Materials Research Proceedings **18** (2021) 203-210 https://doi.org/10.21741/9781644901311-24

Reinforcement Learning for Structural Health Monitoring based on Inspection Data

Simon Pfingstl[1,a,*], Yann Niklas Schoebel[1,b] and Markus Zimmermann[1,c]

[1]Technical University of Munich, Laboratory for Product Development and Lightweight Design, Boltzmannstr. 15, 85748 Garching, Germany

[a]simon.pfingstl@tum.de, [b]yann.schoebel@web.de, [c]zimmermann@tum.de

Keywords: Reinforcement Learning, Structural Health Monitoring, Crack Growth, Inspection Timing

Abstract. Due to uncertainty associated with fatigue, mechanical structures have to be often inspected, especially in aerospace. In order to reduce inspection effort, fatigue behavior can be predicted based on measurement data and supervised learning methods, such as neural networks or particle filters. For good predictions, much data is needed. However, often only a small number of sensors to collect data are available, e.g., on airplanes due to weight limitations. This paper presents a method where data that is collected during an inspection is utilized to compute an update of the optimal inspection interval. For this purpose, we describe structural health monitoring (SHM) as a Markov decision process and use reinforcement learning for deciding when to inspect next and when to decommission the structure before failure. In order to handle the infinite state space of the SHM decision process, we use two different regression models, namely neural networks (NN) and k-nearest neighbors (KNN), and compare them to the deep Q-learning approach, which is state of the art. The models are applied to a set of crack growth data which is considered to be representative of the general damage evolution of a structure. The results show that reinforcement learning can be utilized for such a decision task, where the KNN model leads to the best performance.

Introduction

The lifetime of cyclically loaded metal structures is limited, due to the propagation of cracks in the material which leads to a failure of the structure after a certain amount of load cycles [1, 2]. Hence, aluminum parts in aircraft structures need to be inspected periodically to find and replace parts with significant crack propagation. Especially assemblies like wing and fuselage have to endure high cyclic stresses during their lifetime. Maintenance and health monitoring are significant factors in aircraft operating costs. For example, the average direct maintenance costs of a Boeing 757-200 had a proportion of about 23% of the total flight operating costs in 2017 [3].

Every opportunity to reduce these costs is beneficial to the aerospace industry and can reduce the overall costs per flight hour. Using machine learning (ML) methods can help to reduce maintenance costs. Therefore, a lot of research is currently done in the field of prognostic health monitoring (PHM). PHM tries to predict the time to failure of a structural system often by using ML methods. Usually, data, which indicates the deterioration of a system, serves as the input of the ML model. The predicted output is the time to failure of the structural system. Often continuous or many data points are needed whereas applied sensors are heavy and increase the weight. Especially in the aerospace industry, this leads to an increase of the fuel burn which might be more costly than operating an aircraft without an SHM system.

Therefore, this paper aims to show the application of reinforcement learning (RL) to an SHM task, which uses only data when an inspection is carried out. The feasibility of the approach is demonstrated using a set of published crack growth data [4], where the crack length serves as the damage index. This data is considered to resemble a degrading system, wherein real-world applications this might be for instance the number of repairs during a maintenance service. Furthermore, we propose another way of how to train the best decision policy and compare it to the state of the art approach in terms of the number of inspections, the wasted remaining life cycles, and the number of fails. By using RL, the planning of inspection dates might be improved and the overall number of inspections decreased.

In the following section, we summarize and discuss different PHM approaches. In sections 3 and 4, we explain the fundamentals of RL and describe how we apply it on an SHM task, respectively. After showing the results in section 5, we discuss and summarize them in the last section.

Literature Review

In research, a broad amount of possibilities to improve maintenance policies and lower their costs is investigated. One approach to determine the optimal maintenance policy is by Markov decision processes (MDP) or partially observed Markov decision processes (OMDP) [5, 6]. The research shows that MDPs or OMDPs are suitable to determine the optimal maintenance policy and to increase the overall system effectiveness. Moreover, it has been proven that reinforcement Q-learning can find the optimal action-selection strategy for every finite MDP, given sufficient exploration time and random actions [7]. Therefore, we describe structural health monitoring (SHM) as an MDP and use RL for deciding when to inspect next.

