research, 48, New York, New York, USA: PMLR; 2016, p. 1050–9, URL https://proceedings.mlr.press/v48/gal16.html.

[65] Straub D. Value of information analysis with structural reliability methods. Struct Saf 2014;49:75–85. http://dx.doi.org/10.1016/j.strusafe.2013.08.006, URL https://www.sciencedirect.com/science/article/pii/S0167473013000611, Special Issue In Honor of Professor Wilson H. Tang.

[66] Sutton RS, Barto AG. Reinforcement learning: An introduction. MIT Press; 2018.

[67] Sutton RS, McAllester D, Singh S, Mansour Y. Policy gradient methods for reinforcement learning with function approximation. Adv Neural Inf Process Syst 1999;12.

[68] Schulman J, Moritz P, Levine S, Jordan M, Abbeel P. High-dimensional continuous control using generalized advantage estimation. 2018, arXiv:1506.02438.

[69] HANSEN F, PEDERSEN GK. Jensen's OPERATOR inequality. Bull Lond Math Soc 2003;35(04):553–64. http://dx.doi.org/10.1112/s0024609303002200.