If it ain't broke, ~~don't~~ fix it

Optimizing the predictive aircraft maintenance schedule with Remaining Useful Life prognostics

If it ain't broke, ~~don't~~ fix it

Optimizing the predictive aircraft maintenance schedule with Remaining Useful Life prognostics

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, Prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Thursday 11 April 2024 at 15:00 o'clock

by

Ingeborg Irene DE PATER

Master of Science in Econometrics and Management Science, Erasmus University
Rotterdam, The Netherlands
born in Utrecht, The Netherlands

Keywords: Predictive maintenance, Remaining Useful Life prognostics, Aircraft maintenance, Maintenance scheduling, Optimization

Printed by: Ipskamp Printing

Cover by: Ingeborg de Pater, with the help of deepai.com

ISBN: ....

An electronic copy of this dissertation is available at
https://repository.tudelft.nl/.

Predicting the future is a very difficult business indeed.

J.K. Rowling,
Harry Potter and the prisoner of Azkaban

CONTENTS

ACKNOWLEDGEMENTS

During this PhD, my aim was to predict the future, or, put less poetic, to predict the future failure times of aircraft systems. As I could have known from the Harry Potter books, predicting the future is not an easy undertaking, and I did not even have magic to help me. However, I also learned from the Harry Potter books that I did have something far more valuable than magic: Colleagues, friends and family. Thank you all for your help and support, because without you, this thesis would never have seen the light of day.

Above all, I would like to thank my supervisor Mihaela Mitici. Your guidance, enthusiasm and patience made my PhD a very enjoyable experience. You taught me everything on conducting research, from finding new research directions to managing the review process. Moreover, you taught me all the ins and outs of academic writing by spending days writing papers with me. I really appreciate all the time and effort you put into making my PhD a success. Equally important, I also enjoyed working with you on a personal level very much. You cared for me and the other PhD students not just as researchers, but as humans. You always encouraged me with your kind words if a paper was turned down, if I was stuck in coding, or completely fed up with writing (never my favourite activity). I will miss our small, “gezellige” chats during lunch and over coffee. Thank you very much for all your support over the past four years.

Second, I would like to thank my promotor, Max Mulder. You have the busiest schedule I have ever seen, where days with meetings from 9 am to 5 pm are no exception. But despite this, you always made time for me when I needed your guidance. I enjoyed our friendly chats with the good coffee and tea in your office, and I appreciate your friendly and always encouraging words. Also for you, my personal well-being came in the first place, and my research only in the second place. I appreciated that very much. During the last phase of my PhD, I was very happy with your efficient help with both the writing of the thesis and with all the bureaucracy.

Even though I enjoyed my PhD in itself very much, it were my colleagues who made it a truly great experience. I would like to thank all (former) PhD students and postdocs from ATO. Marie, for learning me how to boulder, Mahdi, for a huge supply of saffron and our friendly chats, Hao, for brightening my work day with our little chats, Simon, for always finding new mathematical events to which we could go together, Mike, for all your tips on hiking and nature, Matt, for sharing your great selection of alcohol with us, Malte, for all your nice birthday parties, Chengpeng, for sharing your passion for drones, George, for cheering me up with your fantastic outfits, Ilias, for sharing your enthusiasm about games, Thomas, for taking over, not entirely voluntarily, my ATO seminars, Iordanis, for being such a nice person to hang out with, Prashant, for learning me about the Indian cuisine, Wenhua, for introducing me to the best Chinese restaurant I have ever been to, and Gulcin, for all your great sweets from Turkey. My special thanks go to Juseong: I really appreciate all your help during the beginning of my PhD, and I always enjoyed going

to conferences together. I would also like to thank all our visiting PhD students and the (former) staff of ATO: Marta, for our “gezellige” chats, Paul, who taught me nearly everything I know about aircraft, Marcia, Alexei, Elise, Alessandro, Felipe, Henk, Vis, Bruno and Nathalie.

In Utrecht, I would like to thank Diogo, for being such a fun and relaxed person to hang around with, and Nishant. I really enjoyed working together with, and learning from, both of you. I would also like my colleagues Jie and Jiayan; I enjoyed sharing our office together very much.

Talking to my friends outside my PhD about everything except my PhD, gave me the energy I needed to continue my research. Thank you Annemarijn, for our wonderful board year together, Iggy, for being such a great flatmate, Emilie, for celebrating my successes and always telling me to be more confident and less modest, Kathelijne, for being my oldest friend, Joost, for I wish you could have added this thesis to your amazing book collection, Bernice, Aletta, Eline, Juultje, Marcella, Naomi, Solveig, Dionne, Mia, Maeve, Simone, Michelle, Olivia, Lisa, David, Joshua, Floris, Laura en Lucas, Susan, Talitha, Laura, Caro, all my other friends from Devi, and everyone I (hopefully none) accidentally forget.

Then, I would like to thank my family. First, I would like to thank my parents, for always wishing the best for me, for always cooking for me like I have not eaten in days, and for giving me a wonderful childhood full of books, cooking, playing, exploration, amazing holidays, juggling, seafood and, most importantly, endless amounts of love. And, of course, I would like to thank my sister Sophie, for being an amazing play mate and role model when I grew up, and for being fully confident in my capability to complete a PhD. It makes me very happy to know that you will always be there for me if I need you.

Last, but not least, I would like to thank Joost. You are always there for me, and by having so much confidence in me, I also get more confidence in myself. We have great times together, and do so many fun things, like going on amazing holidays, long walks and talks, visiting the zoo, cooking and eating with friends, and simply watching our favourite series on the couch. I predict that in the future, when we are both old and wrinkled, we will still visit Blijdorp together!

SUMMARY

Predictive aircraft maintenance is a maintenance strategy that aims to reduce the number of failures, the number of inspections, the number of maintenance tasks and the aircraft maintenance costs. Aircraft are equipped with health monitoring systems, where sensors continuously measure the condition of the aircraft components. In predictive maintenance, these sensor measurements are used to estimate the time left until the failure of these components, called the *Remaining Useful Life* (RUL). These RUL prognostics are subsequently used to optimize the aircraft maintenance schedule. There are several challenges that complicate the implementation of predictive aircraft maintenance in practice. In this thesis, the three main challenges are addressed.

Challenge 1: Developing accurate RUL prognostics for aircraft systems.

The first challenge is to accurately estimate the RUL for aircraft systems. This is challenging because the aircraft are operated under highly-varying operating conditions. Moreover, since most aircraft systems undergo preventive maintenance, there are only very few failure instances available. Most data samples are unlabelled, i.e., the sensor measurements of a flight are available, but the corresponding true RUL is unknown.

In Chapter 2, these unlabelled data samples from non-degraded aircraft systems are used to train a Long Short-term Memory autoencoder, an unsupervised learning neural network. The errors of this autoencoder are used to create a health indicator for the aircraft systems. The RUL is subsequently estimated from the health indicator with a similarity-based matching method. This approach is used to estimate the RUL of aircraft engines, for which only six failure instances are available. Accurate RUL prognostics are obtained with a Root Mean Square Error (RMSE) of 2.67 flights only, which is 19% lower than when using a supervised learning neural network. Moreover, integrating the highly-varying operating conditions of the aircraft in the autoencoder improves the monotonicity of the health indicators by 97%, which is statistically significant.

Training a neural network to accurately estimate the RUL is very time-consuming. In Chapter 3, the training time is significantly reduced by developing a new weight initialization method for the weights in the last layer of a neural network. This new initialization methods minimizes the initial training loss, while constraining the variance of the weights. This method is applied in a Convolutional Neural Network (CNN), that estimates the RUL of aircraft engines. With the proposed method, the training time is significantly reduced: 34% fewer epochs are needed to reach the same validation loss as the best benchmark initialization strategy (Kaiming initialization). Moreover, the new method, combined with transfer learning, also accelerates the training of classification neural networks commonly used in image recognition.

Challenge 2: Quantifying the uncertainty of the RUL prognostics.

The second challenge is to develop *probabilistic RUL prognostics*, i.e., RUL prognostics with quantified uncertainty. In Chapter 4, the Probability Density Function (PDF) of the RUL of aircraft cooling units is estimated. First, several potential health indicators for the cooling units are evaluated. With the best health indicator, aircraft components are identified as healthy or unhealthy with Chebyshev's inequality. Once a cooling unit is diagnosed as unhealthy, the degradation model is identified using a dynamic time-warping clustering approach. Last, the PDF of the RUL is estimated with this degradation model and particle filtering. All cooling units are diagnosed as unhealthy between 40 and 2 flights before failure, at which point the particle filtering method already provides accurate RUL prognostics with a RMSE of only 4.04 flights. Last, the inner workings of the particle filtering method are explainable, and only a few failure instances are needed to estimate the (hyper)parameters of this method.

There are many metrics to evaluate the accuracy of *point RUL prognostics*, i.e., RUL prognostics without quantified uncertainty. However, there is a lack of metrics to evaluate probabilistic RUL prognostics. In Chapter 5, four new metrics to evaluate probabilistic RUL prognostics in the form of a PDF are proposed. The Continuous Ranked Probability Score (CRPS) and the weighted CRPS evaluate the accuracy and sharpness of a single probabilistic RUL prognostic. The α-Coverage and the Reliability Score evaluate the reliability of multiple probabilistic RUL prognostics. These metrics are used to evaluate the probabilistic RUL prognostics of aircraft engines, which are estimated with a CNN with Monte Carlo dropout. It is shown that the four proposed metrics together capture the accuracy, sharpness and reliability of the probabilistic RUL prognostics well.

Challenge 3: Optimizing the aircraft maintenance schedule with RUL prognostics.

The last step, and challenge, in predictive maintenance is to optimize the maintenance schedule for a fleet of aircraft with the RUL prognostics, while integrating several aspects of aircraft maintenance: The limited capacity, the limited number of spare components, the limited maintenance opportunities and the uncertainty of the RUL prognostics.

In Chapter 6, the aircraft maintenance schedule is optimized with imperfect point RUL prognostics without quantified uncertainty. An alarm is triggered for a system if its RUL prognostic falls below an alarm threshold several flights in a row. Once an alarm is triggered, a safety factor is used to schedule maintenance, to avoid failures due to errors in the RUL prognostics. With the alarms, the frequent rescheduling of maintenance tasks is prevented. The alarm threshold and the safety factor are optimized with a genetic algorithm. This method is applied to plan maintenance for 20 aircraft, each equipped with 2 aircraft engines. The RUL of the engines is estimated with a CNN. Due to the safety factor, on average only 1.6% of the maintenance tasks take place after the engine failed, as estimated with a Monte Carlo simulation of the long-term maintenance planning.

Probabilistic RUL prognostics are instead used to optimize the aircraft maintenance schedule in Chapter 7. The renewal-reward process is used to determine the optimal maintenance moment for a single aircraft system. The maintenance planning for the fleet of aircraft is subsequently optimized with a linear program. In the case study, maintenance is planned for a fleet of 50 aircraft, each equipped with one aircraft engine. A CNN with Monte Carlo dropout is used to estimate a PDF of the RUL. The optimal maintenance moment of the engines is usually close to the lower bound of the 99% confidence interval of the RUL prognostic, i.e., when the probability of failure is still small. There are therefore on average only 0.003 engines failures per ten years in the Monte Carlo simulation, while there are on average 61.6 engine failures per ten years in a preventive maintenance strategy without RUL prognostics.

The problem is extended by considering a limited number of spare components and a multi-component aircraft system with redundant components in Chapter 8. Specifically, a linear program is used to optimize the maintenance schedule for aircraft cooling systems, where each cooling system consists of four cooling units. The PDF of the RUL of these cooling units is estimated with particle filtering, as in Chapter 4. With the RUL prognostics, the expected number of maintenance tasks in five years decreases from 135 to 106, making the maintenance more efficient.

Last, the initial aircraft maintenance schedule and all possible future updates of this schedule are jointly optimized with a multi-stage stochastic program in Chapter 9. This stochastic program is solved with the nested Benders decomposition method. To accelerate this method, a new clustering algorithm is developed based the endogenous (decision-dependent) uncertainty in the problem. In the case study, the maintenance planning for up to five aircraft engines and a planning horizon of four weeks (28 days/ stages) is optimized. With the stochastic program, the expected costs are up to 0.89% lower compared to the upper bound solution. Moreover, the clustering algorithm significantly reduces the computational time. For five engines, the same number of iterations of the nested Benders decomposition is executed 25 times faster with the clustering algorithm, than without the clustering algorithm.

In Chapter 7 and 8, a predictive maintenance strategy (with RUL prognostics) is compared with a preventive maintenance strategy (without RUL prognostics). The costs with predictive maintenance are 53% (Chapter 7) and 30% (Chapter 8) lower than with preventive maintenance. In Chapter 6 and 7, a maintenance strategy with the imperfect RUL prognostics from the RUL prognostic models is compared with a maintenance strategy with perfect RUL prognostics without any uncertainty. With perfect RUL prognostics, the costs decrease by 19.5% (Chapter 6) and 14% (Chapter 7). Efforts to further improve the RUL prognostic models are thus worthwhile.

Conclusions and recommendations for future research.

This thesis provides an overall framework for predictive aircraft maintenance, that describes i) how to estimate the RUL from the sensor measurements, ii) how to quantify the uncertainty of the RUL prognostics and iii) how to use these RUL prognostics to optimize the maintenance planning. Future work could extend this framework in several ways. First, the RUL prognostics can be improved by making explainable RUL prognostic models, that can be trained without any failure instances. Second, the proposed maintenance planning models can be extended with the maintenance planning for other aircraft systems, while jointly optimizing the flight schedule.

1 INTRODUCTION

In the early days of aviation, flying was a dangerous endeavour. Only three days before the first successful flight in history by Orville Wright, his brother Wilbur Wright crashed in the dunes in the world's first powered aircraft accident [1]. Flying was dangerous for a number of reasons: Untrained pilots, unsafe aircraft designs and a lack of aircraft maintenance [1, 2]. Following the "if it ain't broke, don't fix it" paradigm, maintenance was usually only performed after parts of the aircraft had failed [2]. This is called *corrective maintenance*.

During the two world wars, the safety of flying rapidly improved due to, amongst others, the mass production of aircraft, research on aircraft safety and new aircraft designs [1]. New maintenance strategies contributed to this increased safety. Specifically, aircraft maintenance transitioned from corrective maintenance to *preventive maintenance*, where the aim is to maintain a component before it fails [2]. Aircraft maintenance soon became a mixture of three types of maintenance: *Hard time* preventive maintenance, where a component is always replaced after a fixed amount of time, *on condition* preventive maintenance, where components are frequently inspected and only replaced when there is degradation, and *condition monitoring maintenance*, where sensors detect degradation/failures [3–5]. These three maintenance processes have become the dominant maintenance practice.

In current practice, aircraft systems undergo frequent inspections during which their degradation is assessed [6, 7]. A system is then only preventively maintained if the degradation exceeds a certain threshold. Other *life-limited* parts of the aircraft are preventively replaced after a fixed operating time [6, 7]. Though these frequent inspections and time-based replacements decrease the number of failures, they also make aircraft maintenance relatively inefficient. Approximately 70% of the aircraft inspections do not find any fault [4]. And with the time-based replacements, some of the remaining life is wasted. For instance, with the time-based replacements of the life-limited aircraft engine parts, between 3% to 15% of the remaining life is usually wasted [6]. This makes aircraft maintenance very expensive. In 2021, the average maintenance cost per flight hour per aircraft was 1,340 dollar, and the total costs of Maintenance, Repair and Overhaul (MRO) for the aircraft made up for on average 11.2% of the total airline operational costs [8].

Even with these expensive frequent inspections and time-based replacements,

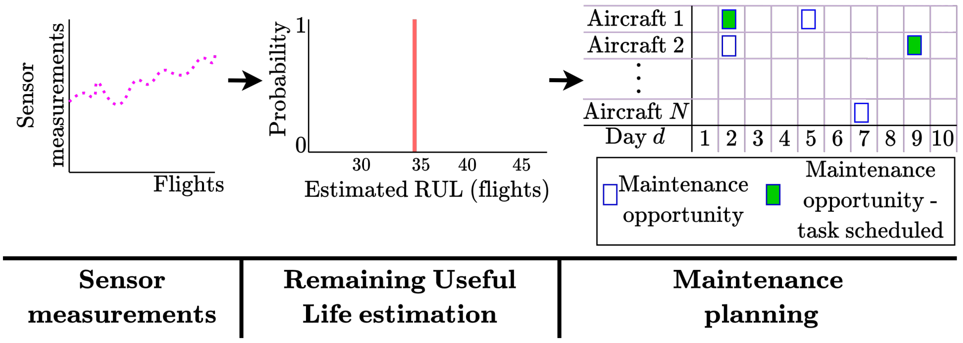


Figure 1.1.: A schematic overview of the predictive aircraft maintenance process.

unexpected failures of aircraft components still occur. These failures usually do not jeopardize the safety of operating the aircraft, due to the fail-safe design of, and redundancies in, most aircraft systems [9, 10]. However, aircraft are not allowed to fly with certain failed components, which results in unscheduled maintenance [10, 11]. Between one third to one fifth of an aircraft's downtime due to maintenance comes from unplanned maintenance [4], which may lead to aircraft delays (costs: 10,000 dollar per hour or more [4]), and even flight cancellations, (costs: 100,000 dollar or more [4]) [10]. Not only scheduled maintenance with inspections and time-based replacements, but also unscheduled maintenance is thus very expensive.

Overall, the current aircraft maintenance practice is expensive and contains a difficult trade-off: More frequent inspections and earlier replacements make aircraft maintenance less efficient (which is costly), but fewer inspections and later replacements cause more unexpected failures (which is also costly) [5]. This trade-off can be mitigated by a new maintenance process, called *predictive maintenance*.

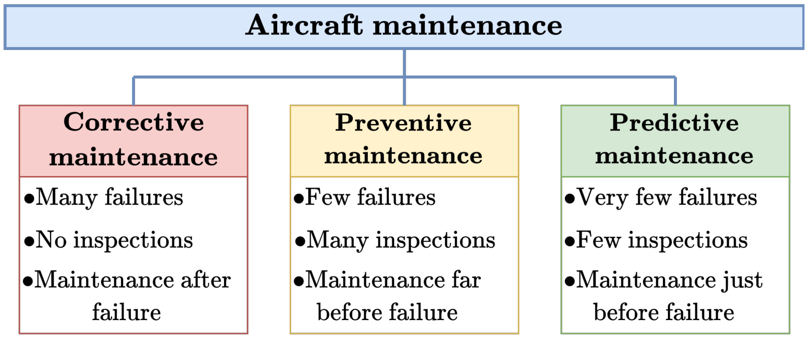


Figure 1.2.: An overview of the characteristics of corrective, preventive and predictive aircraft maintenance.

Aircraft are equipped with a health monitoring system, where sensors continuously measure the condition of the aircraft components [4]. The latest generation of aircraft contain around 300,000 sensors [5]. In predictive maintenance, these sensor measurements are used to estimate the time left until the failure of an aircraft component, the *Remaining Useful Life (RUL)* [4]. These RUL prognostics are then used to optimize the aircraft maintenance schedule, Figure 1.1. This approach is efficient since components are maintained *just before* failure, while only few inspections are needed. And due to the continuous monitoring, predictive maintenance reduces the number of failures in an aircraft [4]. Although predictive aircraft maintenance is very promising, there are still several obstacles that complicate its implementation in practice. The main challenges are discussed in more detail below. Figure 1.2 illustrates the characteristics of corrective, preventive and predictive aircraft maintenance.

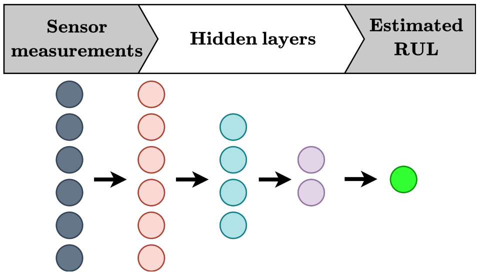
1.1. CHALLENGES FOR PREDICTIVE AIRCRAFT MAINTENANCE

1.1.1. CHALLENGE 1: DEVELOPING ACCURATE RUL PROGNOSTICS FOR AIRCRAFT SYSTEMS

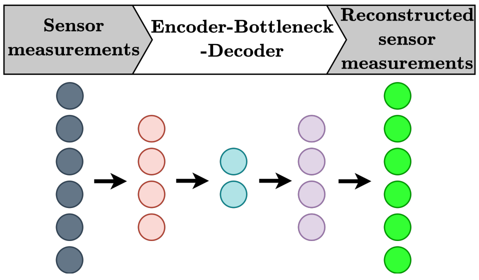
The first step in predictive maintenance is to estimate the RUL of a system from its sensor measurements, see Figure 1.1. There are many studies that estimate the RUL of a variety of systems [12, 13], such as batteries [14–18], bearings [19–24], milling machines [25, 26] and wind turbines [27, 28]. For instance, the RUL of batteries is estimated from temperature, voltage, current, capacity and resistance measurements [14–18], while the RUL of bearings is estimated from vibration measurements [19–24]. Accurately estimating the RUL of aircraft systems, however, is still challenging.

Many studies have successfully used a purely *data-driven approach* to estimate the RUL. Within the data-driven approach, the *supervised learning* models are very popular. In a supervised learning model, the RUL is directly estimated from the sensor measurements. For instance, some studies employ random forests to estimate the RUL directly from the sensor measurements [29, 30]. Even more popular are supervised learning neural networks, see Figure 1.3a. For instance, in [23, 24, 31–33], the sensor measurements are used as input in a Convolutional Neural Network, which directly outputs an estimate of the RUL. Similar, in [34–36], a Long Short-Term Memory Neural Network is used for directly estimating the RUL, while in [37–39], a Gated Recurrent Unit Neural Network is employed instead.

However, many labelled data samples are needed to train an accurate supervised learning model. For aircraft systems, many sensor measurements are available, due to the modern aircraft health monitoring systems. For instance, an aircraft engine generates up to one terabyte of data during just a single flight [40–42]. However, due to preventive maintenance, most aircraft systems are already maintained far before failure. This is especially the case for safety-critical aircraft systems. For these systems, the RUL cannot be observed and remains unknown. These systems therefore have many *unlabelled*, censored data samples, i.e., samples with the sensor



(a) Supervised learning neural network.


Sensor measurements
Encoder-Bottleneck -Decoder
Reconstructed sensor measurements

(b) Unsupervised learning neural network.

Figure 1.3.: A schematic overview of a supervised learning neural network and an unsupervised learning neural network for estimating the RUL.

measurements of a flight, but without the corresponding RUL. These unlabelled data samples cannot be used to train a supervised learning model. For most aircraft systems, only very few failure instances are available i.e., sensor measurements that are gathered from the installation of the system until failure. Consequently, *labelled* data samples, with the sensor measurements of a flight and the corresponding true RUL, are sparse. Only these labelled data samples can be used to train a supervised learning model [43]. Most supervised learning models proposed in literature for estimating the RUL are therefore not suitable for most aircraft systems.

Instead, for most aircraft systems, *data-driven unsupervised learning* models are more suitable to estimate the RUL. An unsupervised learning model is a model that can be trained with the abundantly available *unlabelled* data samples. To develop RUL prognostics, an unsupervised learning model is trained with the unlabelled data samples from non-degraded systems only. As such, the errors of these models are large for degraded systems [44]. A well-known unsupervised learning model that is often employed in predictive maintenance, is the autoencoder, see Figure 1.3b. An autoencoder is trained to first encode the sensor measurements (i.e., to reduce the dimension) and then to decode the sensor measurements again (i.e., to reconstruct the sensor measurements from the reduced dimension). The reconstruction errors are subsequently used to detect increasing degradation and/or to estimate the RUL [45–51]. Another unsupervised learning method employed in predictive maintenance is Principal Component Analysis (PCA), where the unlabelled data samples are used to estimate the weights of the PCA. The difference between the principal components of the sensor measurements and the real sensor measurements is subsequently used to detect and diagnose faults in systems [52–55].

However, it is not straightforward to estimate the RUL of aircraft systems with an unsupervised learning model. First, the measurements of multiple sensors around an aircraft system are recorded at a high-frequency during long flights. Each flight generates a long, multivariate time-series of sensor measurements. Standard unsupervised learning models, such as PCA or an autoencoder, are not suitable to analyse long multivariate time-series [56, 57]. Second, aircraft are operated under highly-varying operating conditions, such as varying altitude and speed, which affect the sensor measurements. The variations in the sensor measurements due to varying operating conditions do not contain any information on the degradation in a system, and are therefore regarded as “noise” in predictive maintenance [12]. With an unsupervised learning model, the noise in the measurements due to the varying operating conditions might lead to a false diagnosis of a healthy system as degraded. As such, one of the major challenges in predictive maintenance is to develop RUL prognostics that are robust to varying operating conditions by distinguishing the degradation signal from the noise caused by the varying operating conditions [43, 58, 59]. Most studies that address this challenge ([60–62]) use a supervised learning model, which is not suitable for aircraft systems with just a few failure instances.

Alternatively, some studies estimate the RUL using a *model-based approach* instead of a data-driven approach [16, 17, 20–22, 26, 63–67]. In a model-based approach, it is assumed that the evolution of the degradation over time follows a certain degradation model. For some components, the degradation in a system is directly measured. With this measured degradation, the parameters of the degradation model are estimated and the degradation is propagated over time to estimate the RUL. For instance, for aircraft fuselage panels, the crack size in the panel directly represents the degradation in the panel [63–65], see Figure 1.4. The growth of the crack size follows Paris’ law, a physical degradation model [63–65]. For other components, the degradation cannot be directly measured. Instead, the sensor measurements are transformed in a health indicator, which represents the degradation in the component. This health indicator in turn is used to estimate the parameters of the degradation model and to estimate the RUL. For instance, for bearings, the Root Mean Square (RMS) of the vibration measurements is often employed as health indicator, while an exponential degradation model is assumed [20, 21].

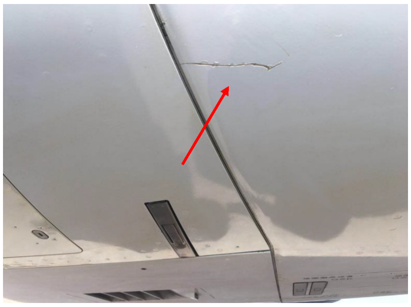


Figure 1.4.: Crack size in the fuselage. This picture is from [68].

For a model-based approach, only few failure instances are needed to estimate the model (hyper)parameters. As such, this approach is very suitable to estimate the RUL of aircraft systems. However, applying this approach to complex aircraft systems is not straightforward. First, many health indicators and degradation models are based on the physics of a system. For complex aircraft systems, however, the physics can be very complicated. The degradation in such a system can be caused by different faults in the many different components of the system, and can often not be measured directly. Moreover, a health indicator with a physical meaning, such as the RMS, and a physical degradation model, such as Paris' law, are often not available for these complex system [60]. The noise in the sensor measurements due to the varying operating conditions further complicates the creation of a health indicator [12]. Finding a reliable health indicator and degradation model for complex aircraft systems therefore remains a formidable challenge.

Last, many studies employ a neural network for estimating the RUL. An additional challenge with this method is that training an accurate neural network is time-consuming. Recently, several improvements have accelerated the training of a neural network. These improvements are both mathematical, such as improved optimization algorithms [69, 70] and new weight initialization strategies [71, 72], and in computer science, such as improved GPUs [73] and new parallelization techniques [74]. However, training deep neural networks (i.e., with many parameters) with large data sets is still time-consuming and requires a large amount of energy [73, 75, 76]. For instance, it is estimated that training the well-known GPT-3 language model with 1,024 A100 GPUs takes 34 days [74]. Though neural networks for estimating the RUL are typically much smaller, the computational time is still large. This complicates the use of neural networks for estimating the RUL.

The first challenge is thus to create an accurate RUL prognostic model for aircraft systems, addressing all difficulties (few failure instances, varying operating conditions, time-consuming to train neural networks) described above. In this thesis, this challenge is addressed in Chapters 2 and 3.

1.1.2. CHALLENGE 2: QUANTIFYING THE UNCERTAINTY OF THE RUL PROGNOSTICS

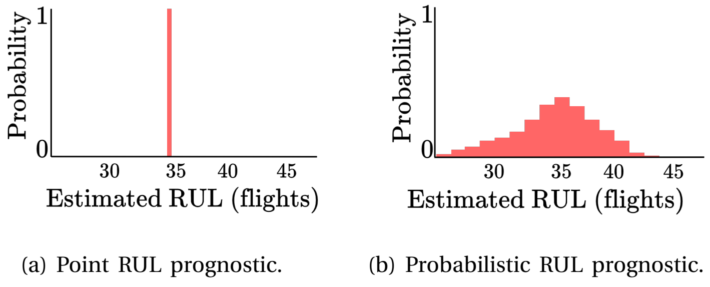


Figure 1.5.: A schematic example of a point and probabilistic RUL prognostic.

Most existing studies estimate *point RUL prognostics* [14, 19, 23–25, 31–34, 36–39, 45, 47–49, 61, 62], i.e., one point is estimated for the RUL, see Figure 1.5a. However, there are several sources of uncertainty for the RUL prognostics. First, the RUL prognostic model itself is often a simplification or approximation of reality. Second, the parameters of the model are uncertain, especially if few (labelled) data samples are available. Uncertainty coming from these two sources is called *epistemic uncertainty* [43, 77, 78]. The uncertainty in the model parameters could be reduced by collecting more (labelled) data samples, but this is often not possible in practice. Other sources of uncertainty are the measurements noise, the unknown future operating conditions (such as the weather, the loading, etc.) and the inherent randomness between the input, the sensor measurements, and the output, the RUL. The uncertainty of these two sources are called *aleatory uncertainty*, and this uncertainty cannot be reduced with more data or more knowledge [43, 77, 78]. RUL prognostics thus always contain some uncertainty.

Some studies therefore not only estimate the RUL, but also try to quantify the uncertainty of the RUL prognostics, for instance by estimating a Probability Density Function (PDF) of the RUL, Figure 1.5b [77]. By quantifying the uncertainty of the RUL prognostics, maintenance planners can make more informed and explainable decisions: The trade-off between using an aircraft system for a longer time (increasing the total lifetime of the system) versus maintaining the aircraft system soon (reducing the risk of failure) is quantified. In this thesis, the RUL prognostics with quantified uncertainty are referred to as *probabilistic RUL prognostics*.

Most studies quantify the uncertainty of the RUL prognostics with a supervised learning neural network, where the RUL is directly estimated from the sensor measurements, see Figure 1.3a. For instance, in [78, 79], a Bayesian neural network is used to estimate a PDF of the RUL directly from the sensor measurements. In [28, 35], a PDF of the RUL is estimated by applying Monte Carlo dropout in a supervised learning neural network. In [80], it is assumed that the PDF of the RUL follows a normal distribution, and a deep Gaussian process is used to estimate the mean and standard deviation of this normal distribution. However, as described in Challenge 1 (Section 1.1.1), supervised learning neural networks are not suitable to estimate the RUL for most aircraft systems with very few failure instances.

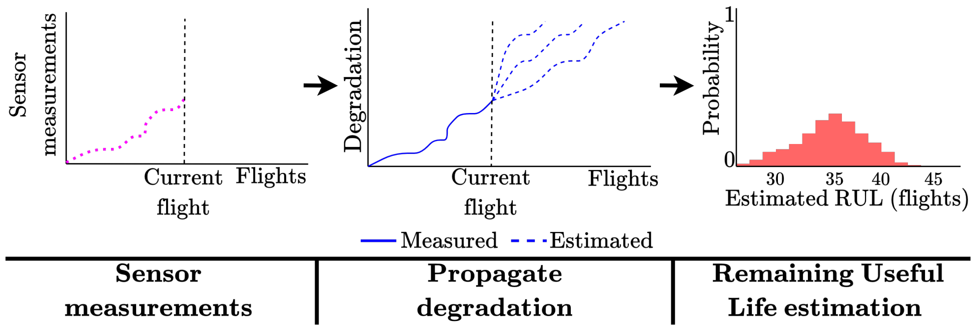


Figure 1.6.: A schematic overview of a model-based approach to estimate probabilistic RUL prognostics.

Other studies therefore estimate the PDF of the RUL using a model-based approach instead of a neural network, as described in Challenge 1 (Section 1.1.1). In a model-based approach, the parameters of a degradation model are estimated with the measured degradation/health indicator. Most methods to estimate these parameters, such as filtering methods [15–17, 20, 21, 63, 64], also quantify the uncertainty of the parameters. With these uncertain parameters, a PDF of the RUL is estimated by propagating the degradation over time along multiple possible paths, each with a corresponding probability, see Figure 1.6. A model-based approach is suitable to develop probabilistic RUL prognostics, as the uncertainty quantification is inherent to most model-based methods. However, applying a model-based approach to estimate the RUL of complex aircraft systems is difficult, see Section 1.1.1.

After quantifying the uncertainty of the RUL prognostics, the quality of the probabilistic RUL prognostics with their quantified uncertainty is assessed. There are several well-known metrics that evaluate the accuracy of point RUL prognostics, such as the Root Mean Square Error (RMSE) or the Mean Absolute Error (MAE). There are also some metrics that evaluate the accuracy of probabilistic RUL prognostics, such as the prognostics horizon, the cumulative relative accuracy and the convergence [81–83]. However, not only the accuracy, but also the sharpness and reliability of the probabilistic RUL prognostics should be assessed. Metrics that measure the sharpness and reliability are crucial for evaluating the trustworthiness of the probabilistic RUL prognostics. There are not yet any commonly accepted metrics, however, for assessing the sharpness and reliability of probabilistic RUL prognostics in the predictive maintenance community.

The second challenge is therefore to generate probabilistic RUL prognostics for aircraft systems with quantified uncertainty. An additional aspect of this challenge is to develop metrics to evaluate these probabilistic RUL prognostics. In this thesis, this challenge is addressed in Chapters 4 and 5.

1.1.3. CHALLENGE 3: OPTIMIZING THE AIRCRAFT MAINTENANCE SCHEDULE WITH RUL PROGNOSTICS

The second step in predictive aircraft maintenance is to optimize the aircraft maintenance schedule with the RUL prognostics, see Figure 1.1. Literature on predictive maintenance usually focuses only on estimating the RUL. Integrating the RUL prognostics in the aircraft maintenance planning is still rare [84–86].

Optimization of the aircraft maintenance schedule is complicated since there are many aspects that need to be integrated in the optimization. Each aircraft contains multiple systems, which in turn contain multiple components. Each of these components have their own individual RUL prognostic. These RUL prognostics are updated over time, as more sensor measurements become available. Due to the aircraft's flight schedule, maintenance can only be scheduled during a few specific moments, called *maintenance opportunities*, see Figure 1.1. Moreover, maintenance is planned for a *fleet* of multiple aircraft. Aircraft are usually maintained in dedicated maintenance hangars, see Figure 1.7. However, there is only limited space in these hangars. Only a few aircraft can therefore be maintained at the same time, i.e., there is limited maintenance capacity. Last, only a limited number of spare components is available to replace components in the fleet of aircraft.

Existing studies on predictive aircraft maintenance planning take some of these aspects into account, but not all of them. In [78, 86, 87], the aircraft predictive

Figure 1.7.: An aircraft in the hangar for maintenance.

maintenance planning is optimized for a single aircraft with one single-component system. In [88], maintenance is optimized for a multi-component system in a single aircraft. Last, in [63, 85, 89, 90], the maintenance planning is optimized for a multi-component system or for multiple systems within a fleet of aircraft, but without considering the spare parts. The objective of these studies is to minimize the expected total costs. These consists of, among others, the cost of corrective maintenance after failure [53, 78, 86, 89, 90], the costs of preventive maintenance before failure [53, 78, 86, 89, 90], the costs of holding inventory [78, 86, 87], and the fixed set-up costs for maintaining an aircraft [63, 85].

Additionally, when integrating probabilistic RUL prognostics in the maintenance schedule, the quantified uncertainty is accordingly included as well. Most studies take this uncertainty into account by first optimizing the initial maintenance planning, using linear programming [85, 86], deep reinforcement learning [89, 90], heuristics [88] or analytical derivations of the optimal maintenance moment [63, 78, 87]. This initial maintenance planning is subsequently updated over time, for instance with a rolling horizon approach [78, 85, 86, 89, 90]. With deep reinforcement learning [89, 90], these future updates of the maintenance planning are already considered when creating the initial maintenance planning. Another method to jointly optimize the initial maintenance planning and the future updates of this planning is stochastic programming, but this method has not yet been used to optimize the predictive aircraft maintenance schedule. In [91, 92], a multi-stage stochastic program is used to schedule maintenance for a generic multi-component system instead. Both reinforcement learning and stochastic programming, however, have one major drawback: The time it takes to find the optimal solution grows exponentially with the size of the problem, which is called the curse of dimensionality [93–95]. This makes optimization under uncertainty a notoriously difficult task [93–95].

Consider, for instance, an aircraft with five aircraft systems, where each system may fail during each flight. After the first flight with these five systems, there are

32 possible outcomes. Here, each outcome represents a different combination of systems that failed during the first flight. Each outcome after the first flight leads to different possible outcomes after the second flight, which gives 243 possible outcomes after flight two. Continuing this gives 59,041 possible outcomes after the tenth flight, and even 2,476,081 possible outcomes after the twentieth flight. Clearly, the number of possible outcomes grows exponentially with the number of flights. When optimizing the initial maintenance planning and the future updates, all these different possible outcomes should be considered. The computational time therefore also grows exponentially with the number of considered aircraft systems and flights, which makes jointly optimizing the initial maintenance schedule and the future updates of this schedule a fundamental problem.

The last challenge is therefore to optimize the maintenance schedule for a fleet of aircraft, with limited hangar space, fixed maintenance opportunities and a limited number of spare parts, while integrating the RUL prognostics and the quantified uncertainty. In this thesis, this challenge is addressed in Chapters 6, 7, 8 and 9.

1.2. APPROACH, OUTLINE, AND SCOPE OF THIS THESIS

The aim of this thesis is to develop an overarching framework for predictive aircraft maintenance. It consists of three parts, following the three challenges, Figure 1.8. First, accurate point RUL prognostics for common aircraft systems are developed (Challenge 1, Chapters 2 and 3). Second, the uncertainty of these RUL prognostics is further quantified, yielding probabilistic RUL prognostics (Challenge 2, Chapters 4 and 5). Third, both the point and probabilistic RUL prognostics are used to optimize the aircraft maintenance planning (Challenge 3, Chapters 6, 7, 8 and 9). In this overarching framework, the RUL prognostic models from the field of data science obtained in Challenge 1 and 2, are combined with optimization methods from the field of operations research in Challenge 3.

Below, the scope of this thesis and each chapter is briefly discussed. All chapters in this thesis are written as scientific papers and can be read independently.

1.2.1. APPROACH FOR CHALLENGE 1: DEVELOPING ACCURATE RUL PROGNOSTICS FOR AIRCRAFT SYSTEMS.

The first challenge of this thesis is to develop accurate point RUL prognostics for common aircraft systems. This challenge is addressed in Chapters 2 and 3.

In Chapter 2, a Long Short-Term Memory (LSTM) autoencoder with local Luong attention is used to estimate the RUL of aircraft engines. This autoencoder, an unsupervised learning neural network, is trained with the unlabelled data samples from non-degraded engines, and is thus suitable when only few failure instances are available. Due to the local Luong attention, the autoencoder accurately encodes and decodes the long time-series of sensor measurements of a full flight. Moreover, the RUL prognostics are robust to the varying operating conditions of the aircraft, since these operating conditions are integrated in various places in the autoencoder.

A good weight initialization strategy accelerates the convergence of the weights in a neural network, and thus reduces the training time [71, 72]. In Chapter 3, a

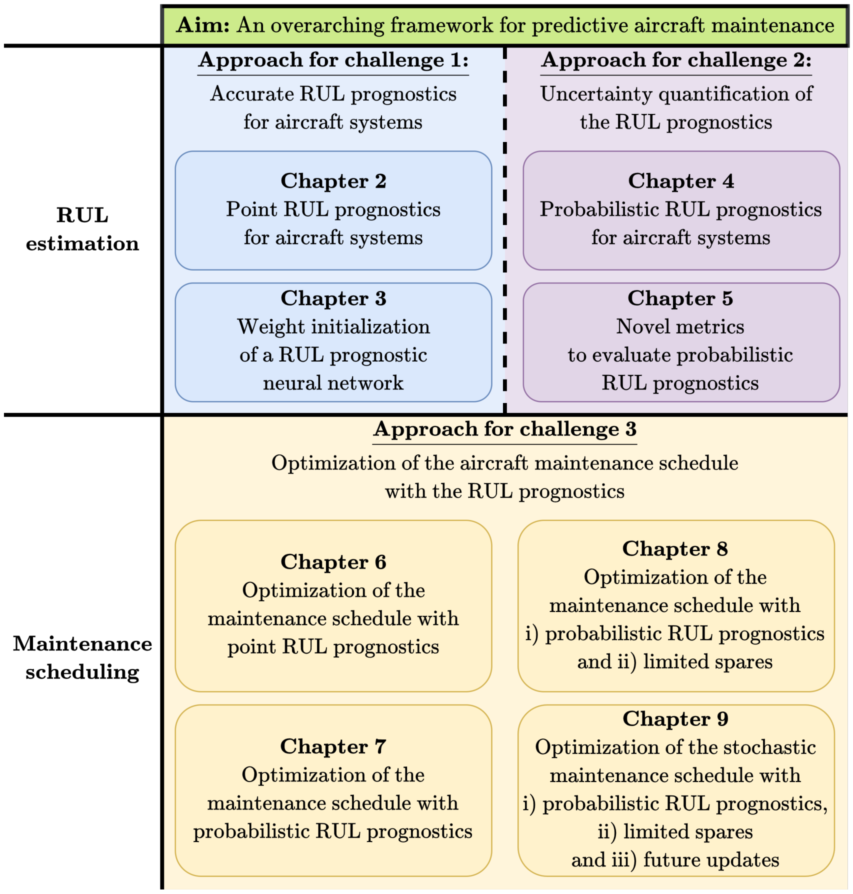

Aim: An overarching framework for predictive aircraft maintenance
RUL estimation
Maintenance scheduling
Approach for challenge 1: Accurate RUL prognostics for aircraft systems
Chapter 2 Point RUL prognostics for aircraft systems
Chapter 3 Weight initialization of a RUL prognostic neural network
Approach for challenge 2: Uncertainty quantification of the RUL prognostics
Chapter 4 Probabilistic RUL prognostics for aircraft systems
Chapter 5 Novel metrics to evaluate probabilistic RUL prognostics
Approach for challenge 3 Optimization of the aircraft maintenance schedule with the RUL prognostics
Chapter 6 Optimization of the maintenance schedule with point RUL prognostics
Chapter 7 Optimization of the maintenance schedule with probabilistic RUL prognostics
Chapter 8 Optimization of the maintenance schedule with i) probabilistic RUL prognostics and ii) limited spares
Chapter 9 Optimization of the stochastic maintenance schedule with i) probabilistic RUL prognostics, ii) limited spares and iii) future updates

Figure 1.8.: Overview of the challenges and related chapters in this thesis.

new weight initialization strategy for the last layer of a neural network is therefore developed. This new weight initialization strategy accelerates the training of a regression neural network that estimates the RUL of aircraft engines. Moreover, combined with transfer learning, it also accelerates the training of common classification neural networks that are used for image recognition.

1.2.2. APPROACH FOR CHALLENGE 2: QUANTIFYING THE UNCERTAINTY OF THE RUL PROGNOSTICS

The second challenge is to quantify the uncertainty of the RUL prognostics for the aircraft systems. This challenge is addressed in Chapters 4 and 5.

In Chapter 4, a model-based approach is combined with a filtering algorithm to estimate a PDF of the RUL of aircraft cooling units. Since no physical degradation model is available for these cooling units, several methods to transform the sensor measurements in a health indicator are first evaluated. Then, two degradation models for the cooling units are identified. A dynamic time-warping clustering method identifies to which of the two models the degradation in a single cooling unit belongs. Last, the RUL of the cooling units is estimated with particle filtering.

In Chapter 5, four new metrics to evaluate the accuracy, sharpness and reliability of probabilistic RUL prognostics, in the form of a PDF, are introduced. The Continuously Ranked Probability Score (CRPS) [96] and the weighted CRPS are used to evaluate the sharpness and accuracy of a single probabilistic RUL prognostic. The α-Coverage and the Reliability Score are introduced to evaluate the reliability of multiple probabilistic RUL prognostics. These two metrics are derived from the reliability diagram [97]. These metrics are used to evaluate the trustworthiness of probabilistic RUL prognostics for aircraft engines.

SCOPE OF CHALLENGES 1 AND 2

There are multiple possible approaches to develop RUL prognostics, and many open challenges to address. To limit the scope of this thesis, a few assumptions and choices regarding the RUL prognostic models were made.

First, only data-driven and model-based approaches are considered for estimating the RUL in this thesis. Though a physical RUL prognostic approach is very promising for some systems (such as aircraft fuselage panels and bearings), no detailed physical degradation model is available for the systems considered in this thesis. Second, the main focus in this thesis is on developing accurate RUL prognostics with limited failure data and highly-varying operating conditions. Since this is already very challenging, some other aspects of RUL prognostic models are not considered, such as the explainability. However, the RUL prognostic methods proposed in Chapters 2 and 4 can be made more explainable with some simple adjustments, as the underlying ideas of these methods are already explainable.