Another approach is to predict the damage progress by mathematical surrogate models. E.g. in [8-11], it has been already proven that the propagation of cracks can be predicted with neuronal networks. Even the propagation under overloads and non-linear damage can be predicted with such algorithms [12, 13]. Moreover, it is possible to perform live monitoring with sensor data of an operating system, e.g. in [14]. If the remaining useful life (RUL) can be predicted by a monitoring system, we commonly refer to this as prognostic health monitoring (PHM). Several ways exist to predict the RUL. One method, which has been very successful in the PHM challenge in 2008 [15, 16], was a similarity-based approach [17]. This method uses run to failure data to define a library of degradation paths. The data of the test system is compared to the library, and the most similar is used to determine the RUL. One disadvantage of PHM methods is that they require continuous or rather many sensor data points during the lifetime of a component to predict the damage progress. However, in most structural parts of an aircraft, there are no sensors to monitor the current condition. Additional sensors would lead to heavier aircraft which will inevitably increase the fuel burn and thus the operating costs. Therefore, this work presents an approach that uses only data that is collected during inspections.

Fundamentals of RL and new Approach

RL is an area of ML, in which a software agent is interacting with a simulated or physical environment. This environment has to be modeled as an MDP. The environment is at a time t in a state S, and the agent selects an action A, which leads the environment in a new state S'. Array $\boldsymbol{A}$ consists of all possible actions. The agent gets feedback on the performed action by a reward R. That allows the agent to learn by itself over time. It does not need a teacher. For finite environments, the agent selects actions, until it reaches a terminal state [18].

Q-learning is a special type of RL, where the agent learns so-called Q-values for an action taken in a state. These values represent the expected reward for performing that action in that

state. First, the Q-values are randomly initialized. Then, in the training step, the values are adopted according to a version of the Bellmann equation (Eq. 1),

$$Q^*(S, A) = (1 - \alpha)\, Q(S, A) + \alpha\, [R + \gamma \max(Q'(S', \boldsymbol{A}))] \quad (1)$$

where Q*, Q, and Q' are the updated, current, and future Q-values, respectively, α is the learning rate and γ the discount factor. In Q-learning, the Q-values are trained based on the current and on the future Q-value, i.e. the not yet updated Q-value and the highest achievable reward for doing an action one timestep in the future (temporal difference learning). Q-learning is based on an ε-greedy strategy. In the learning phase, the agent has the chance ε to perform a random action, while ε decreases over time. If ε is equal to zero, the agent does not execute a random action anymore and will always execute the action with the highest Q-Value ("greedy"). The learning rate α influences how much the new values replace the old. The discount factor γ is responsible for weighting the future Q-values compared to the current ones. The learning rate, the discount factor, and the ε parameter thus influence the speed and outcome of the training

The classic Q-learning stores the Q-values of every state in a table. This is a problem for large or infinite state and/or action spaces, where memory usage can overcome the available memory. To solve this problem, usually, a value approximator is introduced to predict the Q-values of every state. This is usually referred to as deep Q-learning which is frequently associated with a NN as the value approximator. An overview and basic description of deep RL and deep Q-learning, in particular, can be found in [19]. The method achieves very good performance in human interaction problems, e.g. playing Atari or board games [20], and is so effective that computers can beat people in extremely complex games with many possible move combinations like Go or Chess [21].

The state of the art approach uses a built-in NN, which's parameters are trained in the learning phase in order to predict the Q-values. However, this could lead to long training times until the NN predicts the expected Q-values of the Bellmann equation. Furthermore, large steps in the Q-value function might be difficult to approximate and many parameters of the NN would be needed. Another idea, which we want to propose in this paper, is to first train the Q-values by using the default Q-learning approach and store them in a table. Afterward, the learned table can be used to train a mathematical surrogate model. The inputs of the model are the state variables and the outputs are the Q-values for each action. With this approach, a larger variety of mathematical surrogate models can be used, e.g. k-nearest neighbors (KNN), since the data to be trained are already available. A combination of Q-learning and the KNN method was already described in [22, 23], where the authors applied this approach to the Mountain-Car control problem and the Cart-Pole balancing task.