In addition, two main assumptions are made in the RUL prognostic models. First, it is assumed that some data of failure instances are always available. Recently, a few studies broached the subject of fault detection [44, 98] and even RUL estimation [99] if failure data are available, but not labelled. In other words, there are data samples belonging to degraded or failed systems, but these data samples are not labelled as such. However, for safety-critical aircraft systems, there are sometimes no data samples belonging to degraded or failed systems available at all, labelled or unlabelled. Research on estimating the RUL without any failure data, and without a physical approach, is still a new, open research direction. As such, the focus in this thesis is on estimating the RUL with the data of at least a few failure instances, which is already challenging. Second, in this thesis, it is assumed that data samples (labelled or unlabelled) are available. The challenges in the collection of these data samples, such as the storage of sensor data, the type of sensors to install, and privacy issues concerning the sensor data, are not discussed.

1.2.3. Approach for Challenge 3: Optimizing the aircraft maintenance schedule with RUL prognostics

The third and last challenge is to integrate the RUL prognostics in the aircraft maintenance schedule. This challenge is addressed in Chapters 6, 7, 8 and 9.

First, point RUL prognostics are used to optimize the maintenance schedule for a fleet of aircraft in Chapter 6. Here, a maintenance task is only scheduled if the point RUL prognostic falls below an alarm threshold several times in a row, to avoid rescheduling maintenance tasks. Moreover, a safety factor is applied when scheduling the maintenance task, to avoid failures due to errors in the RUL prognostics.

The subsequent chapters instead use probabilistic RUL prognostics to optimize the maintenance schedule. In Chapter 7, the maintenance is scheduled for a single system per aircraft with a linear program. This linear program contains constraints on the limited maintenance capacity and on the limited number of maintenance opportunities. In Chapter 8, this linear program is extended by also considering a limited number of spare components. Moreover, in this chapter, maintenance is scheduled for multi-component aircraft systems with redundant components.

Last, the considered problem is formulated as a multi-stage stochastic linear program with decision-dependent (i.e., endogenous) uncertainty in Chapter 9. By formulating the problem as a stochastic program, the initial maintenance planning and all possible future updates are jointly optimized. This stochastic program is solved to optimality with the Nested Benders decomposition method. A new clustering algorithm, based on the endogenous uncertainty in the problem, is proposed to accelerate this solution method.

Scope of challenge 3

Predictive aircraft maintenance planning is a very broad topic, where many different aspects related to maintenance can be incorporated. The scope of this thesis is limited by considering only a few of these aspects.

First, an aircraft consists of many different components and systems, for which both inspections, maintenance and replacements should be planned. The maintenance tasks for all these different components and systems are often grouped together in the same time period. Second, the maintenance planning is intertwined with other aircraft optimization problems, such as the optimization of the flight schedule and aircraft routes. However, research on combining RUL prognostic models with the optimization of the predictive maintenance planning, for aircraft or other systems, was very limited at the start of this thesis. The focus is therefore on planning maintenance for a singly type of component/system at the time, without considering the other aircraft optimization problems.

Last, the scope of this thesis is limited by testing the proposed methods on three different datasets, two with aircraft engines and one with aircraft cooling units.

REFERENCES

[1] Pigott, P. (2016). *Brace for impact: Air crashes and aviation safety*. Dundurn.

[2] Weerasekera, S. (2020). *Introduction to Maintenance, Repair and Overhaul of aircraft, engines and components*. SAE International.

[3] Ackert, S. P. (2010). *Basics of aircraft maintenance programs for financiers* (tech. rep.). Aircraft Monitor.

[4] *From aircraft health monitoring to aircraft health management* (tech. rep.). (2022). International Air Transport Association (IATA).

[5] Sprong, J. P., Jiang, X., Polinder, H., et al. (2020). Deployment of prognostics to optimize aircraft maintenance – A literature review. *Journal of International Business Research and Marketing, 5*(4), Pages: 26–37.

[6] Ackert, S. (2011). *Engine maintenance concepts for financiers* (tech. rep.). Aircraft Monitor.

[7] Barrera, D. L., Barrera, L., & Barrera, D. L. (2022). *Aircraft maintenance programs*. Springer.

[8] Maintenance Cost Technical Group (MCTG). (2022). *Airline maintenance cost executive commentary (FY2021 data), public version* (tech. rep.). International Air Transport Association (IATA).

[9] Aubin, B. R. (2004). *Aircraft maintenance: The art and science of keeping aircraft safe*. SAE International.

[10] Gerdes, M., Scholz, D., & Galar, D. (2016). Effects of condition-based maintenance on costs caused by unscheduled maintenance of aircraft. *Journal of Quality in Maintenance Engineering, 22*(4), Pages: 394–417.

[11] Van den Bergh, J., De Bruecker, P., Beliën, J., & Peeters, J. (2013). *Aircraft maintenance operations: State of the art* (tech. rep.). KU Leuven.

[12] Lei, Y., Li, N., Guo, L., Li, N., Yan, T., & Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing, 104*, Pages: 799–834.

[13] Jimenez, J. J. M., Schwartz, S., Vingerhoeds, R., Grabot, B., & Salaün, M. (2020). Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics. *Journal of Manufacturing Systems, 56*, Pages: 539–557.

[14] Wei, Y., & Wu, D. (2023). Prediction of state of health and Remaining Useful Life of lithium-ion battery using graph convolutional network with dual attention mechanisms. *Reliability Engineering & System Safety, 230*, Article number: 108947.

[15] Raghavan, N., & Frey, D. D. (2015, June 22-25). Remaining Useful Life estimation for systems subject to multiple degradation mechanisms. *IEEE*

Conference on Prognostics and Health Management (PHM), Austin, Texas, USA, Pages: 1–8.

[16] Dong, G., Chen, Z., Wei, J., & Ling, Q. (2018). Battery health prognosis using Brownian motion modeling and particle filtering. *IEEE Transactions on Industrial Electronics*, *65*(11), Pages: 8646–8655.

[17] An, D., Choi, J.-H., & Kim, N. H. (2013). Prognostics 101: A tutorial for particle filter-based prognostics algorithm using MATLAB. *Reliability Engineering & System Safety*, *115*, Pages: 161–169.

[18] Hu, J., & Chen, P. (2020). Predictive maintenance of systems subject to hard failure based on proportional hazards model. *Reliability Engineering & System Safety*, *196*, Article number: 106707.

[19] Hu, T., Guo, Y., Gu, L., Zhou, Y., Zhang, Z., & Zhou, Z. (2022). Remaining Useful Life estimation of bearings under different working conditions via Wasserstein distance-based weighted domain adaptation. *Reliability Engineering & System Safety*, *224*, Article number: 108526.

[20] Cui, L., Wang, X., Xu, Y., Jiang, H., & Zhou, J. (2019). A novel switching unscented Kalman filter method for Remaining Useful Life prediction of rolling bearing. *Measurement*, *135*, Pages: 678–684.

[21] Li, N., Lei, Y., Lin, J., & Ding, S. X. (2015). An improved exponential model for predicting Remaining Useful Life of rolling element bearings. *IEEE Transactions on Industrial Electronics*, *62*(12), Pages: 7762–7773.

[22] Gebraeel, N. Z., Lawley, M. A., Li, R., & Ryan, J. K. (2005). Residual-life distributions from component degradation signals: A Bayesian approach. *IIE Transactions*, *37*(6), Pages: 543–557.

[23] Cao, Y., Ding, Y., Jia, M., & Tian, R. (2021). A novel temporal Convolutional Network with residual self-attention mechanism for Remaining Useful Life prediction of rolling bearings. *Reliability Engineering & System Safety*, *215*, Article number: 107813.

[24] Li, X., Zhang, W., & Ding, Q. (2019). Deep learning-based Remaining Useful Life estimation of bearings using multi-scale feature extraction. *Reliability Engineering & System Safety*, *182*, Pages: 208–218.

[25] Zhang, J., Jiang, Y., Wu, S., Li, X., Luo, H., & Yin, S. (2022). Prediction of Remaining Useful Life based on bidirectional Gated Recurrent Unit with temporal self-attention mechanism. *Reliability Engineering & System Safety*, *221*, Article number: 108297.

[26] Fan, M., Zeng, Z., Zio, E., Kang, R., & Chen, Y. (2018). A sequential Bayesian approach for Remaining Useful Life prediction of dependent competing failure processes. *IEEE Transactions on Reliability*, *68*(1), Pages: 317–329.

[27] Li, X., Teng, W., Peng, D., Ma, T., Wu, X., & Liu, Y. (2023). Feature fusion model based health indicator construction and self-constraint state-space estimator for Remaining Useful Life prediction of bearings in wind turbines. *Reliability Engineering & System Safety*, *233*, Article number: 109124.

[28] Cao, L., Zhang, H., Meng, Z., & Wang, X. (2023). A parallel GRU with dual-stage attention mechanism model integrating uncertainty quantification for

probabilistic RUL prediction of wind turbine bearings. *Reliability Engineering & System Safety, 235*, Article number: 109197.

[29] Chen, X., Jin, G., Qiu, S., Lu, M., & Yu, D. (2020, October 16-18). Direct Remaining Useful Life estimation based on random forest regression. *Global Reliability and Prognostics and Health Management (PHM) Conference*, Shanghai, China, Pages: 1–7.

[30] Patil, S., Patil, A., Handikherkar, V., Desai, S., Phalle, V. M., & Kazi, F. S. (2018, November 9-15). Remaining Useful Life (RUL) prediction of rolling element bearing using random forest and gradient boosting technique. *Proceedings of the ASME 2018 International Mechanical Engineering Congress and Exposition, 13*, Pittsburgh, Pennsylvania, USA, Pages: 1–7.

[31] Li, X., Ding, Q., & Sun, J.-Q. (2018). Remaining Useful Life estimation in prognostics using deep Convolution Neural Networks. *Reliability Engineering & System Safety, 172*, Pages: 1–11.

[32] Babu, G. S., Zhao, P., & Li, X.-L. (2016, April 16-19). Deep Convolutional Neural Network based regression approach for estimation of Remaining Useful Life. *Proceedings of the 21st International Conference on Database Systems for Advanced Applications (DASFAA)*, Dallas, Texas, USA, Pages: 214–228.

[33] Li, H., Zhao, W., Zhang, Y., & Zio, E. (2020). Remaining Useful Life prediction using multi-scale deep Convolutional Neural Network. *Applied Soft Computing, 89*, Article number: 106113.

[34] Xiang, S., Qin, Y., Zhu, C., Wang, Y., & Chen, H. (2020). Long Short-Term Memory Neural Network with weight amplification and its application into gear Remaining Useful Life prediction. *Engineering Applications of Artificial Intelligence, 91*, Article number: 103587.

[35] Wang, J., Zhang, F., Zhang, J., Liu, W., & Zhou, K. (2023). A flexible RUL prediction method based on poly-cell LSTM with applications to lithium battery data. *Reliability Engineering & System Safety, 231*, Article number: 108976.

[36] Wang, S., Fan, Y., Jin, S., Takyi-Aninakwa, P., & Fernandez, C. (2023). Improved anti-noise adaptive Long Short-Term Memory Neural Network modeling for the robust Remaining Useful Life prediction of lithium-ion batteries. *Reliability Engineering & System Safety, 230*, Article number: 108920.

[37] Chen, J., Jing, H., Chang, Y., & Liu, Q. (2019). Gated Recurrent Unit based Recurrent Neural Network for Remaining Useful Life prediction of nonlinear deterioration process. *Reliability Engineering & System Safety, 185*, Pages: 372–382.

[38] Li, Y., Chen, Y., Shao, H., & Zhang, H. (2023). A novel dual attention mechanism combined with knowledge for Remaining Useful Life prediction based on Gated Recurrent Units. *Reliability Engineering & System Safety, 239*, Article number: 109514.

[39] Xiang, S., Li, P., Huang, Y., Luo, J., & Qin, Y. (2023). Single gated RNN with differential weighted information storage mechanism and its application to machine RUL prediction. *Reliability Engineering & System Safety, 242*, Article number: 109741.

[40] Oyekanlu, E. (2017, December 11-14). Predictive edge computing for time series of industrial iot and large scale critical infrastructure based on open-source software analytic of big data. *IEEE International Conference on Big Data (Big Data)*, Boston, Massachusetts, USA, Pages: 1663–1669.

[41] Weiner, M. (2017). Optimized engine maintenance. *AEROReport, the aviation magazine of MTU Aero Engines.*

[42] Read, B. (2018). Digital takeover. *Royal Aeronautical Society.*

[43] Fink, O., Wang, Q., Svensen, M., Dersin, P., Lee, W.-J., & Ducoffe, M. (2020). Potential, challenges and future directions for deep learning in Prognostics and Health Management applications. *Engineering Applications of Artificial Intelligence*, *92*, Article number: 103678.

[44] Ulmer, M., Zgraggen, J., & Huber, L. G. (2023). A generic machine learning framework for fully-unsupervised anomaly detection with contaminated data. *arXiv preprint arXiv:2308.13352.*

[45] Malhotra, P., TV, V., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). Multi-sensor prognostics using an unsupervised health index based on LSTM encoder-decoder. *arXiv preprint arXiv:1608.06154.*

[46] Ye, Z., & Yu, J. (2021). Health condition monitoring of machines based on Long Short-Term Memory convolutional autoencoder. *Applied Soft Computing, 107,* Article number: 107379.

[47] Gugulothu, N., Tv, V., Malhotra, P., Vig, L., Agarwal, P., & Shroff, G. (2017). Predicting Remaining Useful Life using time series embeddings based on Recurrent Neural Networks. *arXiv preprint arXiv:1709.01073.*

[48] Yu, W., Kim, I. Y., & Mechefske, C. (2019). Remaining Useful Life estimation using a bidirectional Recurrent Neural Network based autoencoder scheme. *Mechanical Systems and Signal Processing, 129*, Pages: 764–780.

[49] Fu, S., Zhong, S., Lin, L., & Zhao, M. (2021). A novel time-series memory auto-encoder with sequentially updated reconstructions for Remaining Useful Life prediction. *IEEE Transactions on Neural Networks and Learning Systems, 33*, Pages: 7114–7125.

[50] Zhai, S., Gehring, B., & Reinhart, G. (2021). Enabling predictive maintenance integrated production scheduling by operation-specific health prognostics with generative deep learning. *Journal of Manufacturing Systems, 61*, Pages: 830–855.

[51] Liu, C., Sun, J., Liu, H., Lei, S., & Hu, X. (2020). Complex engineered system health indexes extraction using low frequency raw time-series data based on deep learning methods. *Measurement, 161*, Article number: 107890.

[52] Tharrault, Y., Mourot, G., & Ragot, J. (2008, June 25-27). Fault detection and isolation with robust principal component analysis. *16th Mediterranean Conference on Control and Automation*, Ajaccio, France, Pages: 59–64.

[53] Nguyen, V. H., & Golinval, J.-C. (2010). Fault detection based on kernel principal component analysis. *Engineering Structures*, *32*(11), Pages: 3683–3691.

[54] Gajjar, S., Kulahci, M., & Palazoglu, A. (2018). Real-time fault detection and diagnosis using sparse principal component analysis. *Journal of Process Control, 67,* Pages: 112–128.

[55] Sarita, K., Devarapalli, R., Kumar, S., Malik, H., Garcia Marquez, F. P., & Rai, P. (2022). Principal component analysis technique for early fault detection. *Journal of Intelligent & Fuzzy Systems, 42*(2), Pages: 861–872.

[56] Vasilev, I. (2019). *Advanced deep learning with Python: Design and implement advanced next-generation AI solutions using Tensorflow and PyTorch.* Packt Publishing Ltd.

[57] Li, H. (2016). Accurate and efficient classification based on common principal components analysis for multivariate time series. *Neurocomputing, 171*, Pages: 744–753.

[58] Ochella, S., Shafiee, M., & Dinmohammadi, F. (2022). Artificial intelligence in Prognostics and Health Management of engineering systems. *Engineering Applications of Artificial Intelligence, 108*, Article number: 104552.

[59] Koutroulis, G., Mutlu, B., & Kern, R. (2022). Constructing robust health indicators from complex engineered systems via anticausal learning. *Engineering Applications of Artificial Intelligence, 113*, Article number: 104926.

[60] Chao, M. A., Kulkarni, C., Goebel, K., & Fink, O. (2022). Fusing physics-based and deep learning models for prognostics. *Reliability Engineering & System Safety, 217*, Article number: 107961.

[61] Zhu, J., Chen, N., & Shen, C. (2020). A new data-driven transferable Remaining Useful Life prediction approach for bearing under different working conditions. *Mechanical Systems and Signal Processing, 139*, Article number: 106602.

[62] Cao, Y., Jia, M., Ding, P., & Ding, Y. (2021). Transfer learning for Remaining Useful Life prediction of multi-conditions bearings based on bidirectional-GRU network. *Measurement, 178*, Article number: 109287.

[63] Yiwei, W., Christian, G., Binaud, N., Christian, B., Haftka, R. T., et al. (2017). A cost driven predictive maintenance policy for structural airframe maintenance. *Chinese Journal of Aeronautics, 30*(3), Pages: 1242–1257.

[64] Wang, Y., Gogu, C., Binaud, N., & Bes, C. (2015, September 7-10). Predicting Remaining Useful Life by fusing SHM data based on extended Kalman filter. *Proceedings of the 25th European Safety and Reliability conference*, Zurich, Switzerland, Pages: 7–10.

[65] Karandikar, J. M., Kim, N. H., & Schmitz, T. L. (2012). Prediction of Remaining Useful Life for fatigue-damaged structures using Bayesian inference. *Engineering Fracture Mechanics, 96*, Pages: 588–605.

[66] Lim, P., Goh, C. K., Tan, K. C., & Dutta, P. (2015). Multimodal degradation prognostics based on switching Kalman filter ensemble. *IEEE Transactions on Neural Networks and Learning Systems, 28*(1), Pages: 136–148.

[67] Li, N., Gebraeel, N., Lei, Y., Fang, X., Cai, X., & Yan, T. (2021). Remaining Useful Life prediction based on a multi-sensor data fusion model. *Reliability Engineering & System Safety, 208*, Article number: 107249.

[68] Li, Y., Han, Z., Xu, H., Liu, L., Li, X., & Zhang, K. (2019). YOLOv3-lite: A lightweight crack detection network for aircraft structure based on depthwise separable convolutions. *Applied Sciences, 9*(18), Article number: 3781.

[69] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

[70] Xie, X., Pu, Y.-F., & Wang, J. (2023). A fractional gradient descent algorithm robust to the initial weights of multilayer perceptron. *Neural Networks, 158*, Pages: 154–170.

[71] Glorot, X., & Bengio, Y. (2010, May 13-15). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, *9*, Sardinia, Italy, Pages: 249–256.

[72] He, K., Zhang, X., Ren, S., & Sun, J. (2015, December 7-13). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, Pages: 1026–1034.

[73] Thompson, N. C., Greenewald, K., Lee, K., & Manso, G. F. (2020). The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*.

[74] Narayanan, D., Shoeybi, M., Casper, J., LeGresley, P., Patwary, M., Korthikanti, V., Vainbrand, D., Kashinkunti, P., Bernauer, J., Catanzaro, B., et al. (2021, November 14-19). Efficient large-scale language model training on GPU clusters using megatron-LM. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, St. Louis, Missouri, USA, Pages: 1–15.

[75] García-Martín, E., Rodrigues, C. F., Riley, G., & Grahn, H. (2019). Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing, 134*, Pages: 75–88.

[76] Desislavov, R., Martínez-Plumed, F., & Hernández-Orallo, J. (2021). Compute and energy consumption trends in deep learning inference. *arXiv preprint arXiv:2109.05472*.

[77] Nemani, V., Biggio, L., Huan, X., Hu, Z., Fink, O., Tran, A., Wang, Y., Zhang, X., & Hu, C. (2023). Uncertainty quantification in machine learning for engineering design and health prognostics: A tutorial. *Mechanical Systems and Signal Processing, 205*, Article number: 110796.

[78] Zhuang, L., Xu, A., & Wang, X.-L. (2023). A prognostic driven predictive maintenance framework based on Bayesian deep learning. *Reliability Engineering & System Safety, 234*, Article number: 109181.

[79] Peng, W., Ye, Z.-S., & Chen, N. (2019). Bayesian deep-learning-based health prognostics toward prognostics uncertainty. *IEEE Transactions on Industrial Electronics*, *67*(3), Pages: 2283–2293.

[80] Biggio, L., Wieland, A., Chao, M. A., Kastanis, I., & Fink, O. (2021). Uncertainty-aware prognosis via deep Gaussian process. *IEEE Access*, *9*, Pages: 123517–123527.

[81] Saxena, A., Celaya, J., Balaban, E., Goebel, K., Saha, B., Saha, S., & Schwabacher, M. (2008, October 6-9). Metrics for evaluating performance of prognostic techniques. *International Conference on Prognostics and Health Management*, Denver, Colorado, USA, Pages: 1–17.

[82] Saxena, A., Celaya, J., Saha, B., Saha, S., & Goebel, K. (2009, March 7-14). Evaluating algorithm performance metrics tailored for prognostics. *IEEE Aerospace conference*, Big Sky, Montana, USA, Pages: 1–13.

[83] Lall, P., Lowe, R., & Goebel, K. (2011, April 18-20). Prognostics and health monitoring of electronic systems. *12th International Conference on Thermal, Mechanical & Multi-Physics Simulation and Experiments in Microelectronics and Microsystems*, Linz, Austria, Pages: 1–17.

[84] Mitici, M., de Pater, I., Barros, A., & Zeng, Z. (2023). Dynamic predictive maintenance for multiple components using data-driven probabilistic RUL prognostics: The case of turbofan engines. *Reliability Engineering & System Safety, 234*, Article number: 109199.

[85] Zeng, J., & Liang, Z. (2023). A dynamic predictive maintenance approach using probabilistic deep learning for a fleet of multi-component systems. *Reliability Engineering & System Safety, 238*, Article number: 109456.

[86] Shoorkand, H. D., Nourelfath, M., & Hajji, A. (2023). A hybrid CNN-LSTM model for joint optimization of production and imperfect predictive maintenance planning. *Reliability Engineering & System Safety, 241,* Article number: 109707.

[87] Nguyen, K. T., & Medjaher, K. (2019). A new dynamic predictive maintenance framework using deep learning for failure prognostics. *Reliability Engineering & System Safety, 188,* Pages: 251–262.

[88] Zhou, K.-L., Cheng, D.-J., Zhang, H.-B., Hu, Z.-t., & Zhang, C.-Y. (2023). Deep learning-based intelligent multilevel predictive maintenance framework considering comprehensive cost. *Reliability Engineering & System Safety, 237,* Article number: 109357.

[89] Lee, J., & Mitici, M. (2022). Deep reinforcement learning for predictive aircraft maintenance using probabilistic Remaining-Useful-Life prognostics. *Reliability Engineering & System Safety, 230*, Article number: 108908.

[90] Tseremoglou, I., & Santos, B. F. (2024). Condition-based maintenance scheduling of an aircraft fleet under partial observability: A deep reinforcement learning approach. *Reliability Engineering & System Safety, 241,* Article number: 109582.

[91] Zhu, Z., & Xiang, Y. (2021). Condition-based maintenance for multi-component systems: Modeling, structural properties, and algorithms. *IISE transactions, 53*(1), Pages: 88–100.

[92] Zhu, Z., Xiang, Y., & Zeng, B. (2021). Multicomponent maintenance optimization: A stochastic programming approach. *INFORMS Journal on Computing, 33*(3), Pages: 898–914.

[93] Sahinidis, N. V. (2004). Optimization under uncertainty: State-of-the-art and opportunities. *Computers & Chemical Engineering, 28*(6-7), Pages: 971–983.

[94] Powell, W. B. (2019). A unified framework for stochastic optimization. *European Journal of Operational Research, 275*(3), Pages: 795–821.

[95] Wiering, M. A., & Van Otterlo, M. (2012). *Reinforcement learning: State-of-the-art.* Springer.

[96] Gneiting, T., & Katzfuss, M. (2014). Probabilistic forecasting. *Annual Review of Statistics and Its Application, 1,* Pages: 125–151.

[97] Zadrozny, B., & Elkan, C. (2002, July 23-26). Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 8,* Edmonton, Alberta, Canada, Pages: 694–699.

[98] Munir, M., Siddiqui, S. A., Dengel, A., & Ahmed, S. (2018). DeepAnT: A deep learning approach for unsupervised anomaly detection in time series. *IEEE Access*, *7*, Pages: 1991–2005.

[99] Calabrese, F., Regattieri, A., Botti, L., Mora, C., & Galizia, F. G. (2020). Unsupervised fault detection and prediction of Remaining Useful Life for online prognostic health management of mechanical systems. *Applied Sciences*, *10*(12), Article number: 4120.

2

HEALTH INDICATORS AND POINT RUL PROGNOSTICS WITH A LSTM AUTOENCODER

Most health indicators and Remaining Useful Life (RUL) prognostics are obtained with supervised learning models. These models must be trained with many labelled data samples (i.e., the true RUL is known). Aircraft systems, however, are often preventively replaced before failure, in which case the true RUL remains unknown. There are thus very few labelled data samples available. Moreover, aircraft fly under highly-varying operating conditions that influence the sensor measurements.

In this chapter, we therefore introduce a Long Short-Term Memory (LSTM) autoencoder to develop a health indicator for an aircraft system instead. An autoencoder is an unsupervised learning model which can be trained with unlabelled data samples (i.e., the true RUL is unknown). The operating conditions of the aircraft are integrated in the autoencoder as well. These health indicators are further used to estimate the RUL of the aircraft system with a similarity-based matching approach.

We illustrate our approach for turbofan engines. In the case study, we show that the consideration of the operating conditions leads to robust health indicators. Moreover, we highlight the benefits of using an unsupervised learning model instead of a supervised learning model when few labelled data samples are available.

Parts of this chapter have been published in:

de Pater, I., & Mitici, M. (2023). Developing health indicators and RUL prognostics for systems with few failure instances and varying operating conditions using a LSTM autoencoder. *Engineering Applications of Artificial Intelligence, 117,* Article number: 105582

de Pater, I., & Mitici, M. (2023, September 3-7). Constructing health indicators for systems with few failure instances using unsupervised learning. *Proceedings of the 33st European Safety and Reliability Conference*, Southampton, UK, Pages: 3066–3073

2.1. Introduction

Complex technical systems are crucial for the safe and reliable operation of machines, vehicles, plants, etc. The unexpected failures of such systems lead to costly unplanned downtime and potential safety risks. To limit the number of failures, systems are often replaced preventively [3, 4]. However, due to such preventive, early replacements, the actual failure time of the system is unobserved. This complicates the estimation of the Remaining Useful Life (RUL).

Especially in aviation, preventive replacement of complex, safety-critical systems is common. Replacing these systems early is preferred over keeping them running for a long time and risking a failure. Consequently, the data-monitoring samples from such systems are often unlabelled, i.e., the corresponding true RUL is unknown. In the rare case when such a system does fail during operation, the failure time is observed and the data-monitoring samples coming from this system are labelled (i.e., the true RUL is known) [5]. This mix of very few labelled data samples, but many unlabelled data samples is often seen for complex aircraft systems.

Common RUL prognostics models are Convolutional Neural Networks [6–8] and Long Short-Term Memory (LSTM) neural networks [9], which directly estimate the RUL. However, such supervised learning methods require the availability of many labelled data samples to train accurate prognostic models. This makes these supervised learning approaches unsuitable for safety-critical aircraft systems.

Instead, accurate RUL prognostics can be obtained by first developing a health indicator using the unlabelled data samples and signal reconstruction, i.e., an autoencoder learns the normal system behaviour from the unlabelled data samples [10, 11]. This autoencoder is then used to detect deviations from the normal system behaviour that emerge due to increasing degradation [10]. This approach has been considered in, for instance, [11, 12]. In [11], a health indicator is developed using the reconstruction error of a LSTM autoencoder and a linear regression model. Similarly, in [12], a health indicator is obtained based on the reconstruction errors of a LSTM autoencoder and a Gaussian distribution. In contrast, in [13–15] and in [16], the embeddings of a recurrent autoencoder and a conditional variational autoencoder, respectively, are used to develop a health indicator. In [17], a health indicator is developed with both the reconstruction error and the embeddings of a LSTM autoencoder. With such health indicators, accurate RUL prognostics can be obtained with just a few labelled data samples.

However, the autoencoders mentioned above cannot be directly applied to develop health indicators for aircraft systems. First, health-monitoring sensor measurements are recorded at a high frequency during a flight. Moreover, the flight itself is several hours long. As such, the time-series of measurements of each flight is long. For the case study presented in this chapter, the aircraft system performs many flights, each containing 60-303 time-steps. In contrast, existing studies consider autoencoders that use fixed-length data samples of 5 [14] to 40 [17] time-steps only.

Also, the conditions under which a system operates are expected to influence the degradation of the system [18]. This is especially the case for aircraft, where the operating conditions vary due to weather conditions, flying routes, flying altitudes etc. [19]. One of the major open challenges for Prognostics and Health Management

(PHM), in aviation and in other application domains, is to develop health indicators and RUL prognostics that are robust to varying operating conditions [3, 4, 10].

When proposing health indicators using an autoencoder, only one study accounts for the operating regime: A high-level cluster of similar operating conditions [16]. The operating conditions of aircraft, however, are highly-varying during each flight. For example, the altitude and speed of an aircraft continuously change during a flight. Considering only a few aggregated clusters of the operating conditions, as in [16], might lead to a loss of information [10].

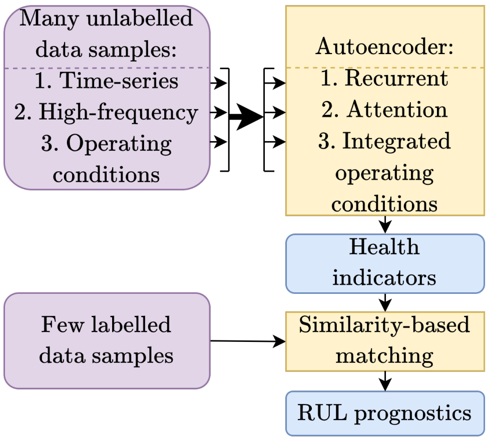


Figure 2.1.: Schematic overview of the consider approach.

To develop a health indicator for systems with high-frequency measurements and highly-varying operating conditions, we propose to use the reconstruction error of a LSTM autoencoder (LSTM-AE) with i) attention and ii) integrated operating conditions. LSTM neural networks are well suited to process varying-length time-series, while avoiding the vanishing gradient problem [20]. However, a standard recurrent autoencoder cannot reconstruct long time-series of sensor data well: The final embedding of the autoencoder cannot contain all relevant features of a long input sample. Moreover, the final embedding contains more information about the last sensor measurements than about the first sensor measurements of the flight [20]. In language-related application with long sentences, these problems have been successfully elevated by implementing attention, giving state-of-the-art results [10]. Inspired by this, we also apply attention in the LSTM-AE.

To develop health indicators robust to the highly-varying operating conditions, we input the operating conditions in the autoencoder. These operating conditions are not encoded and then reconstructed, but merely used “informatively”, i.e., the LSTM-AE is informed on the aircraft operating conditions solely to assist in encoding and reconstructing the sensor data samples. In contrast with [16], no information on the operating conditions is thus lost by clustering the operating conditions.

We apply our methodology to develop health indicators and RUL prognostics for the aircraft turbofan engines of the new N-CMAPSS dataset [21]. An overview of the considered approach is in Figure 2.1. The obtained health indicators have a high monotonicity (0.38), trendability (0.95) and prognosability (0.94). We show that the monotonicity of the health indicators decreases with 97% when the operating conditions are not considered in the LSTM-AE. Also, the monotonicity decreases by 11% when no attention is applied in the LSTM-AE.

Having the health indicators, we divide the lifetime of each engine in a healthy and an unhealthy stage. Last, we estimate the RUL of the engines in the unhealthy stage with a similarity-based matching method, using the health indicators and the few available labelled data samples [11, 22, 23]. Due to the high monotonicity, trendability and prognosability of the health indicators, the overall RMSE of these RUL prognostics in the unhealthy stage is only 2.67 flights.

The main contributions of this chapter are:

1. Safety-critical systems are rarely operated until failure, but are rather replaced preventively, and thus have very few failures. We propose an unsupervised learning approach to construct a health indicator and to estimate the RUL for such systems with very few labelled data samples, i.e., very few failures. We show that our approach outperforms existing supervised learning methods for the considered system. Specifically, the RMSE of the RUL prognostics is 19% lower compared to existing supervised learning methods.

2. We develop a health indicator by integrating the highly-varying operating conditions of the system in a LSTM autoencoder. This makes the health indicators robust to these operating conditions and improves their monotonicity, trendability and prognosability significantly. Moreover, the obtained health indicators have a high trendability, even when the operating conditions in the test set and in the training set differ significantly.

3. We use attention in the LSTM autoencoder to handle the high-frequency measurements gathered during the long flights of the considered system. We show that using attention improves the monotonicity of the health indicators by 11%. This is particularly relevant for novel technical systems whose health is monitored continuously and at a high-frequency.

The remainder of this chapter is organized as follows. We first introduce the proposed methodology to construct the health indicators in Section 2.2. Then, we introduce the considered N-CMAPSS dataset in Section 2.3, and present the resulting health indicators in Section 2.4. Last, we introduce the similarity-based matching approach to develop RUL prognostics in Section 2.5, and analyse the RUL prognostics in Section 2.6. The conclusions are provided in Section 2.7.

2.2. METHODOLOGY - HEALTH INDICATORS WITH A LSTM AUTOENCODER

In Section 2.2.1, we introduce the LSTM autoencoder (LSTM-AE) with attention and integrated operating conditions. In Section 2.2.2, we use the reconstruction errors from this autoencoder to construct a health indicator.

2.2.1. LSTM-AE WITH LOCAL LUONG ATTENTION

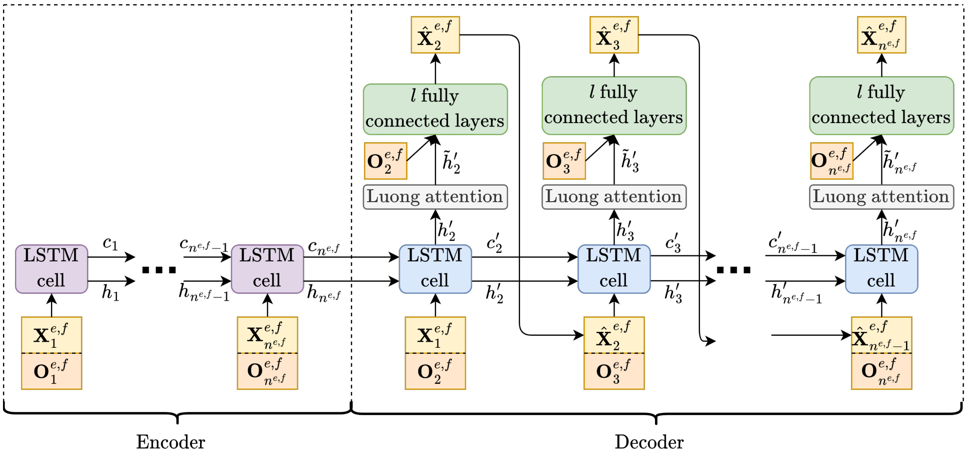


Figure 2.2.: A schematic overview of the considered LSTM-AE with informative operating conditions.

In this section, we introduce the Long Short-Term Memory autoencoder (LSTM-AE). Let $\mathbf{X}^{e,f} = \left\{\mathbf{X}_t^{e,f}, t \in \{1, 2, \ldots, n^{e,f}\}\right\}$ be the multi-sensor measurements of an aircraft system e during a flight f, with $n^{e,f}$ the number of multi-sensor measurements of flight f. Here, $\mathbf{X}_t^{e,f} = \left[X_t^{e,f,1}, X_t^{e,f,2}, \ldots, X_t^{e,f,m^{\mathrm{s}}}\right]$ is the t^{th} multi-sensor measurement of this flight f, with m^{s} the number of considered sensors. The LSTM-AE first consists of an encoder, that maps the multi-sensor measurements $\mathbf{X}^{e,f}$ to an embedding with a smaller dimension (i.e., encodes), and then a decoder, that reconstructs the measurements from this embedding. The objective of the LSTM-AE is to minimize the total absolute reconstruction error $\mathcal{L}$ of each flight:

$$\mathcal{L}^{e,f} = \sum_{t=2}^{n^{e,f}} \left|\hat{\mathbf{X}}_t^{e,f} - \mathbf{X}_t^{e,f}\right|, \tag{2.1}$$

with $\hat{\mathbf{X}}_t^{e,f}$ the reconstructed sensor measurements $\mathbf{X}_t^{e,f}$ at time-step t of system e during flight f, i.e., the output of the LSTM-AE. We train this LSTM-AE solely with the unlabelled sensor data samples from a just-installed aircraft system, i.e., when the system is still considered healthy.

Beside the multi-sensor measurements, also the operating conditions during each flight are available. Let $\mathbf{O}^{e,f} = \left\{\mathbf{O}_t^{e,f}, t \in \{1, 2, \ldots, n^{e,f}\}\right\}$ be the operating conditions during flight f with system e. Here, $\mathbf{O}_t^{e,f} = \left[O_t^{e,f,1}, O_t^{e,f,2}, \ldots, O_t^{e,f,m^{\mathrm{o}}}\right]$ denotes the operating conditions at time-step t during this flight, and m^{o} is the number of operating conditions. The operating conditions are used as input for both the encoder and the decoder. But in contrast with the sensor measurements, the operating conditions are not encoded and then reconstructed, but merely used "informatively": They assist in encoding and decoding the sensor measurements. A schematic overview of the considered LSTM-AE is in Figure 2.2.

Encoder

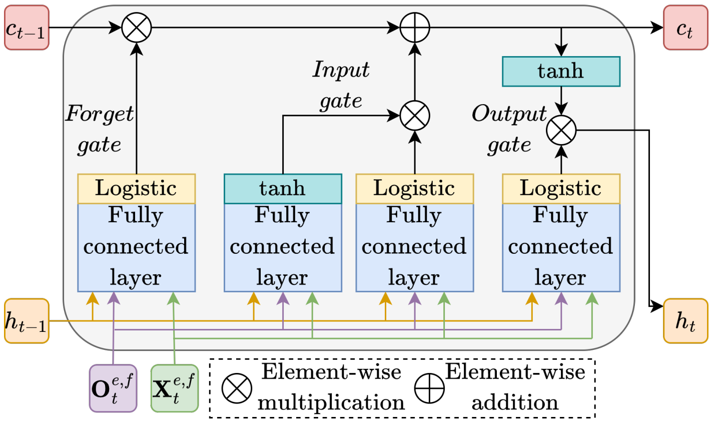


Figure 2.3.: A schematic overview of a LSTM-cell with informative operating conditions.

At each time step t during a flight f, the encoder encodes the multi-sensor measurement $\mathbf{X}_t^{e,f}$ to the short-term state h_t, which has a smaller dimension. The encoder consists of $n^{e,f}$ LSTM-cells for this flight f [24, 25]. At time step t, we consider as input to the LSTM-cell i) the long-term state c_{t-1} and the short-term state h_{t-1} of previous time-step $t-1$, ii) the multi-sensor measurement $\mathbf{X}_t^{e,f}$, and iii) the operating conditions $\mathbf{O}_t^{e,f}$. Each LSTM-cell consists of 3 gates (see Figure 2.3):

The forget gate in the LSTM-cell determines which part of the long-term state is (partly) erased [20, 26]:

$$g_t = \sigma\left(W_g\mathbf{X}_t^{e,f} + V_g\mathbf{O}_t^{e,f} + U_g h_{t-1} + b_g\right). \tag{2.2}$$

Here, W_g, V_g and U_g are the weight matrices connecting $\mathbf{X}_t^{e,f}, \mathbf{O}_t^{e,f}$ and h_{t-1} to the output g_t, respectively, and b_g denotes the bias of this forget gate. Last, $\sigma()$ denotes the logistic activation function used in the forget gate.

The input gate first proposes a new candidate long-term state c_t^{can} [20, 26]:

$$c_t^{\text{can}} = \tanh\left(W_c\mathbf{X}_t^{e,f} + V_c\mathbf{O}_t^{e,f} + U_c h_{t-1} + b_c\right), \tag{2.3}$$

where W_c, V_c and U_c are the weight matrices connecting $\mathbf{X}_t^{e,f}, \mathbf{O}_t^{e,f}$ and h_{t-1} to the output c_t^{can} respectively, and b_c is the bias of this input gate. A tanh (hyperbolic tangent) activation function is considered. Next, the input gate determines which parts of the candidate long-term state are added to the new long-term state [20, 26]:

$$i_t = \sigma\left(W_i \mathbf{X}_t^{e,f} + V_i \mathbf{O}_t^{e,f} + U_i h_{t-1} + b_i\right). \tag{2.4}$$

Again, W_i, V_i and U_i are the weight matrices connecting $\mathbf{X}_t^{e,f}, \mathbf{O}_t^{e,f}$ and h_{t-1} to the output i_t respectively, and b_i is the bias of this gate.

Last, the long-term state, with the output of the forget and input gate, is updated as follows [20, 26]:

$$c_t = \left(c_{t-1} \otimes g_t\right) \oplus \left(i_t \otimes c_t^{\text{can}}\right), \tag{2.5}$$

where $\oplus$ denotes the element-wise addition operator and $\otimes$ denotes the element-wise multiplication operator.

The output gate constructs the short-term state h_t, by first determining which parts of the long-term state c_t are transferred to the short-term state [20, 26]:

$$p_t = \sigma\left(W_p \mathbf{X}_t^{e,f} + V_p \mathbf{O}_t^{e,f} + U_p h_{t-1} + b_p\right). \tag{2.6}$$

Here, W_p, V_p and U_p are the weight matrices connecting $\mathbf{X}_t^{e,f}, \mathbf{O}_t^{e,f}$ and h_{t-1} to the output p_t respectively, and b_p is the bias of this layer. The new short-term state h_t is now constructed as [20, 26]:

$$h_t = p_t \otimes \tanh(c_t). \tag{2.7}$$

DECODER

The decoder reconstructs the sensor measurements $\mathbf{X}_t^{e,f}, t \in \{2, \ldots, n^{e,f}\}$ using the short-term states from the encoder. We first obtain at each time-step t of flight f the short-term state h'_t of the decoder with a LSTM-cell. Next, local Luong attention is used to update the short-term state h'_t to the augmented short-term state $\tilde{h}'_t$. Last, we input the augmented short-term state together with the operating conditions at time-step t to a fully connected neural network. This network outputs the reconstructed sensor measurements $\hat{\mathbf{X}}_t^{e,f}$ (see Figure 2.2).

Recurrent layer The first layer of the decoder consists of $n^{e,f} - 1$ LSTM-cells. At time-step t of flight f, we consider the decoder short-term state h'_{t-1} and the decoder long-term state c'_{t-1} of time-step $t-1$ as input to the LSTM-cell. If $t = 2$, we consider the last short-term state $h_{n^{e,f}}$ and long-term state $c_{n^{e,f}}$ of the encoder as input instead. Moreover, we input the previous reconstructed sensor measurements $\hat{\mathbf{X}}_{t-1}^{e,f}$ during the testing phase. During the training phase, we use teacher forcing instead [20], i.e., we input the true sensor measurements $\mathbf{X}_{t-1}^{e,f}$. If $t = 2$, we always input the true sensor measurements from time-step $t = 1$. Last, we use the operating conditions $\mathbf{O}_t^{e,f}$ of time-step t as input, to assist in decoding the sensor measurements. The output of the LSTM-cell is the decoder short-term state h'_t and the decoder long-term state c'_t.

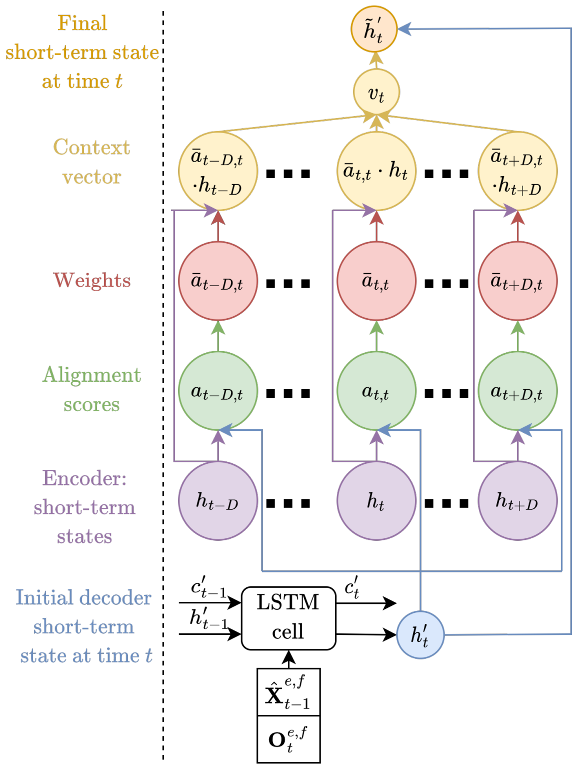

Final short-term state at time t
Context vector
Weights
Alignment scores
Encoder: short-term states
Initial decoder short-term state at time t
$\tilde{h}'_t$
v_t
$\bar{a}_{t-D,t} \cdot h_{t-D}$
$\bar{a}_{t,t} \cdot h_t$
$\bar{a}_{t+D,t} \cdot h_{t+D}$
$\bar{a}_{t-D,t}$
$\bar{a}_{t,t}$
$\bar{a}_{t+D,t}$
$a_{t-D,t}$
$a_{t,t}$
$a_{t+D,t}$
h_{t-D}
h_t
h_{t+D}
c'_{t-1}
h'_{t-1}
LSTM cell
c'_t
h'_t
$\hat{\mathbf{X}}^{e,f}_{t-1}$
$\mathbf{O}^{e,f}_t$

Figure 2.4.: Schematic overview of the local Luong attention mechanism.

Local Luong attention The dimension of the last encoder short-term state $h_{n^{e,f}}$ is too small to contain all relevant features of a long flight. Moreover, the last encoder hidden state contains more information about the last sensor measurements than about the first sensor measurements of the flight [20]. We therefore use local Luong attention [27] to update the decoder short-term states with all the encoder short-term states $h_t, t \in \{1, 2, \ldots, n^{e,f}\}$. A schematic overview of the local Luong attention mechanism is in Figure 2.4.

First, we compute how well the initial short-term state h'_t of the decoder aligns with the encoder short-term states $h_j, j \in \{t-D, t-D+1, \ldots, t+D\}$. Here, D is the window size of the local attention mechanism. The alignment score $a_{j,t}$ between the decoder short-term state h'_t and the encoder short-term state h_j is [27]:

$$a_{j,t} = h_t'^T W_a h_j,\ j \in \{t-D, \ldots, t+D\}. \tag{2.8}$$

Here, W_a is the weight matrix belonging to the attention mechanism, and T denotes the transpose. With these alignment scores, the weights $\bar{a}_{j,t}$ are derived with the softmax function as follows [27]:

$$\bar{a}_{j,t} = \frac{e^{a_{j,t}}}{\sum_{k=t-D}^{t+D} e^{a_{k,t}}},\ j \in \{t-D,\dots,t+D\}. \tag{2.9}$$

Next, the weights are used to derive the context vector v_t [27]:

$$v_t = \sum_{j=t-D}^{t+D} \bar{a}_{j,t} h_j \tag{2.10}$$

With this context vector, the decoder short-term state is updated with one fully connected layer as follows [27]:

$$\tilde{h}'_t = \tanh\left(W_h \left[v_t, h'_t\right]\right), \tag{2.11}$$

with W_h a weight matrix belonging to this fully connected layer.

Fully connected layers Last, we reconstruct the sensor measurements at time-step t with l fully connected layers. As input, we use both the augmented short-term state $\tilde{h}'_t$ and the operating conditions $\mathbf{O}_t^{e,f}$ at time-step t. By adding the operating conditions as input, we ensure that it is not useful for the augmented short-term states $\tilde{h}'_t$ to contain any information on the current operating conditions. We thus truly aim to encode and decode only the sensor measurements. Here, the first $l-1$ layers apply the tanh activation function. The last layer has a linear activation function instead, and contains m^s nodes.

2.2.2. Constructing a health indicator with the reconstruction errors of the LSTM-AE

We train the LSTM-AE only with the unlabelled sensor data samples of just-installed aircraft systems, i.e., from systems that are considered healthy. We therefore expect that the reconstruction errors increase when a system degrades over time [11]. These reconstruction errors of the trained LSTM-AE are used to derive a health indicator.

Let $\mathcal{L}_f^{e,s}$ denote the mean reconstruction loss of a sensor s during flight f performed by a system e:

$$\mathcal{L}_f^{e,s} = \frac{1}{n^{e,f}-1} \sum_{t=2}^{n^{e,f}} \left| \hat{X}_t^{e,f,s} - X_t^{e,f,s} \right|, \tag{2.12}$$

with $\hat{X}_t^{e,f,s}$ the t^{th} reconstructed measurement of sensor s during flight f performed by system e. Let $\mathcal{L}^{e,s} = \{\mathcal{L}_f^{e,s}, f \in \{1,2,\dots,F^e\}\}$ be the time-series of the reconstruction loss for a sensor s that monitors system e. Then, we define $\lambda^e = \{\lambda_f^e, f \in \{1,2,\dots,F^e\}\}$, the cumulative reconstruction error, as the health indicator of an engine e, with

$$\lambda_f^e = \sum_{s=1}^{m^s} \mathcal{L}_f^{e,s}. \tag{2.13}$$

2.3. Case study - aircraft engines

In this section, we first describe the considered data set in Section 2.3.1. Then, we describe the preprocessing of the data in Section 2.3.2, while the dataset is illustrated in Section 2.3.3. Last, we introduce the three metrics that are used to evaluate the health indicators in Section 2.3.4.

2.3.1. Aircraft engines in the N-CMAPSS data set

We consider dataset DS02 of the *new* N-CMAPSS data set [21]. Here, the degradation of aircraft turbofan engines is simulated with the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) model of NASA [21]. There are some key differences between this new data set and the previous C-MAPSS data set [28].

First, dataset DS02 of N-CMAPSS contains a limited number of engines: The training set contains only 6 engines. The test set contains 3 engines, namely engine 11, 14 and 15. For each engine e in the training and the test set of DS02, sensor measurements are available during each flight f from engine installation until engine failure (i.e., run-to-failure instances). Let F^e denote the number of flights performed by engine e. Besides the sensor measurements, N-CMAPSS also contains the operating conditions of the flights of the aircraft. A total of $m^{\mathrm{o}} = 4$ operating conditions are available: The altitude of the aircraft (alt), the flight Mach number (Mach), the throttle-resolver angle (TRA) and the total temperature at the fan inlet (T2). Last, N-CMAPSS contains high-frequency sensor measurements, with one measurement per sensor/operating condition per second.

In the beginning of the engine's lifetime, the N-CMAPSS simulator generates sensor measurements using a linear, slow degradation model. Afterwards, an exponential, accelerated degradation model is used to simulate the sensor measurements instead [21]. The degradation of the engines when the linear, slow degradation model is used is still small, so we consider these sensor measurements as coming from "healthy", just-installed engines. We thus train our LSTM-AE only with the sensor measurements obtained with this slow degradation model. Let f^e_{a} be the last flight for which the sensor measurements are generated with the slow degradation model.

2.3.2. Data preprocessing

The training dataset consists of 6 engines, which together perform 446 flights. With 28 sensors and $4,311 \leq n^{e,f} \leq 18,169$ measurements per sensor per flight, we have a total of 147 million data points in the training data set. However, most of the measurements of the 28 sensors are highly correlated. For example, the flow out of the low pressure turbine and the flow out of the high pressure turbine have a correlation of 1.00. To reduce the computational load when training the LSTM-AE, without comprising the information contained in the sensor measurements, we select only one of two or more sensors that have a correlation of 0.99 or higher. This results in the selection of $m^{\mathrm{s}} = 13$ sensors from the available 28 sensors (see Table 2.1). With this, the number of sensor measurements considered in the training dataset is reduced to 68 million.

Symbol	Description	Unit
Wf	Fuel flow	pps
Nf	Physical fan speed	rpm
T24	Total temperature at LPC outlet	°R
T30	Total temperature at HPC outlet	°R
T48	Total temperature at HPT outlet	°R
T50	Total temperature at LPT outlet	°R
P2	Total pressure at fan inlet	psia
P50	Total pressure at LPT outlet	psia
W21	Fan flow	pps
W50	Flow out of LPT	lbm/s
SmFan	Fan stall margin	-
SmLPC	LPC stall margin	-
SmHPC	HPC stall margin	-

Table 2.1.: Sensors selected based on the correlation. LPC - Low Pressure Combustor. HPC - High Pressure Combustor. HPT - High Pressure Turbine. LPT - Low Pressure Turbine.

To further reduce the computational load for training the LSTM-AE, we aggregate the sensor measurements and operating conditions per minute. In other words, we consider the mean measurement and operating condition per minute. This reduces the number of sensor measurements in the training set to 1.143 million.

Moreover, both the sensor measurements and the operating conditions are normalized using min-max normalization, to ensure that the measurements have the same relative scale:

$$\frac{2\cdot(X_t^{e,f,s}-X_{\text{min}}^s)}{X_{\text{max}}^s-X_{\text{min}}^s}-1, \tag{2.14}$$

$$\frac{2\cdot(O_t^{e,f,o}-O_{\text{min}}^o)}{O_{\text{max}}^o-O_{\text{min}}^o}-1, \tag{2.15}$$

where $X_{\text{min}}^s/O_{\text{min}}^o$ and $X_{\text{max}}^s/O_{\text{max}}^o$ are the minimum and maximum measurement of sensor s/operating condition o in the training set respectively.

Last, there are only 101 flights in the training set where the sensor measurements are generated with the linear, slow degradation model. We therefore use data augmentation to increase the number of data samples for training the LSTM-AE [29]. For each flight $f \le f_{\text{a}}^e$ performed by engine e, we consider time-windows with a size of $60, 70, 80, \ldots, n^{e,f}-10, n^{e,f}$ time-steps. These time-windows are rolled over flight f of engine e with a step size (i.e., stride) of 5 minutes. In this way, we subtract for each time-window with a size of $60, 70, 80, \ldots, n^{e,f}-10, n^{e,f}$ time-steps, several time-series of multi-sensor measurements (i.e., data samples) from flight f of engine e. With this approach, 25,433 data samples are obtained to train the LSTM-AE.

2.3.3. Illustration of N-CMAPSS data set

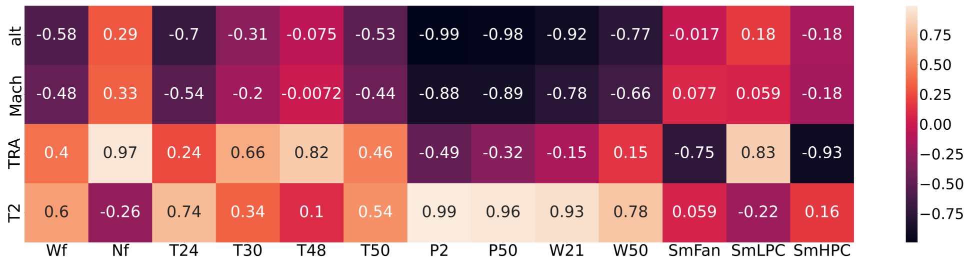
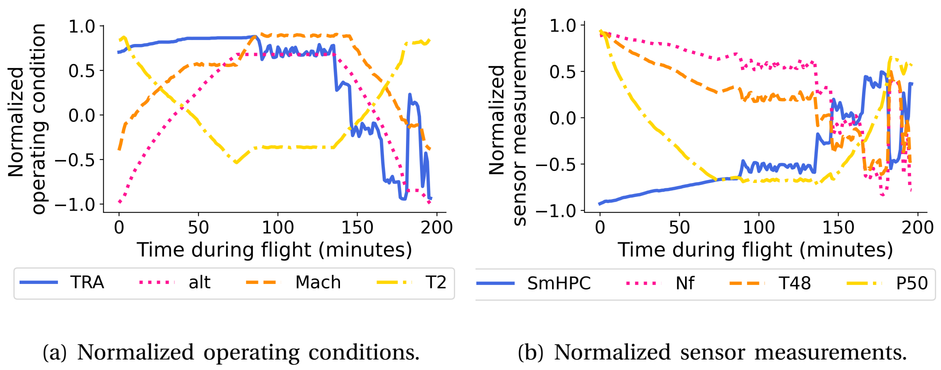


Figure 2.5.: Heatmap of the correlation between the sensor measurements and operating conditions - training engines of DS02, N-CMAPSS.



(a) Normalized operating conditions. (b) Normalized sensor measurements.

Figure 2.6.: Normalized operating conditions and normalized sensor measurements - flight 1, training engine 2 of DS02, N-CMAPSS.

Figure 2.6a shows the normalized operating conditions during the first flight of engine 2 from the training dataset. Figure 2.6b shows the normalized sensor measurements of sensors SmHPC, Nf, T48 and P50. These figures and the correlation heatmap in Figure 2.5 show that the sensor measurements are highly correlated with the operating conditions. For example, the correlation between the total pressure at the LPT outlet (P50) and the altitude of the aircraft (alt) is -0.98 (see Figure 2.5).

Figure 2.7 shows the mean normalized sensor measurement per flight, for all flights performed by engine 2 and for sensors SmHPC, Nf, T48 and P50. These mean sensor measurements do not exhibit a clear trend towards failure. A more extensive analysis is thus necessary to obtain a health indicator.

2.3.4. Metrics to evaluate the health indicators

We evaluate the health indicators with the monotonicity ($\mathcal{M}$), trendability ($\mathcal{T}$) and prognosability ($\mathcal{P}$) metrics as follows:

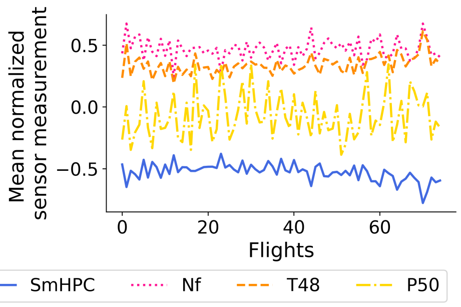


Figure 2.7.: Mean normalized sensor measurement per flight - training engine 2 of DS02, N-CMAPSS.

Monotonicity We measure the monotonicity $\mathcal{M}$ of the health indicator λ^e of an engine e as follows [17]:

$$\mathcal{M} = \frac{1}{F^e - 1}\left|\sum_{f=1}^{F^e-1} I(\lambda^e_{f+1} - \lambda^e_f)\right|, \tag{2.16}$$

$$I(x) = \begin{cases} 1 & x > 0 \\ -1 & x \leq 0 \end{cases}.$$

Trendability We consider the Spearman correlation coefficient between the health indicator λ^e and the flights $\{1, \ldots, F^e\}$ to measure the trendability $\mathcal{T}$ for engine e [30]:

$$\mathcal{T} = \frac{F^e \sum_{f=1}^{F^e} r_f^{\lambda^e} f - \left(\sum_{f=1}^{F^e} r_f^{\lambda^e}\right)\left(\sum_{f=1}^{F^e} f\right)}{\sqrt{\left(F^e \sum_{f=1}^{F^e}\left(r_f^{\lambda^e}\right)^2\right) - \left(\sum_{f=1}^{F^e} r_f^{\lambda^e}\right)^2} \cdot \sqrt{\left(F^e \sum_{f=1}^{F^e} f^2\right) - \left(\sum_{f=1}^{F^e} f\right)^2}}, \tag{2.17}$$

where $r_f^{\lambda^e}, f \in \{1, 2, \ldots, F^e\}$ is the rank sequence of the health indicator λ^e.

Prognosability We consider the following prognosability metric $\mathcal{P}$, which is also called the consistency [17, 30]:

$$\mathcal{P} = \exp\left[\frac{-\mathrm{STD}(\lambda^e_{F^e}, e \in E^{\text{test}})}{\frac{1}{|E^{\text{test}}|}\sum_{e \in E^{\text{test}}}\left|\lambda^e_1 - \lambda^e_{F^e}\right|}\right], \tag{2.18}$$

where $\mathrm{STD}(\lambda^e_{F^e}, e \in E^{\text{test}})$ is the standard deviation of the last health indicator values $\lambda^e_{F^e}, e \in E^{\text{test}}$ (with F^e the last flight of engine e), and E^{test} is the set with the test engines of dataset DS02. The closer this metric is to zero, the better the prognosability is.

2.4. Results - health indicator for aircraft engines

In this section, we present the health indicators developed for the engines in DS02, N-CMAPSS. We first describe the hyperparameters of the LSTM-AE in Section 2.4.1, and then the sensor selection in Section 2.4.2. Next, we present the health indicators from the LSTM-AE in Section 2.4.3. Last, we compare our approach with other methods in Section 2.4.4.

2.4.1. Hyperparameters of LSTM-AE

Table 2.2 shows the considered hyperparameters for the LSTM-AE. The architecture is derived using a grid search. After training, we select the weights that provide the lowest validation loss. Using a computer with an Intel Core i7 processor (8th generation), 4 CPU cores and 8GB of RAM memory, it took on average 4.04 minutes to train the LSTM-AE for one epoch.

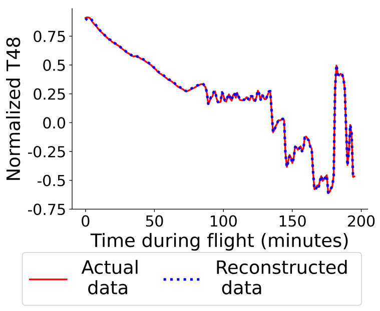
Hyperparameter	Value
Hyperparameters - architecture	
Hidden size h_t, c_t, h'_t and c'_t	4
Window-size D	5
Number of fully connected layers l	3
Number of nodes first $l-1$ fully connected layers	128
Hyperparameters - optimization	
Optimizer	Adam [31]
Number of epochs	100
Training-Validation split	90%-10%
Initial learning rate	0.01
Decrease learning rate when no improvement in validation loss for ... epochs in a row	10
Decrease learning rate by	$\frac{1}{10}$

Table 2.2.: Considered hyperparameters of the LSTM-AE.

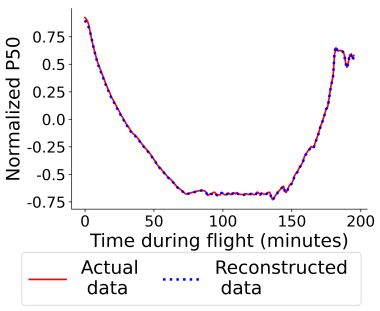
2.4.2. Sensor selection for constructing a health indicator

Figure 2.8 shows the reconstructed measurements of sensors T48 and P50 for the first and the last flight of training engine 2. During the first flight, the reconstructed measurements are very close to the actual measurements for both sensors. In contrast, the reconstructed measurements of sensor T48 deviate considerably from the actual measurements during the last flight of engine 2 (Figure 2.8c). This is expected, since the engine is severely degraded just before failure, while we train the LSTM-AE with sensor measurements from non- or slightly-degraded engines only. This does not hold, however, for all sensors. The reconstructed measurements of sensor P50 are still very close to the actual sensor measurements (Figure 2.8d).

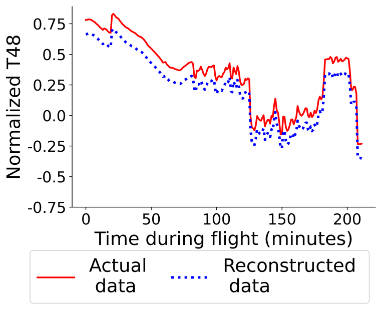
The different trends towards the time of failure of sensors T48 and P50 are also shown in Figure 2.9. The reconstruction loss of sensor T48 monotonically increases



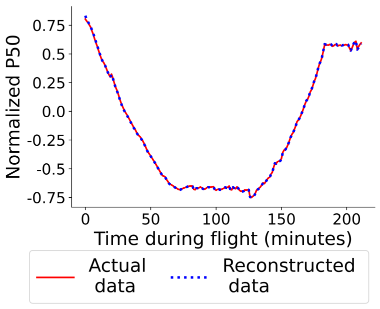
(a) First flight - sensor T48.



(b) First flight - sensor P50.

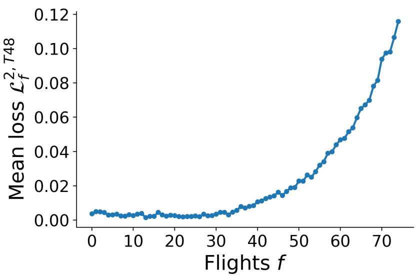


(c) Last flight - sensor T48.

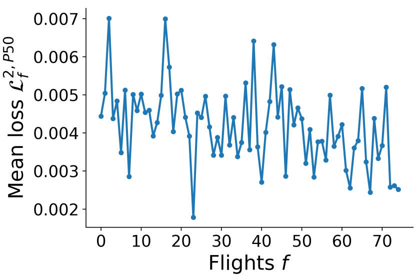


(d) Last flight - sensor P50.

Figure 2.8.: Actual and reconstructed sensor measurements with the LSTM-AE - first and last flight, training engine 2 of DS02, N-CMAPSS.



(a) Sensor T48 $\left(\mathcal{C}^{\mathrm{T48}} = 0.95\right)$.

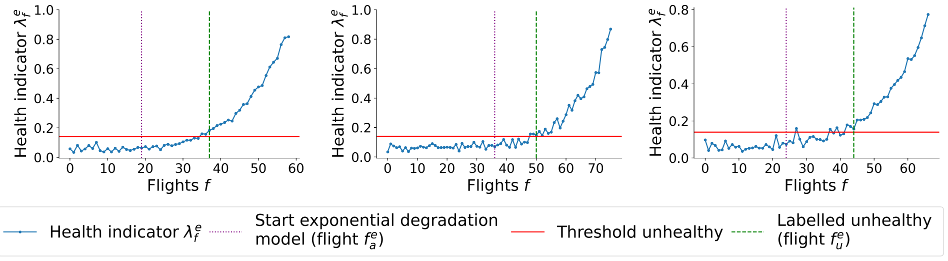


(b) Sensor P50 $\left(\mathcal{C}^{\mathrm{P50}} = 0.01\right)$.

Figure 2.9.: Mean loss $\mathcal{L}_f^{e,s}$ per flight with the LSTM-AE - training engine 2 of DS02, N-CMAPSS.

towards failure, while the reconstruction loss of sensor P50 resembles random noise. Let $\mathcal{C}^{e,s}$ be the Spearman correlation coefficient (see eq. (2.17)) between the reconstruction loss $\mathcal{L}^{e,s}$ of sensor s monitoring engine e and the flights f, i.e., the operating time. Let $\mathcal{C}^s$ be the mean over the Spearman correlation coefficients $\mathcal{C}^{e,s}$ for sensor s, where the mean is taken over all training engines e. This mean correlation is close to 1 for sensors for which the loss clearly increases towards failure, while it is close to 0 for sensors for which the loss shows no trend towards failure. To construct a health indicator, we include in eq. (2.13) only those sensors for which the mean Spearman correlation $\mathcal{C}^s$ between the reconstruction loss and the flights is 0.5 or larger. In this way, the health indicator is not constructed with sensors that are very weakly correlated with the time to failure, such as sensor P50. These sensors add little to no information on the degradation to the health indicator.

2.4.3. Health indicators of the test engines



(a) Engine 11. (b) Engine 14. (c) Engine 15.

Figure 2.10.: Health indicator with the LSTM-AE - test engines 11, 14 and 15 of DS02, N-CMAPSS.

Figure 2.10 shows the obtained health indicators, and Table 2.3 shows the sensors selected to construct the health indicators and the corresponding metrics. The three test engines fail when the health indicator equals roughly 0.8/0.9, which is reflected by the high prognosability of 0.94. The increasing trend of the health indicators towards failure is reflected by the mean trendability of 0.95. Some noise is visible in the health indicators, which is reflected by a quite low mean monotonicity of 0.38.

The operating conditions for test engine 11 are similar to the operating conditions of the engines in the training set, while the operating conditions of test engines 14 and 15 are different than in the training set [21]. The monotonicity for engine 14 and 15 is indeed lower than for engine 11. However, the trendability is still high for all three test engines. Our approach thus achieves a high trendability and prognosability even when the operating conditions are different than in the training set.

2.4.4. Comparison with other autoencoders

We compare the health indicators from the proposed approach with the health indicators from several other methods. Here, we consider other recurrent and non-recurrent autoencoders and the LSTM-AE without attention or operating conditions. The results for these other methods are in Table 2.3 as well.

		Engine 11		Engine 14		Engine 15		Mean		
	Selected sensors	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{M}$	$\mathcal{T}$	$\mathcal{P}$
Proposed method										
LSTM-AE	W50, SmFan, SmLPC, SmHPC, Wf, T24, T30, T48, T50	0.48	**0.97**	0.31	**0.91**	**0.36**	**0.97**	**0.38**	**0.95**	0.94
Proposed method with other recurrent autoencoders										
GRU-AE	W50, SmFan, SmLPC, SmHPC, Wf, T24, T30, T48, T50	0.21	0.94	0.23	0.79	0.12	0.93	0.18	0.89	0.94
BiGRU-AE	W50, SmLPC, SmHPC, T48, T50	0.24	0.88	0.23	0.84	0.03	0.79	0.17	0.84	**0.95**
BiLSTM-AE	W50, SmFan, SmLPC, SmHPC, Wf, T24, T30, T48, T50	**0.52**	**0.97**	**0.33**	0.90	0.30	**0.97**	**0.38**	0.94	0.94
Proposed method without operating conditions (no o.c.) or without attention (no att.)										
LSTM-AE -no o.c.	SmLPC	0.00	0.65	0.01	0.53	0.03	0.39	0.01	0.52	0.67
LSTM-AE -no att.	W50, SmFan, SmLPC, SmHPC, Wf, T24, T30, T48, T50	0.48	**0.97**	0.23	0.88	0.30	**0.97**	0.34	0.94	0.94
Other non-recurrent autoencoders										
1D-CAE	SmHPC	0.10	0.55	0.01	0.54	0.09	0.28	0.07	0.45	0.77
FAE	W21, W50, SmFan, SmLPC, SmHPC, Nf, T24, T30, T48, T50, P50	0.38	0.77	0.12	0.52	0.18	0.82	0.23	0.70	0.89

Table 2.3.: Evaluation of the health indicators for various autoencoders - test engines 11, 14 and 15 of DS02, N-CMAPSS. $\mathcal{M}$ - Monotonicity. $\mathcal{T}$ - trendability. $\mathcal{P}$ - prognosability. The best results are denoted in bold.

Other recurrent autoencoders We compare the obtained health indicators with the health indicators from four other recurrent autoencoders: The Gated Recurrent Unit autoencoder (GRU-AE), the bidirectional GRU-AE (BiGRU-AE) and the bidirectional LSTM-AE (BiLSTM-AE). For each recurrent autoencoder, we also implement local Luong attention and integrate the operating conditions.

For the BiLSTM-AE, we consider a bidirectional LSTM encoder [20, Chapter 8]. However, in the decoder, the reconstructed sensor measurements of time-step t are used as input in the LSTM-cell at time-step $t+1$. We therefore cannot consider a bidirectional decoder as well. For the GRU-AE, we replace each LSTM-cell in the proposed LSTM-AE by a GRU-cell [20, Chapter 7]. Similar, for the BiGRU-AE, we replace each LSTM-cell in the BiLSTM-AE with a GRU-cell.

Table 2.3 shows the monotonicity, trendability and prognosability for the health indicators of the different recurrent autoencoders. With the (Bi)GRU-AE, the monotonicity and the trendability are considerably lower than with the LSTM-AE. The prognosability with the BiGRU-AE is 0.95, which is slightly higher than the prognosability of 0.94 with the LSTM-AE.

The monotonicity and the prognosability are the same with the LSTM-AE and the BiLSTM-AE. With the LSTM-AE, however, the mean trendability of 0.95 is slightly higher than the mean trendability of 0.94 for the BiLSTM-AE.

The same subset of sensors is selected to create the health indicators for the GRU-AE, the LSTM-AE and the BiLSTM-AE. For the BiGRU-AE, however, less sensors are selected to create the health indicators. For this autoencoder, four sensors, namely SmFan, Wf, T24, and T30, are not selected.

No operating conditions or no attention We also analyse the health indicators when the operating conditions are *not* incorporated in the LSTM-AE, i.e., when the operating conditions $\mathbf{O}^{e,f}$ are completely removed from the autoencoder. When not considering the operating conditions, no sensor had a Spearman trendability for the training engines of 0.5 or higher. Instead, we only include sensor SmLPC, which has the highest trendability of all sensors. Without incorporating the operating conditions, the monotonicity, trendability and prognosability of the health indicators decrease considerably, as shown in Table 2.3.

Table 2.3 also shows the metrics when we do incorporate the operating conditions, but when no attention is used. The prognosability is the same with and without attention. The monotonicity, however, is lower when not using attention (0.34 instead of 0.38). Also the trendability is slightly lower (0.94 instead of 0.95).

Other non-recurrent autoencoders Last, we compare our method with two standard, non-recurrent autoencoders, namely a one-dimensional Convolutional autoencoder (1D-CAE) and a fully connected autoencoder (FAE). Both the sensor measurements and the operating conditions are selected as input, though only the sensor measurements are reconstructed. To construct fixed-length input samples, we consider a sliding time-window with a fixed size of 16 time-steps and a stride of 1. To create the health indicator value for a flight f of an engine e, we use the mean reconstruction loss $\mathcal{L}_f^{e,s}$ over all time-windows of size 16 and stride 1 of this flight f.

The one-dimensional convolutional encoder consists of two blocks, each with two convolutional layers and one max pooling layer with a pooling size of 2. The filters of the convolutional layer have a size of 4×1 and a stride of 1. The first three convolutional layers have 8 channels, while the last convolutional layer has only 1 channel. The convolutional decoder consists of the same structure, but instead of pooling layers we use interpolating layers. We consider zero-padding for all convolutional layers. Moreover, all layers use the ReLU activation function, except the last layer of the decoder, which uses the linear activation function.

The encoder of the FAE consists of two fully connected layers. The number of neurons is halved for each subsequent fully connected layer. The decoder consists of the same structure, only here the number of neurons is doubled in each fully connected layer. Each layer applies the ReLU activation function, except the last layer of the decoder, which uses the linear activation function.

Table 2.3 shows the results for the 1D-CAE and the FAE. For the 1D-CAE, no sensor had a Spearman trendability for the training engines of 0.5 or higher. Instead, we only include sensor SmHPC, which has the highest trendability of all sensors. The trendability and prognosability are considerably higher when considering a recurrent autoencoder instead of the 1D-CAE or the FAE. This shows the added value of processing the time-series of sensor measurements with a recurrent autoencoder.

2.5. METHODOLOGY - ONLINE RUL PROGNOSTICS USING SIMILARITY-BASED MATCHING

In this section, we show how the health indicators developed in Section 2.4 are used for health state division (Section 2.5.1) and to obtain RUL prognostics (Section 2.5.2).

2.5.1. HEALTH STATE DIVISION USING CHEBYSHEV'S INEQUALITY

Before we estimate the RUL, we first diagnose an engine as healthy or unhealthy. This is called health state division or diagnostics [30]. An engine is diagnosed as unhealthy once its health indicator crosses a threshold η times in a row. This threshold is determined using Chebyshev's inequality [32, 33]. For our application, this inequality states that:

$$P\left(\left|\lambda_f^e - \mu\right| \geq k\sigma\right) \leq \frac{1}{k^2}, \tag{2.19}$$

where $k > 0$, $P(\cdot)$ denotes the probability, and μ is the mean and σ is the standard deviation of the health indicator values λ_f^e of the training engines e and flights f for which the sensor measurements are simulated using the slow, linear degradation model (i.e., $f \leq f_{\text{a}}^e$). Thus, an engine is diagnosed as unhealthy as soon as the threshold $\mu + k\sigma$ is exceeded by the health indicator λ_f^e η times in a row. The probability that this occurs while the sensor measurements are generated using the slow, linear degradation model is less than $\left(\frac{1}{k^2}\right)^\eta$. Let f_{u}^e be the flight during which an engine e is diagnosed as unhealthy. We estimate the RUL from flight f_{u}^e onwards.

2.5.2. SIMILARITY-BASED MATCHING METHOD FOR RUL PROGNOSTICS

Once an engine is diagnosed as unhealthy, we estimate its RUL after each flight using a similarity-based health indicator matching approach [11, 22]. These are online RUL prognostics, since the RUL prognostics are updated every time more sensor measurements become available.

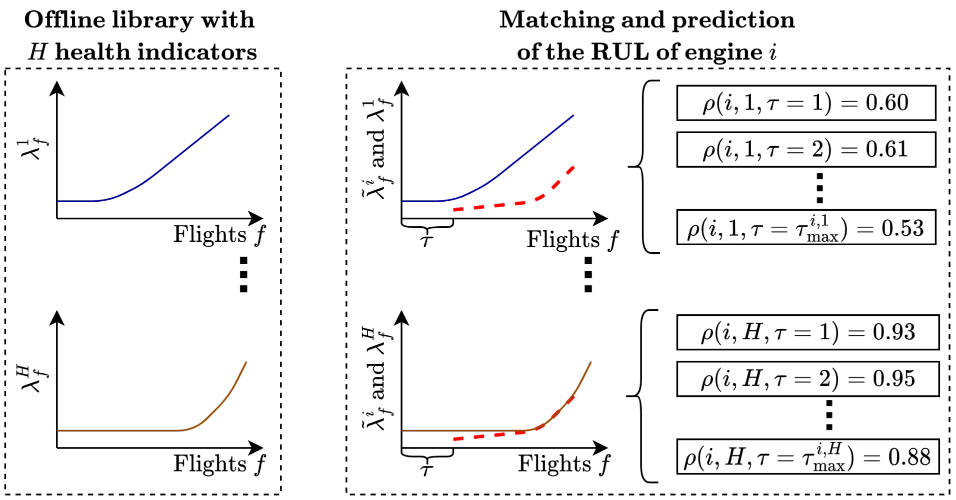
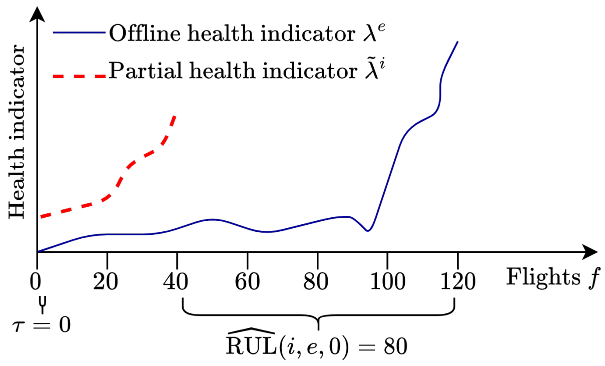

Offline library with H health indicators
λ^1_f
Flights f
λ^H_f
Flights f
Matching and prediction of the RUL of engine i
$\tilde{\lambda}^i_f$ and λ^1_f
τ
Flights f
$\rho(i,1,\tau=1)=0.60$
$\rho(i,1,\tau=2)=0.61$
$\rho(i,1,\tau=\tau^{i,1}_{\max})=0.53$
$\tilde{\lambda}^i_f$ and λ^H_f
τ
Flights f
$\rho(i,H,\tau=1)=0.93$
$\rho(i,H,\tau=2)=0.95$
$\rho(i,H,\tau=\tau^{i,H}_{\max})=0.88$

Figure 2.11.: Illustration of the similarity-based matching method to estimate the RUL of engine i, with H health-indicators from the training set in the library.

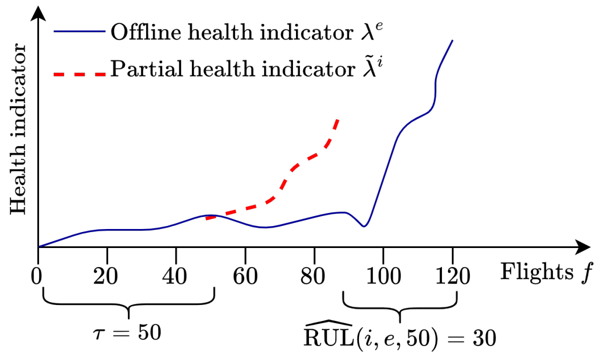
Let $\tilde{\lambda}^i = \left\{\lambda^i_f, f \in \{\tilde{f} - M, \ldots, \tilde{f}\}\right\}$ denote a partial health indicator available for engine i, using the sensor measurements available up to a flight $\tilde{f} \geq f^e_{\mathrm{u}}$. Here, M is the fixed length of the partial health indicators. Our aim is to estimate the RUL of engine i at flight $\tilde{f}$. For this, we consider a library with for each training engine e the offline health indicator λ^e. To estimate the RUL, we match the partial health indicator $\tilde{\lambda}^i$ with all the offline health indicators λ^e in the library. A schematic overview of the matching procedure is in Figure 2.11.

When matching, we determine the similarity between the partial health indicator $\tilde{\lambda}^i$ and the offline indicators λ^e in the library as the average Euclidean distance between these indices. To maximize the similarity between $\tilde{\lambda}^i$ and λ^e, i.e., to identify the best matches between $\tilde{\lambda}^i$ and λ^e, $\tilde{\lambda}^i$ is shifted along λ^e in the positive time-direction for τ flights.

Figure 2.12 shows an example of a matching between the partial health indicator $\tilde{\lambda}^i$ and a health indicator λ^e from the library. In this example, the partial health indicator is $M = 40$ flights long, while the offline health indicator λ^e is 120 flights long. When $\tau = 0$, the Euclidean distance is based on the first 40 values of both health indicators, so between $\tilde{\lambda}^i$ and $\{\lambda^e_f, f \in \{1,2,\ldots,40\}\}$. However, the similarity between these two health indicators is very small (see Figure 2.12a). In Figure 2.12b, we therefore shift $\tilde{\lambda}^i$ forward with $\tau = 50$ flights. Now, the Euclidean distance between $\tilde{\lambda}^i$ and $\{\lambda^e_f, f \in \{51,\ldots,90\}\}$ decreases. Let $\tau^{i,e}_{\max}$ ($\tau^{i,e}_{\max} = |\lambda^e| - M$ flights) denote the maximum number of flights $\tilde{\lambda}^i$ can be shifted forward when matching with λ^e, with $|\lambda^e|$ the length of the offline health indicator.



(a) Matching procedure with *no* time-lag τ.



(b) Matching procedure with a time-lag of $\tau = 50$.

Figure 2.12.: The iterative process of matching the partial health indicator $\tilde{\lambda}^i$ with the offline health indicator λ^e, with different values for the time-lag τ.

Given time-lag τ, the average Euclidean distance $d(\tilde{\lambda}^i, \lambda^e, \tau)$ between λ^e and $\tilde{\lambda}^i$ is:

$$d(\tilde{\lambda}^i, \lambda^e, \tau) = \frac{1}{M}\sqrt{\sum_{f=1}^{M}\left(\lambda^e_{f+\tau} - \tilde{\lambda}^i_f\right)^2}, \tag{2.20}$$

with a corresponding preliminary RUL prognostic of (see also Figure 2.12):

$$\widehat{\text{RUL}}(i,e,\tau) = \left|\lambda^e\right| - M - \tau, \tag{2.21}$$

and a similarity score of [11, 22]:

$$\rho(i,e,\tau) = \exp\left(\frac{-d(\tilde{\lambda}^i, \lambda^e, \tau)}{\gamma}\right), \tag{2.22}$$

where $\gamma > 0$ is a parameter that influences the scaling of the score with respect to the Euclidean distance. The score $\rho(i,e,\tau)$ is higher when $\tilde{\lambda}^i$ and λ^e are more similar, given the time-lag τ.

Let $\tilde{\rho}$ denote the highest similarity score of engine i, obtained across all training engines e in the library and all time-lags τ (see also Figure 2.11), i.e.,:

$$\tilde{\rho} = \max_{\substack{e \in E^{\text{train}}, \\ \tau \in \left\{0,1,\dots,\tau_{\max}^{i,e}\right\}}} \left\{\rho\left(i,e,\tau\right)\right\}, \tag{2.23}$$

where E^{train} denotes the set with all training engines. To estimate the RUL of engine i, we include all preliminary RUL prognostics $\widehat{\text{RUL}}(i,e,\tau)$ for which the score $\rho(i,e,\tau)$ is high enough, i.e., when [11]:

$$\rho\left(i,e,\tau\right) \geq \alpha \cdot \tilde{\rho}, \tag{2.24}$$

with $\alpha \in [0,1]$. Let Π^i be the set of all combinations (e,τ) of training engines e and time lags τ, such that $\rho(i,e,\tau) \geq \alpha \cdot \tilde{\rho}$. With this, the weight of RUL prognostic $\widehat{\text{RUL}}(i,e,\tau)$, $(e,\tau) \in \Pi^i$, is computed as:

$$p(i,e,\tau) = \frac{\rho(i,e,\tau)}{\sum_{(\epsilon,T)\in\Pi^i} \rho(i,\epsilon,T)}. \tag{2.25}$$

Finally, the estimated RUL of engine i after flight $\tilde{f}$ is [22]:

$$\text{RUL}^i = \sum_{(e,\tau)\in\Pi^i} p(i,e,\tau) \cdot \widehat{\text{RUL}}(i,e,\tau). \tag{2.26}$$

2.6. Results - Online RUL prognostics for aircraft engines

In this section, we present the RUL prognostics for the test engines of dataset DS02. First, we analyse the health state division and the RUL prognostics in Section 2.6.1. Then, we compare our results with the results of the other autoencoders in Section 2.6.2. Moreover, we compare our RUL prognostics with the RUL prognostics of common supervised learning models that directly estimate the RUL in Section 2.6.3. Last, we analyse the RUL prognostics for a decreasing number of offline health indicators in the library in Section 2.6.4.

2.6.1. Health state division and RUL prognostics

Table 2.4 shows the flight f_{u}^e during which the test engines of DS02 are diagnosed as unhealthy. Each test engine is labeled as unhealthy between 29 to 40 flights before failure (see also Figure 2.10). This is 1 to 12 flights after the last flight f_{a}^e during which the sensor measurements are generated using the linear degradation model.

Figure 2.13 shows the RUL prognostics for the test engines of DS02. These RUL prognostics are generated as soon as the engine is diagnosed as unhealthy, and then updated after each flight. The RUL of engine 11 is slightly overestimated when this engine is diagnosed as unhealthy, with a prediction error of -6 flights. In contrast, the RUL for engine 14 is slightly underestimated (a prediction error of 6 flights) after it is diagnosed as unhealthy. However, the RUL prognostics of all test engines

Method	Engine 11	Engine 14	Engine 15
Flight f_{a}^{e}	18	35	23
Proposed method			
LSTM-AE f_{u}^{e}	**30**	**36**	**32**
Proposed method with other recurrent autoencoders			
GRU-AE f_{u}^{e}	44	53	48
BiGRU-AE f_{u}^{e}	46	66	54
BiLSTM-AE f_{u}^{e}	**30**	**36**	**32**
Proposed method without attention (no att.)			
LSTM-AE- no att. f_{u}^{e}	**30**	**36**	37

Table 2.4.: Health state division: Flight f_{a}^{e}, after which the sensor measurements are generated using the exponential degradation model, and flight f_{u}^{e}, during which the engine is diagnosed as unhealthy - test engines 11, 14 and 15 of DS02, N-CMAPSS. The best results are denoted in bold.

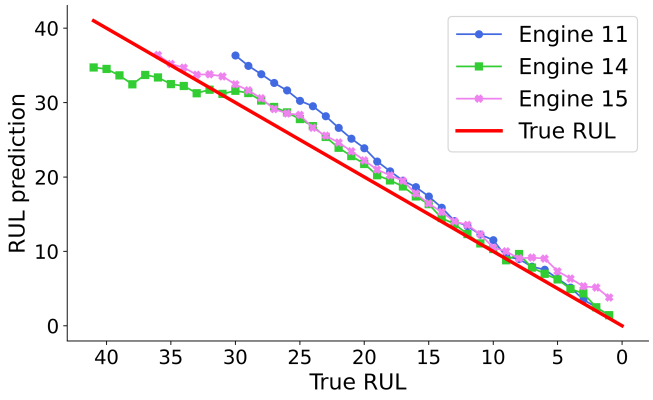


Figure 2.13.: RUL prognostics with the LSTM-AE - test engines 11, 14 and 15 of DS02, N-CMAPPS. The first RUL prognostic is made when the engine is declared unhealthy, and is updated after every flight.

quickly converge to the true RUL as the engines approach their failure time. Table 2.5 shows the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE) with these RUL prognostics. The results show that RUL is well estimated for all three engines, with a RMSE between 2.12 and 3.50 flights only.

The hyperparameters of the health state division and the similarity-based matching, used to obtain these RUL results, are derived using a grid search with leave-one-out cross validation in the training set [34]. Here, the goal is to minimize the RMSE for the training engines. For the health state division, we obtain $\eta = 3$ and $k = 5$ (see Section 2.5.1). For the similarity-based matching method, we obtain $M = 10$, $\lambda = 0.01$ and $\alpha = 0.7$ (see Section 2.5.2).

	Engine 11		Engine 14		Engine 15		All	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Proposed method								
LSTM-AE	2.89	3.50	1.89	2.41	1.95	2.12	2.18	2.67
Proposed method with other recurrent autoencoders								
GRU-AE	1.06	1.22	3.76	4.27	0.98	1.24	2.12	3.87
BiGRU-AE	1.24	1.72	3.02	3.45	1.73	2.80	1.91	2.36
BiLSTM-AE	2.72	3.41	2.42	3.02	2.77	2.96	2.62	3.12
Proposed method without attention (no att.)								
LSTM-AE-no att.	2.76	3.50	2.49	3.31	2.09	2.28	2.45	3.10

Table 2.5.: Evaluation of the RUL prognostics for our proposed approach (LSTM-AE) versus other approaches - test engines 11, 14 and 15 of DS02, N-CMAPSS.

2.6.2. COMPARISON WITH THE RUL PROGNOSTICS OF OTHER AUTOENCODERS

We also analyse the health state division and the RUL prognostics with the health indicators of the best autoencoders of Section 2.4.4: Specifically, we consider the other recurrent autoencoders, and the LSTM-AE without attention.

The test engines are diagnosed as unhealthy during the same flights for the BiLSTM-AE and the LSTM-AE (see Table 2.4). Moreover, test engine 11 and 14 are diagnosed as unhealthy during the same flights for the LSTM-AE with and without attention. However, test engine 15 is diagnosed as unhealthy 5 flights later when no attention is used. For the BiGRU-AE and the GRU-AE, the engines are diagnosed as unhealthy after a later flight: For the GRU-AE, the engines are labelled as unhealthy 15 to 23 flights before failure, while for the BiGRU-AE, the engines are labelled as unhealthy only 10 to 13 flights before failure. This late diagnosis as unhealthy is expected, given the relatively low monotonicity and trendability of the (Bi)GRU-AE. It is, however, preferable if an engine is diagnosed as unhealthy far before failure, provided that the degradation in the engines is large enough to accurately estimate the RUL of the engines.