Proposed RL Framework for an SHM Decision Process

First, we describe the maintenance decision task as an MDP. Figure 1 shows the MDP of the SHM problem, where two kinds of states (*in service* and *out of service*) are present.

Materials Research Proceedings **18** (2021) 203-210 https://doi.org/10.21741/9781644901311-24

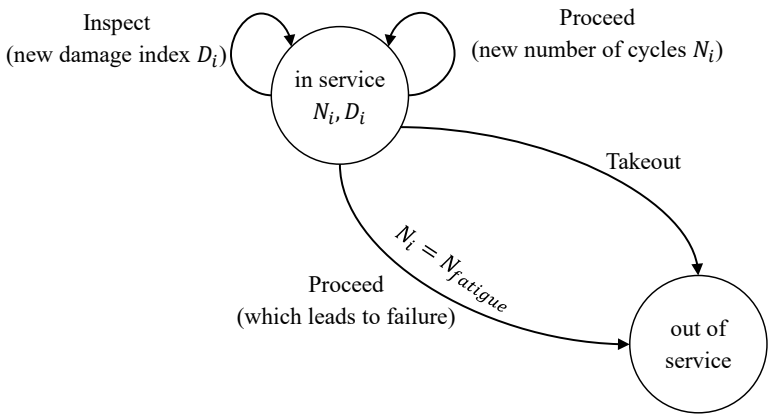


Figure 1: Markov decision problem for the maintenance decision task with crack length as the damage index.

A state S consists of two variables, the number of cycles N and the damage index D. A new state is entered if one or both variables change their values. However, two inspections in a row are not allowed, since this would not change the state and could lead to an infinite loop. In total, three actions are available to change the state:

- *Proceed.* The number of cycles increases by one. The agent does not get any new information about the damage index. Therefore, the last observed damage index stays the same.
- *Inspect.* The number of cycles stays the same, whereas the damage index gets updated by the current one, which corresponds to the current number of cycles.
- *Takeout.* The structure gets decommissioned and the episode ends.

Second, in order to use RL and to train an agent, we defined four different rewards:

- *Move.* The agent receives a positive reward for every proceed-action if the action does not lead to a structural failure. This aims to encourage the agent to use the structure as long as possible.
- *Maintenance.* The agent receives a negative reward for every usually costly inspection since the aim is to achieve a long life with a minimal number of inspections.
- *Fail.* The agent receives a large negative reward if the action leads to a structural failure.
- *Takeout.* The agent receives a mediocre reward, which is lower than the fail reward but larger than the maintenance reward.

Since the damage index is a float number, we have an infinite number of possibles states. Moreover, we cannot assure that all states seen by the agent in the learning phase cover all states occurring in the operating phase. This is why we need a value approximator, where the inputs are the state variables (i.e. the number of cycles and the damage index) and the outputs are the Q-values for each action. In this paper, we compare the state of the art approach (built-in NN) with the proposed approach, where we first train the Q-values by using the default Q-learning approach and then train a mathematical surrogate model on it. For the latter, we use two different models, KNN and NN. Therefore, we compare three different models in total.

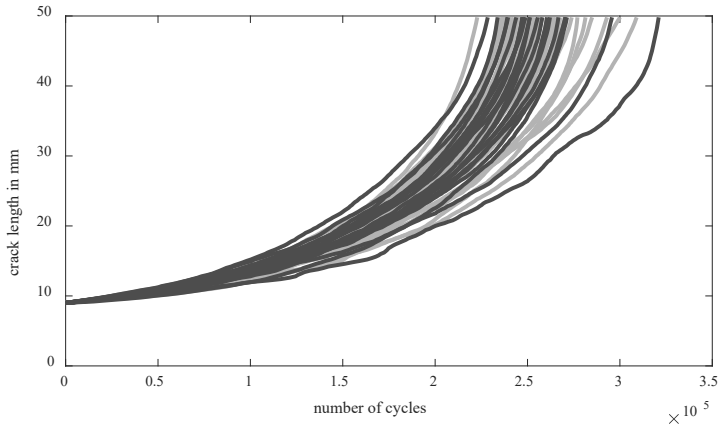


Figure 2: Crack propagation lines split up into a training (light gray) and a test set (dark gray).