Table 2.5 also shows the MAE and the RMSE of the RUL prognostics for the other considered autoencoders. For all four considered autoencoders, we find that $\eta = 3$, $k = 5$ and $\lambda = 0.01$. The fixed length equals $M = 10$ for the GRU-AE, and $M = 5$ for the other autoencoders. The parameter α is 0.1 for the BiGRU-AE, 0.2 for the LSTM-AE without attention, 0.4 for the BiLSTM-AE and 0.5 for the GRU-AE.

The RUL prognostics with the LSTM-AE have a lower overall RMSE and MAE then the RUL prognostics with the BiLSTM-AE. This is as expected, since the health indicators of the LSTM-AE have a slightly higher trendability.

We cannot directly compare the RUL prognostics of the (Bi)GRU-AE and the LSTM-AE, since the engines are diagnosed as unhealthy much closer to failure when considering the (Bi)GRU-AE. In general, we expect that the RUL prognostics improve when an engine degrades over time. We thus expect that the RMSE is lower when an engine is diagnosed as unhealthy later. Nevertheless, the overall RMSE of the

LSTM-AE (2.67 flights) is better than the overall RMSE of the GRU-AE (3.87 flights) and only slightly worse than the overall RMSE of the BiGRU-AE (2.36 flights).

For each test engine, the RMSE with the LSTM-AE without attention is larger than or equal to the RMSE with the LSTM-AE with attention. This also holds for test engine 15, even though engine 15 is diagnosed as unhealthy 5 flights later when not using attention. Moreover, the overall RMSE equals 3.10 flights without attention, while it only equals 2.67 flights with attention. This shows the benefits of incorporating attention in the autoencoder.

2.6.3. Comparison with other, supervised learning methods

Last, we compare our results with the results of neural networks that directly output a RUL prognostic, i.e., supervised learning methods. Here, we train two benchmark neural networks to directly estimate the RUL: The one-dimensional convolutional neural network (1D-CNN) and the LSTM neural network (LSTM-NN). These two neural networks are also used as benchmark in [29], and we thus use the same architecture and hyperparameters as in [29]. However, to allow for a fair comparison, we use the same sensors that are used as input to our approach (see Table 2.1) as input to the benchmark neural networks.

There is no straightforward method for health state division when using a supervised learning method. However, RUL prognostics usually improve when the true RUL becomes smaller. The comparison of the RUL prognostics would thus not be fair if we do not use any health state division for the supervised learning methods. Instead, we apply the health state division of the proposed methodology also to the benchmark neural networks. For example, engine 11 is diagnosed as unhealthy at flight 30 with the proposed approach. We thus also estimate the RUL of engine 11 with the benchmark neural networks from flight 30 onward.

	Engine			
	11	14	15	All
Proposed methodology				
LSTM-AE	**3.50**	**2.41**	**2.12**	**2.67**
Supervised learning neural networks				
1D-CNN	4.09	5.07	2.84	4.16
LSTM-NN	4.24	3.32	2.22	3.31

Table 2.6.: RMSE for the test engines 11,14 and 15 of DS02, N-CMAPSS, with various methodologies. The best results are denoted in bold.

Table 2.6 shows the RMSE of the RUL prognostics with the various methodologies. The RMSE of the RUL prognostics is lowest for all test engines when considering our proposed approach. This shows that our approach works well for the considered data set with limited failure instances, compared to a supervised learning method.

2.6.4. IMPACT OF THE NUMBER OF AVAILABLE LABELLED DATA SAMPLES ON THE RUL PROGNOSTICS

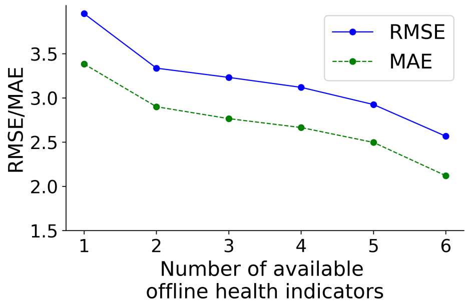


Figure 2.14.: Learning curve of the RMSE and MAE for the LSTM-AE - test engines of DS02, N-CMAPSS.

Due to preventive maintenance, most aircraft systems are replaced before their failure. There are therefore only limited labelled data samples available. In this section, we study the impact of the size of the library in the matching approach, i.e., the number of offline health indicators in the library, on the accuracy of the RUL prognostics. The health indicators are constructed using unlabelled data samples from the beginning of an engine's lifetime only. In real life, there are enough unlabelled data samples to train an autoencoder. We thus use the same online and offline health indicators as in Section 2.4.

Size of library	# of libraries	Set of offline libraries $(\cdot)$
1	6	$\{(e_1) : e_1 \in E^{\text{train}}\}$
2	15	$\{(e_1, e_2) : e_1 \in E^{\text{train}}, e_2 \in E^{\text{train}} \setminus \{e_1\}\}$
3	20	$\{(e_1, e_2, e_3) : e_1 \in E^{\text{train}}, e_2 \in E^{\text{train}} \setminus \{e_1\}, e_3 \in E^{\text{train}} \setminus \{e_1, e_2\}\}$
4	15	$\{(e_1, e_2, e_3, e_4) : e_1 \in E^{\text{train}}, e_i \in E^{\text{train}} \setminus \{e_j, j = 1,2,\ldots,i-1\}, i = 2,3,4\}$
5	6	$\{(e_1, e_2, e_3, e_4, e_5) : e_1 \in E^{\text{train}}, e_i \in E^{\text{train}} \setminus \{e_j, j = 1,2,\ldots,i-1\}, i = 2,3,4,5\}$
6	1	E^{train}

Table 2.7.: Overview of the considered libraries. Here, E^{train} is the set with all training engines.

Figure 2.14 shows the RMSE and the MAE of the RUL prognostics for an increasing number of available offline health indicators in the library. This is called the learning curve. We consider for each number of available offline health indicators, all possible combinations of historical health indicators that give a library of this size. For instance, we consider the following 15 libraries if two health indicators are available:

$$\{(1,2),(1,3),(1,4),(1,5),(1,6),(2,3),(2,4),(2,5),(2,6),(3,4),(3,5),(3,6),(4,5),(4,6),(5,6)\}.$$

Here, $1,2,\dots,6$ denotes the offline health indicator from the first, second, ..., sixth training engine respectively. An overview of all considered libraries is in Table 2.7. With each library, we estimate the RUL of the test engines.

As expected, the RUL prognostics improve when the size of the library increases. The decrease in the RMSE and MAE is highest when we consider two offline health indicators in the library, instead of just one. However, even when a library consists of just one offline health indicator, the RUL is well estimated with a RMSE of only 3.95 flights, and a MAE of only 3.38 flights. This shows that our approach works well when only very few labelled data samples are available.

2.7. CONCLUSION

In aviation, safety-critical aircraft systems usually undergo preventive maintenance. Consequently, only very few labelled sensor data samples, with as label the true RUL, are available. Many labelled data samples, however, are required to train supervised learning models that directly estimate the RUL. In this chapter, we therefore instead propose to construct a health indicator by training a LSTM autoencoder (LSTM-AE) with unlabelled data samples (i.e., the corresponding true RUL is unknown). The reconstruction errors of the LSTM-AE increase as the degradation in a system increases, and are therefore used to construct a health indicator. The sensor measurements of aircraft systems are generated at a high frequency during flights of several hours. Each data sample thus consists of a long time-series of multi-sensor measurements. We apply attention in the LSTM-AE to handle these long time-series. Moreover, aircraft are operated under highly-varying operating conditions. To create robust health indicators, we thus integrate the operating conditions in the LSTM-AE.

Next, we divide the lifetime of each engine in a healthy and an unhealthy stage by applying Chebyshev's inequality to the health indicators. Then, we use the health indicators and the few available labelled data samples in a similarity-based matching approach to estimate the RUL of the engines in the unhealthy stage.

We apply this approach to the aircraft engines in the new N-CMAPSS dataset [21]. The obtained health indicators have a high monotonicity (0.38), prognosability (0.94) and trendability (0.95). Moreover, the health indicators are indeed robust to the varying operating conditions. The trendability is also high for engines with operating conditions deviating from the operating conditions in the training set. Also the obtained RUL prognostics are accurate, with a RMSE of only 2.67 flights. Moreover, our approach outperforms supervised learning methods, that directly estimate the RUL, with a decrease in the RMSE of 19%.

Our proposed methodology is illustrated for aircraft engines. However, the described methodology is also suitable for applications in other fields, such as wind turbine gearboxes, bearings in industrial applications or batteries. For future research, we therefore plan to apply this methodology for other components and systems in other industries as well.

REFERENCES

[1] de Pater, I., & Mitici, M. (2023). Developing health indicators and RUL prognostics for systems with few failure instances and varying operating conditions using a LSTM autoencoder. *Engineering Applications of Artificial Intelligence*, *117*, Article number: 105582.

[2] de Pater, I., & Mitici, M. (2023, September 3-7). Constructing health indicators for systems with few failure instances using unsupervised learning. *Proceedings of the 33st European Safety and Reliability Conference*, Southampton, UK, Pages: 3066–3073.

[3] Ochella, S., Shafiee, M., & Dinmohammadi, F. (2022). Artificial intelligence in Prognostics and Health Management of engineering systems. *Engineering Applications of Artificial Intelligence*, *108*, Article number: 104552.

[4] Koutroulis, G., Mutlu, B., & Kern, R. (2022). Constructing robust health indicators from complex engineered systems via anticausal learning. *Engineering Applications of Artificial Intelligence*, *113*, Article number: 104926.

[5] Berghout, T., Mouss, L.-H., Kadri, O., Saïdi, L., & Benbouzid, M. (2020). Aircraft engines Remaining Useful Life prediction with an adaptive denoising online sequential extreme learning machine. *Engineering Applications of Artificial Intelligence*, *96*, Article number: 103936.

[6] de Pater, I., Reijns, A., & Mitici, M. (2022). Alarm-based predictive maintenance scheduling for aircraft engines with imperfect Remaining Useful Life prognostics. *Reliability Engineering & System Safety*, *221*, Article number: 108341.

[7] Shen, S., Lu, H., Sadoughi, M., Hu, C., Nemani, V., Thelen, A., Webster, K., Darr, M., Sidon, J., & Kenny, S. (2021). A physics-informed deep learning approach for bearing fault detection. *Engineering Applications of Artificial Intelligence*, *103*, Article number: 104295.

[8] de Pater, I., & Mitici, M. (2022, July 6-8). Novel metrics to evaluate probabilistic Remaining Useful Life prognostics with applications to turbofan engines. *Proceedings of the 7th European Conference of the Prognostics and Health Management (PHM) Society*, *7*, Turin, Italy, Pages: 96–109.

[9] Xiang, S., Qin, Y., Zhu, C., Wang, Y., & Chen, H. (2020). Long Short-Term Memory Neural Network with weight amplification and its application into gear Remaining Useful Life prediction. *Engineering Applications of Artificial Intelligence*, *91*, Article number: 103587.

[10] Fink, O., Wang, Q., Svensen, M., Dersin, P., Lee, W.-J., & Ducoffe, M. (2020). Potential, challenges and future directions for deep learning in Prognostics and Health Management applications. *Engineering Applications of Artificial Intelligence*, *92*, Article number: 103678.

[11] Malhotra, P., TV, V., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). Multi-sensor prognostics using an unsupervised health index based on LSTM encoder-decoder. *arXiv preprint arXiv:1608.06154*.

[12] Ye, Z., & Yu, J. (2021). Health condition monitoring of machines based on Long Short-Term Memory convolutional autoencoder. *Applied Soft Computing, 107*, Article number: 107379.

[13] Gugulothu, N., Tv, V., Malhotra, P., Vig, L., Agarwal, P., & Shroff, G. (2017). Predicting Remaining Useful Life using time series embeddings based on Recurrent Neural Networks. *arXiv preprint arXiv:1709.01073*.

[14] Yu, W., Kim, I. Y., & Mechefske, C. (2019). Remaining Useful Life estimation using a bidirectional Recurrent Neural Network based autoencoder scheme. *Mechanical Systems and Signal Processing, 129*, Pages: 764–780.

[15] Fu, S., Zhong, S., Lin, L., & Zhao, M. (2021). A novel time-series memory auto-encoder with sequentially updated reconstructions for Remaining Useful Life prediction. *IEEE Transactions on Neural Networks and Learning Systems, 33*, Pages: 7114–7125.

[16] Zhai, S., Gehring, B., & Reinhart, G. (2021). Enabling predictive maintenance integrated production scheduling by operation-specific health prognostics with generative deep learning. *Journal of Manufacturing Systems, 61*, Pages: 830–855.

[17] Liu, C., Sun, J., Liu, H., Lei, S., & Hu, X. (2020). Complex engineered system health indexes extraction using low frequency raw time-series data based on deep learning methods. *Measurement, 161*, Article number: 107890.

[18] Wei, Y., Wu, D., & Terpenny, J. (2021). Learning the health index of complex systems using dynamic conditional variational autoencoders. *Reliability Engineering & System Safety, 216*, Article number: 108004.

[19] Wang, J., Zeng, Z., Zhang, H., Barros, A., & Miao, Q. (2022). An hybrid domain adaptation diagnostic network guided by curriculum pseudo labels for electro-mechanical actuator. *Reliability Engineering & System Safety, 228*, Article number: 108770.

[20] Vasilev, I. (2019). *Advanced deep learning with Python: Design and implement advanced next-generation AI solutions using Tensorflow and PyTorch*. Packt Publishing Ltd.

[21] Arias Chao, M., Kulkarni, C., Goebel, K., & Fink, O. (2021). Aircraft engine run-to-failure dataset under real flight conditions for prognostics and diagnostics. *Data, 6*(1), Article number: 5.

[22] Yu, W., Kim, I. Y., & Mechefske, C. (2020). An improved similarity-based prognostic algorithm for RUL estimation using an RNN autoencoder scheme. *Reliability Engineering & System Safety, 199*, Article number: 106926.

[23] Lyu, J., Ying, R., Lu, N., & Zhang, B. (2020). Remaining Useful Life estimation with multiple local similarities. *Engineering Applications of Artificial Intelligence, 95*, Article number: 103849.

[24] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), Pages: 1735–1780.

[25] Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation, 12*(10), Pages: 2451–2471.

[26] Géron, A. (2018). *Neural networks and deep learning*. O'Reilly.

[27] Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

[28] Saxena, A., & Goebel, K. (2008). *Turbofan engine degradation simulation data set*, NASA Prognostics Data Repository, NASA Ames Research Center, Moffett Field, California, USA.

[29] Chao, M. A., Kulkarni, C., Goebel, K., & Fink, O. (2022). Fusing physics-based and deep learning models for prognostics. *Reliability Engineering & System Safety*, *217*, Article number: 107961.

[30] Lei, Y., Li, N., Guo, L., Li, N., Yan, T., & Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing*, *104*, Pages: 799–834.

[31] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[32] Singh, J., Darpe, A., & Singh, S. P. (2020). Bearing Remaining Useful Life estimation using an adaptive data-driven model based on health state change point identification and K-means clustering. *Measurement Science and Technology*, *31*(8), Article number: 085601.

[33] Kong, X., & Yang, J. (2019). Remaining Useful Life prediction of rolling bearings based on RMS-MAVE and dynamic exponential regression model. *IEEE Access*, *7*, Pages: 169705–169714.

[34] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

3 An improved weight initialization strategy for neural networks, applied to point RUL prognostics

Throughout this dissertation, we use neural networks to estimate the Remaining Useful Life (RUL) of aircraft systems. Training these neural networks is time-consuming. A good weight initialization strategy may accelerate the convergence of the weights in a neural network and reduce the training time. In this chapter, we therefore propose a new method to initialize the weights in the last layer of a neural network.

We first derive analytically a constraint on the weights that accelerates the convergence of the weights during the back-propagation algorithm. We then use constrained linear regression to analytically derive the initial weights and the initial bias of the last layer that minimize the initial training loss, given the derived constraint.

We apply our proposed weight initialization approach to a Convolutional Neural Network that estimates the RUL of aircraft engines. With our weight initialization strategy, the initial training and validation loss are relatively small, the weights do not get stuck in a local optimum, and the convergence of the weights is accelerated. We also apply our approach to two neural networks commonly used in image classification, combined with transfer learning.

Parts of this chapter have been published in:

de Pater, I., & Mitici, M. (2023). A mathematical framework for improved weight initialization of neural networks using Lagrange multipliers. *Neural Networks, 166*, Pages: 579–594

3.1. Introduction

Neural networks have become increasingly popular in the last few decades, with applications in a wide range of domains [2], such as image classification [3], time-series prediction [4], object detection [5] and natural language processing [6]. The weights of these neural networks are usually optimized using gradient descent, until the weights converge such that the loss is close to a minimum [7]. In the last years, many improvements have been proposed to accelerate the convergence of the weights such as, for instance, improved optimizers [8, 9], good weight initialization strategies [10, 11] and the use of specialized hardware such as GPUs. However, training an accurate neural network is still computationally intensive, consumes a large amount of energy and takes a long time. This is especially problematic for deep neural networks trained with large data sets. Methods that accelerate the convergence of the weights in a neural network are therefore still needed.

One way to speed up the training of a neural network is a good weight initialization method. Most papers on weight initialization focus on initializing the weights such that the convergence of the weights during the back-propagation algorithm is accelerated. Here, the goal is to mitigate the vanishing or exploding gradient problem, and to prevent that the weights get stuck in a local optimum [12]. This is also the aim of the most popular approaches for weights initialization, namely "Xavier" [10] and "Kaiming" initialization [11]. In [10], it is shown that the variance of both the forward-propagated outputs and the backward-propagated gradients should be equal for all layers in the neural network after the weight initialization to prevent the vanishing/exploding gradient problem. From this requirement, the required variance of the weights is derived for each layer. In [10], this required variance is derived for a neural network with the sigmoid or tanh activation function, while in [11], this required variance is derived for a neural network with the ReLU activation function. In both papers, the weights are then randomly chosen from a distribution, such as uniform or normal, with a mean of zero and the required variance. These weight initialization strategies are very successful when training deep neural networks. In this chapter, we therefore also consider the requirement that the variance of the outputs and gradients of each layer of the neural network are equal after the weight initialization.

Other studies on weight initialization use the characteristics of the training data instead. In [13], the weights are initialized with the orthogonal projection of the input correlation matrix using the singular value decomposition. In [14], this method is extended by using layer sequential unit variance: Random weights are first orthonormalized, and then the variance of the output of each layer is normalized to one. In [15], the weights are sampled from an interval that is determined based on the characteristics of the data. In this chapter, we also use the characteristics of the training data to initialize the weights.

Overall, most existing studies on weight initialization do not consider the initial loss [10, 11, 13, 14]. Due to the random starting point of the weights, however, many updating steps and thus many epochs might be necessary to achieve convergence. In contrast, there are few studies that focus on finding initial weights that minimize the initial training loss, i.e., on finding weights close to an optimum, instead. For example, in [16, 17] linear regression is used to initialize the weights of the last layer, the authors of [18] use linear discriminant analysis to initialize the weights such that the difference between classes is maximized and in [19], the initial training loss is minimized using the singular value decomposition. However, these studies in turn do not consider the convergence of the weights when initializing the weights of the last layer. To address this, we propose a weight initialization strategy that combines both goals: We initialize the weights and bias of the last layer of a neural network such that i) the weight convergence during the back-propagation algorithm is accelerated and ii) given that the weight convergence is accelerated, we initialize the weights close to an optimum point by minimizing the initial training loss.

Our approach is inspired by techniques from the field of neural networks with random weights (NNRW), such as extreme learning machines. In NNRW, the weights are optimized without gradient descent or other iterative methods [20]. Instead, the weights of the first layers of the neural network are randomly chosen. The weights of the last layer are then optimized with the objective to minimize the loss function using, for instance, Ridge regression [20], the matrix inverse [21] or inequality constrained least-squares [22]. Inspired by NNRW, we optimize the initial weights and the initial bias of the last layer of the neural network using a regression model.

However, in contrast with NNRWs, we further optimize the weights with the back-propagation algorithm. To also mitigate the vanishing/exploding gradient problem, we analytically derive a novel tight constraint on the weights of the last layer to ensure that the variance of the output/gradients of the last layer is equal to the variance of the output/gradients of the other layers [10]. This novel tight constraint holds without assuming that the mean of the weights and bias is zero, as commonly done in most weight initialization strategies. Given this constraint, we next derive analytically the optimal initial weights and bias of the last layer, i.e., the weights and bias that give the lowest initial training loss, using Lagrange. We derive this constraint, and the corresponding optimal initial weights and bias, for neural networks that solve a regression problem and that use a specific activation function.

We first apply our proposed approach to estimate the Remaining Useful Life (RUL) of aircraft engines using a Convolutional Neural Network. This is a regression problem. Compared to the random Xavier weight initialization method [10], the weights indeed start at point that gives a relatively small initial training and validation loss. The initial validation loss is 2.4 times smaller with our approach. Moreover, the weights quickly converge from this small initial training loss, due to the novel constraint derived on the weights. Last, the weights do not get stuck in a local optimum. Overall, the training of the neural network is therefore much faster when using our approach. The smallest validation loss obtained after 198 epochs following Xavier [10], is already obtained after 49 epochs with our approach. We thus need 75% fewer epochs to reach the same result. The best benchmark strategy is Kaiming weight initialization [11]. Here, the initial validation loss is 2.4 times smaller with our approach as well. Moreover, the minimum validation loss reached after 148 epochs using Kaiming initialization, is already reached after 97 epochs with our approach, i.e., we need 34% fewer epochs to reach the same validation loss. We also show that with the new initialization technique, we can relax the assumption that the mean of the weights is zero (as assumed in [10, 11]). Last, we find that only a small part of the training set can be used to initialize the weights. This makes the weight initialization 42 times faster, while the weights converge at the same rate.

We next adjust our approach so that it can be applied to neural networks that solve classification problems as well, with any type of activation function. We then illustrate our approach using ResNet-18 and ResNet-34 [23] to classify the images in the CIFAR-100 dataset [24]. When training ResNet-18 and ResNet-34 from scratch, we achieve a slightly faster convergence of the weights. However, we obtain the best results when combining our approach with transfer learning. Combined with transfer learning, the initial accuracy of the validation set after applying our approach is already 53% and 55% for ResNet-18 and ResNet-34 respectively. This leads to a faster convergence of the weights. Using ResNet-18 and the best benchmark method (LeCun initialization [25]), the highest validation accuracy of 81.32% is obtained after 49 epochs. Using our approach, the same accuracy is already obtained after just 13 epochs. Similarly, for ResNet-34, the highest validation accuracy of 84.18% is obtained after 50 epochs using LeCun initialization, while the same accuracy is already obtained after 14 epochs with our approach.

The remainder of this chapter is structured as follows. We first introduce our proposed weight initialization methodology for neural networks in Section 3.2. We apply this methodology to a case study with a regression problem in Section 3.3. Last, we apply our approach to classification neural networks, namely ResNet-18 and ResNet-34, in Section 3.4. We then discuss the conclusions, the limitations and future research directions in Section 3.5. The Python code for the proposed weight initialization method is in [1].

3.2. Methodology - Weight initialization in the last layer of the neural network

In Section 3.2.1 we discuss the layout of the considered neural network. We derive the constraints on the weights in the last layer in Section 3.2.2, and we integrate these constraints in the linear regression problem using the Lagrange multiplier in Section 3.2.3. We summarize the full weight initialization procedure in Section 3.2.4.

3.2.1. Neural network for a regression problem

We use a neural network with L layers to solve a regression problem, i.e., the true label of each sample is one numerical value. Let S be the training set for the neural network, with N training samples. The true label of a training sample $i \in S$ is denoted by y_i, and the vector with all true labels is denoted by $\mathbf{y} = [y_1, y_2, \dots, y_N]$. We assume that the activation function $f(\cdot)$ used throughout the neural network has a unit derivative at 0, i.e., $f'(0) = 1$, and that this derivative exists. For example, this could be the hyperbolic tangent activation function (tanh). Moreover, we assume that the last layer L of this neural network is a fully connected layer. This is often the case for regression neural networks, e.g., [26–28]. Figure 3.1 shows a schematic overview of this last layer.

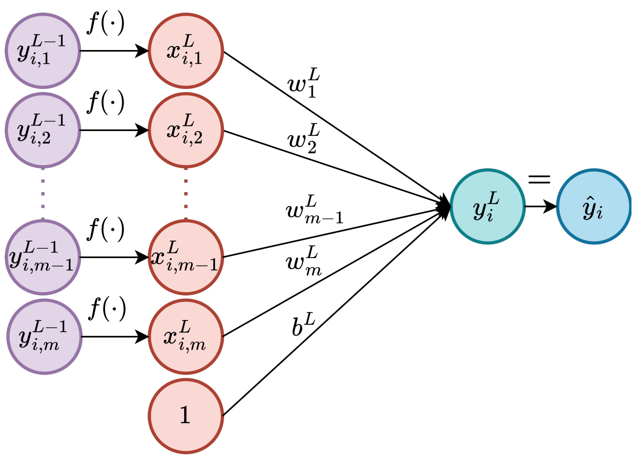


Figure 3.1.: Schematic overview of the last layer of the assumed neural network for a training sample $i \in S$.

Let $\mathbf{y}^{L-1} = [\mathbf{y}_1^{L-1}, \mathbf{y}_2^{L-1}, \ldots, \mathbf{y}_m^{L-1}]$ be the matrix with the output of the $(L-1)^{\text{th}}$ layer, with m the number of output nodes. Here, $\mathbf{y}_j^{L-1} = [y_{1,j}^{L-1}, y_{2,j}^{L-1}, \ldots, y_{N,j}^{L-1}]^T$ is the vector with the output of the j^{th} node of the $(L-1)^{\text{th}}$ layer for all training samples $i \in S$, and T denotes the transpose. Then, $\mathbf{x}^L = f(\mathbf{y}^{L-1}) = [\mathbf{x}_1^L, \mathbf{x}_2^L, \ldots, \mathbf{x}_m^L]$ is the matrix with the activated input of layer L. Here, $\mathbf{x}_j^L = [x_{1,j}^L, x_{2,j}^L, \ldots, x_{N,j}^L]^T$ is the vector with the j^{th} hidden input state of layer L for all training samples $i \in S$. Last, the weights of layer L are denoted by $\mathbf{w}^L = [w_1^L, w_2^L, \ldots, w_m^L]$, while b^L denotes the bias. The output y_i^L of layer L for training sample i is then $y_i^L = \sum_{j=1}^m w_j^L x_{i,j}^L + b^L$.

We assume that the considered neural network applies a linear activation function to the output, i.e., the output y_i^L of the last layer directly is the estimated label $\hat{y}_i$ of training sample i. This linear activation function is also often applied in neural networks that solve a regression problem, e.g., [26–28]. The vector with estimated labels for the training set S is denoted by $\hat{\mathbf{y}} = \mathbf{y}^L = [y_1^L, y_2^L, \ldots, y_N^L]$. The objective of the regression task is to minimize the loss function. As loss function, we use the squared error of the output:

$$\text{Loss} = \sum_{i \in S} \left(y_i - \hat{y}_i\right)^2. \tag{3.1}$$

In this chapter, we randomly initialize the weights of the first $L-1$ layers from a normal distribution following [10] (i.e., Xavier initialization). Given these random weights, the aim of this study is to initialize the weights $\mathbf{w}^L$ and bias b^L of the last layer such that the loss is minimized. We therefore perform one forward pass with all training samples in S, and obtain the hidden input states $\mathbf{x}^L$ of layer L. Now, the weights $\mathbf{w}^L$ and bias b^L that minimize the loss function can easily be obtained with the least squares solution of a linear regression of the actual labels $\mathbf{y}$ on the hidden states $\mathbf{x}^L$. The objective of this linear regression is:

$$\min_{b^L,\mathbf{w}^L} \sum_{i\in S} \left(y_i - \hat{y}_i\right)^2 = \min_{b^L, w_j^L, j=1,2,\ldots,m} \sum_{i\in S} \left(y_i - \left(\sum_{j=1}^{m} w_j^L x_{i,j}^L + b^L\right)\right)^2. \tag{3.2}$$

However, initializing the weights in this way might cause the gradients to vanish or explode during the back-propagation algorithm. This in turn hinders the convergence of the weights. To avoid the vanishing or exploding gradient problem, the authors of [10] found that the variance of the outputs and gradients of each layer in the neural network should be equal after weight initialization. In the next sections, we rewrite this requirement in a constraint on the weights of the last layer, and integrate this constraint in the linear regression problem.

3.2.2. Constraints on the weights of the last layer of the neural network

To avoid the vanishing or the exploding gradient problem, the variance (Var) of i) the forward-propagated output and ii) the backward-propagated gradients of each layer should be equal. For the last layer L, this means that [10]:

$$\mathrm{Var}\left(\hat{y}\right) = \mathrm{Var}\left(x^L\right) \tag{3.3}$$

$$\mathrm{Var}\left(\frac{\partial \mathrm{Loss}}{\partial y^L}\right) = \mathrm{Var}\left(\frac{\partial \mathrm{Loss}}{\partial y^{L-1}}\right), \tag{3.4}$$

where $\hat{y}$, x^L, $\frac{\partial \mathrm{Loss}}{\partial y^L}$ and $\frac{\partial \mathrm{Loss}}{\partial y^{L-1}}$ represent the random variable of any element in $\hat{\mathbf{y}}$, $\mathbf{x}^L$, $\frac{\partial \mathrm{Loss}}{\partial \mathbf{y}^L}$ and $\frac{\partial \mathrm{Loss}}{\partial \mathbf{y}^{L-1}}$ respectively [11]. In [10], these two requirements are used to derive the desired variance for the weights in each layer. The authors assume in this derivation that the initial weights are independent and identically distributed random variables. The weights are then randomly sampled from a distribution, usually normal or uniform, with mean zero and the desired variance. This strategy is commonly called "Xavier initialization".

In this chapter, we cannot directly use the same derivation as in [10], since we do not assume that the mean of the weights is zero. Moreover, regarding the weights as independent and identically distributed random variables, as in [10], becomes problematic for the considered approach. We therefore use the two requirements to derive a constraint on the weights in the last layer using the rules of the variance of a linear function instead. An advantage of this method is that fewer assumptions are made, and that it suits a linear regression approach more naturally. To verify our approach, we show in Appendix 3.C how the same constraints can be derived using the same derivation as in [10].

Requirement 1: $\mathrm{Var}\left(\hat{y}\right) = \mathrm{Var}\left(x^L\right)$

As in [10, 11], we assume that the hidden states in $\mathbf{x}^L$ are independently and identically distributed. Specifically, the variance of x_j^L, representing a random variable of any element in $\mathbf{x}_j^L$, is equal for all nodes $j \in \{1,2,\dots,m\}$. This assumption from [10] still holds, since these hidden states come from the randomly initialized weights. In contrast with [10, 11], however, we treat the initialized weights and the initialized bias of the last layer as constant numbers instead of random variables, i.e., given the initialized bias and weights, we assume that the variance of $\hat{y}$ comes from the variance of x^L only. This interpretation fits a linear regression approach well, since the weights are not sampled from a distribution. Since we impose that $\hat{y} = y^L$, it follows that [29]:

$$\begin{aligned} \mathrm{Var}(\hat{y}) &= \mathrm{Var}\left(b^L + \sum_{j=1}^{m} w_j^L x_j^L\right) \\ &= \sum_{j=1}^{m} \left(w_j^L\right)^2 \mathrm{Var}\left(x^L\right) \end{aligned} \tag{3.5}$$

The first requirement states that $\mathrm{Var}\left(\hat{y}\right) = \mathrm{Var}\left(x^L\right)$. This gives the following constraint on the sum of the squared weights:

$$\sum_{j=1}^{m} \left(w_j^L\right)^2 = 1 \tag{3.6}$$

In [10], it is assumed that the expected value of the weights is zero. If we would also assume that the mean of the weights is zero, then eq. (3.6) states that the empirical variance of the weights should equal $\frac{1}{m}$ (i.e., $\frac{1}{m}\sum_{j=1}^{m}\left(w_j^L\right)^2 = \frac{1}{m}$). This is then the same as the constraint on the variance of the weights in eq. (10) in [10]. However, we do not use that the mean of the weights is zero in our derivation. In this chapter, we therefore first consider the case where the mean of the weight is not restricted. In Section 3.3.4, we instead follow [10, 11] and assume that the mean of the weights in the last layer is zero.

Requirement 2: $\mathrm{Var}\left(\frac{\partial \mathrm{Loss}}{\partial y^L}\right) = \mathrm{Var}\left(\frac{\partial \mathrm{Loss}}{\partial y^{L-1}}\right)$

The second requirement states that the variance of the gradients is equal throughout the neural network. To derive a constraint on the weights from this requirement, we first write out $\frac{\partial \mathrm{Loss}}{\partial \mathbf{y}_j^{L-1}}$ of one hidden state $j, j \in \{1,2,\dots,m\}$:

$$\frac{\partial \mathrm{Loss}}{\partial \mathbf{y}_j^{L-1}} = \frac{\partial \mathrm{Loss}}{\partial \mathbf{y}^L}\frac{\partial \mathbf{y}^L}{\partial \mathbf{x}^L}\frac{\partial \mathbf{x}^L}{\partial \mathbf{y}_j^{L-1}} = \frac{\partial \mathrm{Loss}}{\partial \mathbf{y}^L} w_j^L f'\left(\mathbf{y}_j^{L-1}\right), \tag{3.7}$$

with $f'(\cdot)$ the derivative of the considered activation function. As in [10], we assume that this activation function has a unit derivative at 0 (see Section 3.2.1). Since the weight initialization in all layers before layer L is still random, we follow [10] and assume that $\mathbf{y}_j^{L-1} \approx \mathbf{0}$, and thus that $f'(\mathbf{y}_j^{L-1}) \approx \mathbf{1}$ (Taylor expansions around $\mathbf{y}_j^{L-1} = 0$). This gives:

$$\frac{\partial \mathrm{Loss}}{\partial \mathbf{y}_j^{L-1}} \approx \frac{\partial \mathrm{Loss}}{\partial \mathbf{y}^L} w_j^L \tag{3.8}$$

Using this, we derive the variance:

$$\mathrm{Var}\left(\frac{\partial\mathrm{Loss}}{\partial y_j^{L-1}}\right) = \mathrm{Var}\left(\frac{\partial\mathrm{Loss}}{\partial y^L} w_j^L\right) = \left(w_j^L\right)^2 \mathrm{Var}\left(\frac{\partial\mathrm{Loss}}{\partial y^L}\right), \tag{3.9}$$

where $\frac{\partial\mathrm{Loss}}{\partial y_j^{L-1}}$ represent the random variable of any element in $\frac{\partial\mathrm{Loss}}{\partial \mathbf{y}_j^{L-1}}$. We are, however, interested in the variance of $\frac{\partial\mathrm{Loss}}{\partial y^{L-1}}$ for the random variable of any element in $\frac{\partial\mathrm{Loss}}{\partial \mathbf{y}^{L-1}}$, over all hidden states j. Since w_j^L usually does not equal w_i^L if $i \neq j$, we cannot assume that the variance of $\frac{\partial\mathrm{Loss}}{\partial y_j^{L-1}}$ is the same for all $j \in \{1,2,\ldots,m\}$. Instead, we use that the variance of $\frac{\partial\mathrm{Loss}}{\partial y^{L-1}}$ is the variance of a mixture distribution of all random variables $\frac{\partial\mathrm{Loss}}{\partial y_j^{L-1}}$, for $j \in \{1,2,\ldots,m\}$. The variance of m mixture distributions, where each distribution has weight $\frac{1}{m}$, mean $\mu_j = \mathbb{E}[\frac{\partial\mathrm{Loss}}{\partial y_j^{L-1}}]$, and variance $(w_j^L)^2\mathrm{Var}\left(\frac{\partial\mathrm{Loss}}{\partial y^L}\right)$, and where the total mean is $\mu = \mathbb{E}[\frac{\partial\mathrm{Loss}}{\partial y^{L-1}}]$, equals [30]:

$$\sum_{j=1}^{m} \frac{1}{m}\left(\left(w_j^L\right)^2 \mathrm{Var}\left(\frac{\partial\mathrm{Loss}}{\partial y^L}\right) + \mu_j^2\right) - \mu^2$$

The expected value μ_j of $\frac{\partial\mathrm{Loss}}{\partial y_j^{L-1}}$ is:

$$\begin{aligned}\mathbb{E}\left[\frac{\partial\mathrm{Loss}}{\partial y_j^{L-1}}\right] &= \mathbb{E}\left[\frac{\partial\mathrm{Loss}}{\partial y^L} w_j^L\right] \\ &= \mathbb{E}\left[-2(y-\hat{y})\, w_j^L\right] \\ &= -2w_j^L\left(\mathbb{E}[y] - \mathbb{E}[\hat{y}]\right).\end{aligned} \tag{3.10}$$

Here, y represent the random variable of any element in $\mathbf{y}$. In Section 3.2.3, we show that the expected value $\mathbb{E}[\hat{y}]$, given the optimal bias and weights in the last layer, equals $\frac{1}{N}\sum_{i\in S} y_i = \mathbb{E}[y]$. The expected value μ_j of $\frac{\partial\mathrm{Loss}}{\partial y_j^{L-1}}$ is thus zero, and the total mean μ is zero as well. This gives:

$$\mathrm{Var}\left(\frac{\partial\mathrm{Loss}}{\partial y^{L-1}}\right) = \sum_{j=1}^{m} \frac{1}{m}(w_j^L)^2 \mathrm{Var}\left(\frac{\partial\mathrm{Loss}}{\partial y^L}\right) \tag{3.11}$$

We therefore derive the constraint that:

$$\frac{1}{m}\sum_{j=1}^{m}\left(w_j^L\right)^2 = 1. \tag{3.12}$$

If we would also assume that the mean of the weights is zero, then eq. (3.12) states that the empirical variance of the weights should equal 1. This is then the same as the constraint on the variance of the weights in eq. (11) of [10].

Final constraint As in [10], we derive two different, conflicting constraints on the weights. As a compromise, we therefore average the two constraints [10]:

$$\sum_{j=1}^{m}\left(w_j^L\right)^2 = \frac{1+m}{2} \tag{3.13}$$

3.2.3. LAGRANGE RELAXATION OF THE CONSTRAINED LINEAR REGRESSION PROBLEM

To initialize the weights in the last layer, we thus solve the following constrained linear regression problem:

$$\begin{aligned} \min_{b^L, w_j^L, j=1,2,\ldots,m} \quad & \sum_{i\in S}\left(y_i - b^L - \sum_{j=1}^{m} w_j^L x_{i,j}^L\right)^2 \\ \text{such that} \quad & \sum_{j=1}^{m}\left(w_j^L\right)^2 = \frac{1+m}{2}. \end{aligned} \tag{3.14}$$

This constrained regression problem can be solved exactly using the Lagrange multiplier. The Lagrange function $\mathcal{L}(\lambda, b^L, w_j^L, j = 1,2,\ldots,m)$ is:

$$\mathcal{L}\left(\lambda, b^L, w_j^L, j = 1,2,\ldots,m\right) = \sum_{i\in S}\left(y_i - b^L - \sum_{j=1}^{m} w_j^L x_{i,j}^L\right)^2 + \lambda\left(\sum_{j=1}^{m}\left(w_j^L\right)^2 - \frac{1+m}{2}\right). \tag{3.15}$$

This minimization problem is similar to the minimization problem in Ridge linear regression [31]. In Ridge regression, however, the value of the Lagrange multiplier λ is often chosen directly by the user instead.

The derivation of the solution of the Lagrange function in terms of λ, $\mathbf{w}^L$ and b^L is well-known (see [31]). We therefore only give the final solution here. For completeness, we give the full derivation of this solution in Appendix 3.A. In Appendix 3.A, we first use the singular value decomposition of the centered hidden state to derive the optimal value for λ. Given this value for λ, the optimal value of the weights is (see Appendix 3.A):

$$\mathbf{w}^L = \left(\left(\mathbf{x}^c\right)^T \mathbf{x}^c + \lambda \mathbf{I}\right)^{-1}\left(\mathbf{x}^c\right)^T \mathbf{y}^c, \tag{3.16}$$

with $\mathbf{y}^c$ a $N \times 1$ vector with the centered true label of all training samples in S, and $\mathbf{x}^c$ a $N \times m$ matrix with the centered hidden states for each training sample and each input node. Here, for one sample $i \in S$, we define the j^{th} centered hidden state as:

$$x_{i,j}^c = x_{i,j}^L - \bar{x}_j^L,, \tag{3.17}$$

with $\bar{x}_j^L$ the mean value of the j^{th} hidden state over all training samples $i \in S$, i.e., $\bar{x}_j^L = \frac{1}{N}\sum_{i\in S} x_{i,j}^L$. The centered true label of a sample i is:

$$y_i^c = y_i - \frac{1}{N}\sum_{i\in S} y_i. \tag{3.18}$$

Last, given these weights, the optimal value of the bias is (see Appendix 3.A):

$$b^L = \frac{1}{N}\sum_{i\in S} y_i - \sum_{j=1}^{m} w_j^L \bar{x}_j^L. \tag{3.19}$$

Note that with this value for b^L, the expected value of $\frac{\partial \text{Loss}}{\partial y_j^{L-1}}$ in eq. (3.10) is zero, since $\mathbb{E}[\hat{y}] = \mathbb{E}[y]$, as shown below:

$$\begin{aligned}
\mathbb{E}[\hat{y}] &= \mathbb{E}\left[b^L + \sum_{j=1}^{m} w_j^L x_j^L\right] \\
&= \mathbb{E}\left[\frac{1}{N}\sum_{i\in S} y_i - \sum_{j=1}^{m} w_j^L \bar{x}_j^L + \sum_{j=1}^{m} w_j^L x_j^L\right] \\
&= \frac{1}{N}\sum_{i\in S} y_i - \sum_{j=1}^{m} w_j^L \bar{x}_j^L + \sum_{j=1}^{m} w_j^L \mathbb{E}[x_j^L] \\
&= \frac{1}{N}\sum_{i\in S} y_i \\
&= \mathbb{E}[y],
\end{aligned} \tag{3.20}$$

where we use that $\mathbb{E}[x_j^L] = \frac{1}{N}\sum_{i\in S} x_{i,j}^L = \bar{x}_j^L$.

3.2.4. Procedure for the weight initialization of a neural network

To initialize the weights of the neural network, we follow the steps below:

1. Randomly initialize the weights of the first $L-1$ layers using Xavier initialization: Sample the weights from a uniform or normal distribution with a mean of zero and the variance as derived in [10].
2. Perform one forward pass to compute the hidden states $x_{i,j}^L$ for all $j \in \{1, 2, \ldots, m\}$ and for all $i \in S$.
3. Solve the constrained minimization problem in eq. (3.14):
   a) Center the hidden input states $x_{i,j}^L$ following eq. (3.17) and center the true labels y_i following eq. (3.18).
   b) Calculate the optimal value of the Lagrange multiplier λ using the singular value decomposition of the centered input values $\mathbf{x}^c$ (see Appendix 3.A).
   c) Initialize the weights of the last layer as in eq.(3.16).
   d) Initialize the bias of the last layer as in eq. (3.19).

The Python code for the proposed weight initialization procedure is in [1].

3.2.5. ASSUMING THE WEIGHTS MUST HAVE ZERO MEAN

In Xavier initialization [10] and Kaiming initialization [11], it is assumed that the expected value of the initialized weights is zero. Most studies therefore initialize the weights from a normal or uniform distribution with a mean of zero. To analyze the potential benefits of this restriction for our approach, we also impose here that the mean of the weights is zero. With this assumption, the final constraint on the sum of the squared weights in eq. (3.13) becomes a constraint on the empirical variance of the weights instead. This is the same as the constraint on the variance of the weights in eq. (12) of [10] (Xavier initialization).

With this extra assumption, our constrained linear regression problem becomes:

$$\begin{aligned} \min_{b^L, w_j^L, j=1,2,\ldots,m} \quad & \sum_{i\in S}\left(y_i - b^L - \sum_{j=1}^{m} w_j^L x_{i,j}^L\right)^2 \\ \text{such that} \quad & \sum_{j=1}^{m}\left(w_j^L\right)^2 = \frac{1+m}{2}, \\ & \sum_{j=1}^{m} w_j^L = 0, \end{aligned} \tag{3.21}$$

with the Lagrange function, $\mathcal{L}(\lambda_1, \lambda_2, b^L, w_j^L, j=1,\ldots,m)$:

$$\begin{aligned} \mathcal{L}\left(\lambda_1, \lambda_2, b^L, w_j^L, j=1,\ldots,m\right) = & \sum_{i\in S}\left(y_i - b^L - \sum_{j=1}^{m} w_j^L x_{i,j}^L\right)^2 + \\ & \lambda_1\left(\sum_{j=1}^{m}\left(w_j^L\right)^2 - \frac{1+m}{2}\right) + \lambda_2 \sum_{j=1}^{m} w_j^L. \end{aligned} \tag{3.22}$$

We solve this Lagrange function for λ_1, λ_2, $\mathbf{w}^L$ and b^L in Appendix 3.B. We first derive the optimal value of λ_1 and λ_2 using the singular value decomposition of $\mathbf{x}^c$. Given these optimal values, we derive the following optimal value for the weights:

$$\mathbf{w}^L = \left(\left(\mathbf{x}^c\right)^T \mathbf{x}^c + \lambda_1 \mathbf{I}\right)^{-1}\left(\left(\mathbf{x}^c\right)^T \mathbf{y}^c - \frac{1}{2}\lambda_2 \mathbf{1}\right), \tag{3.23}$$

with $\mathbf{x}^c$ and $\mathbf{y}^c$ as defined before in eq. (3.17) and (3.18). Given these weights, the bias is calculated as in eq. (3.19).

3.3. CASE STUDY AND RESULTS FOR REGRESSION PROBLEMS

We apply our proposed methodology to estimate the Remaining Useful Life (RUL, time left until failure) of aircraft engines in the C-MAPSS dataset [32]. This dataset contains simulated sensor measurements of aircraft turbofan engines. In total, the measurements of 21 sensors around the engine are considered, such as the pressure at the High Pressure Combustor or the physical fan speed of an engine.

For each sensor, one measurement per engine per flight is simulated by the NASA Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) simulator. Over time, the health of each engine degrades. This degradation is captured by the sensor measurements. Our goal is to develop a model that uses these sensor measurements to estimate the RUL of the engines in the C-MAPPS dataset. The C-MAPPS dataset is widely used in literature to develop RUL prognostic models, with 250 papers published on this dataset [33–35]. More information on this dataset is in [36].

In this chapter, we consider subset FD001 of the C-MAPSS dataset. The training set of FD001 contains 100 engines. For each engine, the sensor measurements are simulated for each flight from the installation of this engine until failure. Subset FD001 also contains a test set with 100 test engines. For each test engine, the sensor measurements are terminated somewhere before the failure of the engine. The goal is to estimate the RUL of the test engine at this point.

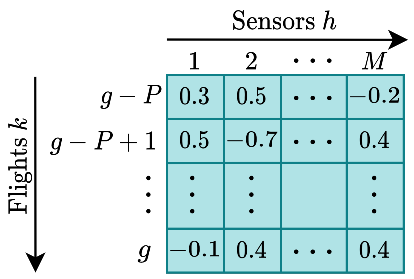


Figure 3.2.: A schematic example of a data sample **Z** that is used as input to the CNN.

We use a Convolutional Neural Network (CNN) to estimate the RUL. Seven of the 21 sensors in the C-MAPSS dataset have constant measurements over time. We thus use the remaining $M = 14$ sensors as input to the CNN. We first normalize the measurements of these 14 sensors using min-max normalization [37]. After a flight g of an engine, we then use a data sample **Z** with these normalized sensor measurements of the past P flights as input to the CNN, i.e.,:

$$\mathbf{Z} = [\mathbf{z}_{g-P}, \mathbf{z}_{g-P+1}, \ldots, \mathbf{z}_g], \tag{3.24}$$

where P is the window size, and z_k are the normalized sensor measurements belonging to flight k:

$$\mathbf{z}_k = [\hat{z}_{k1}, \hat{z}_{k2}, \ldots, \hat{z}_{kM}], \tag{3.25}$$

with $\hat{z}_{kh}$ the normalized sensor measurement of sensor h and flight k. Figure 3.2 shows an example of such an input sample **Z**. The true label of this data sample is the RUL of the considered engine after flight g.

The considered CNN has been proposed in [37, 38]. This CNN consists of 5 convolutional layers. The first 4 convolutional layers each have 10 one-dimensional kernels of size 10×1. The last convolutional layer has one one-dimensional kernel of size 3×1. Same padding is applied to all convolutional layers. After the 5 convolutional layers, two fully connected layers are added. The first fully connected layer has 100 nodes as output, and uses a dropout rate of 0.5. The last connected layer uses these $m = 100$ nodes as input, and outputs the RUL prognostic. This CNN has 45.372 parameters. All layers use the tanh activation function, except the last layer, which uses a linear activation function. Last, following [37, 38], we use the common piece-wise linear RUL target function, where we aim to estimate a RUL of 125 flights when the actual RUL is larger than 125 flights.

We split the engines in the training set in 80 engines for training the neural network, and 20 engines for the validation. Moreover, we use a window size of $P = 30$ in eq. (3.24). With this window size, we create $N = 13890$ data samples for the training set S, and 3841 data samples for the validation set. The weights are further optimized using the Adam optimizer [8], with a batch size of 512 samples, 200 epochs and a learning rate of 0.001. The considered loss function is the (Mean) Squared Error (as in eq. (3.1)). When estimating the RUL of the test engines in the test set, we use the weights that give the lowest validation loss.

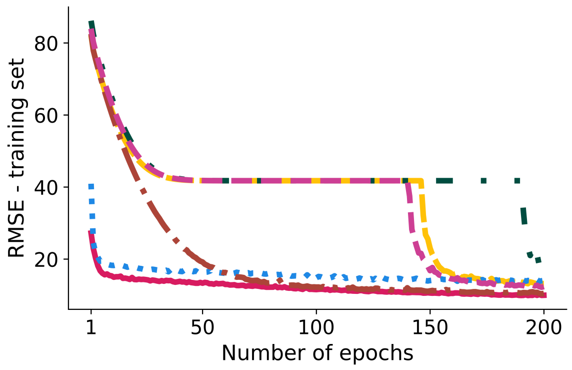
3.3.1. BENCHMARK STRATEGIES

In this section, we compare the convergence of the weights of the neural network for several weight initialization strategies. We refer to our proposed approach to initialize the weights of the last layer, that combines linear regression with Lagrange, as the "Lagrangian LR" strategy. To avoid the vanishing/exploding gradient problem, we impose a constraint on the weights in the proposed strategy. To evaluate the effectiveness of this constraint, we also initialize the weights and biases following our proposed strategy, but without any constraint on the weights of the last layer in eq. (3.14). Instead, we initialize the weights and bias of the last layer with the least squares solution of the linear regression of the true label y_i on the estimated label $\hat{y}_i = b^L + \sum_{j=1}^{m} w_j^L x_{i,j}^L$ (see eq. (3.2)). We refer to this benchmark strategy as "LR" (Linear Regression). As other benchmark strategies, we use Xavier initialization [10] in all layers, including the last layer, Kaiming initialization [11], LeCun initialization [25] and, last, orthogonal initialization [13].

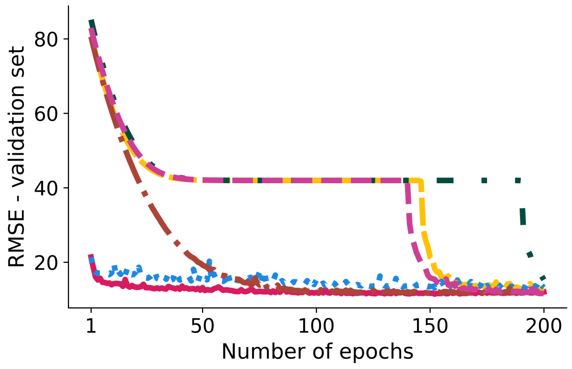
3.3.2. COMPARISON OF DIFFERENT WEIGHT INITIALIZATION STRATEGIES

Figure 3.3 shows the RMSE of the training and validation set after each epoch for the considered strategies. Table 3.1 shows the corresponding minimum value of the RMSE of the training and validation set after various number of epochs. We implement the neural network in Python with PyTorch, and train the neural network on a computer with 4 Intel Core i7 CPU cores. With this computer, it takes 32 seconds to calculate the singular value decomposition of $\mathbf{x}^c$, and it takes less than 1 second to find an optimal value of λ. In total, the weight initialization of the last layer with the proposed approach takes 35 seconds. Training the neural network for one epoch takes on average 10 seconds. Thus, initializing the weights with the proposed approach takes as long as training the neural network for roughly 4 epochs.

After the initialization of the weights using the proposed Lagrangian LR approach,



(a) RMSE of the training set for all epochs.



(b) RMSE of the validation set for all epochs.

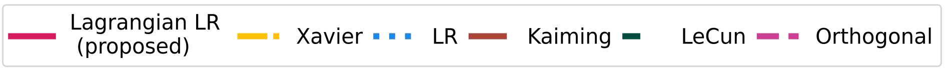


Figure 3.3.: RMSE of the training and validation set after each epoch, for the proposed strategy and the benchmark strategies.

the RMSE of the training set is only 39.0 with and without dropout, while the RMSE of the validation set is only 38.9. The training data is thus not overfitted when initializing the weights with the proposed approach. The weights of the neural network subsequently quickly convergence during the back-propagation algorithm. The lowest validation RMSE of 11.4 is therefore obtained after 152 epochs, after which the validation RMSE does not decrease any further, and even slightly increases. At this point, the weights of the neural network have thus converged. Using the weights that give the lowest validation loss, the RMSE obtained in the test set is only 12.5. Overall, the weights of the neural network thus start at a good point with the considered approach, i.e., with a small initial training and validation loss, and quickly converge during the back-propagation algorithm.

When we initialize the weights of the last layer following the Xavier strategy as

Initialization strategy	Min. RMSE	Number of epochs 1	10	25	50	75	100	150	200	Test RMSE
Proposed strategy										
Lagrangian	Train.	**27.2**	**15.3**	**14.3**	**13.3**	**12.5**	**11.5**	**10.6**	**9.9**	**12.5**
LR (Proposed)	Valid.	21.4	**14.0**	**13.2**	**12.5**	**12.1**	**11.7**	**11.5**	**11.4**	
Benchmark strategies										
Xavier	Train.	82.1	61.4	45.5	41.8	41.8	41.8	23.0	13.0	13.0
[10]	Valid.	80.6	61.8	46.0	42.1	42.0	42.0	21.4	12.6	
LR - Linear	Train.	40.9	18.3	17.0	16.3	15.9	14.8	13.9	13.7	13.2
Regression	Valid.	**21.3**	16.7	15.8	14.8	14.8	14.0	13.4	13.0	
Kaiming	Train.	82.5	60.5	37.6	19.2	13.7	12.2	11.1	10.4	12.7
[11]	Valid.	80.6	60.8	38.0	19.3	13.9	12.1	11.7	11.7	
LeCun	Train.	86.2	65.0	47.0	41.8	41.8	41.8	41.8	16.9	14.2
[25]	Valid.	85.1	65.4	47.6	42.1	42.0	42.0	42.0	15.3	
Orthogonal	Train.	84.0	63.3	46.3	41.8	41.8	41.8	16.8	12.2	12.6
[13]	Valid.	82.9	63.7	46.8	42.0	42.0	42.0	15.8	11.6	

Table 3.1.: The minimum (Min.) RMSE obtained after a various number of epochs for the training (Train.) and validation (Valid.) set, for the proposed strategy and the benchmark strategies. The RMSE of the test set is calculated with the weights that give the lowest validation loss. The lowest RMSE is denoted in bold.

well, the RMSE of the training set is 90.5 with and without dropout, while the RMSE of the validation set is 92.0. The weights of the neural network thus start at a worse point compared to the proposed strategy: The initial validation loss is 2.4 times larger. During the back-propagation algorithm, the weights initially quickly converge, until the weights of the neural network get stuck in a local optimum at epoch 46, when the RMSE of the training set is 41.8. At epoch 147, the neural network escapes the local optimum and the weights of the neural network then quickly further converge, obtaining a minimum validation RMSE of 12.6 after 198 epochs, with a corresponding test RMSE of 13.0. In contrast, our proposed Lagrangian LR strategy already obtains a validation RMSE of 12.6 after only 49 epochs. Thus, we need 75% fewer epochs to reach the same validation loss when using the Lagrangian LR strategy. Comparing the Xavier initialization [10] to the proposed Lagrangian LR strategy, we conclude that the Lagrangian LR strategy i) provides an initial starting point of the weights with a lower training and validation loss, ii) avoids getting stuck in a local minimum, and iii) obtains a quicker converge of the weights.

With the LR strategy, the RMSE of the training set after the weight initialization is 48.3 with and 20.8 without dropout, while the RMSE of the validation set is 19.0. Moreover, the weights of the neural network quickly converge during the first few epochs, leading to a considerable decrease in the RMSE of the training and validation set. However, this convergence is slow compared to the convergence of the weights with the proposed strategy: After 200 epochs, the validation RMSE is 11.4 with the proposed strategy, while it still is 13.0 with the LR strategy. The results with the LR strategy are therefore even worse than with Xavier initialization, where a lower loss for both the training and the validation set is obtained within 200 epochs. It is thus important to not only focus on initializing the weights close to an optimum, but also on initializing the weights such that the vanishing/exploding gradient problem with the weights is mitigated.

The initial loss of the other three benchmark strategies is large, with a validation RMSE around 90 after the weight initialization. Moreover, just as with the Xavier initialization strategy, the neural network gets stuck in a local optimum with the Lecun [25] and orthogonal [13] initialization strategies. However, in the end, the neural network converges to a similar final training, validation and test accuracy for all considered benchmark strategies. The best performing benchmark strategy is the Kaiming weight initialization [11]. For this strategy, the initial RMSE of the training set is 90.4 with and without dropout, while the initial validation RMSE is 91.8. The initial validation RMSE is thus 2.4 times larger than with the proposed strategy. Using Kaiming initialization, however, the weights quickly converge to an optimum without getting stuck in a local optimum. After 148 epochs, the minimum validation RMSE of 11.7 is already reached. However, this validation RMSE is already reached after 97 epochs with the proposed approach, i.e., we need 34% fewer epochs to reach the same validation loss with the proposed approach.

3.3.3. Initialization of the weights with only a part of the training set

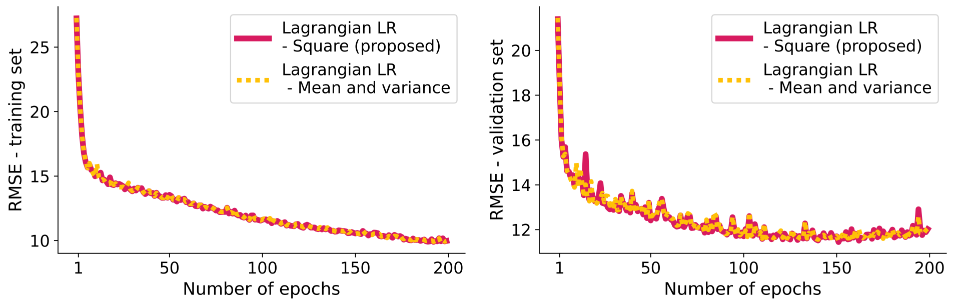
Calculating the singular value decomposition of large training sets is time-consuming. For large training sets, it might therefore be beneficial to initialize the weights with only a part of the training set instead. In this section, we therefore analyze the convergence of the weights if we only employ 10% of the training set to initialize the weights of the last layer. Here, we use 10-fold cross validation by splitting the training set randomly in 10 non-overlapping subsets. For each validation split, we use one subset to initialize the weights of the neural network. We then subsequently train the neural network with the full training set.

The singular value decomposition with 10% of the training set takes on average only 0.41 seconds, while initializing the weights takes on average only 0.83 seconds. This is 42 times faster than when considering the full dataset (see Section 3.3). Table 3.2 shows the mean, minimum and maximum value of the minimum RMSE of the validation and training set after a various number of epochs. Here, the mean, minimum and maximum are taken over the results of the 10 validation splits. We also show the minimum RMSE of the validation and training set when considering the full training set to initialize the weights. The mean training and validation RMSE from the cross-validation is very close to the training and validation RMSE when using the full training set for the weight initialization (a maximum of 0.1 difference). Moreover, after the first 10 epochs, the minimum training and validation RMSE are close together for all 10 validation splits (a maximum of 0.3 difference after the first 10 epochs). For the considered dataset, the results are therefore very similar when using only 10% of the training dataset or the full dataset to initialize the weights, while initializing the weights with only 10% of the training set is 42 times faster.

			Number of epochs						
			1	10	25	50	100	150	200
10% of the training data set used for weight initialization with 10-fold cross validation	Mean RMSE	Train.	27.2	15.4	14.3	13.3	11.6	10.5	9.9
		Valid.	21.4	14.0	13.3	12.5	11.7	11.5	11.5
	Min. RMSE	Train.	27.1	15.2	14.1	13.3	11.5	10.5	9.8
		Valid.	21.0	13.9	13.1	12.5	11.6	11.4	11.3
	Max. RMSE	Train.	27.4	15.5	14.4	13.3	11.6	10.6	10.0
		Valid.	21.7	14.3	13.4	12.6	11.8	11.6	11.6
Full training set used for weight initialization	RMSE	Train.	27.2	15.3	14.3	13.3	11.5	10.6	9.9
		Valid.	21.4	14.0	13.2	12.5	11.7	11.5	11.4

Table 3.2.: The minimum RMSE obtained after various number of epochs for the training (Train.) and validation (Valid.) set using our proposed strategy. Here, we report the results when using the full training set to initialize the weights, and the results of the 10-fold cross validation, where only 10% of the training set is used for weight initialization in each validation split. For the 10-fold cross validation, we report the mean, minimum (min) and maximum (max) value of the minimum RMSE over the 10 performed validation splits.

3.3.4. Weight initialization with a mean weight of zero



(a) RMSE of the training set for all epochs (b) RMSE of the validation set for all epochs

Figure 3.4.: The RMSE of the training set after each epoch, for the proposed strategy with only a restriction on the sum of the squared weights (Lagrangian LR - Square), and for a Lagrangian regression with a restriction on the empirical variance and mean of the weights (Lagrangian LR - Mean and variance).

In this section we analyze the benefits of assuming that the mean of the weights has to be zero. Figure 3.4 shows the RMSE of the training and validation set after each epoch for the proposed approach (Lagrangian LR - Square), and for the proposed approach with the additional restriction that the mean of the weights is zero (Lagrangian LR - Mean and variance, see Section 3.2.5). The RMSE of the training and validation set is nearly equal for both strategies. For our approach, there is thus no additional benefit in assuming that the mean of the weights is zero for the last layer, in contrast with the Xavier [10] and Kaiming [11] weight initialization strategies.

3.4. Case study and results for classification problems

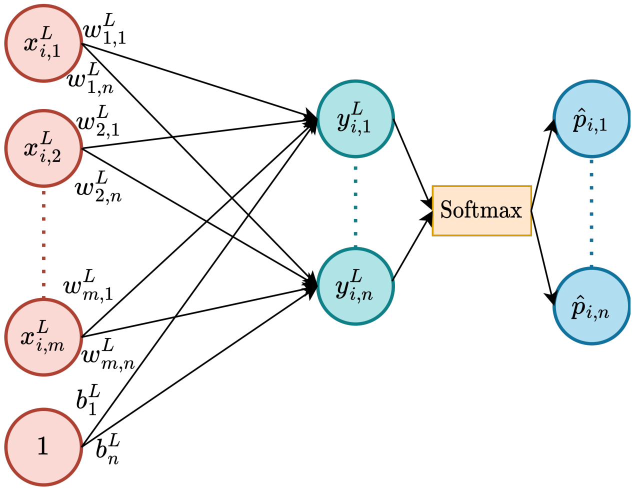


Figure 3.5.: Schematic overview of the last layer of assumed neural network for classification problems for a training sample $i \in S$.

The focus of this chapter is on neural networks that solve a regression problem. For completeness, we apply in this section a variant of our method to neural networks for classification problems. Neural networks that solve a classification problem have a different activation function in the last layer, a different loss function and often a different activation function throughout the neural network than considered in the methodology in Section 3.2. Thus, we cannot directly employ the mathematical derivations from Section 3.2 to these neural networks as well. Instead, we adjust our methodology for classification neural networks as follows.

We first introduce the notation for the last layer of a classification neural network. An overview of this notation is in Figure 3.5. Let $y_i \in \{1, 2, \dots, n\}$ be the true label for a sample $i \in S$. Here, n denotes the number of classes. We assume that the last layer L of a classification neural network still is a fully connected layer. This layer still has m input nodes, but now has n, instead of one, output nodes. As before, let $\mathbf{x}_i^L$ denote the vector with the activated input of layer L for sample $i \in S$. Here, $\mathbf{x}_i^L = [x_{i,1}^L, x_{i,2}^L, \dots, x_{i,m}^L]$, with $x_{i,j}^L$ the j^{th} hidden input state of layer L for sample i.

Let $\mathbf{W}^L = [\mathbf{w}_1^L, \mathbf{w}_2^L, \dots, \mathbf{w}_n^L]$ denote the matrix with the weights of the last layer L. Here, $\mathbf{w}_h^L = [w_{1,h}^L, w_{2,h}^L, \dots, w_{m,h}^L]$ $(h \in \{1,2,\dots,n\})$ denotes the weights connecting the hidden input state $\mathbf{x}_i^L$ of a sample i to the h^{th} output node of this last layer L. Similarly, let $\mathbf{b}^L = [b_1^L, \dots, b_n^L]$ denote the vector with the biases of the last layer L, where b_h^L denotes the bias belonging to the h^{th} output node of layer L. For a sample i, the value $y_{i,h}^L$ of this h^{th} output node is calculated as $y_{i,h}^L = \sum_{j=1}^m w_{j,h}^L x_{i,j}^L + b_h^L$. The total output of layer L for a sample i is denoted by $\mathbf{y}_i^L = [y_{i,1}^L, y_{i,2}^L, \dots, y_{i,n}^L]$.

We use the Softmax activation function to convert the output values $\mathbf{y}_i^L$ to probabilities. The estimated probability $\hat{p}_{i,h}$ that sample i belongs to class h is:

$$\hat{p}_{i,h} = \frac{\exp\left(y_{i,h}^L\right)}{\sum_{g=1}^n \exp\left(y_{i,g}^L\right)} \tag{3.26}$$

The loss of these estimated probabilities is calculated by the cross entropy:

$$\text{Loss} = -\sum_{i\in S}\sum_{h=1}^{n} \mathcal{I}\left(y_i = h\right)\log\left(\hat{p}_{i,h}\right), \tag{3.27}$$

where $\mathcal{I}\left(y_i = h\right)$ is 1 if $y_i = h$, and zero otherwise. Last, we assign each sample i to the class h with the largest estimated probability $\hat{p}_{i,h}$. Let $\hat{y}_i$ denote the class to which sample i is assigned. From this, we calculate the accuracy:

$$\text{Accuracy} = 100 \cdot \frac{\sum_{i\in S}\mathcal{I}\left(y_i = \hat{y}_i\right)}{N} \tag{3.28}$$

As before, we initialize the weights of the first $L-1$ layers of the neural network randomly, using several weight initialization methods. The aim is now to find the weights $\mathbf{W}^L$ and the bias $\mathbf{b}^L$ of the last layer L such that the cross entropy loss is minimized. However, we again want to prevent the exploding/vanishing gradient problem. With a logistic regression, we do not know analytically the optimal value of the sum of the squared weights. We also do not know how to analytically derive a value for λ given such a sum. However, we can still apply Ridge logistic regression with a randomly chosen λ to regularize the weights. In Ridge logistic regression, we minimize the following function [31]:

$$\min_{\mathbf{W}^L, \mathbf{b}^L} = -\sum_{i\in S}\sum_{h=1}^{n} \mathcal{I}\left(y_i = h\right)\log\left(\hat{p}_{i,h}\right) + \lambda\left(\sum_{h=1}^{n}\sum_{j=1}^{m}\left(w_{j,h}^L\right)^2\right). \tag{3.29}$$

This is very similar to the Lagrangian function in eq. (3.15). The main difference is that we now consider another loss function and that we choose λ ourselves. Future research should be conducted to derive a good value for λ (either empirically or analytically) in classification problems. To initialize the weights in the last layer L, we thus perform a forward pass to obtain the hidden input states $\mathbf{x}_i^L$ for all samples $i \in S$. We then use the Ridge logistic regression of the actual labels y_i on the hidden states $\mathbf{x}_i^L$ (for all samples $i \in S$) to find the weights $\mathbf{W}^L$ and the biases $\mathbf{b}^L$ that minimize the loss given λ.

3.4.1. Case study with the CIFAR-100 dataset

We test the above approach on the images of the CIFAR-100 dataset [24]. Each image in this dataset has a size of 32 by 32 pixels, and belongs to one out of a hundred classes, such as “bicycle” or “camel”. The objective is to correctly classify the images. The dataset is divided into 10.000 test images, and 50.000 training images. We further divide the training images into 45.000 images for training, and 5.000 images for validation. We preprocess the data by scaling the input images using z-score standardization. Moreover, to prevent overfitting, we apply Random Augmentation [39] on the 45.000 training images, with 3 operations of a magnitude of 15.

With this training images, we train two well-known neural networks to classify the images, namely ResNet-18 and ResNet-34 [23]. These neural networks are both variants on the general ResNet developed in [23]. ResNet-18 is the smallest variant, with 18 layers and over 11 million parameters, while ResNet-34 is a larger variant with 34 layers and over 21 million parameters. The last layer of both neural networks is a single fully connected layer, to which we apply our approach. ResNets are initially developed for the ImageNet dataset, which has relatively large images. In [23], all images in this dataset are cropped to have a size of 224 by 224 pixels. Following this, we therefore resize the images in the CIFAR-100 dataset to a size of 224 to 224 pixels as well, using bilinear interpolation. We train both neural networks for 100 epochs using stochastic gradient descent with a momentum of 0.9, a batch size of 256 and weight decay of 1^{-4}, to prevent overfitting. The initial learning rate is 0.01, and is multiplied by 0.1 after every 25 epochs.

We implement the logistic regression with Ridge using the Scikit-learn package [40], with the “lbfgs” solver. This solver applies a quasi-Newton method [31] to optimize the weights of the logistic regression. We select λ from $\{1, 10, 100, 1000, 10000\}$. Specifically, we first calculate the sum of the squared weights in the last layer L when we initialize all weights of the neural network with Xavier initialization. Let the value of this sum be denoted by α. We then select the smallest value of $\lambda \in \{1, 10, 100, 1000, 10000\}$ for which the sum of the squared weights in the last layer is equal to or smaller than this value α. For both ResNet-18 and ResNet-34, this procedure gives $\lambda = 1000$. Last, we perform the logistic regression on all training images without any random augmentation. With this, it takes between 105 and 143 seconds to initialize the weights in the last layer with Ridge logistic regression for ResNet-18, and between 135 and 211 seconds for ResNet-34. Training the neural network for one epoch on 1 NVIDIA Tesla V100S GPU takes between 80 and 90 seconds for ResNet-18, and between 90 and 100 seconds for ResNet-34. The logistic regression thus takes roughly as long as training the neural network for two epochs.

Since we do not analytically derive the optimal value for λ, we are not restricted to applying the proposed methodology after Xavier weight initialization only. Instead, we initialize the weights in the last layer with Ridge logistic regression in combination with each benchmark weight initialization method. For simplicity, we use the same value of λ for all weight initialization methods.

3.4.2. RESULTS FOR THE CIFAR-100 DATASET WITH TRAINING A NEURAL NETWORK FROM SCRATCH

Weight initialize	Init. last layer	Max. acc.(%)	Number of epochs 0	10	25	50	75	100	Test acc.(%)
			ResNet-18 [23]						
Kaiming (original) [11]	[11]	Train	1.01	22.44	45.59	55.38	56.93	57.14	62.20
		Valid.	1.00	30.42	49.98	61.30	62.16	62.20	
	Prop.	Train	**3.72**	**24.82**	**46.64**	**56.92**	**58.28**	**58.64**	**63.47**
		Valid.	**9.34**	**34.72**	**52.26**	**62.46**	**63.30**	**63.30**	
Xavier [10]	[10]	Train.	1.01	27.73	51.46	63.90	66.13	66.13	67.24
		Valid.	1.00	38.83	54.66	66.10	67.14	67.14	
	Prop.	Train.	**3.73**	**31.14**	**52.84**	**65.38**	**66.82**	**67.36**	**67.83**
		Valid.	**5.43**	**39.00**	**55.58**	**67.46**	**68.20**	**68.32**	
LeCun [25]	[25]	Train.	1.01	28.51	51.71	63.83	65.67	66.00	67.06
		Valid.	1.22	37.26	53.44	66.52	67.68	67.84	
	Prop.	Train.	**3.72**	**31.06**	**52.65**	**65.01**	**66.75**	**67.09**	**67.47**
		Valid.	**9.34**	**40.26**	**55.94**	**67.02**	**68.08**	**68.32**	
Orthogonal [13]	[13]	Train.	1.05	28.13	51.87	64.17	66.14	66.42	**67.50**
		Valid.	1.12	34.32	55.44	66.08	67.06	67.08	
	Prop.	Train.	**4.30**	**30.82**	**53.13**	**65.29**	**66.94**	**67.43**	67.20
		Valid.	**5.96**	**37.30**	**56.48**	**67.04**	**68.04**	**68.04**	

Table 3.3.: The maximum accuracy (max. acc.) in percent obtained with ResNet-18 after a various number of epochs for the training (Train.) and validation (Valid.) set. Here, we consider several weight initialization strategies, with two different ways to initialize the weights in the last layer ("Init. last layer"); According to the considered weight initialization strategy, or with the proposed strategy of Ridge logistic regression ("Prop."). The accuracy of the test set is calculated with the weights that give the lowest validation loss. The highest accuracy per weight initialization strategy is denoted in bold.

Table 3.3 and Table 3.4 shows the maximum accuracy after training ResNet-18 and ResNet-34 for several number of epochs, respectively. The accuracy after epoch 0 is the initial accuracy, i.e., the accuracy after the weight initialization without any training of the neural network. The initial validation accuracy is between 4.12% and 9.34% for ResNet-18, and between 3.02% and 7.22% for ResNet-34. This initial accuracy is relatively small compared to the initial loss of the regression problem, which is already halfway between the loss with random weight initialization and the final obtained loss after training. This might be because we consider a more complicated problem, with 100 instead of only one possible output. Moreover, we consider a more complicated neural network: In the regression problem, the neural network has 45.372 parameters, while ResNet-18 and ResNet-34 have over 11

Weight initialize	Init. last layer	Max. acc.(%)	Number of epochs 0	10	25	50	75	100	Test acc.(%)
ResNet-34 [23]									
Kaiming (original) [11]	[11]	Train.	0.94	24.23	51.59	65.04	67.62	67.90	66.04
		Valid.	0.86	33.18	54.42	65.00	65.66	65.66	
	Prop.	Train.	**2.79**	**28.50**	**54.00**	**68.64**	**70.42**	**70.90**	**67.23**
		Valid.	**7.22**	**38.76**	**56.24**	**66.14**	**67.12**	**67.26**	
Xavier [10]	[10]	Train.	0.94	30.38	56.87	72.77	75.53	76.00	69.52
		Valid.	0.86	37.94	57.70	69.28	70.16	70.16	
	Prop.	Train.	**2.78**	**34.37**	**58.93**	**75.65**	**77.88**	**78.73**	**69.84**
		Valid.	**7.20**	**40.96**	**59.64**	**69.62**	**70.38**	**70.38**	
LeCun [25]	[25]	Train.	0.94	31.38	57.45	73.25	75.80	76.09	68.56
		Valid.	0.86	38.38	60.14	69.46	70.22	70.26	
	Prop.	Train.	**2.79**	**34.30**	**58.96**	**75.53**	**77.69**	**78.04**	**69.86**
		Valid.	**7.22**	**40.42**	**60.56**	**70.16**	**70.42**	**70.50**	
Orthogonal [13]	[13]	Train.	1.00	31.23	57.85	74.12	76.71	77.35	69.13
		Valid.	1.02	39.08	**59.54**	69.40	70.06	70.24	
	Prop.	Train.	**2.37**	**34.95**	**59.12**	**76.19**	**78.13**	**78.66**	**69.99**
		Valid.	**6.84**	**40.60**	58.12	**69.42**	**70.34**	**70.38**	

Table 3.4.: The maximum accuracy (max. acc.) in percent obtained with ResNet-34 after a various number of epochs for the training (Train.) and validation (Valid.) set. Here, we consider several weight initialization strategies, with two different ways to initialize the weights in the last layer("Init. last layer"); According to the considered weight initialization strategy, or with the proposed strategy of Ridge logistic regression (Proposed). The accuracy of the test set is calculated with the weights that give the lowest validation loss. The highest accuracy per weight initialization strategy is denoted in bold.

million and 21 million parameters respectively. With many more randomly initialized parameters, the characteristics of the input might have largely disappeared in the hidden state of the last layer for both ResNets, compared to the hidden state of the last layer of the regression neural network. This also might be the reason that ResNet-34 has a lower initial validation accuracy than ResNet-18.

However, for both neural networks, the initial accuracy with the proposed weight initialization method is still higher than without, for all considered initialization methods. This leads to a slightly higher training and validation accuracy throughout the training process: For both ResNet-18 and ResNet-34 and with all initialization methods, the training and validation accuracy is slightly higher when applying Ridge logistic regression to initialize the weights of the last layer. The only exception is the validation accuracy after 25 epochs with orthogonal initialization for ResNet-34, which is higher when not using Ridge logistic regression in the beginning. However, also in this case, applying Ridge logistic regression gives a higher validation accuracy after 0, 10, 50, and 100 epochs. With the proposed methodology, we thus obtain a slightly faster convergence of the weights.

For all initialization methods, the final training and validation accuracy are slightly higher with the proposed methodology. The test accuracy is also higher when the weights are initialized with Ridge logistic regression, except when we use orthogonal initialization in ResNet-18: In this case, the test accuracy is 67.50% when using the orthogonal initialization without Ridge logistic regression, while it is 67.20% when using orthogonal initialization with Ridge logistic regression. The highest test accuracy of 67.83% for ResNet-18 is obtained when combining Xavier initialization with our approach, while the highest test accuracy of 69.99% for ResNet-34 is obtained when combining orthogonal initialization with our approach.

3.4.3. Results for the CIFAR-100 dataset with transfer learning

Classification neural networks are often trained using transfer learning., i.e., the weights of the neural network are (partly) initialized with the weights from another neural network with (partly) the same structure, trained on another dataset (called the source dataset). The weights are then fine-tuned for the target dataset using a gradient descent method [7].

In this section, we test how our approach works performs combined with transfer learning. We therefore initialize all weights, except the weights of the last layer, in ResNet-18 and ResNet-34 with the pre-trained weights of Torchvision in Pytorch. These weights are obtained by training the neural networks to classify the images in the ImageNet-1K dataset [41]. This dataset contains 1000 classes, instead of 100. Because there are different classes in the ImageNet-1K dataset and in the CIFAR-100 dataset, we cannot initialize the weights of the last layer with the weights from Torchvision. Instead, we initialize the weights in the last layer with the proposed methodology (Ridge logistic regression), and with all benchmark methods. For the Ridge logistic regression, we use the same λ as in Section 3.4.2. We then train the neural network for 50 epochs using stochastic gradient descent with a momentum of 0.9, a batch size of 256 and a weight decay of 1^{-4}. The initial learning rate is still 0.01, but we now multiply this learning rate with 0.1 after 10 and 30 epochs.

The results are in Table 3.5. The initial validation accuracy after epoch 0, i.e., without training the neural network, is already 53% and 55% for ResNet-18 and ResNet-34 respectively. Because of this, the weights converge fast to a (local) optimum: Throughout the training process, both the training and the validation accuracy are higher with the proposed weight initialization method than with the benchmark methods. For both ResNets, LeCun initialization [25] is the best benchmark method. For ResNet-18, LeCun initialization obtains its highest validation accuracy of 81.32% after 49 epochs, while we already obtain this validation accuracy in 13 epochs. For ResNet-34, LeCun initialization obtains its highest validation accuracy of 84.18% after 50 epochs, while we already obtain this validation accuracy after 14 epochs. Moreover, with the proposed methodology, we obtain a higher final training, validation and test accuracy within 50 epochs than with all benchmark

Weight initialization last layer	Max acc.(%)	Number of epochs 0	10	20	30	40	50	Test acc.(%)
ResNet-18 [23]								
Proposed	Train.	**20.70**	**73.04**	**81.59**	**83.46**	**84.27**	**84.76**	**81.76**
	Valid.	**53.68**	**78.24**	**81.86**	**81.86**	**81.86**	**81.86**	
Kaiming [11]	Train.	1.01	70.55	79.34	81.50	82.43	82.93	80.25
	Valid.	1.16	75.98	80.84	80.84	81.06	81.06	
Xavier [10]	Train.	1.02	70.96	79.71	81.92	83.00	83.27	79.97
	Valid.	0.98	76.02	80.52	80.74	80.80	80.98	
LeCun [25]	Train.	0.95	71.38	80.00	82.04	83.14	83.50	80.54
	Valid.	1.38	76.48	80.86	81.06	81.24	81.32	
Orthogonal [13]	Train.	1.06	71.54	80.22	82.09	82.99	83.51	80.92
	Valid.	1.20	76.48	80.88	81.28	81.36	81.44	
ResNet-34 [23]								
Proposed	Train.	**22.04**	**78.67**	**88.07**	**90.09**	**90.93**	**90.96**	**84.57**
	Valid.	**55.12**	**80.60**	**84.86**	**85.24**	**85.24**	**85.28**	
Kaiming [11]	Train.	0.86	71.63	83.81	86.24	87.37	87.76	82.34
	Valid.	0.94	76.56	83.04	83.04	83.26	83.26	
Xavier [10]	Train.	0.86	73.27	85.16	86.99	88.39	88.59	82.91
	Valid.	0.94	77.62	82.78	82.78	82.98	82.98	
LeCun [25]	Train.	0.86	76.51	87.12	88.96	90.13	90.38	83.03
	Valid.	0.94	78.14	83.54	84.12	84.12	84.18	
Orthogonal [13]	Train.	0.90	76.73	87.36	89.03	90.02	90.34	83.10
	Valid.	0.88	78.22	83.98	83.98	84.06	84.10	

Table 3.5.: The maximum accuracy (max. acc.) in percent obtained after a various number of epochs for the training (Train.) and validation (Valid.) set when using transfer learning. For the weights in the last layer, we consider several benchmark weight initialization strategies and the proposed strategy of Ridge logistic regression (Proposed). The accuracy of the test set is calculated with the weights that give the lowest validation loss. The highest accuracy is denoted in bold.

methods. The convergence of the weights is thus accelerated by combining Ridge logistic regression with transfer learning.

3.5. Conclusions

In this chapter, we introduce a new initialization method for the weights in the last layer of a neural network. We assume that this neural network solves a regression problem and that it uses an activation function that has a unit derivative of 1 at 0. Here we focus both on i) accelerating the convergence of the weights during the back-propagation algorithm by mitigating the vanishing/exploding gradient problem, and ii) initializing the weights close to an optimum point by minimizing the initial training loss. To accelerate the convergence of the weights, we impose that the variance of the outputs and the gradients of each layer in the neural network should be equal after the weight initialization, following [10]. From this requirement, we analytically derive a constraint on the weights in the last layer. We then analytically derive the optimal weights and bias of the last layer, i.e., the weights and bias that minimize the initial training loss, while fulfilling this derived constraint.

We apply this initialization strategy to a CNN that estimates the RUL of aircraft engines. Our proposed strategy initializes the weights such that the initial training and validation loss are relatively small. Moreover, the proposed strategy prevents that the weights of the CNN get stuck in a local optimum. The weights therefore converge very fast. The minimum validation loss obtained with Xavier initialization [10] after 198 epochs is already obtained after only 49 epochs with our approach. Moreover, compared to the best benchmark strategy (Kaiming initialization [11]), we need 34% fewer epochs to reach the same validation loss. To further analyse our proposed methodology, we also show that it is sufficient for the considered data set to use only a part of the training set for the weight initialization. Moreover, we show that with our proposed initialization approach, it is not necessary to assume that the mean of the weights is zero, as in the weight initialization strategies of [10, 11].

Last, we adjust our proposed methodology to apply it to a neural network with any type of activation function that solves a classification problem, by using logistic regression with Ridge regularization. We apply this to ResNet-18 and ResNet-34 [23], and classify the images in the CIFAR-100 dataset [24]. When training the ResNets from scratch, we obtain a slightly higher initial accuracy and a slightly faster convergence of the weights with our approach. However, our approach works best when combined with transfer learning. In this case, the initial validation accuracy is already 53% and 55% for ResNet-18 and ResNet-34 respectively. This leads to a faster weight convergence and a higher test accuracy than with the benchmark methods.

LIMITATIONS AND FUTURE RESEARCH

In this chapter, we consider a specific type of problem (i.e., regression problem), with a specific type of neural network (i.e., specific type of activation function). Many problems in machine learning, however, do not satisfy these assumptions. They either do not solve a regression problem and/or use other activation functions. For future research, we therefore plan to extend both the mathematical analysis and the experiments to other types of problems, neural networks and datasets. Specifically, we would like to consider other activation functions, and to empirically find a good value for λ when this is not possible analytically.

Moreover, we have also discussed applying the considered methodology to classification neural networks with any type of activation function. Here, we choose the value of the Lagrangian multiplier λ ourselves. Future work could further develop the approach for these classification neural networks. We find it particularly interesting to determine a good value for λ, either analytically or empirically.

Appendix 3.A. Solution of the minimization problem (eq. (3.14)) for λ

In this appendix, we solve the constrained linear regression problem of eq. (3.14) to find the optimal value of λ following [31]. The Lagrange function $\mathcal{L}(\lambda, b^L, w_j^L, j = 1,2,\ldots,m)$ of the minimization problem is:

$$\mathcal{L}\left(\lambda, b^L, w_j^L, j = 1,2,\ldots,m\right) = \sum_{i\in S}\left(y_i - b^L - \sum_{j=1}^{m} w_j^L x_{i,j}^L\right)^2 + \lambda\left(\sum_{j=1}^{m}\left(w_j^L\right)^2 - \frac{1+m}{2}\right). \quad (3.30)$$

Scaling of the inputs

The solution of this constrained minimization problem is not equivalent to scaling the inputs or the outputs [31]. Following [31], we first normalize the variables:

$$\begin{aligned}
&\mathcal{L}\left(\lambda, b^L, w_j^L, j = 1,2,\ldots,m\right) && (3.31)\\
&= \sum_{i\in S}\left(y_i - b^L - \sum_{j=1}^{m} w_j^L \bar{x}_j^L + \sum_{j=1}^{m} w_j^L \bar{x}_j^L - \sum_{j=1}^{m} w_j^L x_{i,j}^L\right)^2 + \lambda\left(\sum_{j=1}^{m}\left(w_j^L\right)^2 - \frac{1+m}{2}\right)\\
&= \sum_{i\in S}\left(y_i - b^L - \sum_{j=1}^{m} w_j^L \bar{x}_j^L - \sum_{j=1}^{m} w_j^L\left(x_{i,j}^L - \bar{x}_j^L\right)\right)^2 + \lambda\left(\sum_{j=1}^{m}\left(w_j^L\right)^2 - \frac{1+m}{2}\right),
\end{aligned}$$

with $\bar{x}_j^L$ the mean value of the j^{th} hidden state over all training samples $i \in S$, i.e., $\bar{x}_j^L = \frac{1}{N}\sum_{i\in S} x_{i,j}^L$. We define the centered weight w_j^c and the centered bias b^c as:

$$w_j^c = w_j^L, \quad (3.32)$$

$$b^c = b^L + \sum_{j=1}^{m} w_j^L \bar{x}_j^L. \quad (3.33)$$

With this, we obtain the following expression for the Lagrange function:

$$\begin{aligned}
\mathcal{L}\left(\lambda, b^L, w_j^L, j = 1,2,\ldots,m\right) &= \mathcal{L}\left(\lambda, b^c, w_j^c, j = 1,2,\ldots,m\right) && (3.34)\\
&= \sum_{i\in S}\left(y_i - b^c - \sum_{j=1}^{m} w_j^c\left(x_{i,j}^L - \bar{x}_j^L\right)\right)^2 + \lambda\left(\sum_{j=1}^{m}(w_j^c)^2 - \frac{1+m}{2}\right).
\end{aligned}$$

Following [31], we first analyse the optimal value of b^c. The derivative of the Lagrange function with respect to b^c is:

$$\frac{\partial}{\partial b^c}\mathcal{L}\left(\lambda, b^c, w_j^c, j = 1,2,\ldots,m\right) = -2\sum_{i\in S}\left(y_i - b^c - \sum_{j=1}^{m} w_j^c\left(x_{i,j}^L - \bar{x}_j^L\right)\right) \quad (3.35)$$

To find the optimum, we set this derivative equal to zero:

$$\sum_{i\in S}(y_i - b^c) - \sum_{i\in S}\sum_{j=1}^{m} w_j^c\left(x_{i,j}^L - \bar{x}_j^L\right) = 0 \quad (3.36)$$

First, let us analyse $\sum_{i\in S}\sum_{j=1}^{m} w_j^c\left(x_{i,j}^L - \bar{x}_j^L\right)$:

$$\begin{aligned}\sum_{i\in S}\sum_{j=1}^{m} w_j^c\left(x_{i,j}^L - \bar{x}_j^L\right) &= \sum_{j=1}^{m}\sum_{i\in S} w_j^c x_{i,j}^L - \sum_{j=1}^{m}\sum_{i\in S} w_j^c \bar{x}_j^L \\ &= \sum_{j=1}^{m} w_j^c N \bar{x}_j^L - \sum_{j=1}^{m} N w_j^c \bar{x}_j^L \\ &= 0.\end{aligned} \tag{3.37}$$

Using this in eq. (3.36) gives the optimal value for b^c [31]:

$$b^c = \frac{1}{N}\sum_{i\in S} y_i. \tag{3.38}$$

We therefore center the input and output values of the linear regression as:

$$x_{i,j}^c = x_{i,j}^L - \bar{x}_j^L, \tag{3.39}$$

$$y_i^c = y_i - \frac{1}{N}\sum_{i\in S} y_i. \tag{3.40}$$

This gives the following Lagrange function:

$$\begin{aligned}\mathcal{L}\left(\lambda, b^c, w_j^c, j = 1,2,\ldots,m\right) &= \mathcal{L}\left(\lambda, w_j^c, j = 1,2,\ldots,m\right) \\ &= \sum_{i\in S}\left(y_i^c - \sum_{j=1}^{m} w_j^c x_{i,j}^c\right)^2 + \lambda\left(\sum_{j=1}^{m}\left(w_j^c\right)^2 - \frac{1+m}{2}\right).\end{aligned} \tag{3.41}$$

In matrix form, this is:

$$\mathcal{L}\left(\lambda, \mathbf{w}^c\right) = \left(\mathbf{y}^c - \mathbf{x}^c\mathbf{w}^c\right)^T\left(\mathbf{y}^c - \mathbf{x}^c\mathbf{w}^c\right) + \lambda\left(\left(\mathbf{w}^c\right)^T\left(\mathbf{w}^c\right) - \frac{1+m}{2}\right), \tag{3.42}$$

with $\mathbf{y}^c$ a $N\times 1$ vector with the centered true label of all training samples in S, $\mathbf{w}^c$ a $m\times 1$ vector with the centered weights of the last layer L, and $\mathbf{x}^c$ a $N\times m$ matrix with the centered hidden states for each training sample and each input node.