The feasibility of using RL for a maintenance decision task is demonstrated by using a set of published crack propagation data [4], where the crack length serves as the damage index. This data is considered to resemble a degrading system. This might be in real-world applications for instance the number of repairs during an inspection. We assume that the overall damage of a structural system is degrading continuously, even if maintenance tasks are executed. Figure 2 shows the 68 crack propagation lines, which are split up into a training set (line 1-47) and a test set (line 48-68). The training set is used to learn the Q-values and to set the parameters of the mathematical surrogate models, whereas the test set is used to evaluate the models.

The number of layers and the number of neurons can be optimized using a hyperparameter optimizer. In this case, we used a fully connected NN with 128 neurons in the first and 64 neurons in the second hidden layer. The output layer has three nodes (one for each action Q-value). A 5-fold cross-validation within the training set led to an optimal KNN parameter at k = 250. The Q-values are randomly initialized between -5 and 0. For this study, we used the following rewards: $R_{move} = 1$, $R_{maint} = -5$, $R_{fail} = -500$, $R_{takeout} = -50$. These rewards lead to a behavior, where the agent tries to maximize the usage time of the structure. These values could also be optimized, to get a certain behavior of the agent, e.g. most important is safety or a small number of inspections.

Results

Table 1 shows the evaluation of the test lines 48, 59, and 68, which represent a slow, medium, and fast crack growth, respectively. The blue point indicates an inspection, whereas the red point represents the takeout. It can be seen that the agent with the built-in NN behaves quite statically. The RL model has learned to execute the takeout always at about 225,000 cycles. No inspection is done, neither for fast nor for slow crack growth. In contrast, the agent with the afterward trained NN executes many inspections beginning at about 235,000. This leads to longer usage but also many inspections. For a fast crack growth, the agent misses decommissioning the structure (see figure on the right bottom in Table 1). Also, the agent of the KNN model executes inspections. However, the first inspection is carried out a little earlier than for the NN model leading to no missed takeouts. If the crack length is rather long, the agent executes the takeout action. If the observed crack length is rather short, the agent proceeds some cycles until it inspects next.

Table 1: Comparison of the results.

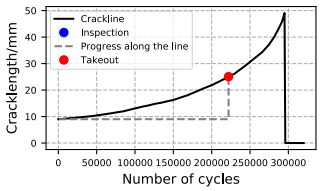
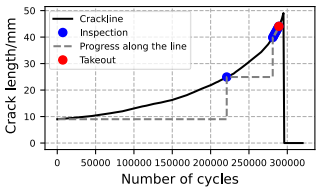
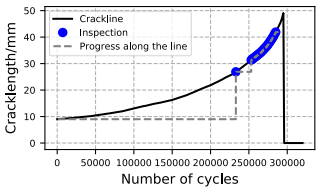
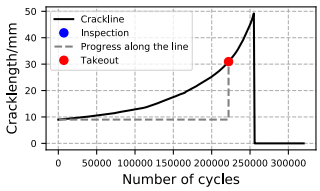
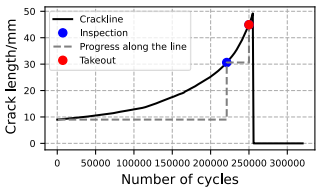
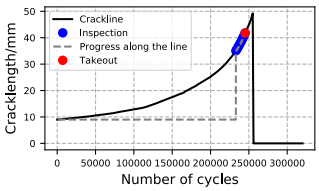
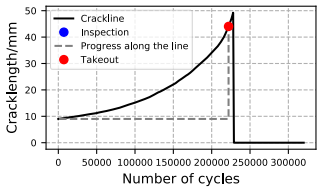
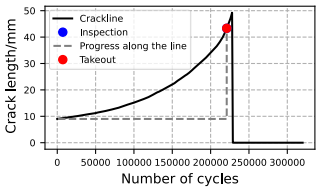
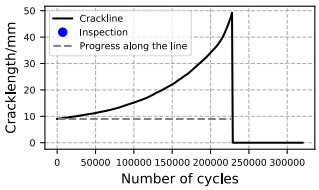
Test line	Built-in NN	KNN	NN
48			
59			
68			