SOLUTION OF THE LAGRANGE FUNCTION USING THE SINGULAR VALUE DECOMPOSITION

To solve the Lagrange function, we solve the system of equations:

$$\nabla_{\mathbf{w}^c}\mathcal{L}(\lambda, \mathbf{w}^c) = \mathbf{0}, \tag{3.43}$$

$$\nabla_{\lambda}\mathcal{L}(\lambda, \mathbf{w}^c) = 0, \tag{3.44}$$

where $\mathbf{0}$ is a $m\times 1$ vector with zeros. Given λ, we solve the first gradient $\nabla_{\mathbf{w}^c}\mathcal{L}(\lambda, \mathbf{w}^c) = \mathbf{0}$ with respect to $\mathbf{w}^c$. Solving this gradient gives the well-known solution of Ridge linear regression [31]:

$$\nabla_{\mathbf{w}^c}\mathcal{L}(\lambda, \mathbf{w}^c) = \mathbf{0} \tag{3.45}$$

$$\Rightarrow -2\left(\mathbf{x}^c\right)^T\left(\mathbf{y}^c - \mathbf{x}^c\mathbf{w}^c\right) + 2\lambda\mathbf{w}^c = \mathbf{0}$$
$$\Rightarrow \mathbf{w}^c = \left(\left(\mathbf{x}^c\right)^T\mathbf{x}^c + \lambda\mathbf{I}\right)^{-1}\left(\mathbf{x}^c\right)^T\mathbf{y}^c,$$

with $\mathbf{I}$ a $m \times m$ identity matrix.

We then use the singular value decomposition to solve the gradient with respect to λ as well. The singular value decomposition of $\mathbf{x}^c$ is [42]:

$$\mathbf{x}^c = \mathbf{UDV}^T, \tag{3.46}$$

where $\mathbf{U}$ is an orthogonal $N \times N$ matrix (so $\mathbf{U}^{-1} = \mathbf{U}^T$) and $\mathbf{V}$ is an orthogonal $m \times m$ matrix (so $\mathbf{V}^{-1} = \mathbf{V}^T$). Moreover, $\mathbf{D}$ is a $N \times m$ "diagonal" matrix, with the singular values s of $\mathbf{x}^c$ on the diagonal [42]. Using this decomposition, the optimal value for the centered weights w^c becomes [31]:

$$\begin{aligned}\mathbf{w}^c &= \left(\left(\mathbf{x}^c\right)^T\mathbf{x}^c + \lambda\mathbf{I}\right)^{-1}\left(\mathbf{x}^c\right)^T\mathbf{y}^c \\ &= \left(\mathbf{VD}^T\mathbf{DV}^T + \lambda\mathbf{VV}^T\right)^{-1}\left(\mathbf{UDV}^T\right)^T\mathbf{y}^c \\ &= \left(\mathbf{V}\left(\mathbf{D}^T\mathbf{D} + \lambda\mathbf{I}\right)\mathbf{V}^T\right)^{-1}\mathbf{VD}^T\mathbf{U}^T\mathbf{y}^c \\ &= \mathbf{V}\left(\mathbf{D}^T\mathbf{D} + \lambda I\right)^{-1}\mathbf{D}^T\mathbf{U}^T\mathbf{y}^c\end{aligned} \tag{3.47}$$

The sum of the centered weights $(\mathbf{w}^c)^T\mathbf{w}^c$ thus becomes:

$$\begin{aligned}\left(\mathbf{w}^c\right)^T\mathbf{w}^c &= \left(\mathbf{V}\left(\mathbf{D}^T\mathbf{D} + \lambda I\right)^{-1}\mathbf{D}^T\mathbf{U}^T\mathbf{y}^c\right)^T\mathbf{V}\left(\mathbf{D}^T\mathbf{D} + \lambda I\right)^{-1}\mathbf{D}^T\mathbf{U}^T\mathbf{y}^c \\ &= \left(\mathbf{y}^c\right)^T\mathbf{UD}\left(\mathbf{D}^T\mathbf{D} + \lambda I\right)^{-2}\mathbf{D}^T\mathbf{U}^T\mathbf{y}^c.\end{aligned} \tag{3.48}$$

Let $\mathbf{b} = \mathbf{U}^T\mathbf{y}^c$, a vector of size $N \times 1$. Let p_i denote the i^{th} element of a vector p, and let s_i be the i^{th} singular value of $\mathbf{x}^c$. Then, it follows that:

$$\begin{aligned}\left(\mathbf{w}^c\right)^T\mathbf{w}^c &= \mathbf{b}^T\mathbf{D}\left(\mathbf{D}^T\mathbf{D} + \lambda I\right)^{-2}\mathbf{D}^T\mathbf{b} \\ &= \sum_{j=1}^{m}\frac{b_j^2 s_j^2}{(s_j^2 + \lambda)^2}.\end{aligned} \tag{3.49}$$

We set this equal to the final constraint (eq. (3.13)):

$$\sum_{j=1}^{m}\frac{b_j^2 s_j^2}{(s_j^2 + \lambda)^2} = \frac{1+m}{2}, \tag{3.50}$$

which can be solved numerically.

Appendix 3.B. Solution of the minimization problem with a mean weight of zero (eq. (3.21)) for λ

In this appendix, we solve the constrained linear regression problem of eq. (3.21) (with a mean weight of zero) to find the optimal value of λ. The Lagrange function

$\mathcal{L}(\lambda_1, \lambda_2, b^L, w_j^L, j = 1, 2, \ldots, m)$ of this problem is:

$$\mathcal{L}\left(\lambda_1, \lambda_2, b^L, w_j^L, j = 1, \ldots, m\right) = \sum_{i \in S} \left(y_i - b^L - \sum_{j=1}^{m} w_j^L x_{i,j}^L \right)^2 + \lambda_1 \left(\sum_{j=1}^{m} \left(w_j^L\right)^2 - \frac{1+m}{2} \right) + \lambda_2 \sum_{j=1}^{m} w_j^L. \tag{3.51}$$

We scale the hidden states and true labels in the same way as in Appendix 3.A, eq. (3.39) and (3.40):

$$\begin{aligned}\mathcal{L}\left(\lambda_1, \lambda_2, b^L, w_j^L, j = 1, 2, \ldots, m\right) &= \mathcal{L}\left(\lambda_1, \lambda_2, w_j^c, j = 1, 2, \ldots, m\right) \\ &= \sum_{i \in S} \left(y_i^c - \sum_{j=1}^{m} w_j^c x_{i,j}^c \right)^2 + \lambda_1 \left(\sum_{j=1}^{m} \left(w_j^c\right)^2 - \frac{1+m}{2} \right) \\ &+ \lambda_2 \sum_{j=1}^{m} w_j^c.\end{aligned} \tag{3.52}$$

In matrix form, this normalized Lagrange function becomes:

$$\mathcal{L}\left(\lambda_1, \lambda_2, \mathbf{w}^c\right) = \left(\mathbf{y}^c - \mathbf{x}^c \mathbf{w}^c\right)^T \left(\mathbf{y}^c - \mathbf{x}^c \mathbf{w}^c\right) + \lambda_1 \left(\left(\mathbf{w}^c\right)^T \mathbf{w}^c - \frac{1+m}{2} \right) + \lambda_2 \mathbf{1}^T \mathbf{w}^c, \tag{3.53}$$

with $\mathbf{1}$ a vector with ones of size $m \times 1$.

SOLUTION OF THE LAGRANGE FUNCTION WITH A MEAN WEIGHT OF ZERO

To solve the Lagrange function, we solve the system of equations:

$$\nabla_{\mathbf{w^c}} \mathcal{L}\left(\lambda_1, \lambda_2, \mathbf{w}^c\right) = \mathbf{0}, \tag{3.54}$$

$$\nabla_{\lambda_1} \mathcal{L}\left(\lambda_1, \lambda_2, \mathbf{w}^c\right) = 0 \tag{3.55}$$

$$\nabla_{\lambda_2} \mathcal{L}\left(\lambda_1, \lambda_2, \mathbf{w}^c\right) = 0 \tag{3.56}$$

Given λ_1 and λ_2, we solve the first gradient $\nabla_{\mathbf{w}^c} \mathcal{L}(\lambda_1, \lambda_2, \mathbf{w}^c) = \mathbf{0}$ with respect to $\mathbf{w}^c$:

$$\begin{aligned}&\nabla_{\mathbf{w}^c} \mathcal{L}\left(\lambda_1, \lambda_2, \mathbf{w}^c\right) = \mathbf{0} \\ &\Rightarrow -2\left(\mathbf{x}^c\right)^T \left(\mathbf{y}^c - \mathbf{x}^c \mathbf{w}^c\right) + 2\lambda_1 \mathbf{w}^c + \lambda_2 \mathbf{1} = 0 \\ &\Rightarrow \mathbf{w}^c = \left(\left(\mathbf{x}^c\right)^T \mathbf{x}^c + \lambda_1 \mathbf{I} \right)^{-1} \left(\left(\mathbf{x}^c\right)^T \mathbf{y}^c - \frac{1}{2}\lambda_2 \mathbf{1} \right).\end{aligned} \tag{3.57}$$

We again use the singular value decomposition of $\mathbf{x}^c$ (eq. (3.46)) to solve the Lagrange function for λ_1 and λ_2. First, we rewrite the optimal value of $\mathbf{w}^c$:

$$\begin{aligned}\mathbf{w}^c &= \left(\mathbf{V}\mathbf{D}^T\mathbf{D}\mathbf{V}^T + \lambda_1 \mathbf{V}\mathbf{V}^T\right)^{-1} \left(\left(\mathbf{U}\mathbf{D}\mathbf{V}^T\right)^T \mathbf{y}^c - \frac{1}{2}\lambda_2 \mathbf{1} \right) \\ &= \left(\mathbf{V} \left(\mathbf{D}^T\mathbf{D} + \lambda_1 \mathbf{I}\right)^{-1} \mathbf{V}^T \right) \left(\mathbf{V}\mathbf{D}^T\mathbf{U}^T\mathbf{y}^c - \frac{1}{2}\lambda_2 \mathbf{1} \right)\end{aligned} \tag{3.58}$$

$$= \mathbf{V}\left(\mathbf{D}^T\mathbf{D}+\lambda_1\mathbf{I}\right)^{-1}\mathbf{D}^T\mathbf{U}^T\mathbf{y}^c - \frac{1}{2}\lambda_2\mathbf{V}\left(\mathbf{D}^T\mathbf{D}+\lambda_1\mathbf{I}\right)^{-1}\mathbf{V}^T\mathbf{1}.$$

We use this result to analyse $\nabla_{\lambda_2}\mathcal{L}(\lambda_1,\lambda_2,\mathbf{w}^c) = \mathbf{1}^T\mathbf{w}^c$ and to find the optimal value of λ_2. Let v_j^s be the sum of the j^{th} column of $\mathbf{V}$. It then follows that:

$$\begin{aligned}\mathbf{1}^T\mathbf{w}^c &= \mathbf{1}^T\left(\mathbf{V}\left(\mathbf{D}^T\mathbf{D}+\lambda_1\mathbf{I}\right)^{-1}\mathbf{D}^T\mathbf{U}^T\mathbf{y}^c - \frac{1}{2}\lambda_2\mathbf{V}\left(\mathbf{D}^T\mathbf{D}+\lambda_1\mathbf{I}\right)^{-1}\mathbf{V}^T\mathbf{1}\right)\\ &= \sum_{j=1}^m \frac{s_j v_j^s b_j}{s_j^2+\lambda_1} - \frac{1}{2}\lambda_2\sum_{j=1}^m \frac{(v_j^s)^2}{s_j^2+\lambda_1}.\end{aligned}$$

3

This expression should equal zero (eq. (3.55)):

$$\begin{aligned}&\sum_{j=1}^m \frac{s_j v_j^s b_j}{s_j^2+\lambda_1} - \frac{1}{2}\lambda_2\sum_{j=1}^m \frac{(v_j^s)^2}{s_j^2+\lambda_1} = 0\\ \Rightarrow &\frac{1}{2}\lambda_2 = \frac{\sum_{j=1}^m \frac{s_j v_j^s b_j}{s_j^2+\lambda_1}}{\sum_{j=1}^m \frac{(v_j^s)^2}{s_j^2+\lambda_1}}.\end{aligned}$$

With this, we analyse $\nabla_{\lambda_1}\mathcal{L}(\lambda_1,\lambda_2,\mathbf{w}^c) = (\mathbf{w}^c)^T\mathbf{w}^c - \frac{1+m}{2}$. First, we derive that:

$$\begin{aligned}\left(\mathbf{w}^c\right)^T\cdot\mathbf{w}^c &= \left(\left(\mathbf{y}^c\right)^T\mathbf{U}\mathbf{D}\left(\mathbf{D}^T\mathbf{D}+\lambda_1\mathbf{I}\right)^{-1}\mathbf{V}^T - \frac{1}{2}\lambda_2\mathbf{1}^T\mathbf{V}\left(\mathbf{D}^T\mathbf{D}+\lambda_1\mathbf{I}\right)^{-1}\mathbf{V}^T\right)\\ &\quad\left(\mathbf{V}\left(\mathbf{D}^T\mathbf{D}+\lambda_1\mathbf{I}\right)^{-1}\mathbf{D}^T\mathbf{U}^T\mathbf{y}^c - \frac{1}{2}\lambda_2\mathbf{V}\left(\mathbf{D}^T\mathbf{D}+\lambda_1\mathbf{I}\right)^{-1}\mathbf{V}^T\mathbf{1}\right)\\ &= \left(\mathbf{y}^c\right)^T\mathbf{U}\mathbf{D}\left(\mathbf{D}^T\mathbf{D}+\lambda_1\mathbf{I}\right)^{-2}\mathbf{D}^T\mathbf{U}^T\mathbf{y}^c - \lambda_2\left(\mathbf{y}^c\right)^T\mathbf{U}\mathbf{D}\left(\mathbf{D}^T\mathbf{D}+\lambda_1\mathbf{I}\right)^{-2}\mathbf{V}^T\mathbf{1}\\ &\quad+\left(\frac{1}{2}\lambda_2\right)^2\mathbf{1}^T\mathbf{V}\left(\mathbf{D}^T\mathbf{D}+\lambda_1\mathbf{I}\right)^{-2}\mathbf{V}^T\mathbf{1}\\ &= \sum_{j=1}^m \frac{b_j^2 s_j^2}{\left(s_j^2+\lambda_1\right)^2} - \lambda_2\sum_{j=1}^m \frac{b_j s_j v_j^s}{\left(s_j^2+\lambda_1\right)^2} + (\frac{1}{2}\lambda_2)^2\sum_{j=1}^m \frac{\left(v_j^s\right)^2}{\left(s_j^2+\lambda_1\right)^2}\\ &= \sum_{j=1}^m\left(\frac{\frac{1}{2}\lambda_2 v_j^s - b_j s_j}{s_j^2+\lambda_1}\right)^2.\end{aligned} \tag{3.59}$$

This expression should equal $\frac{1+m}{2}$, i.e,:

$$\sum_{j=1}^m\left(\frac{\frac{1}{2}\lambda_2 v_j^s - b_j s_j}{s_j^2+\lambda_1}\right)^2 = \frac{1+m}{2}. \tag{3.60}$$

Given the optimal value for $\frac{1}{2}\lambda_2$ in eq. (3.59), we can solve this numerically to find the optimal value for λ_1. Using this, we can directly calculate the optimal value for λ_2 from eq. (3.59).

APPENDIX 3.C. DERIVATION OF THE CONSTRAINTS ON THE WEIGHTS FOLLOWING [10]

In this appendix, we derive the same constraints on the weights as in Section 3.2.2. However, we now follow the same derivation, with the same assumptions, as in [10].

REQUIREMENT 1: $\mathrm{VAR}\left(\hat{y}\right) = \mathrm{VAR}\left(x^L\right)$

We first derive the variance of y^L in terms of the variance of x^L following [10]. In contrast with our derivation, where we regarded each weight as a constant number, we now regard each weight as a random variable. Let w^L represent the random variable of any element in $\mathbf{w}^L$. Four key assumptions are made in the derivation in [10]: i) the hidden states x^L are independent and identically distributed, ii) the weights $w_j^L, j = 1,2,\ldots,m$ are independent and identically distributed, iii) the hidden states x^L are independent of the weights w^L, and iiii) $\mathbb{E}[x^l] = 0$, due to the considered activation function. Note that only the first assumption is made in our derivation as well. Then, it holds that:

$$\begin{aligned}
\mathrm{Var}\left(y^L\right) &= \mathrm{Var}\left(b^L + \sum_{j=1}^{m} w_j^L x_j^L\right) \qquad (3.61)\\
&= m \cdot \mathrm{Var}\left(w^L x^L\right)\\
&= m\left(\mathrm{Var}\left(w^L\right)\mathrm{Var}\left(x^L\right) + \mathbb{E}\left[w^L\right]^2 \mathrm{Var}\left(x^L\right) + \mathrm{Var}\left(w^L\right)\mathbb{E}\left[x^L\right]^2\right)\\
&= m\left(\left(\mathrm{Var}\left(w^L\right) + \mathbb{E}\left[w^L\right]^2\right)\mathrm{Var}\left(x^L\right)\right)\\
&= m\mathbb{E}\left[\left(w^L\right)^2\right]\mathrm{Var}\left(x^L\right).
\end{aligned}$$

Note that we deviate here from [10], since we do not assume that the expected value of a weight is zero. Since $\mathrm{Var}\left(\hat{y}\right) = \mathrm{Var}\left(x^L\right)$ should hold, this gives:

$$m\mathbb{E}\left[\left(w^L\right)^2\right] = 1. \qquad (3.62)$$

We similarly derive that the sum of the squared weights equals 1 in Section 3.2.2.

REQUIREMENT 2: $\mathrm{VAR}\left(\frac{\partial \mathrm{LOSS}}{\partial y^L}\right) = \mathrm{VAR}\left(\frac{\partial \mathrm{LOSS}}{\partial y^{L-1}}\right)$

In Section 3.2.2 we derived that:

$$\frac{\partial \mathrm{Loss}}{\partial \mathbf{y}_j^{L-1}} \approx \frac{\partial \mathrm{Loss}}{\partial \mathbf{y}^L} w_j^L.$$

We now additionally assume, following [10], that the weights of the last layer and the gradient $\frac{\partial \mathrm{Loss}}{\partial y^L}$ are independent of each other, and that the variance of $\frac{\partial \mathrm{Loss}}{\partial y_j^{L-1}}$ is the same for each node $j \in \{1,2,\ldots,m\}$. This gives:

$$\mathrm{Var}\left(\frac{\partial \mathrm{Loss}}{\partial y^{L-1}}\right) = \mathrm{Var}\left(\frac{\partial \mathrm{Loss}}{\partial y^L} w^L\right) \qquad (3.63)$$

$$
\begin{aligned}
&= \mathrm{Var}\left(w^L\right)\mathrm{Var}\left(\frac{\partial \mathrm{Loss}}{\partial y^L}\right) + \mathbb{E}\left[w^L\right]^2 \mathrm{Var}\left(\frac{\partial \mathrm{Loss}}{\partial y^L}\right) + \mathrm{Var}\left(w^L\right)\mathbb{E}\left[\frac{\partial \mathrm{Loss}}{\partial y^L}\right]^2 \\
&= \left(\mathrm{Var}\left(w^L\right) + \mathbb{E}\left[w^L\right]^2\right)\mathrm{Var}\left(\frac{\partial \mathrm{Loss}}{\partial y^L}\right) \\
&= \mathbb{E}\left[\left(w^L\right)^2\right]\mathrm{Var}\left(\frac{\partial \mathrm{Loss}}{\partial y^L}\right).
\end{aligned}
$$

Since $\mathrm{Var}\left(\frac{\partial \mathrm{Loss}}{\partial y^L}\right) = \mathrm{Var}\left(\frac{\partial \mathrm{Loss}}{\partial y^{L-1}}\right)$ should hold, it thus follows that:

$$
\mathbb{E}\left[\left(w^L\right)^2\right] = 1. \qquad (3.64)
$$

We similarly derive that the sum of the squared weights is m in Section 3.2.2.

REFERENCES

[1] de Pater, I., & Mitici, M. (2023). A mathematical framework for improved weight initialization of neural networks using Lagrange multipliers. *Neural Networks, 166,* Pages: 579–594.

[2] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of Convolutional Neural Networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems, 33,* Pages: 6999–7019.

[3] Pan, X., Xu, J., Pan, Y., Wen, L., Lin, W., Bai, K., Fu, H., & Xu, Z. (2022). Afinet: Attentive feature integration networks for image classification. *Neural Networks, 155,* Pages: 360–368.

[4] Martínez, F., Charte, F., Frías, M. P., & Martínez-Rodríguez, A. M. (2022). Strategies for time series forecasting with generalized regression neural networks. *Neurocomputing, 491,* Pages: 509–521.

[5] Zhao, Z.-Q., Zheng, P., Xu, S.-t., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems, 30*(11), Pages: 3212–3232.

[6] Roh, J., Park, S., Kim, B.-K., Oh, S.-H., & Lee, S.-Y. (2021). Unsupervised multi-sense language models for natural language processing tasks. *Neural Networks, 142,* Pages: 397–409.

[7] Vasilev, I. (2019). *Advanced deep learning with Python: Design and implement advanced next-generation AI solutions using Tensorflow and PyTorch.* Packt Publishing Ltd.

[8] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

[9] Xie, X., Pu, Y.-F., & Wang, J. (2023). A fractional gradient descent algorithm robust to the initial weights of multilayer perceptron. *Neural Networks, 158,* Pages: 154–170.

[10] Glorot, X., & Bengio, Y. (2010, May 13-15). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 9,* Sardinia, Italy, Pages: 249–256.

[11] He, K., Zhang, X., Ren, S., & Sun, J. (2015, December 7-13). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *Proceedings of the IEEE International Conference on Computer Vision (ICCV),* Santiago, Chile, Pages: 1026–1034.

[12] Narkhede, M. V., Bartakke, P. P., & Sutaone, M. S. (2022). A review on weight initialization strategies for neural networks. *Artificial intelligence review, 55*(1), Pages: 291–322.

[13] Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

[14] Mishkin, D., & Matas, J. (2015). All you need is a good init. *arXiv preprint arXiv:1511.06422*.

[15] Adam, S. P., Karras, D. A., Magoulas, G. D., & Vrahatis, M. N. (2014). Solving the linear interval tolerance problem for weight initialization of neural networks. *Neural Networks*, *54*, Pages: 17–37.

[16] Aguirre, D., & Fuentes, O. (2019, September 17-19). Improving weight initialization of ReLU and output layers. *Artificial Neural Networks and Machine Learning – Proceedings of the 28th International Conference on Artificial Neural Networks*, Munich, Germany, Pages: 170–184.

[17] Yam, Y., Chow, T. W., & Leung, C.-T. (1997). A new method in determining initial weights of feedforward neural networks for training enhancement. *Neurocomputing*, *16*(1), Pages: 23–32.

[18] Chumachenko, K., Iosifidis, A., & Gabbouj, M. (2022). Feedforward neural networks initialization based on discriminant learning. *Neural Networks*, *146*, Pages: 220–229.

[19] Yam, Y.-F., & Chow, T. W. (1995). Determining initial weights of feedforward neural networks based on least squares method. *Neural Processing Letters*, *2*(2), Pages: 13–17.

[20] Cao, W., Wang, X., Ming, Z., & Gao, J. (2018). A review on neural networks with random weights. *Neurocomputing*, *275*, Pages: 278–287.

[21] Kim, M. (2021). The generalized extreme learning machines: Tuning hyperparameters and limiting approach for the Moore–Penrose generalized inverse. *Neural Networks*, *144*, Pages: 591–602.

[22] Fernández-Navarro, F., Riccardi, A., & Carloni, S. (2014). Ordinal neural networks without iterative tuning. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(11), Pages: 2075–2085.

[23] He, K., Zhang, X., Ren, S., & Sun, J. (2016, June 27-30). Deep residual learning for image recognition. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, Nevada, USA, Pages: 770–778.

[24] Krizhevsky, A., Hinton, G., et al. (2009). *Learning multiple layers of features from tiny images* (tech. rep.). [Data set]. University of Toronto.

[25] LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K.-R. (2012). Efficient BackProp. In *Neural networks: Tricks of the trade* (Pages: 9–48). Springer.

[26] Vural, N. M., Ilhan, F., Yilmaz, S. F., Ergüt, S., & Kozat, S. S. (2021). Achieving online regression performance of LSTMs with simple RNNs. *IEEE Transactions on Neural Networks and Learning Systems*, *33*, Pages: 7632–7643.

[27] Yan, W. (2012). Toward automatic time-series forecasting using neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(7), Pages: 1028–1039.

[28] Kim, S., Lu, P. Y., Mukherjee, S., Gilbert, M., Jing, L., Čeperić, V., & Soljačić, M. (2020). Integration of neural network-based symbolic regression in deep

learning for scientific discovery. *IEEE Transactions on Neural Networks and Learning Systems, 32*(9), Pages: 4166–4177.

[29] Heij, C., Heij, C., de Boer, P., Franses, P. H., Kloek, T., van Dijk, H. K., et al. (2004). *Econometric methods with applications in business and economics.* Oxford University Press.

[30] Frühwirth-Schnatter, S., & Frèuhwirth-Schnatter, S. (2006). *Finite mixture and Markov switching models.* Springer.

[31] Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.

[32] Saxena, A., & Goebel, K. (2008). *Turbofan engine degradation simulation data set,* NASA Prognostics Data Repository, NASA Ames Research Center, Moffett Field, California, USA.

[33] Vollert, S., & Theissler, A. (2021, September 7-10). Challenges of machine learning-based RUL prognosis: A review on NASA's C-MAPSS data set. *26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* Vasteras, Sweden, Pages: 1–8.

[34] de Pater, I., & Mitici, M. (2022, July 6-8). Novel metrics to evaluate probabilistic Remaining Useful Life prognostics with applications to turbofan engines. *Proceedings of the 7th European Conference of the Prognostics and Health Management (PHM) Society, 7,* Turin, Italy, Pages: 96–109.

[35] Lee, J., & Mitici, M. (2023). Deep reinforcement learning for predictive aircraft maintenance using probabilistic Remaining-Useful-Life prognostics. *Reliability Engineering & System Safety, 230,* Article number: 108908.

[36] Ramasso, E., & Saxena, A. (2014, September 29 - October 2). Review and analysis of algorithmic approaches developed for prognostics on CMAPSS dataset. *Proceedings of the Annual Conference of the Prognostics and Health Management (PHM) Society 2014, 6,* Fort Worth, Texas, USA, Pages: 1–11.

[37] Li, X., Ding, Q., & Sun, J.-Q. (2018). Remaining Useful Life estimation in prognostics using deep Convolution Neural Networks. *Reliability Engineering & System Safety, 172,* Pages: 1–11.

[38] de Pater, I., Reijns, A., & Mitici, M. (2022). Alarm-based predictive maintenance scheduling for aircraft engines with imperfect Remaining Useful Life prognostics. *Reliability Engineering & System Safety, 221,* Article number: 108341.

[39] Cubuk, E. D., Zoph, B., Shlens, J., & Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. *Advances in Neural Information Processing Systems, 33,* Pages: 18613–18624.

[40] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12,* Pages: 2825–2830.

[41] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015).

ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV), 115*(3), Pages: 211–252.

[42] Poole, D. (2011). *Linear algebra: A modern introduction* (3rd ed.). Brooks/Cole/Thomson Learning.

4
MODEL-BASED PROBABILISTIC RUL PROGNOSTICS WITH CLUSTERING

In the previous two chapters, we develop point Remaining Useful Life (RUL) prognostics, i.e., we estimate one number for the RUL. For maintenance planners, it is challenging to make maintenance decisions based on point RUL prognostics, as the uncertainty of the RUL prognostics is not quantified. In this chapter, we therefore estimate a Probability Density Function (PDF) of the RUL instead.

In contrast with the previous two chapters, we use a model-based approach. We first cluster the health indicators of several components offline using dynamic time-warping. Within each cluster, the degradation follows a specific trend. We therefore propose for each cluster a degradation model and corresponding failure threshold. Using particle filtering, we subsequently estimate the RUL of a component, using the degradation model and failure threshold of its corresponding cluster.

In the case study, we apply our approach to estimate the RUL of aircraft cooling units using leave-one-out cross validation. Only a single cooling unit is assigned to a different cluster in the cross validation, than in the offline clustering. Using the assigned clusters, we are able to accurately estimate the RUL.

Parts of this chapter have been published in:

Mitici, M., & de Pater, I. (2021). Online model-based Remaining-Useful-Life prognostics for aircraft cooling units using time-warping degradation clustering. *Aerospace*, *8*(6), Article number: 168
de Pater, I., & Mitici, M. (2021, June 28 - July 2). Model-based Remaining-Useful-Life prognostics for aircraft cooling units. *Proceedings of the European Conference of the Prognostics and Health Management (PHM) Society*, *6*, Virtual, Pages: 1–8

4.1. Introduction

The costs of aircraft maintenance, repair and overhaul (MRO) account for 9% of the total airline operational costs [3]. Predictive maintenance is a new maintenance strategy that aims to reduce these maintenance costs and, in particular, to reduce the costs of unexpected failures. Here, the sensor measurements of aircraft components are analyzed to estimate the time left until failure (Remaining Useful Life, RUL).

In the past years, several RUL prognostic methods have been developed [4], mainly using data-driven or model-based approaches. Data-driven RUL prognostic methods mainly use neural networks to estimate the RUL, such as Feed Forward Neural Networks [5], Convolutional Neural Networks [6, 7] and Recurrent Neural Networks [8, 9]. By contrast, in a model-based approach, it is assumed that the degradation inside a component follows a degradation model, such as a Wiener processes [10, 11] or a physics-based model [12, 13]. The parameters of this model are estimated with, for instance, a Kalman filter [14] or a particle filter [15].

Some RUL prognostic methods are developed to immediately predict the RUL, from the moment a component is installed. Other studies, however, divide the lifetime of a component in several stages, such as “healthy” and “unhealthy”, or “healthy”, “degradation” and “critical” [4]. The delay time model [16–18] similarly assumes that a component is healthy (i.e., in a good state) for some time after installment. After the degradation in a component passes a certain threshold, it is seen as unhealthy (i.e., there is a defect), at which point the RUL can be predicted. Also in this study, we first identify when a component becomes unhealthy, and only then predict the RUL.

In most RUL prognostic methods, it is implicitly assumed that there is only one degradation mechanism that governs the failure of components [19]. However, systems are often complex, consisting of multiple components, where each component is subject to various operational factors and may fail according to various fault modes. It is therefore expected that the degradation of a system may follow more than one degradation mechanism, and may thus exhibit multiple degradation trends [19, 20]. To capture these various degradation trends towards failure, several degradation models should be integrated in the RUL prognostic method.

Only few studies develop RUL prognostics using multiple degradation trends. Data-driven approaches with multiple degradation trends are considered in [21, 22]. In [21], a Long Short-Term Memory Neural Network is developed such that degradation patterns under multiple operational conditions and belonging to several fault modes are detected. With this approach, the RUL of aircraft engines is estimated. In [22], a deep learning approach is combined with a genetic algorithm to estimate the RUL of aircraft engines operating under multiple conditions and failing under several fault modes. Machine-learning approaches, however, are “black-box” models, of which the inner workings are hard to explain [23]. This is an obstacle for the implementation of purely data-driven RUL prognostic methods in the maintenance practice [24]. In this chapter, we therefore focus on a model-based prognostic approach that considers multiple degradation trends.

Model-based RUL prognostic approaches that incorporate multiple degradation models and fault modes are considered in [19, 25–29]. In [25], four physics-based failure models of subsea pipelines are incorporated in one dynamic Bayesian network. In [19], the authors consider multiple degradation trends for the resistance in batteries, each with its own degradation model. With these degradation models and a particle filtering algorithm, the authors estimate the RUL of batteries. A switching Kalman filter to model multiple degradation trends within the degradation process of a single component is used in [28, 29] for bearings and in [26] for engines. In these studies, multiple degradation models are integrated in one comprehensive degradation process. The degradation trend of a single component is thus not explicitly identified. In contrast, we first cluster the degradation trends of components and propose a degradation model for each cluster. A separate RUL prognostic model is developed for each cluster, based on this degradation model. In this way, different components are associated with different degradation models.

For model-based RUL prognostic methods, two frequently considered degradation models are the exponential and the linear degradation model [4, 30]. Linear degradation models are developed in [31–34]. In [31], a linear model with Brownian drift and random shocks, together with a particle filtering algorithm, is used to estimate the RUL of milling machines. A linear degradation model is also used in [32] for batteries, in [33] for aircraft engines, and in [34] for engine bleed valves. Exponential degradation models are used in [35–39]. An exponential model is used together with a particle filtering algorithm to estimate the RUL of bearings in [35], and to estimate the RUL of batteries in [36]. An exponential model for the degradation of bearings is also considered in [37], for batteries in [38] and for railway turnout systems in [39]. Similarly, in this chapter, we consider clusters of components for which the degradation follows a linear or an exponential model.

In this chapter, we propose an approach to obtain online, model-based RUL prognostics for aircraft components by exploiting the knowledge obtained from clusters of component degradation trends. First, using the sensor monitoring data, we construct a health indicator which is used to diagnose components as healthy or unhealthy. As soon as a component is diagnosed as unhealthy, a cluster-specific degradation model is selected for this component based on a dynamic time-warping clustering of a library of health indicators. These degradation models, together with a particle filtering algorithm, are further used to obtain RUL prognostics. We illustrate our approach for the case of several aircraft cooling units, originating from a fleet of aircraft. We consider operational aircraft data, i.e., the sensor measurements have been collected during the actual operation of the aircraft. The results show that our proposed cluster-based RUL prognostic approach provides accurate RUL prognostics for these cooling units.

The main contributions of this study are as follows:

- We propose an end-to-end methodology that employs the sensor measurement to first diagnose of a component as *healthy* or *unhealthy*, and that subsequently estimates the PDF of the RUL. Using a health indicator, a component is diagnosed as healthy or unhealthy. Once a component reaches the unhealthy stage, a degradation model is selected based on the similarity between the degradation trend of this component and the degradation trend of the clusters in a library of health indicators. This approach exploits the potential to learn from the degradation trends in other components;

- We consider multiple degradation models. A clustering method for the health indicators of the components is proposed. Each obtained cluster is associated with one degradation model and a corresponding failure threshold.
- Our proposed approach is illustrated using operational aircraft data, i.e., sensor measurements that are collected during the actual operation of aircraft. In general, case studies are conducted with simulated data or data generated in a laboratory, which may not be as noisy as the aircraft operational data.

The remainder of this chapter is organized as follows. In Section 4.2 we provide a model-based RUL prognostic method using a dynamic time-warping algorithm and a particle filtering algorithm. For the time-warping algorithm, we consider several components which are clustered based on the similarity between their degradation trends. In Section 4.3 we illustrate our methodology for several aircraft cooling units. Section 4.4 provides conclusions and recommendations for further research.

4.2. Methodology - Online model-based RUL prognostics

In this section we provide an approach to obtain online RUL prognostics for aircraft components. We first construct a health indicator from the sensor monitoring data. Based on this health indicator, we online diagnose a component as *healthy* or *unhealthy* (step 1). Once a component is diagnosed as unhealthy, a degradation model and corresponding failure threshold is selected for this component using a dynamic time-warping algorithm that clusters a library of health indicators (step 2 and 3). After selecting a degradation model, the RUL is estimated with a particle filtering algorithm (step 4). This process is illustrated in Figure 4.1.

Let a stochastic process $\left\{X^i_f, f \in \mathbb{N}\right\}$ characterize the degradation of a component i over time, where X^i_f denotes the actual degradation in component i after flight f. Let $D > 0$ denote a failure threshold, i.e., component i has failed if the degradation exceeds this failure threshold (if $X^i_f > D$). After a flight $f^{i,\text{cur}} > 0$, the RUL of a component (in flights) is defined as follows:

$$\text{RUL} = \min\{f : X^i_{f^{i,\text{cur}}+f} \geq D, f \in \mathbb{N}\}. \tag{4.1}$$

4.2.1. Step 1: Constructing a health indicator and defining the health stage

Let $\mathcal{I}^i_f$ denote the health indicator value of component i after flight f, and let y^i_f be the sensor measurements of this component after flight f. We assume that $\mathcal{I}^i_f$ is a function $h(\cdot)$ of the sensor measurements $y^i_f, y^i_{f-1}, \ldots, y^i_{f-N}$ of the past N flights, i.e.,:

$$\mathcal{I}^i_f = h\left(y^i_f, y^i_{f-1}, \ldots, y^i_{f-N}\right). \tag{4.2}$$

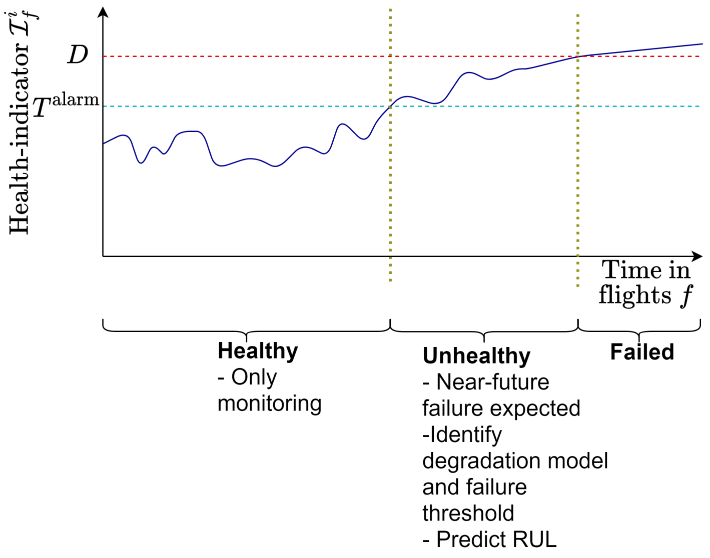

Health-indicator $\mathcal{I}_f^i$
D
T^{alarm}
Time in flights f
Healthy
- Only monitoring
Unhealthy
- Near-future failure expected
-Identify degradation model and failure threshold
- Predict RUL
Failed

Figure 4.1.: A component i is defined as healthy or unhealthy, based on the health indicator $\mathcal{I}_f^i$ that is updated every flight f. Once a component is diagnosed as unhealthy, a degradation model is identified and the RUL is estimated.

Here, the sensor measurements y_f in turn depend on the actual degradation X_f^i of component i after flight f.

We diagnose the component as *healthy* or *unhealthy* by analyzing whether $\mathcal{I}_f^i$ exceeds an alarm threshold T^{alarm}. This alarm threshold is defined by Chebyshev's inequality [40–42], which states that for any probability distribution with a specified mean μ and standard deviation σ, at most $\frac{1}{k^2}$ percent of the values from this distribution fall outside the $\mu \pm k\sigma$ interval, $k > 0$. This implies that:

$$p\left(\left|\mathcal{I}_f^i - \mu\right| \geq k\sigma\right) \leq \frac{1}{k^2}, \tag{4.3}$$

where $p(x)$ denotes the probability of an event x, while μ is the mean and σ is the standard deviation of the health indicator values while an aircraft component is healthy. We thus use the following alarm threshold T^{alarm}:

$$T^{\text{alarm}} = \mu + k\sigma. \tag{4.4}$$

As soon as $\mathcal{I}_f^i > T^{\text{alarm}}$, a component is diagnosed as *unhealthy* (see Figure 4.1). Let $f^{i,\text{alarm}}$ denote the first flight when a component i is diagnosed as unhealthy. Last, we define $f^{i,\text{fail}}$ as the flight during which the component i fails.

4.2.2. Step 2: Selecting a degradation model for a component

As soon as a component is diagnosed as unhealthy, we select a degradation model and a corresponding failure threshold D based on the clusters in a library of health indicators. As a last step, we use this selected degradation model together with a particle filtering algorithm to estimate the RUL.

We consider an *offline* library of n health indicators. This library contains the health indicators of n aircraft components that have already failed in the past. Each health indicator is constructed starting m flights before the component is diagnosed as unhealthy (see Section 4.2.1) until the component has failed.

Let $\mathcal{I}^i$ denote the health indicator for a component i in the offline library ($1 \leq i \leq n$). A health indicator is a time-series consisting of the health indicator values after flight $f \in [f^{i,\text{alarm}} - m, f^{i,\text{fail}}]$:

$$\mathcal{I}^i = \left\{\mathcal{I}^i_f, f \in \left[f^{i,\text{alarm}} - m, f^{i,\text{alarm}} - m + 1, \ldots, f^{i,\text{fail}}\right]\right\}. \tag{4.5}$$

We group the health indicators $\mathcal{I}^i, i \in \{1, 2, \ldots, n\}$ in several clusters, where the underlying degradation of the components in each cluster follows a similar degradation trend towards failure. For this, we propose i) a Dynamic Time Warping (DTW) algorithm [43, 44] to determine the minimum distance between any two health indicators $\mathcal{I}^i$ and $\mathcal{I}^j$ of component i and component j, respectively, and, ii) we cluster the n health indicators based on this minimum distance.

i) Dynamic time-warping [43, 44] to determine the minimum distance between two health indicators

To define the distance between the health indicators of two components, we first define a warping path p between the two health indicators $\mathcal{I}^i$ of component i and $\mathcal{I}^j$ of component j. In a warping path p, each health indicator value in $\mathcal{I}^i$ is connected to a health indicator value in $\mathcal{I}^j$:

Definition 1 (Warping path [44]) *A warping path p between $\mathcal{I}^i$ and $\mathcal{I}^j$ is a sequence $p = (p_1, p_2, \ldots, p_L)$ with $p_l = (f_l, g_l)$ (for $l \in \{1, 2, \ldots, L\}$) with*

$$f_l \in \left\{f^{i,alarm} - m, f^{i,alarm} - m + 1, \ldots, f^{i,fail}\right\} \tag{4.6}$$

$$g_l \in \left\{f^{j,alarm} - m, f^{j,alarm} - m + 1, \ldots, f^{j,fail}\right\}. \tag{4.7}$$

The following four conditions for p have to hold:

a $p_1 = (f^{i,alarm} - m, f^{j,alarm} - m)$

b $p_L = (f^{i,fail}, f^{j,fail}|)$.

c $f_1 \leq f_2 \leq \ldots \leq f_L$, *and* $g_1 \leq g_2 \leq \ldots \leq g_L$.

d $p_{l+1} - p_l \in \{(1,0), (0,1), (1,1)\}$ *for* $l \in \{1, 2, \ldots, L-1\}$.

Condition a and b in Definition 1 imply that the first and last elements of health indicators $\mathcal{I}^i$ and $\mathcal{I}^j$ are warped (aligned), respectively. Condition c ensures that the path is monotonically increasing for both health indicators. Last, condition d ensures that no element of $\{\mathcal{I}^i\}$ and $\{\mathcal{I}^j\}$ can be omitted in the warping path, i.e., the elements in the sequence need to be contiguous.

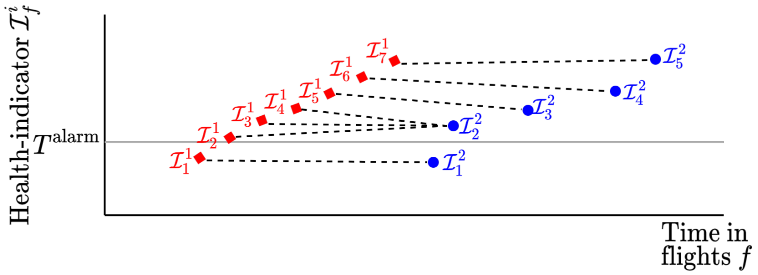


Figure 4.2.: An example of a time-warping path between two health indicators, $\{\mathcal{I}^1_f, f \in [1,2,3,4,5,6,7]\}$ and $\{\mathcal{I}^2_g, g \in [1,2,3,4,5]\}$.