Table 2 compares the three approaches in terms of the number of inspections, the wasted remaining life cycles, and the number of fails evaluated on the test set. The best is if all three measures are zero. The behaviors of the different approaches explained before are consistent with Table 2. The values of the built-in NN show that, due to the lack of inspections, the agent wastes much RUL. The KNN model is better than the NN model since its values are lower in all three measures. This is due to the fact that the Q-values change rapidly at high numbers of cycles. This is a result of the high negative reward when missing the takeout. Apparently, the KNN model can handle this discontinuity better, since it is based on the trained neighbor values. For a regression model like NN, it is difficult to picture this behavior.

Table 2: Comparison of the three approaches evaluated on the test set.

Method	Number of inspections	Wasted RUL	Number of fails
Built-in NN	0	760,000	0
KNN	92	144,000	0
NN	320	153,000	4

Discussion and Conclusion

The results show that is is possible to use RL to automate the maintenance planning of cyclically loaded parts. One advantage of using RL is that an agent can be trained even though it is not known how to optimally inspect the structure. Even if the trained model will not be applied for

instance due to safety laws, it can be a valuable insight of when to carry out inspections. However, the proposed study is a very simplified example of a real maintenance task. The assumption that a real structure behaves like a continuously degrading system might not be valid, since the structural system is repaired after an inspection. Moreover, a big problem could be data acquisition. Therefore, it would be an interesting task to work together with an aircraft operator to find out if the currently measured data can be used as an input for the proposed method.

Furthermore, the study reveals that using models other than the built-in NN can be beneficial, especially if the Q-values change rapidly. Additionally, the training phase which includes the reinforcement learning phase and the training of a mathematical surrogate model is shorter compared to the start of the art approach. However, the KNN model predicts the Q-values slower than the NN model, since distances to all neighbors have to be computed.

Overall, this work was able to show that Reinforcement Learning has a high potential in tasks like SHM.

References

[1] Subra Suresh, Fatigue of Materials, Second edition, Cambridge University Press, Cambridge, 1998.

[2] Stephen J. Burns, The Theory of Materials Failure, by Richard M. Christensen, (56), 2015.

[3] International Civil Aviation Organization (ICAO), Airlines Operating costs and productivity, Teheran, 2017.

[4] D. A Virkler, B. M. Hillberry, P. K. Goel, The Statistical Nature of Fatigue Crack Propagation., Journal of Engineering Materials and Technology 101, 1979, pp. 148–153.

[5] Mohammad M. AlDurgam, Salih O. Duffuaa, Optimal joint maintenance and operation policies to maximise overall systems effectiveness, International Journal of Production Research 51 (5), 2013, pp. 1319–1330. https://doi.org/10.1080/00207543.2012.659351

[6] G. K. Chan, S. Asgarpoor, Optimum maintenance policy with Markov processes, Electric Power Systems Research 76 (6-7), 2006, pp. 452–456. https://doi.org/10.1016/j.epsr.2005.09.010

[7] Francisco S. Melo, Convergence of Q-learning: a simple proof, Institute for Systems and Robotics, Lisboa, Portugal.

[8] K. Zarrabi, W. W. Lu, A. K. Hellier, An Artificial Neural Network Approach to Fatigue Crack Growth, AMR 275, 2011, pp. 3–6. DOI: https://doi.org/10.4028/www.scientific.net/AMR.275.3

[9] Hassaan Bin Younis, Khurram Kamal, Muhammad Fahad Sheikh, Amir Hamza, Tayyab Zafar, Prediction of fatigue crack growth rate in aircraft aluminum alloys using radial basis function neural network, Archieves of Computational Materials Science and Surface Engineering, pp. 825–830. https://doi.org/10.1109/ICACI.2018.8377568

[10] R. M. V. Pidaparti, M. J. Palakal, Neural network approach to fatigue-crack-growth predictions under aircraft spectrum loadings, Journal of Aircraft 32 (4), 1995, pp. 825–831. https://doi.org/10.2514/3.46797

[11] J. R. Mohanty, B. B. Verma, D.R.K. Parhi, P. K. Ray, Application of artificial neural network for predicting fatigue crack propagation life of aluminum alloys, Archives of Computational Materials Science and Surface Engineering (1/3), 2009, pp. 133–138.

[12] Wei Zhang, Zhangmin Bao, Shan Jiang, Jingjing He, An Artificial Neural Network-Based Algorithm for Evaluation of Fatigue Crack Propagation Considering Nonlinear Damage Accumulation, Materials (Basel, Switzerland) 9 (6), 2016. https://doi.org/10.3390/ma9060483

[13] J. R. Mohanty, B. B. Verma, , P. K. Ray, D.R.K. Parhi, Prediction of mode-I overload-induced fatigue crack growth rates using neuro-fuzzy approach, Expert Systems with Applications 37 (4), 2010, pp. 3075–3087. https://doi.org/10.1016/j.eswa.2009.09.022

[14] J.P.M. Smeulers, R. Zeelen, A. Bos, PROMIS - A generic PHM methodology applied to aircraft subsystems, 6-3153-6-3159. https://doi.org/10.1109/AERO.2002.1036156

[15] A. Saxena, K. Goebel, PHM08 Challenge Data Set. NASA Ames Research Center, Moffett Field, CA, 2008, Available online at NASA Ames Prognostics Data Repository http://ti.arc.nasa.gov/project/prognostic-data-repository, checked on 7/19/2020.

[16] Emmanuel Ramasso, Abhinav Saxena, Performance Benchmarking and Analysis of Prognostic Methods for CMAPSS Datasets, International Journal of Prognostics and Health Management (5), 2014.

[17] Tianyi Wang, Jianbo Yu, David Siegel, Jay Lee, A similarity-based prognostics approach for Remaining Useful Life estimation of engineered systems, International Conference on Prognostics and Health Management, 2008, pp. 1–6. https://doi.org/10.1109/PHM.2008.4711421

[18] Richard S. Sutton, Andrew G. Barto, Reinforcement learning. An introduction, MIT Press, Cambridge, 1998. https://doi.org/10.1109/TNN.1998.712192

[19] Yuxi Li, Deep Reinforcement Learning: An Overview, 2018, arXiv-ID: 1701.07274v6 [cs.LG]

[20] Volodymyr Mnih; Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare et al., Human-level control through deep reinforcement learning (518), Nature (7540), 2015, pp. 529–533.

[21] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche et al., Mastering the game of Go with deep neural networks and tree search, Nature 529 (7587), 2016, pp. 484–489. https://doi.org/10.1038/nature16961

[22] José Antonio Martín H, Javier de Lope, Darío Maravall, The kNN-TD Reinforcement Learning Algorithm, J. Mira (Ed.): Methods and models in artificial and natural computation. A homage to Professor Mira's scientific legacy International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2009, Santiago de Compostela, Spain, June 22-26, proceedings, part I. 1st ed. Berlin: Springer (Lecture notes in computer science, 5601), 2009, pp. 305–314.

[23] José Antonio Martín H, Javier de Lope, Darío Maravall, Robust high performance reinforcement learning through weighted k-nearest neighbors, Neurocomputing 74 (8), 2011, pp. 1251–1259. https://doi.org/10.1016/j.neucom.2010.07.027