Figure 4.2 shows an example of a warping path for two health indicators, $\left\{\mathcal{I}^1_f, f \in [1,2,3,4,5,6,7]\right\}$ and $\left\{\mathcal{I}^2_g, g \in [1,2,3,4,5]\right\}$. Here, $\mathcal{I}^1$ contains 7 health indicator values, while $\mathcal{I}^2$ contains only 5 health indicator values. There are three health indicator values of the first component, namely $\mathcal{I}^1_2, \mathcal{I}^1_3$ and $\mathcal{I}^1_4$, connected to the health indicator value $\mathcal{I}^2_2$ of the second component. The warping path p for these two health indicators is:

$$p = \left((\mathcal{I}^1_1,\mathcal{I}^2_1),(\mathcal{I}^1_2,\mathcal{I}^2_2),(\mathcal{I}^1_3,\mathcal{I}^2_2),(\mathcal{I}^1_4,\mathcal{I}^2_2),(\mathcal{I}^1_5,\mathcal{I}^2_3),(\mathcal{I}^1_6,\mathcal{I}^2_4),(\mathcal{I}^1_7,\mathcal{I}^2_5)\right).$$

The aim is to find a warping path p between $\mathcal{I}^i$ and $\mathcal{I}^j$ that minimizes the “distance” between the two health indicators. For this, we first define the Euclidean distance between $\mathcal{I}^i_f$ (the health indicator value of component i after flight f) and $\mathcal{I}^j_g$ (the health indicator value of component j after flight g) as follows:

$$d\left(\mathcal{I}^i_f,\mathcal{I}^j_g\right) = \left(\mathcal{I}^i_f - \mathcal{I}^j_g\right)^2. \tag{4.8}$$

Here, we interpret $d\left(\mathcal{I}^i_f,\mathcal{I}^j_g\right)$ as a metric for the dissimilarity between the health indicator values $\mathcal{I}^i_f$ and $\mathcal{I}^j_g$, i.e., the larger the distance, the more dissimilar the health indicator values are. With this, we define $\mathcal{D}\left(p,\mathcal{I}^i,\mathcal{I}^j\right)$ as the “distance” between two health indicators $\mathcal{I}^i$ and $\mathcal{I}^j$, given warping path p:

$$\mathcal{D}\left(p,\mathcal{I}^i,\mathcal{I}^j\right) = \sum_{l=1}^{L} d\left(\mathcal{I}^i_{p_l[1]},\mathcal{I}^j_{p_l[2]}\right), \tag{4.9}$$

where $p_l[1]$ and $p_l[2]$ denote the first element f_l and the second element g_l of p_l.

Our aim is to find the warping path p between $\mathcal{I}^i$ and $\mathcal{I}^j$ that minimizes this distance $\mathcal{D}(p,\mathcal{I}^i,\mathcal{I}^j)$. We define the minimum distance $W\left(\mathcal{I}^i,\mathcal{I}^j\right)$ between $\mathcal{I}^i$ and $\mathcal{I}^j$ as follows:

$$W\left(\mathcal{I}^i,\mathcal{I}^j\right) = \min_p \left\{\mathcal{D}\left(p,\mathcal{I}^i,\mathcal{I}^j\right) : p \text{ is a warping path between } \mathcal{I}^i \text{ and } \mathcal{I}^j\right\}. \tag{4.10}$$

We find the minimum distance $W(\mathcal{I}^i,\mathcal{I}^j)$ with dynamic programming, following [44]. Let $\Delta\left(f',g'\right) = W\left(\tilde{\mathcal{I}}^i_{f'},\tilde{\mathcal{I}}^j_{g'}\right)$ be the minimum distance between a part of the health indicator $\mathcal{I}^i$ and a part of the health indicator $\mathcal{I}^j$:

$$\tilde{\mathcal{I}}^i_{f'} = \left\{\mathcal{I}^i_f, f \in \left[f^{i,\text{alarm}} - m, f^{i,\text{alarm}} - m + 1, \ldots, f'\right]\right\} \tag{4.11}$$

$$\tilde{\mathcal{I}}^j_{g'} = \left\{\mathcal{I}^i_g, g \in \left[f^{j,\text{alarm}} - m, f^{j,\text{alarm}} - m + 1, \ldots, g'\right]\right\}. \tag{4.12}$$

With this, we can easily calculate [44]:

$$\Delta(f', f^{j,\text{alarm}} - m) = \sum_{f=1}^{f'} d\left(\mathcal{I}^i_f, \mathcal{I}^j_{f^{j,\text{alarm}} - m}\right), \tag{4.13}$$

for all $f' \in \{f^{i,\text{alarm}} - m, f^{i,\text{alarm}} - m + 1, \ldots, f^{i,\text{fail}}\}$, while [44]

$$\Delta(f^{i,\text{alarm}} - m, g') = \sum_{g=1}^{g'} d\left(\mathcal{I}^i_{f^{i,\text{alarm}} - m}, \mathcal{I}^j_g\right),$$

for all $g' \in \{f^{j,\text{alarm}} - m, f^{j,\text{alarm}} - m + 1, \ldots, f^{j,\text{fail}}\}$.

We can now solve the following recursive equation until $f' = f^{i,\text{fail}}$ and $g' = f^{j,\text{fail}}$ to find the minimum distance between two health indicators [44]:

$$\Delta\left(f',g'\right) = \min\left\{\Delta\left(f'-1,g'-1\right), \Delta\left(f'-1,g'\right), \Delta\left(f',g'-1\right)\right\} + d\left(\mathcal{I}^i_{f'},\mathcal{I}^j_{g'}\right).$$

We assume that the minimum distance between two health indicators specifies how similar the health indicators are, i.e., the smaller the minimum distance, the more similar the degradation trend in the health indicators are. The dynamic time-warping algorithm was originally developed to determine the similarity between speech patterns in the field of automatic speech recognition.

ii) Clustering the health indicators of a library of n components

After determining the minimum distance $W\left(\mathcal{I}^i,\mathcal{I}^j\right)$ between the health indicators of all pairs of components $i,j \in \{1,2,\ldots,n\}, i \neq j$ in the offline library, we construct a graph $G(V,E,\delta)$. Here, each node $v^i \in V$ (where V denotes all nodes in the graph) corresponds to the health indicator of component $i \in \{1,2,\ldots,n\}$. There are thus n nodes, i.e., $|V| = n$. E denotes all edges in the graph. We construct an edge between node v^i and the closest δ nodes v^j, i.e., to the nodes of the δ components j for which the distance $W\left(\mathcal{I}^i,\mathcal{I}^j\right)$ between the respective health indicators is the smallest, with $j \in \{1,2,\ldots,n\}, i \neq j$. To prevent that all components are connected to the components with the "short" health indicators with few values, we divide the distance $W\left(\mathcal{I}^i,\mathcal{I}^j\right)$ between component i and j with the maximum number of health indicator values in $\mathcal{I}^i$ and $\mathcal{I}^j$. All components for which the nodes in this graph are connected, are considered to be in the same cluster.

We assume that all components in the same cluster have a similar degradation trend with the same functional form, and that all components in the same cluster have the same failure threshold D. These cluster-specific degradation models and failure thresholds are ultimately used to estimate the RUL for a component using a particle filtering algorithm.

4.2.3. Step 3: Online clustering of (non-failed) components

Let component i be an non-failed component for which we receive new sensor measurements after each flight, i.e., that is online monitored. Let an offline library consist of n health indicators $\left\{I^j : j \in \{1,2,\dots,n\}, j \neq i\right\}$. These health indicators come from components that have failed in the past, and for which we have all sensor measurements until failure. Let C be the set of clusters in the offline library, which are obtained using dynamic time-warping (see Section 4.2.2). As soon as the health indicator of the online monitored component i exceeds an alarm threshold T^{alarm} (i.e., as soon as the component is diagnosed as unhealthy), component i is assigned to a cluster in the set C using dynamic time-warping (Section 4.2.2).

The partial health indicator $\tilde{\mathcal{I}}^i$ of component i consists of the health indicator values obtained m flights before the threshold T^{alarm} is reached (flight $f^{i,\text{alarm}} - m$) until the current, most recently available measurement at flight $f^{i,\text{cur}}$:

$$\tilde{\mathcal{I}}^i = \left\{\mathcal{I}^i_f, f \in \left[f^{i,\text{alarm}} - m, f^{i,\text{alarm}} - m + 1, \dots, f^{i,\text{cur}}\right]\right\}. \tag{4.14}$$

Figure 4.3a illustrates $\tilde{\mathcal{I}}^i$ corresponding to an online monitored component i. We calculate the minimum distance $\mathcal{W}(\tilde{\mathcal{I}}^i,\tilde{\mathcal{I}}^j)$ between the partial health indicator $\tilde{\mathcal{I}}^i$ of the online component i and the partial health indicator $\tilde{\mathcal{I}}^j$ of each component j in the offline library ($j \in \{1,2,\dots,n\}$). Here, the partial health indicator $\tilde{\mathcal{I}}^j$ for component j from the offline library is defined as:

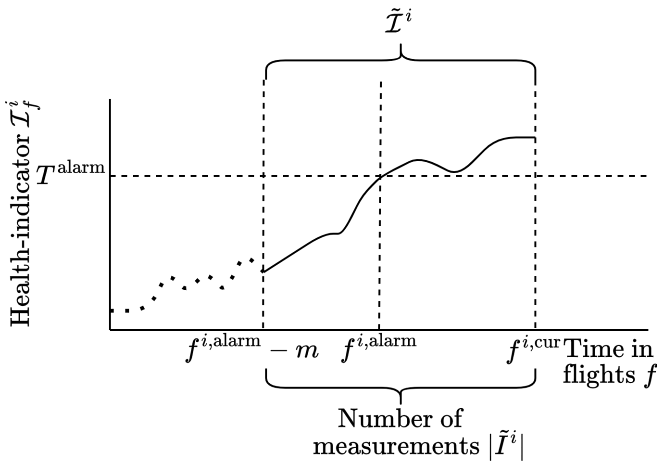
$$\tilde{\mathcal{I}}^j = \left\{\mathcal{I}^j_g, g \in \left[f^{j,\text{alarm}} - m, f^{j,\text{alarm}} - m + 1 \dots, \min\left(f^{j,\text{fail}}, f^{j,\text{alarm}} - m + \left|\tilde{\mathcal{I}}^i\right|\right)\right]\right\}, \tag{4.15}$$

where $\left|\tilde{\mathcal{I}}^i\right|$ is the length of the partial health indicator $\tilde{\mathcal{I}}^i$. Figure 4.3b shows an example for a library with two components $j \in \{1,2\}$.

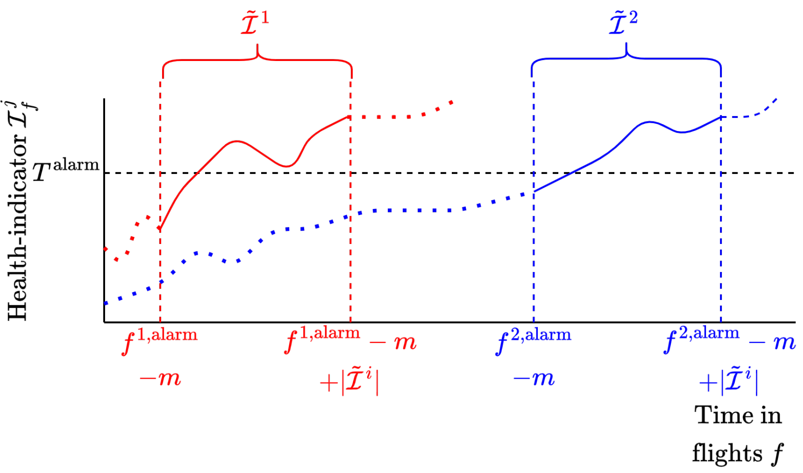
Lastly, we assign component i to a cluster $\tilde{c} \in C$ such that the average minimum distance $\mathcal{W}\left(\tilde{\mathcal{I}}^i,\tilde{\mathcal{I}}^j\right)$ between the partial health indicator of component i and all partial health indicators of the components j in cluster $\tilde{c}$ is minimized:

$$\tilde{c} = \text{argmin}_{c \in C}\left(\frac{1}{|c|}\sum_{j \in c}\mathcal{W}\left(\tilde{\mathcal{I}}^i,\tilde{\mathcal{I}}^j\right)\right). \tag{4.16}$$

We assume that component i has the same degradation model and failure threshold as the degradation model and failure threshold specific to cluster $\tilde{c}$. This degradation model is further used to estimate the RUL for component i.



(a) $\tilde{\mathcal{I}}^i$ of component i.



$\tilde{\mathcal{I}}^1$
$\tilde{\mathcal{I}}^2$
Health-indicator $\mathcal{I}_f^j$
T^{alarm}
$f^{1,\text{alarm}} - m$
$f^{1,\text{alarm}} - m + |\tilde{\mathcal{I}}^i|$
$f^{2,\text{alarm}} - m$
$f^{2,\text{alarm}} - m + |\tilde{\mathcal{I}}^i|$
Time in flights f

(b) $\tilde{\mathcal{I}}^j$ of component $j \in \{1,2\}$.

Figure 4.3.: Illustration of online clustering for component i. The library consists of the health indicators for components $j \in \{1,2\}$.

4.2.4. Step 4: RUL prognostics

Together with the degradation model and the failure threshold D identified in Section 4.2.3, we use a particle filtering algorithm [45] to estimate the RUL of an online monitored component i after a current flight $f^{i,\text{cur}}$. For the ease of notation, we do not include the dependency of component i in the notation of the particle filtering algorithm (i.e., $X_f = X_f^i, \mathcal{I}_f = \mathcal{I}_f^i$, etc.).

We consider the following recurrent function $r(\cdot)$ for the degradation X_f after flight f of the online monitored component:

$$X_f = r\left(X_{f-1}, \omega_f\right), \tag{4.17}$$

where X_f is the actual degradation level of a component after flight f and ω_f represents the noise of flight f of the online monitored component. We assume that the noise after all flights is a collection of independent and identically distributed (i.i.d.) random variables following the normal (Gaussian) distribution. In general, r has a certain form, which depends on the type of the component and the degradation in the component. In our case, the function r is the degradation model from the cluster $\tilde{c}$ to which the online monitored component is assigned. Similar, we assume that the health indicator value after a flight f is, through the measurements, a function $g(\cdot)$ of the degradation X_f:

$$\mathcal{I}_f = g\left(X_f, \nu_f\right) \tag{4.18}$$

where ν_f represents the measurement noise of flight f of the online monitored component. We again assume that the measurement noise after all flights is a collection of independent and identically distributed (i.i.d.) random variables following the normal (Gaussian) distribution.

We estimate the RUL of the online monitored component using the particle filtering algorithm [36, 45, 46]. This algorithm has four steps: i) prediction, ii) updating, iii) approximation and resampling, and iv) estimating the RUL.

i) The prediction step

In the prediction step, we are interested in the prior Probability Density Function (PDF) of the degradation X_f of component i after flight f, given the health indicator values $\mathcal{I}_{f-1},\ldots,\mathcal{I}_1$ of component i from flight 1 up to flight $f-1$. Here, $\mathcal{I}_1$ denotes the first health indicator value of the health indicator of component i, i.e., $\mathcal{I}_1 = \mathcal{I}^i_{f^{i,\text{alarm}}-m}$. We derive this PDF from the following two conditional probabilities:

$$p_{X_f|X_{f-1}}\left(x_f|x_{f-1}\right) \tag{4.19}$$

$$p_{X_{f-1}|\mathcal{I}_{f-1},\mathcal{I}_{f-2},\ldots,\mathcal{I}_1}\left(x_{f-1}|\iota_{f-1},\iota_{f-2},\ldots,\iota_1\right), \tag{4.20}$$

Here, eq. (4.19) is the transition PDF to reach the future degradation X_f, given the current degradation X_{f-1}. Eq. (4.20) is the posterior PDF of the degradation X_{f-1} of the component after flight $f-1$, given the past health indicator values $\mathcal{I}_{f-1},\ldots,\mathcal{I}_1$.

Using the Chapman-Kolmogorov equation [45], this gives the following prior PDF for the degradation X_f after flight f [46]:

$$p_{X_f|\mathcal{I}_{f-1},\mathcal{I}_{f-2},\ldots,\mathcal{I}_1}(x_f|\iota_{f-1},\iota_{f-2},\ldots,\iota_1) = \int p_{X_f|X_{f-1}}(x_f|x_{f-1}) \cdot p_{X_{f-1}|\mathcal{I}_{f-1},\ldots,\mathcal{I}_1}(x_{f-1}|\iota_{f-1},\ldots,\iota_1)\,dx_{f-1}. \tag{4.21}$$

ii) The updating step

With the sensor measurements of flight f, we calculate the health indicator value $\mathcal{I}_f$ of flight f. With this, using Bayes' theorem, the posterior PDF of the degradation X_f after flight f becomes [46]:

$$p_{X_f|\mathcal{I}_f,\mathcal{I}_{f-1},\ldots,\mathcal{I}_1}(x_f|\iota_f,\iota_{f-1},\ldots,\iota_1) = \frac{p_{\mathcal{I}_f|X_f}(\iota_f|x_f)\,p_{X_f|\mathcal{I}_{f-1},\ldots,\mathcal{I}_1}(x_f|\iota_{f-1},\ldots,\iota_1)}{p_{\mathcal{I}_f|\mathcal{I}_{f-1},\ldots,\mathcal{I}_1}(\iota_f|\iota_{f-1},\ldots,\iota_1)}. \tag{4.22}$$

iii) The approximation and resampling step

Since eq. (4.21) and eq. (4.22) are hard to analyse analytically, we instead approximate the posterior PDF of X_f numerically using Importance Sampling [47].

In the beginning of the particle filtering algorithm (flight $f=0$), we initialize M particles, where each particle j has a weight $w_0^j = \frac{1}{M}$ after flight $f=0$, an initial value of the degradation and a different set of model parameters for the function r (eq. (4.17)). These model parameters are drawn from a prior distribution for the model parameters of the recurrence function r.

After flight f, we estimate for each particle j the degradation x_f^j after flight f, with i) the recurrence function $r(\cdot)$ in eq. (4.17), ii) the model parameters of $r(\cdot)$ belonging to particle j and iii) the past estimated degradation x_{f-1}^j. Then, eq. (4.22) with the posterior PDF of the degradation X_f of the online monitored component after flight f is approximated as [46]:

$$p_{X_f|\mathcal{I}_f,\ldots,\mathcal{I}_1}\left(x_f|\iota_f,\ldots,\iota_1\right) \approx \sum_{j=1}^{M} \hat{w}_f^j \cdot \delta\left(x_f - x_f^j\right), \tag{4.23}$$

where $\delta(\cdot)$ is the Dirac function, and $\hat{w}_f^j$ is the normalized weight of the j^{th} particle, $j \in \{1,2,\ldots,M\}$ after flight f. The (non-normalized) weight of a particle j is calculated with the new health indicator value $\mathcal{I}_f$ as follows [46]:

$$w_f^j = w_{f-1}^j \frac{p_{\mathcal{I}_f|X_f^j}(\iota_f|x_f^j)\cdot p_{X_f^j|X_{f-1}^j}(x_f^j|x_{f-1}^j)}{\tilde{p}_{X_f^j|X_{f-1}^j,\mathcal{I}_f,\ldots,\mathcal{I}_1}(x_f^j|x_{f-1}^j,\iota_f,\ldots,\iota_1)}, \tag{4.24}$$

In most studies [36, 45, 46, 48, 49], the importance density function $\tilde{p}_{X_f^j|X_{f-1}^j,\mathcal{I}_f,\ldots,\mathcal{I}_1}(x_f^j|x_{f-1}^j,\iota_f,\ldots,\iota_1)$ is simply chosen as the prior $p_{X_f^j|X_{f-1}^j}(x_f^j|x_{f-1}^j)$. Following this, we update the weights with:

$$w_f^j = w_{f-1}^j p_{\mathcal{I}_f|X_f^j}(\iota_f|x_f^j). \tag{4.25}$$

The probability $p_{\mathcal{I}_f|X_f^j}(\iota_f|x_f^j)$ can be calculated from eq. (4.18). The exact calculation depends on the form of $g(\cdot)$ and the assumed distribution of the measurement noise. Finally, the weights are normalized as [46]:

$$\hat{w}_f^j = \frac{w_f^j}{\sum_{h=1}^{M} w_f^h}. \tag{4.26}$$

After flight f, these normalized weights are also used to resample the particles. The probability to resample a particle j equals $\hat{w}_f^j$. Each new particle receives an equal weight of $\frac{1}{M}$ [46, 50]. Then, we estimate the posterior PDF of the degradation X_{f+1} with these new particles and with the new health indicator value for flight $f+1$.

iv) Estimating the RUL

After all available measurements of component i (from flight $f^{i,\text{alarm}} - m$ up to the current flight, at which moment the RUL is estimated, $f^{i,\text{cur}}$) are used to estimate the PDF of the degradation $X_{f^{i,\text{cur}}}$ after the most recent flight $f^{i,\text{cur}}$, we predict the RUL by propagating the degradation of each particle j over time, using the recurrence function $r(\cdot)$ (eq. (4.17)) and the model parameters of particle j. With the failure threshold D, the RUL RUL^j as estimated with particle j is:

$$\text{RUL}^j = \min\{f : x_f^j \geq D\}. \tag{4.27}$$

The RUL estimate of each particle has a probability of $\frac{1}{M}$. All particles together give the PDF of the RUL of component i, as estimated after flight $f^{i,\text{cur}}$.

4.3. CASE STUDY AND RESULTS FOR COOLING UNITS (CUS)

To illustrate the approach as introduced in Section 4.2, we consider 8 cooling units (CUs) originating from a fleet of aircraft operated by a large European airline. For each CU, a time-series of measurements is recorded during the operation of these aircraft until the components have failed (run-to-failure data).

The CU consists of a compressor, a condenser, an evaporator and a flash tank (see Figure 4.4). After many days of operations, the CU becomes clogged with burned oil, moist and sludge from the compressor. This accelerates the compressor wear. Long-term exposure to these conditions negatively affects the health of the CU, which, in time, leads to a failure [51].

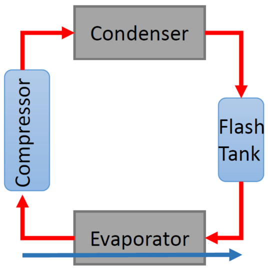


Figure 4.4.: Schematic representation of a cooling unit.

4.3.1. HEALTH INDICATOR FOR THE CUS

For each CU, run-to failure measurements (i.e., measurements until the failure of the component) are available after some initial usage of the component. The

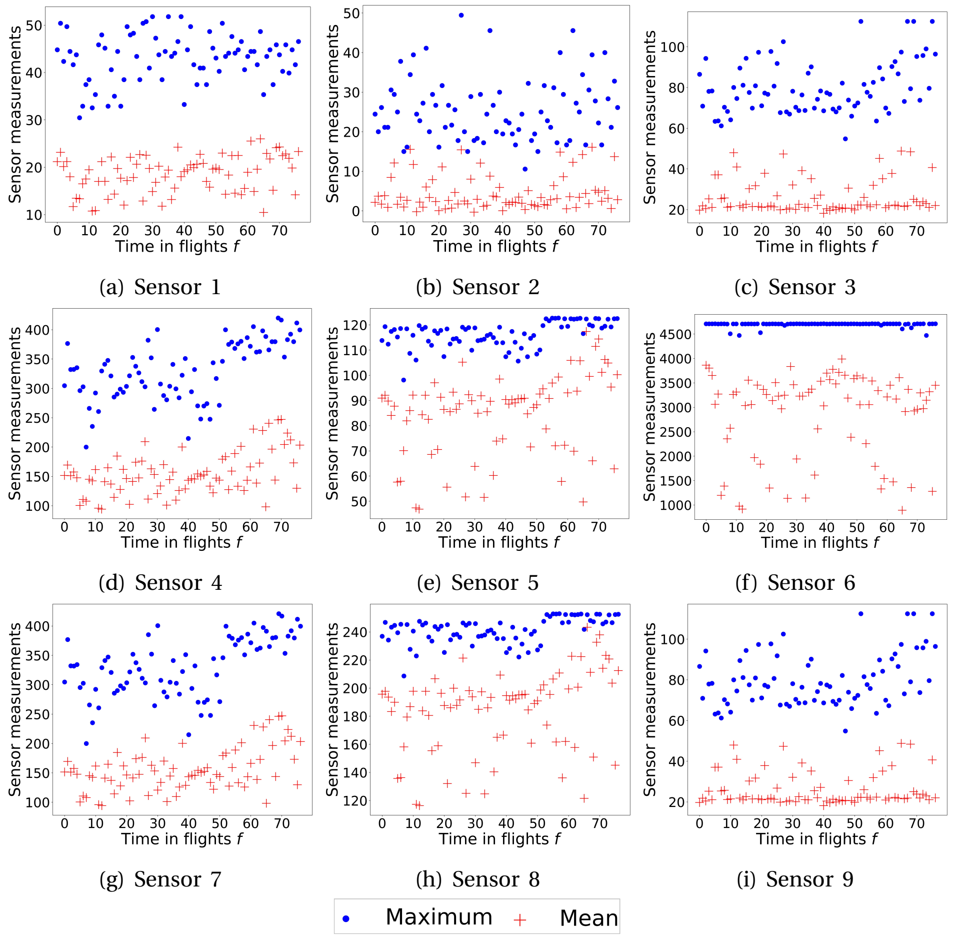


Figure 4.5.: Mean and maximum sensor measurement per flight for one CU for all nine available sensors. The CU fails at flight 77.

time-series with sensor measurements consists of 50 flights (short time-series) up to 390 flights (long time-series). For each CU, there are nine sensors $S = \{1, 2, \ldots, 9\}$, each generating a measurement every 10 seconds during each flight. For the purpose of our analysis, the data sets are anonymized, and thus it is unknown what each sensor measures. The sensor data shows an increasing trend in the mean and maximum sensor measurements towards failure. As an example, Figure 4.5 shows the mean and maximum sensor measurement per flight until the moment of failure for one CU and for each of the nine available sensors.

Let $y_{f,b}^{i,s}$ denote the b^{th} measurement during flight f for CU i generated by sensor

s. We first normalize the sensor measurements for each sensor $s \in \{1,2,\dots,9\}$:

$$\hat{y}^{i,s}_{f,b} = \frac{y^{i,s}_{f,b} - \min_s}{\max_s - \max\limits_{b \in 1,\dots,B^i_f} \left(y^{i,s}_{f,b} \right)}, \tag{4.28}$$

where $\min_s$ and $\max_s$ denote the available minimum and maximum measurement generated by sensor s, respectively, and B^i_f denotes the total number of measurements recorded during flight f of CU i. This normalization enhances the increase in the mean and maximum measurements towards failure.

With these normalized sensor measurements, we consider several health indicators, as described in [4, 52]. We construct each health indicator for the measurements of each sensor, and calculate for each sensor the correlation coefficient (trendability) between the health indicator and the time to failure for the last 50 flights before failure (where 50 flights is the length of the shortest time-series of measurements). Table 4.1 gives an overview of the considered health indicators and the corresponding correlation coefficients. An extensive description of the health indicators is in [52].

Health indicator	Sensors								
	1	2	3	4	5	6	7	8	9
$\text{RMS}_f = \sqrt{\frac{1}{B^i_f} \sum_{b=1}^{B^i_f} \left(\hat{y}^{i,s}_{f,b} \right)^2}$	-0.06	0.12	0.27	0.45	0.46	-0.13	0.44	**0.47**	0.26
$\Delta\text{RMS}_f = \text{RMS}_f - \text{RMS}_{f-1}$	**0.02**	0.01	**0.02**	**0.02**	0.01	-0.01	**0.02**	0.01	**0.02**
Peak-to-peak$_f$ = max($\hat{y}$) - min($\hat{y}$)	-0.08	0.11	0.24	0.43	0.38	-0.16	**0.44**	0.38	0.25
Crest Factor$_f$ = $\frac{\max(\hat{y})}{\text{RMS}_f}$	-0.05	0.06	0.08	-0.02	-0.06	0.08	-0.02	-0.06	**0.13**
Kurtosis$_f$	-0.03	**-0.11**	0.05	**-0.11**	0.05	0.10	**-0.11**	0.05	0.05
Skewness$_f$	**-0.13**	0.09	0.05	-0.11	-0.03	-0.00	-0.11	-0.03	0.05

Table 4.1.: Overview of considered health indicators. For each sensor, the correlation coefficient between the health indicator and the time to failure is given. The highest absolute correlation coefficient per health indicator is denoted in bold. Here, $\max(\hat{y}) = \max\limits_{b \in 1,\dots,B^i_f} \left(\hat{y}^{i,s}_{f,b} \right)$ and $\min(\hat{y}) = \min\limits_{b \in 1,\dots,B^i_f} \left(\hat{y}^{i,s}_{f,b} \right)$

The highest correlation coefficients are obtained for the Root Mean Square (RMS) health indicator. We thus construct a health indicator based on the RMS of the measurements [15, 52, 53]. A health indicator based on the RMS of measurements is often employed in literature [4, 52], for example for health monitoring of gearboxes [54], turbine cutting tools [53] and rolling element bearings [55]. The RMS of sensor s of CU i during flight f is:

$$\text{RMS}^{i,s}_f = \sqrt{\frac{1}{B^i_f} \sum_{b=1}^{B^i_f} \left(\hat{y}^{i,s}_{f,b} \right)^2}. \tag{4.29}$$

Now, a health indicator $\mathcal{I}_f^i$ for CU i during flight f is obtained as the moving average of the maximum RMS obtained by the sensors $s \in S'$ during a single flight:

$$\mathcal{I}_f^i = \frac{1}{N} \sum_{l=f-N}^{f} \max_{s \in S'} \left(\text{RMS}_l^{i,s} \right), \tag{4.30}$$

where $N = 10$ and where $S' \subseteq S$ is the set of sensors for which the moving average of the RMS of the last 50 flights before failure has a correlation coefficient with the time of failure of at least 0.70. We therefore include sensor 4, 5, 7 and 8 in the health indicator. This final health indicator enhances the increase in the mean and the maximum measurements towards failure as observed in Figure 4.5.

Figure 4.6 shows the health indicator and the alarm threshold T^{alarm} when considering the run-to-failure data of the 8 CUs. For the alarm threshold, we assume $k = 2$ (see eq. (4.4)), and we estimate the mean μ and the standard deviation σ of the health indicators with the measurements of the first five flights of each cooling unit. With this, we obtain an alarm threshold of $T^{\text{alarm}} = 11.54$. When analyzing all CUs, the alarm threshold is reached between 2 to 40 flights before the actual failure time.

Figure 4.6 also shows that the degradation trends towards failure differs across the components. Also, the failure threshold differs among the 8 CUs. In the next section, we therefore cluster the health indicators to identify the main degradation models and the corresponding failure thresholds.

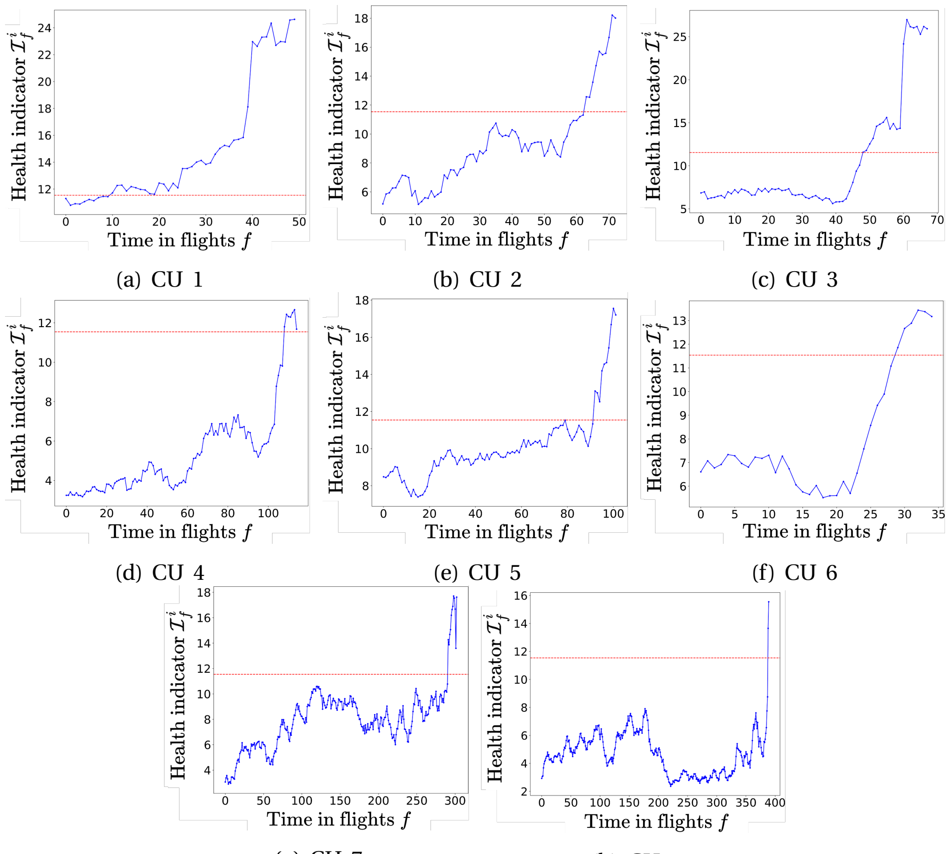
4.3.2. CLUSTERS FOR THE HEALTH INDICATORS

We cluster the $n = 8$ health indicators with dynamic time-warping (see Section 4.2.2) and identify cluster-specific degradation models for the CUs. For this, we use the run-to-failure data of the 8 CUs, where we start the health indicator $m = 10$ flights before the CU reaches the alarm threshold until failure.

Figure 4.7 shows the minimum distance $W(\mathcal{I}^i, \mathcal{I}^j)$, $i, j \in \{1, 2, \ldots, 8\}$, between the health indicators of the 8 CUs, normalized with the maximum length of the health indicators $\mathcal{I}^i$ and $\mathcal{I}^j$. Here, the minimum distances $W(\mathcal{I}^i, \mathcal{I}^j)$ for any two components varies between 0.44 (high similarity between the degradation trends) to 4.24 (large dissimilarities between the degradation trends).

Using the distance $W(\mathcal{I}^i, \mathcal{I}^j), i, j \in \{1, 2, \ldots, 8\}$, between the 8 health indicators, we construct the graph $G(V, E)$, with $|V| = 8$ nodes. Here, a node $v^i \in V$ in the graph corresponds to the health indicator of component $i, i \in \{1, 2, \ldots, 8\}$. For each CU i and a given δ, an edge is constructed between those δ nodes that have the smallest distance to node v^i (CU i). For example, if $\delta = 2$ and CU $i = 4$, an edge is constructed between node v^4 and node v^6 and between node v^4 and v^8, since the distances between node v^4 and nodes v^6 and v^8 are the smallest (Figure 4.7).

In Figure 4.8, we set $\delta = 1$. This results in two clusters of CUs, one cluster with CUs $\{4, 6, 8\}$, and one cluster with CUs $\{1, 2, 3, 5, 7\}$. The health indicator in Figure 4.6 shows that the CU 4, 6 and 8 fail soon after the health indicator reaches the alarm threshold, when the health indicator reaches a value of approximately 15. The health indicator shows a monotonic, very sharp increase in the last flights before failure. In contrast, CUs 1, 2, 3, 5 and 7 fail when the health indicator reaches a value



(a) CU 1 (b) CU 2 (c) CU 3

(d) CU 4 (e) CU 5 (f) CU 6

(g) CU 7 (h) CU 8

Figure 4.6.: Health indicator for the aircraft cooling units (CUs).

of approximately 20. The degradation trend, as represented by health indicator, of these CUs seems to slightly accelerate towards failure.

For $\delta = 2$ (see Figure 4.9), the same two clusters $\{4,6,8\}$ and $\{1,2,3,5,7\}$ emerge. However, the CUs in Figure 4.9 are now connected by multiple edges.

For $\delta = 3$ (see Figure 4.10), one large cluster emerges. Here, the CUs in cluster $\{4,6,8\}$ and cluster $\{1,2,3,5,7\}$ are still grouped together (see Figure 4.8), and connected to each other through CU 2 and 3 only.

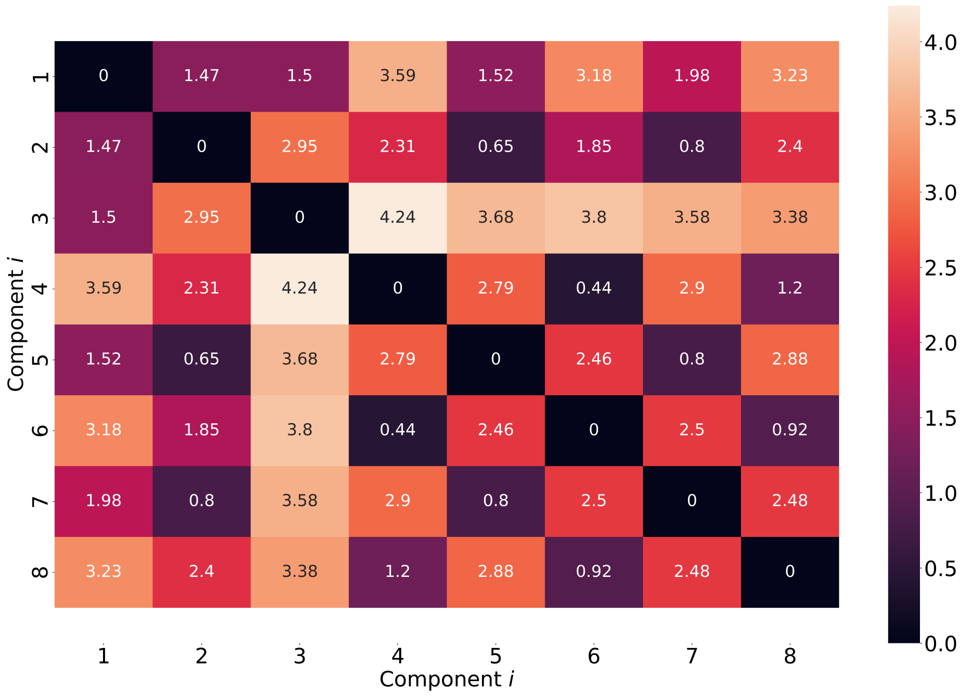
Finally, for $\delta = 4$ (see Figure 4.11), all CUs form one large cluster, which several edges between CUs 4, 6 and 8, and CU 1, 2, 3, 5 and 7.

Following the clusters for $\delta = 1$ and $\delta = 2$, we define the following two clusters:

$$\text{Cluster } 1 = \{4,6,8\} \tag{4.31}$$

$$\text{Cluster } 2 = \{1,2,3,5,7\} \tag{4.32}$$

For the two clusters in eq. (4.31) and (4.32), we next define the cluster-specific



Component i	1	2	3	4	5	6	7	8
1	0	1.47	1.5	3.59	1.52	3.18	1.98	3.23
2	1.47	0	2.95	2.31	0.65	1.85	0.8	2.4
3	1.5	2.95	0	4.24	3.68	3.8	3.58	3.38
4	3.59	2.31	4.24	0	2.79	0.44	2.9	1.2
5	1.52	0.65	3.68	2.79	0	2.46	0.8	2.88
6	3.18	1.85	3.8	0.44	2.46	0	2.5	0.92
7	1.98	0.8	3.58	2.9	0.8	2.5	0	2.48
8	3.23	2.4	3.38	1.2	2.88	0.92	2.48	0

Figure 4.7.: Heat map of the minimum Euclidean distance $W(\mathcal{I}^i, \mathcal{I}^j)$ for the degradation trends of components $i, j \in \{1, 2, \ldots, 8\}$.

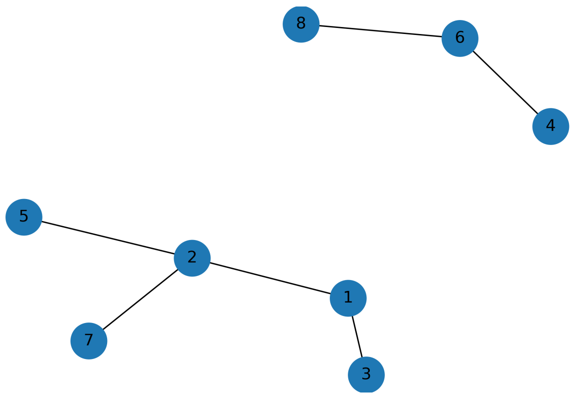


Figure 4.8.: Clustering graph $G(V, E)$ for $\delta = 1$.

degradation models and the cluster-specific failure threshold.

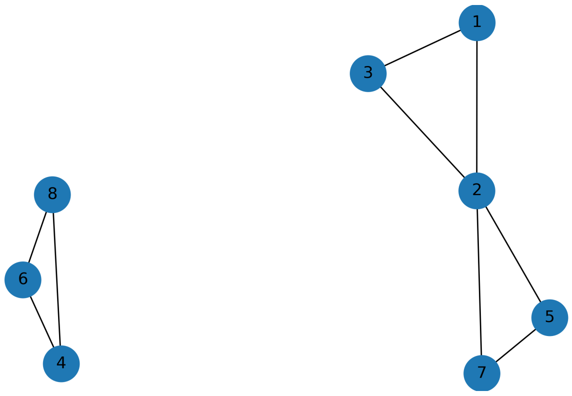


Figure 4.9.: Clustering graph $G(V,E)$ for $\delta = 2$.

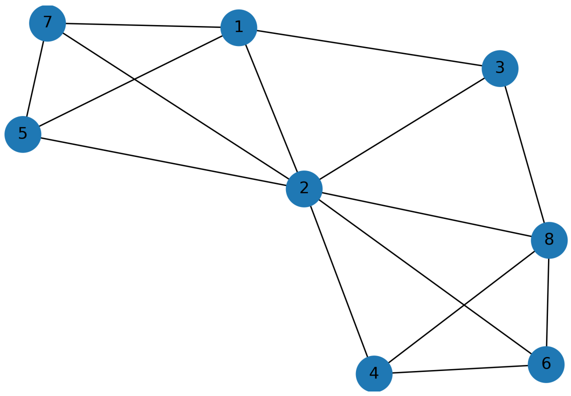


Figure 4.10.: Clustering graph $G(V,E)$ for $\delta = 3$.

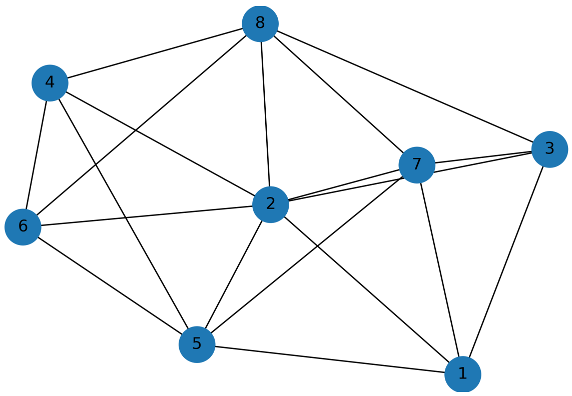


Figure 4.11.: Clustering graph $G(V,E)$ for $\delta = 4$.

4.3.3. CLUSTER 1 - LINEAR DEGRADATION MODEL

The health indicators of CUs 4, 6 and 8 in Cluster 1 are given in Figures 4.6d, 4.6f and 4.6h. All these CUs fail when their health indicator reaches a value of approximately 15. Thus, we consider a failure threshold $D = 15$ for Cluster 1.

Once the health indicators of the CUs in Cluster 1 reach the alarm threshold T^{alarm}, the indicators have a very sharp, but monotonically increasing trend towards failure. We thus consider the following linear degradation model for Cluster 1 [11]:

$$X_t^i = \alpha^i + \beta^i t + \omega^i \tag{4.33}$$

where X_t^i is the degradation of component i at time t, α^i is the initial degradation, β^i is a model parameter and ω^i is the noise in the degradation process. As in [36], we ignore the degradation noise ω^i because it is handled through the uncertainty in the model parameters and in the measurements [56]. Linear degradation models are often considered for prognostics [30], for example for milling machines [31], batteries [32], aircraft engines [33] and engine bleed valves [34].

Following the degradation model in eq. (4.33), we consider the derivative:

$$dX_t^i = \beta^i dt. \tag{4.34}$$

When re-writing the above equation in the form of equations (4.17) and (4.18) with $dt = 1$ flight, we obtain that:

$$X_f^i = X_{f-1}^i + \beta_f^i \tag{4.35}$$

$$\mathcal{I}_f^i = X_f^i + \nu_f^i \tag{4.36}$$

where $\nu_f^i \sim N(0, \sigma_{\nu_f^i})$ is the measurement noise at flight f. Here, we assume that the health indicator directly represents the degradation in the CU. The prior distributions of the model parameters, with which the particles are initialized, are $\beta_0^i \sim U(0.01, 1)$ and $\sigma_{\nu_0^i} \sim U(1, 2)$. The initial degradation of each particle is initialized with the minimum observed health indicator value of the partial health indicator (up to the current flight $f^{i,\text{cur}}$ at which moment the RUL is predicted) of component i. These initial distributions are determined such that sample impoverishment, the degeneracy problem and the non-convergence of the particle filtering algorithm are avoided [56].

4.3.4. CLUSTER 2 - EXPONENTIAL DEGRADATION MODEL

The health indicators of CU 1, 2, 3, 5 and 7 in Cluster 2 are given in Figure 4.6. All these CUs fail when their health indicator reaches a value of approximately 20. Thus, we consider a failure threshold $D = 20$ for Cluster 2.

Moreover, the increments of the health indicator are steadily increasing towards failure for these CUs. We thus consider the following exponential degradation model for the CUs in Cluster 2:

$$X_t^i = \alpha^i + \exp(\beta^i t) + \omega^i \tag{4.37}$$

where X_t^i is the degradation of component i at time t, α^i is the initial degradation, β^i is a model parameter and ω^i is the noise in the degradation process. As before, we ignore the degradation noise ω^i [36]. Exponential degradation models are considered in many prognostic studies [30], for bearings [35, 37], batteries [36, 38] and railway turnout systems [39].

Taking the logarithm of eq. (4.37) gives:

$$S_t^i = \ln\left(X_t^i - \alpha_i\right) = \beta^i t. \tag{4.38}$$

As in [11], we assume that $\alpha_i = 0$. Following the degradation model in eq. (4.38), we consider the derivative:

$$dS_t^i = \beta^i dt. \tag{4.39}$$

Re-writing the above equation in the form of equations (4.17) and (4.2) with $dt = 1$ flight, we obtain:

$$S_f^i = S_{f-1}^i + \beta_f^i \tag{4.40}$$

$$\mathcal{I}_f^i = X_f^i + \nu_f^i, \tag{4.41}$$

where $\nu_f^i \sim N(0, \sigma_{\nu_f^i})$ is the measurement noise at flight f. As before, we assume that the health indicator directly represents the degradation in the CU. The initial distributions of the model parameters are initialized as $\beta_0^i \sim U(0.01, 0.1)$ and $\sigma_{\nu_0^i} \sim U(1,2)$, while the initial degradation is again the minimum value of the partial health indicator. The parameter distributions are again initialized such that sample impoverishment, the degeneracy problem and the non-convergence of the particle filtering algorithm are avoided [56].

Lastly, with the degradation models introduced above, we apply the particle filtering algorithm (see Section 4.2) to estimate the RUL of CUs.

4.3.5. RUL estimation

In this section, we estimate the RUL of the 8 CUs using leave-one-out cross validation. Specifically, we select one CU $i \in \{1, 2, \ldots, 8\}$ for which we predict the RUL. We consider the partial health indicator of this CU i up to the moment of generating a RUL prognostic. We determine the minimum distance between this partial health indicator and the partial health indicators of the CUs in Cluster 1 and Cluster 2 (see eq. (4.31) and (4.32)), which do not include CU i in the leave-one-out cross validation. We assign CU i to the cluster for which the average distance between the cluster and CU i is minimum (see Section 4.2.3). We then assume that CU i follows the degradation model and the failure threshold of this cluster. We estimate the RUL of CU i using particle filtering with 10,000 particles.

We assume that the clusters remain the same when taking one CU out. When considering $\delta = 1$, this is indeed the case, except when taking out CU 2. In this case, the cluster with CU 1, 3, 5 and 7 (and 2 in the offline clustering) splits in two separate clusters, one cluster with CU 1 and 3, and one cluster with CU 5 and 7.

	RUL prognostic as soon as CU diagnosed as unhealthy			RUL prognostic 10 flights before failure (RUL = 10)	
CU i	Actual RUL at $f^{i,\text{alarm}}$	Assigned cluster	Mean estimated RUL	Assigned cluster	Mean estimated RUL
1	40	2 (Exp.)	36	2 (Exp.)	2
3	20	1 (Lin.)	10	2 (Exp.)	7
7	12	2 (Exp.)	11	2 (Exp.)	6
2	10	2 (Exp.)	9	2 (Exp.)	9
5	10	2 (Exp.)	8	2 (Exp.)	8
4	7	1 (Lin.)	10	-	-
6	6	1 (Lin.)	7	-	-
8	2	1 (Lin.)	1	-	-
RMSE			4.04		4.34

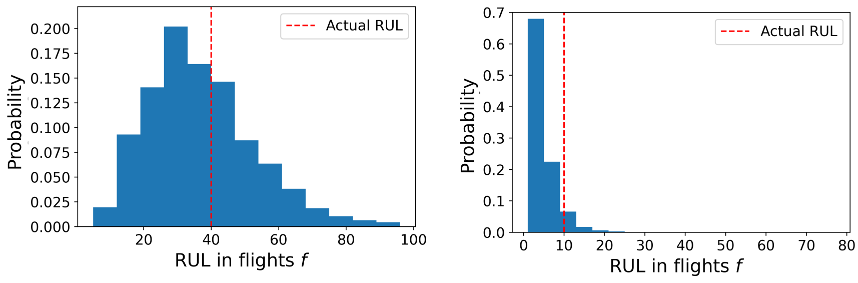
Table 4.2.: The mean RUL prognostics (in flights) for the CUs. The RUL is estimated once the CU is diagnosed as unhealthy and 10 flights before failure.

Table 4.2 shows for each CU the cluster it is assigned to as soon as T^{alarm} is reached, i.e., as soon the CU is diagnosed as unhealthy. CU 1 is diagnosed as unhealthy and an alarm is triggered at an early stage, at 40 flights before the actual failure time. CUs 2, 3, 4, 5, 6, 7 are diagnosed as unhealthy up to 6 flights before the actual failure. For CU 8, an alarm is triggered only 2 flights before the actual failure.

When the CU is diagnosed as unhealthy, CUs 1, 2, 5 and 7 are assigned to Cluster 2, which assumes an exponential degradation model, whereas CUs 3, 4, 6 and 8 are assigned to Cluster 1, which assumes a linear degradation model. Only CU 3 is assigned to a different cluster (namely cluster 1) than in the offline clustering, which leads to an underestimation of the RUL.

Table 4.2 also shows the assigned clusters 10 flights before the time of failure. CUs 4, 6, and 8 fail within less than 10 flights as soon as they are diagnosed as unhealthy, and we thus do not estimate the RUL at 10 flights before failure. 10 flights before failure, CUs 1, 2, 3, 5 and 7 are all assigned to the same, exponential cluster, as in the offline clustering. Our approach thus identifies well the underlying linear or exponential degradation trend of upcoming failures.

Last, Table 4.2 shows that the RUL of the CUs is accurately estimated, with a Root Mean Square Error (RMSE) with the mean estimated RUL of only 4.04 flights (when the CUs are diagnosed as unhealthy) and 4.34 flights (10 flights before failure). Figure 4.12 illustrates the estimated PDF of the RUL of CU 1, the moment it is diagnosed as unhealthy and 10 flights before failure. The largest absolute prediction error with the mean estimated RUL is made for CU 3 at the moment this CU is diagnosed as unhealthy at 20 flights before failure, at which moment we estimate that the mean RUL is 10 flights (i.e., the absolute prediction error is 10 flights). This is also the only CU that is assigned to the wrong cluster in Table 4.2. The clustering thus seems to contribute to the accuracy of the RUL prognostics.



(a) The estimated PDF of the RUL of CU 1 the moment CU 1 is diagnosed as unhealthy (actual RUL is 40 flights),

(b) The estimated PDF of the RUL of CU 1 when the actual RUL is 10 flights.

Figure 4.12.: The estimated PDF of the RUL of CU 1 at two moments in time.

4.4. CONCLUSIONS

In this chapter, we propose an online, model-based RUL prognostic approach for aircraft components. By clustering the health indicators of the components, we have determined cluster-specific degradation models and failure thresholds. Together with a particle filtering algorithm, these degradation models have been used to estimate the PDF of the RUL of aircraft components. To illustrate the approach, we estimate the PDF of the RUL and the mean RUL of several aircraft cooling units for which sensor measurements are recorded during the operation of the aircraft. The results show that the proposed method is able to identify the degradation models of the cooling units and to accurately estimate the RUL. From a practical point of view, our RUL estimation results have the potential to support aircraft maintenance planners with the maintenance task scheduling.

As future work, we plan to optimize the maintenance schedule with the RUL prognostics, in order to evaluate the impact of predictive maintenance on the maintenance costs and the aircraft availability.

REFERENCES

[1] Mitici, M., & de Pater, I. (2021). Online model-based Remaining-Useful-Life prognostics for aircraft cooling units using time-warping degradation clustering. *Aerospace*, *8*(6), Article number: 168.

[2] de Pater, I., & Mitici, M. (2021, June 28 - July 2). Model-based Remaining-Useful-Life prognostics for aircraft cooling units. *Proceedings of the European Conference of the Prognostics and Health Management (PHM) Society*, *6*, Virtual, Pages: 1–8.

[3] Maintenance Cost Technical Group (MCTG). (2020). *Airline maintenance cost executive commentary (FY2019 data), public version* (tech. rep.). International Air Transport Association (IATA).

[4] Lei, Y., Li, N., Guo, L., Li, N., Yan, T., & Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing*, *104*, Pages: 799–834.

[5] Khumprom, P., Grewell, D., & Yodo, N. (2020). Deep neural network feature selection approaches for data-driven prognostic model of aircraft engines. *Aerospace*, *7*(9), Article number: 132.

[6] Li, X., Ding, Q., & Sun, J.-Q. (2018). Remaining Useful Life estimation in prognostics using deep Convolution Neural Networks. *Reliability Engineering & System Safety*, *172*, Pages: 1–11.

[7] Babu, G. S., Zhao, P., & Li, X.-L. (2016, April 16-19). Deep Convolutional Neural Network based regression approach for estimation of Remaining Useful Life. *Proceedings of the 21st International Conference on Database Systems for Advanced Applications (DASFAA)*, Dallas, Texas, USA, Pages: 214–228.

[8] Xiang, S., Qin, Y., Zhu, C., Wang, Y., & Chen, H. (2020). Long Short-Term Memory Neural Network with weight amplification and its application into gear Remaining Useful Life prediction. *Engineering Applications of Artificial Intelligence*, *91*, Article number: 103587.

[9] Chen, J., Jing, H., Chang, Y., & Liu, Q. (2019). Gated Recurrent Unit based Recurrent Neural Network for Remaining Useful Life prediction of nonlinear deterioration process. *Reliability Engineering & System Safety*, *185*, Pages: 372–382.

[10] Zhang, Z., Si, X., Hu, C., & Lei, Y. (2018). Degradation data analysis and Remaining Useful Life estimation: A review on Wiener-process-based methods. *European Journal of Operational Research*, *271*(3), Pages: 775–796.

[11] Si, X.-S., Wang, W., Chen, M.-Y., Hu, C.-H., & Zhou, D.-H. (2013). A degradation path-dependent approach for Remaining Useful Life estimation with an exact and closed-form solution. *European Journal of Operational Research*, *226*(1), Pages: 53–66.

[12] Dalla Vedova, M. D., Germanà, A., Berri, P. C., & Maggiore, P. (2019). Model-based fault detection and identification for prognostics of electromechanical actuators using genetic algorithms. *Aerospace*, *6*(9), Article number: 94.

[13] Dong, T., & Kim, N. H. (2021). Methods of identifying correlated model parameters with noise in prognostics. *Aerospace*, *8*(5), Article number: 129.

[14] Al-Mohamad, A., Hoblos, G., & Puig, V. (2020). A hybrid system-level prognostics approach with online RUL forecasting for electronics-rich systems with unknown degradation behaviors. *Microelectronics Reliability*, *111*, Article number: 113676.

[15] Nesci, A., Martin, A. D., Jacazio, G., & Sorli, M. (2020). Detection and prognosis of propagating faults in flight control actuators for helicopters. *Aerospace*, *7*(3), Article number: 20.

[16] Arts, J., & Basten, R. (2018). Design of multi-component periodic maintenance programs with single-component models. *IISE transactions*, *50*(7), 606–615.

[17] Christer, A. (1982). Modelling inspection policies for building maintenance. *Journal of the Operational Research Society*, *33*, 723–732.

[18] Christer, A., & Waller, W. (1984). Delay time models of industrial inspection maintenance problems. *Journal of the Operational Research Society*, *35*(5), 401–406.

[19] Raghavan, N., & Frey, D. D. (2015, June 22-25). Remaining Useful Life estimation for systems subject to multiple degradation mechanisms. *IEEE Conference on Prognostics and Health Management (PHM)*, Austin, Texas, USA, Pages: 1–8.

[20] Zhang, Z., Si, X., Hu, C., & Kong, X. (2015). Degradation modeling–based Remaining Useful Life estimation: A review on approaches for systems with heterogeneity. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, *229*(4), Pages: 343–355.

[21] Zheng, S., Ristovski, K., Farahat, A., & Gupta, C. (2017, June 19-21). Long Short-Term Memory network for Remaining Useful Life estimation. *IEEE International Conference on Prognostics and Health Management (ICPHM)*, Dallas, Texas, USA, Pages: 88–95.

[22] Ellefsen, A. L., Bjørlykhaug, E., Æsøy, V., Ushakov, S., & Zhang, H. (2019). Remaining Useful Life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering & System Safety*, *183*, Pages: 240–251.

[23] Lee, J., & Mitici, M. (2020). An integrated assessment of safety and efficiency of aircraft maintenance strategies using agent-based modelling and stochastic Petri Nets. *Reliability Engineering & System Safety*, *202*, Article number: 107052.

[24] Omri, N., Al Masry, Z., Mairot, N., Giampiccolo, S., & Zerhouni, N. (2020). Industrial data management strategy towards an SME-oriented PHM. *Journal of Manufacturing Systems*, *56*, Pages: 23–36.

[25] Cai, B., Shao, X., Liu, Y., Kong, X., Wang, H., Xu, H., & Ge, W. (2019). Remaining Useful Life estimation of structure systems under the influence of multiple causes: Subsea pipelines as a case study. *IEEE Transactions on Industrial Electronics*, *67*(7), Pages: 5737–5747.

[26] Lim, P., Goh, C. K., Tan, K. C., & Dutta, P. (2015). Multimodal degradation prognostics based on switching Kalman filter ensemble. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(1), Pages: 136–148.

[27] Reuben, L. C. K., & Mba, D. (2014). Diagnostics and prognostics using switching Kalman filters. *Structural Health Monitoring*, *13*(3), Pages: 296–306.

[28] Cui, L., Wang, X., Xu, Y., Jiang, H., & Zhou, J. (2019). A novel switching unscented Kalman filter method for Remaining Useful Life prediction of rolling bearing. *Measurement*, *135*, Pages: 678–684.

[29] Lim, C. K. R., & Mba, D. (2015). Switching Kalman filter for failure prognostic. *Mechanical Systems and Signal Processing*, *52*, Pages: 426–435.

[30] Ye, Z.-S., & Xie, M. (2015). Stochastic modelling and analysis of degradation for highly reliable products. *Applied Stochastic Models in Business and Industry*, *31*(1), Pages: 16–32.

[31] Fan, M., Zeng, Z., Zio, E., Kang, R., & Chen, Y. (2018). A sequential Bayesian approach for Remaining Useful Life prediction of dependent competing failure processes. *IEEE Transactions on Reliability*, *68*(1), Pages: 317–329.

[32] Dong, G., Chen, Z., Wei, J., & Ling, Q. (2018). Battery health prognosis using Brownian motion modeling and particle filtering. *IEEE Transactions on Industrial Electronics*, *65*(11), Pages: 8646–8655.

[33] Li, N., Gebraeel, N., Lei, Y., Fang, X., Cai, X., & Yan, T. (2021). Remaining Useful Life prediction based on a multi-sensor data fusion model. *Reliability Engineering & System Safety*, *208*, Article number: 107249.

[34] Baptista, M., Henriques, E. M., de Medeiros, I. P., Malere, J. P., Nascimento Jr, C. L., & Prendinger, H. (2019). Remaining Useful Life estimation in aeronautics: Combining data-driven and Kalman filtering. *Reliability Engineering & System Safety*, *184*, Pages: 228–239.

[35] Li, N., Lei, Y., Lin, J., & Ding, S. X. (2015). An improved exponential model for predicting Remaining Useful Life of rolling element bearings. *IEEE Transactions on Industrial Electronics*, *62*(12), Pages: 7762–7773.

[36] An, D., Choi, J.-H., & Kim, N. H. (2013). Prognostics 101: A tutorial for particle filter-based prognostics algorithm using MATLAB. *Reliability Engineering & System Safety*, *115*, Pages: 161–169.

[37] Gebraeel, N. Z., Lawley, M. A., Li, R., & Ryan, J. K. (2005). Residual-life distributions from component degradation signals: A Bayesian approach. *IIE Transactions*, *37*(6), Pages: 543–557.

[38] Saha, B., Goebel, K., & Christophersen, J. (2009). Comparison of prognostic algorithms for estimating Remaining Useful Life of batteries. *Transactions of the Institute of Measurement and Control*, *31*(3-4), Pages: 293–308.

[39] Eker, O. F., Camci, F., Guclu, A., Yilboga, H., Sevkli, M., & Baskan, S. (2011). A simple state-based prognostic model for railway turnout systems. *IEEE Transactions on Industrial Electronics*, *58*(5), Pages: 1718–1726.

[40] Shakya, P., Kulkarni, M. S., & Darpe, A. K. (2014). A novel methodology for online detection of bearing health status for naturally progressing defect. *Journal of Sound and Vibration*, *333*(21), Pages: 5614–5629.

[41] Qian, Y., Yan, R., & Hu, S. (2014). Bearing degradation evaluation using recurrence quantification analysis and Kalman filter. *IEEE Transactions on Instrumentation and Measurement*, *63*(11), Pages: 2599–2610.

[42] Singh, J., Darpe, A., & Singh, S. P. (2020). Bearing Remaining Useful Life estimation using an adaptive data-driven model based on health state change point identification and K-means clustering. *Measurement Science and Technology*, *31*(8), Article number: 085601.

[43] Rabiner, L. (1993). *Fundamentals of speech recognition*. PTR Prentice Hall.

[44] Müller, M. (2007). *Information retrieval for music and motion*. Springer.

[45] Djuric, P. M., Kotecha, J. H., Zhang, J., Huang, Y., Ghirmai, T., Bugallo, M. F., & Miguez, J. (2003). Particle filtering. *IEEE Signal Processing Magazine*, *20*(5), Pages: 19–38.

[46] Qian, Y., & Yan, R. (2015). Remaining Useful Life prediction of rolling bearings using an enhanced particle filter. *IEEE Transactions on Instrumentation and Measurement*, *64*(10), Pages: 2696–2707.

[47] Glynn, P. W., & Iglehart, D. L. (1989). Importance sampling for stochastic simulations. *Management Science*, *35*(11), Pages: 1367–1392.

[48] Zio, E., & Peloni, G. (2011). Particle filtering prognostic estimation of the Remaining Useful Life of nonlinear components. *Reliability Engineering & System Safety*, *96*(3), Pages: 403–409.

[49] Li, N., Lei, Y., Liu, Z., & Lin, J. (2014, June 22-25). A particle filtering-based approach for Remaining Useful Life predication of rolling element bearings. *International Conference on Prognostics and Health Management*, Cheney, Washington, USA, Pages: 1–8.

[50] Doucet, A., & Johansen, A. M. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan & B. Rozovskii (Eds.). Oxford University Press.

[51] de Pater, I., & Mitici, M. (2021). Predictive maintenance for multi-component systems of repairables with Remaining-Useful-Life prognostics and a limited stock of spare components. *Reliability Engineering & System Safety*, *214*, Article number: 107761.

[52] Zhu, J., Nostrand, T., Spiegel, C., & Morton, B. (2014, September 29 - October 2). Survey of condition indicators for condition monitoring systems. *Proceedings of the Annual Conference of the Prognostics and Health Management (PHM) Society*, *6*, Fort Worth, Texas, USA, Pages: 1–13.

[53] Liu, Y., Hu, X., & Zhang, W. (2019). Remaining Useful Life prediction based on health index similarity. *Reliability Engineering & System Safety*, *185*, Pages: 502–510.

[54] Večeř, P., Kreidl, M., & Šmíd, R. (2005). Condition indicators for gearbox condition monitoring systems. *Acta Polytechnica*, *45*(6).

[55] Hu, L., Hu, N.-q., Fan, B., Gu, F.-s., & Zhang, X.-y. (2015). Modeling the relationship between vibration features and condition parameters using relevance vector machines for health monitoring of rolling element bearings under varying operation conditions. *Mathematical Problems in Engineering*, Article number: 123730.

[56] Jouin, M., Gouriveau, R., Hissel, D., Péra, M.-C., & Zerhouni, N. (2016). Particle filter-based prognostics: Review, discussion and perspectives. *Mechanical Systems and Signal Processing*, *72*, Pages: 2–31.

5 NOVEL METRICS TO EVALUATE PROBABILISTIC RUL PROGNOSTICS

In the previous chapter, we estimated the Probability Density Function (PDF) of the Remaining Useful Life (RUL). However, standard metrics to evaluate point RUL prognostics, such as the Root Mean Square Error, are not suitable to evaluate probabilistic RUL prognostics. In this chapter, we therefore propose four new metrics to evaluate probabilistic RUL prognostics in the form of a PDF.

First, we propose to use the Continuous Ranked Probability Score (CRPS) and the weighted CRPS to evaluate the accuracy and sharpness of an individual probabilistic RUL prognostic. Then, we evaluate the reliability of all probabilistic RUL prognostics with the α-Coverage and the Reliability Score.

We test these metrics for aircraft turbofan engines, where we estimate the PDF of the RUL of these engines using a Convolutional Neural Network with Monte Carlo dropout. The proposed metrics are suitable to evaluate the accuracy, sharpness and reliability of these probabilistic RUL prognostics.

Parts of this chapter have been published in:

de Pater, I., & Mitici, M. (2022, July 6-8). Novel metrics to evaluate probabilistic Remaining Useful Life prognostics with applications to turbofan engines. *Proceedings of the 7th European Conference of the Prognostics and Health Management (PHM) Society, 7*, Turin, Italy, Pages: 96–109

5.1. Introduction

Maintenance is undergoing a paradigm shift from time-based maintenance, where tasks are scheduled at fixed time intervals, to predictive maintenance. Under predictive maintenance, sensors continuously measure the condition of components. These measurements are used to estimate the Remaining Useful Life (RUL) of components. In turn, the RUL prognostics are integrated in the maintenance planning. Predictive maintenance has the potential to reduce the maintenance costs, while maintaining the reliability of assets [2].

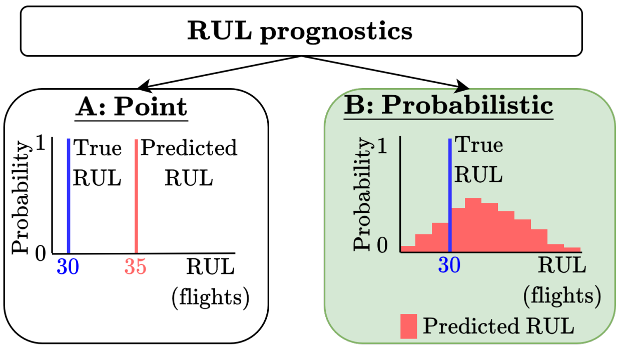


Figure 5.1.: A) Point RUL prognostics, B) Probabilistic RUL prognostics.

Most studies focus on developing *point* RUL prognostics, i.e., one value for the RUL prognostic. For example, a prognostic may indicate that the RUL equals 30 flight cycles for an aircraft component (see Figure 5.1-A). Point RUL prognostics for turbofan engines are developed in [3, 4] using a Convolutional Neural Network (CNN) and in [5] using a Long Short-Term Memory neural network. In [6], point RUL prognostics are obtained for aircraft landing gear brakes using linear regression.

For reliability purposes, however, it is key that the uncertainty associated with the estimated RUL is also determined. In this line, several studies estimate the Probability Density Function (PDF) of RUL, i.e., they develop probabilistic RUL prognostics (see Figure 5.1-B). In [7] and [8], the RUL distribution of turbofan engines is obtained using a Long Short-Term Memory neural network and Deep Gaussian processes, respectively. In [9] the RUL distribution of aircraft cooling units is estimated using particle filtering. Probabilistic RUL prognostics for nuclear components are developed in [10] using Gaussian Process regression. Last, in [11] the RUL distribution is estimated using a noisy Gamma deterioration process.

To evaluate probabilistic RUL prognostics, well-established metrics such as the Root Mean Square Error (RMSE) or the Mean Absolute Error (MAE) are not directly applicable. In principle, the RMSE and MAE could be computed relative to the mean of the estimated RUL distribution. However, this would disregard the variance and sharpness of the estimates, and give little indication of the actual trustworthiness of the RUL prognostics. In [12, 13], a few metrics are proposed to evaluate probabilistic RUL prognostics such as the prognostic horizon, probabilistic $\alpha-\lambda$, (cumulative) relative accuracy and convergence. These metrics evaluate the accuracy of the RUL prognostics, and specifically how this accuracy changes over the lifetime of components. For an example of their usage, see [14]. However, these metrics all require a sequence of RUL prognostics over the lifetime of each component. Yet, for many publicly available degradation test sets, such as the C-MAPSS data set on turbofan engines [15], only one RUL prognostic per test instance can be determined. As such, the prognostic horizon, probabilistic $\alpha - \lambda$, relative accuracy and convergence cannot be used to evaluate these single probabilistic RUL prognostics. Most importantly, these metrics do not explicitly quantify the reliability of the probabilistic RUL prognostics.

In this chapter, we propose two novel metrics to evaluate the accuracy and sharpness of probabilistic RUL prognostics, namely the Continuously Ranked Probability Score (CRPS) and the weighted CRPS. The weighted CRPS uses different penalties when the RUL is overestimated or underestimated. Depending on the type of component, these penalties can be adjusted. For example, for safety-critical components it is important that the RUL is not overestimated, to avoid failures. In such cases, the weighted CRPS applies a larger penalty for overestimating the RUL than for underestimating the RUL. We also propose novel metrics to explicitly evaluate the reliability of the probabilistic RUL prognostics, namely the α-Coverage, the Reliability Diagram and the Reliability Score. The Reliability Diagram provides a visual interpretation of the performance of the prognostics. We illustrate our metrics with the probabilistic RUL prognostics for turbofan engines. We estimate a PDF of the RUL of these engines with a CNN with Monte Carlo dropout.

In Section 5.2, we introduce the CNN with Monte Carlo dropout to estimate a PDF of the RUL for turbofan engines. We next propose metrics to evaluate these estimated RUL distributions in Section 5.3. Last, we illustrate the proposed metrics in a case study in Section 5.4.

5.2. PROBABILISTIC RUL PROGNOSTICS FOR TURBOFAN ENGINES

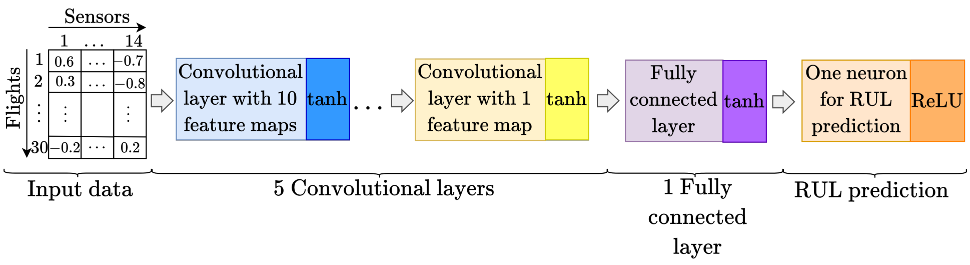


Figure 5.2.: Schematic overview of the CNN architecture for dataset FD001.

In this section, we generate *probabilistic* RUL prognostics for aircraft turbofan engines using a Convolutional Neural Network (CNN) and Monte Carlo dropout.

Specifically, we estimate the PDF of the RUL of an engine, and not just one point value for the RUL. We apply our methodology to the turbofan engines in the C-MAPSS dataset, which is generated using the NASA Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) program [15]. The dataset contains measurements of 21 sensors that monitor the degradation of the turbofan engines. The C-MAPSS dataset consists of four data subsets, each with a different number of operational and fault conditions (see Table 5.1). Each subset contains a training set, with run-to-failure instances, and a test set. For each failure instance in the test set, the data is terminated somewhere before failure with the aim to estimate the RUL. More information on this publicly available data set can be found in [16].

	FD001	FD002	FD003	FD004
Training instances	100	260	100	249
Testing instances	100	259	100	248
Operating conditions	1	6	1	6
Fault conditions	1	1	2	2

Table 5.1.: C-MAPSS datasets for turbofan engines.

We select the 14 out of 21 sensors available in the C-MAPSS dataset that have non-constant measurements. The remaining 7 sensors exhibit constant measurements over time and are thus not considered for RUL estimation. The selected sensor measurements are normalized using min-max normalization [4] with respect to the operating condition [17]. We also include the history of the operating conditions in the input of the CNN, i.e., the number of flights spent in each operating condition, as in [17].

The architecture and hyperparameters of the CNN are similar to the CNN proposed in [4]. Specifically, the CNN consists of 5 convolutional layers, where the first four convolutional layers each have 10 kernels of size 10×1 (i.e., one-dimensional kernels). The last convolutional layer has one kernel of size 3×1, combining all 10 feature maps into one feature map. This last feature map is flattened in a flatten layer, and connected to a fully connected layer. All these layers use the hyperbolic tangent (tanh) activation function. Last, one single neuron is attached to the fully connected layer to estimate the RUL using the Rectified Linear Unit (ReLU) activation function. A schematic overview of this CNN is in Figure 5.2. The weights of the CNN are optimized using the Adam optimizer [18] with a batch size of 512 samples, and a maximum of 250 training epochs. The learning rate is 0.001 for the first 200 epochs, and 0.0001 for the last 50 epochs. A cut-off value R_{early} of 125 flights is applied. We use a window size of 30 flights for FD001 and FD003, of 20 flights for FD002 and of 15 flights for FD004.

To obtain a probability distribution of the RUL using a CNN, we additionally apply Monte Carlo dropout [8, 19]. During the training phase, we apply a dropout rate of $\rho = 0.5$ in each layer, with the exception of the last convolutional layer before the flatten layer, and the first convolutional layer [20]. During the testing phase, we also use dropout and estimate the RUL of each test instance i for $M_i > 1$ times, each time randomly selecting neurons to be dropped. This is illustrated in Figure 5.3. The

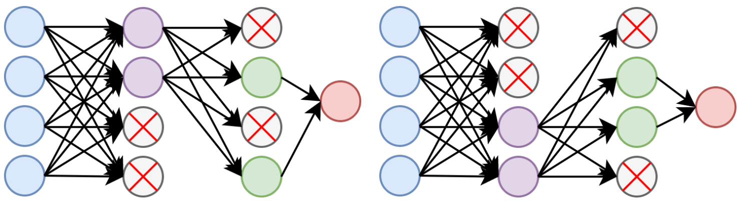
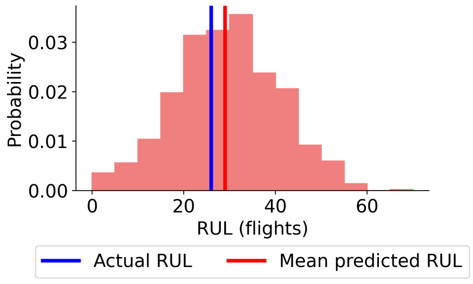

(a) First pass (b) Second pass

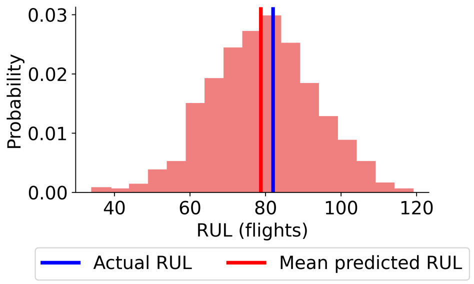
Figure 5.3.: Monte Carlo dropout during two different passes through the network, in a neural network with two fully connected layers.

PDF of the RUL for a test instance i is now created with the M_i RUL prognostics, where each individual prognostic is assigned a probability of $\frac{1}{M_i}$.

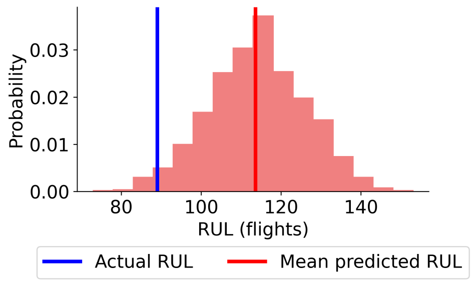
Figure 5.4 shows the obtained PDF of the RUL for engines $i \in \{53, 4, 86, 67\}$ of test set FD001. The PDF of the RUL of engine 53 is well centered around the actual RUL, and the variance is relatively low. The PDF of the RUL of engine 4 is also well centered around the actual RUL, but the variance is larger, suggesting a larger uncertainty about the prediction. In contrast, the PDFs of the RUL of engines 86 and 67 are not well centered around the actual RUL. Moreover, the actual RUL of engine 67 falls outside the estimated PDF for the RUL.



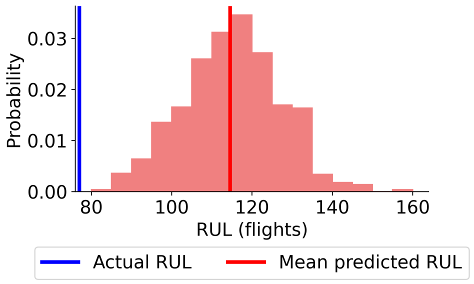
(a) PDF of RUL for engine 53, FD001.



(b) PDF of RUL for engine 4, FD001.



(c) PDF of RUL for engine 86, FD001.



(d) PDF of RUL for engine 67, FD001.

Figure 5.4.: The estimated PDF of the RUL of four individual engines in the test set of FD001.

5.2.1. METRICS OFTEN USED TO EVALUATE POINT RUL PROGNOSTICS

The metrics often used to assess the accuracy of point RUL prognostics are the Mean Absolute Error (MAE), the Root Mean Square Error (RMSE) and the Mean Score. These metrics are computed based on the actual RUL vs. the estimated point RUL. When the PDF of the RUL is estimated instead, the MAE, RMSE and the Mean Score can be computed based on the actual RUL vs. the *mean* estimated RUL.

Formally, let N be the number of test instances in one C-MAPSS test set, and let y_i be the actual RUL for test instance i. Let $\hat{y}_{ij}$, $j \in \{1,2,\ldots,M_i\}$, be the j^{th} RUL prognostic for engine i. Let $\bar{y}_i$ be the mean estimated RUL of test instance i:

$$\bar{y}_i = \frac{1}{M_i}\sum_{j=1}^{M_i} \hat{y}_{ij}. \tag{5.1}$$

Then, when considering probabilistic RUL prognostics, we can evaluate the accuracy of the mean estimated RUL as follows:

$$\text{MAE}^p = \frac{1}{N}\sum_{i=1}^{N} |\bar{y}_i - y_i|. \tag{5.2}$$

$$\text{RMSE}^p = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (\bar{y}_i - y_i)^2}. \tag{5.3}$$

$$\text{Mean Score}^p = \frac{1}{N}\sum_{i=1}^{N} s_i, \tag{5.4}$$

with

$$s_i = \begin{cases} e^{-\frac{\bar{y}_i - y_i}{\gamma}} - 1, & \bar{y}_i - y_i < 0 \\ e^{\frac{\bar{y}_i - y_i}{\delta}} - 1, & \bar{y}_i - y_i \geq 0 \end{cases},$$

with γ and δ user-defined metrics. For the C-MAPSS data set, $\gamma = 13$ and $\delta = 10$ are usually applied [4].

Test set	RMSE^p	MAE^p	Mean Score^p
FD001	12.76	9.22	2.78
FD002	14.74	11.14	3.55
FD003	11.89	9.07	2.43
FD004	18.03	13.44	8.03

Table 5.2.: RMSE^p (in flights), MAE^p (in flights) and Mean Score^p with respect to the mean RUL prognostic - C-MAPSS dataset.

Table 5.2 shows the RMSE^p, MAE^p and Mean Score^p of our probabilistic RUL prognostics, estimated with a CNN with Monte Carlo dropout. Training the neural network took between 12.1 (FD001) to 27.3 (FD002) seconds per epoch on a computer with an Intel Core i7 processor at 2.11 GHz and 8Gb RAM. Our results are comparable with state-of-the-art RUL prognostic results in [5].

However, these metrics do not fully capture the quality of the probabilistic RUL prognostics. The reliability and sharpness of the RUL prognostics is not evaluated, e.g, the variance of the generated PDF of the RUL. For example, for engine 4 (see Figure 5.4b) the absolute error with the mean estimated RUL is only 3.2 flights, and the Score with the mean estimated RUL is only 0.28. The mean estimated RUL is thus very close to the actual RUL. However, the standard deviation of the PDF of the RUL is large ($\sigma = 13.6$), suggesting a large uncertainty in the prediction. This large variance is not reflected in the mean estimated RUL, and thus neither in the RMSE^p, MAE^p and Mean Score^p metrics. Similarly, for engine 86 (see Figure 5.4c), the absolute error with the mean estimated RUL is 24.6 flights, and the score value with the mean estimated RUL is 10.67, which shows that the mean estimated RUL is far off the actual RUL. However, the actual RUL still falls within the PDF of the RUL. This is again not reflected in the mean RUL estimation and thus in the three metrics above. To analyze the full estimated PDF of the RUL with the corresponding uncertainty estimates, we introduce four additional metrics that characterize the reliability, the sharpness and the accuracy associated with the PDFs of the RUL.

5.3. Novel metrics to evaluate probabilistic RUL prognostics

In this section, we introduce the following novel metrics to characterize the reliability, the sharpness and the accuracy of probabilistic RUL prognostics (i.e, when estimating the PDF of the RUL): the Continuous Ranked Probability Score (CRPS), the weighted CRPS (CRPS^W), the α-Coverage, and the Reliability Score (RS). The Python code to calculate the proposed metrics is provided in [1].

5.3.1. Continuous Ranked Probability Score (CRPS)

The Continuous Ranked Probability Score (CRPS) evaluates i) if the estimated RUL distribution is centered around the actual RUL of a component i, i.e., the accuracy of the RUL prognostic, and ii) if the variance of the RUL distribution is low, i.e., the sharpness of the RUL prognostic. A probabilistic RUL prognostic for a component i is best when all RUL prognostics $\hat{y}_{ij}$, $j \in \{1,2,\ldots,M_i\}$ are close to the actual RUL y_i.

The CRPS has been used to evaluate probabilistic predictions for applications such as flight delays [21], sea level pressure and surface temperature [22] and electricity prices [23]. However, to the best of our knowledge, this metric has not yet been used to evaluate probabilistic RUL prognostics.

Let $F_{\hat{y}_i}(x)$ denote the estimated, empirical CDF of the RUL of a component i. The CRPS is defined as follows:

$$\text{CRPS} = \frac{1}{N}\sum_{i=1}^{N} \text{CRPS}_i, \tag{5.5}$$

$$\text{CRPS}_i = \int_{-\infty}^{\infty} \left(F_{\hat{y}_i}(x) - \mathcal{I}\{y_i \leq x\}\right)^2 dx,$$

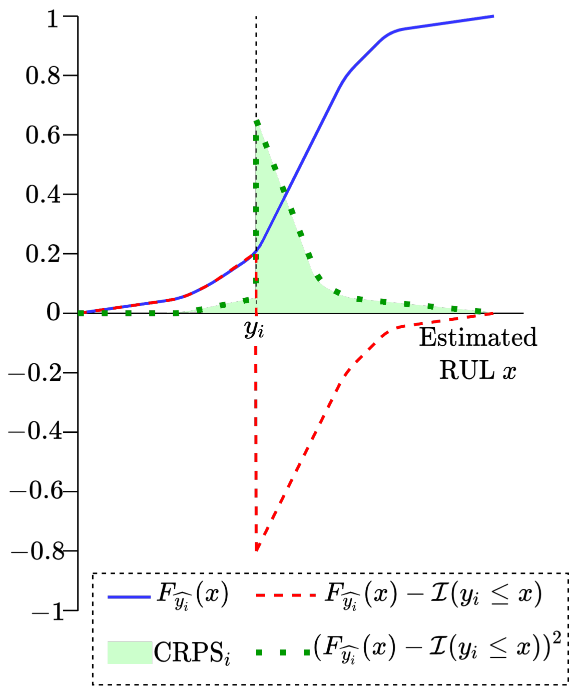


Figure 5.5.: Illustration of the CRPS$_i$ metric for a single component i.

$$\text{with} \quad \mathcal{I}\{y_i \leq x\} = \begin{cases} 1, & y_i \leq x \\ 0, & y_i > x. \end{cases}$$

Intuitively, the CRPS$_i$ for a component i can be seen as a probabilistic generalization of the absolute error $|y_i - \hat{y}_i|$. Specifically, when calculating the CRPS of a point RUL prognostic, we obtain the absolute error of this point RUL prognostic. The smaller the CRPS metric is, the closer the RUL prognostic is to the actual RUL. In an ideal case when a perfect RUL prognostic without uncertainty (i.e., a point RUL prognostic) is obtained, the CRPS equals zero. A comprehensive explanation of this metric can be found in [24].

Figure 5.5 shows a graphical representation of the CRPS for a single component i. The blue solid line represents the empirical CDF of the RUL prognostic of this component i. The light-green area is the CRPS for this component i. This area (i.e., the CRPS value) is small if the accuracy and sharpness of the probabilistic RUL prognostic are high. In general, if the prognostics are accurate and sharp, then most RUL estimates are located close to the true RUL y_i, and the tails of the distribution are small and low. This leads to a low CRPS value. Conversely, the CRPS value increases if the true RUL y_i falls outside the estimated RUL distribution, i.e., if the RUL prognostic is inaccurate.

5.3.2. WEIGHTED CRPS (CRPS^W)

For most components and systems, overestimating the RUL is much more detrimental than underestimating the RUL [4]. Overestimating the failure time is less desirable since missing a component failure may have severer consequences than replacing this component too early. We thus propose the weighted CRPS metric, which considers different penalties for the RUL being overestimated or underestimated. Depending on the type of component, these penalties can be adjusted. In the case of safety-critical components, for example, larger penalties are applied to overestimating the RUL. This is because a RUL overestimation may lead to a failure. The weighted CRPS is defined as follows:

$$\mathrm{CRPS}^W = \frac{1}{N}\sum_{i=1}^{N}\mathrm{CRPS}_i^W, \tag{5.6}$$

$$\begin{aligned}\mathrm{CRPS}_i^W &= (2-\beta)\int_{-\infty}^{y_i}(F_{\hat{y}_i}(x)-\mathcal{I}\{y_i\le x\})^2dx+\beta\int_{y_i}^{\infty}(F_{\hat{y}_i}(x)-\mathcal{I}\{y_i\le x\})^2dx,\\ &= (2-\beta)\int_{-\infty}^{y_i}(F_{\hat{y}_i}(x))^2dx+\beta\int_{y_i}^{\infty}(F_{\hat{y}_i}(x)-1)^2dx,\end{aligned}$$

with $0\le\beta\le2$ an user-specific parameter. The magnitude of the penalty is specified through the weight β. The weight β can be specified by the user, and depends on the domain and on the application.

5.3.3. α-COVERAGE

The CRPS evaluates the accuracy and sharpness of the probabilistic RUL prognostics. It is, however, also important to verify the trustworthiness and reliability of the RUL prognostics. To address this, we introduce the coverage of RUL prognostics, similar to [10]. In this chapter, however, we construct the coverage of the probabilistic RUL prognostics without assuming that these prognostics follow a specific distribution, such as the Gaussian (normal) distribution.

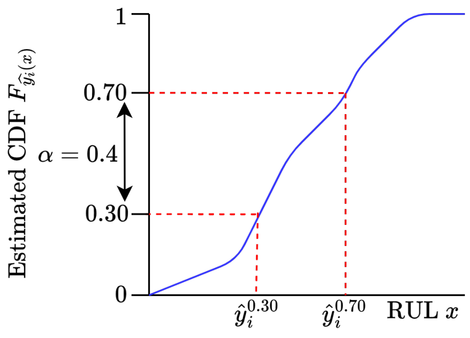


Figure 5.6.: Illustration of the percentiles with the estimated CDF of the RUL of a test instance i.

To calculate the coverage, we first construct a confidence interval around the median of the estimated RUL distribution with width α. For example, let us assume that we have $M_i = 1000$ RUL prognostics for a test instance i, i.e., $\hat{y}_{ij}, j \in \{1, 2, \dots, M_i\}$. Let us consider the confidence interval around the median with width $\alpha = 0.4 = 40\%$. Then, this confidence interval is $[\hat{y}_i^{0.30}, \hat{y}_i^{0.70}]$, with $\hat{y}_i^{0.30}$ the RUL prognostic belonging to the $50\% - 0.5\alpha = 30^{\text{th}}$ percentile. In our example, when we sort all $M_i = 1000$ prognostics from small to large, this is the $j = 300^{\text{th}}$ RUL prognostic $\hat{y}_{i,300}$. Also, $\hat{y}_i^{0.70}$ is the RUL prognostic belonging to the $50\% + 0.5\alpha = 0.70^{\text{th}}$ percentile. In our example, when we sort all $M_i = 1000$ prognostics from small to large, this is the $j = 700^{\text{th}}$ RUL prognostic $\hat{y}_{i,700}$. The estimated probability that the actual RUL y_i of component i is within the confidence interval $[\hat{y}_i^{0.30}, \hat{y}_i^{0.70}]$ is $\alpha = 40\%$. This example is illustrated in Figure 5.6.

We construct a confidence interval with width $\alpha = 0.4$ for all $i \in \{1, 2, \cdots, N\}$ test instances. It is expected that for $\alpha = 40\%$ of the test instances, the actual RUL y_i is within the confidence interval $[\hat{y}_i^{0.30}, \hat{y}_i^{0.70}]$. If the actual RUL of more than 40% of the test instances falls within the confidence interval $[\hat{y}_i^{0.30}, \hat{y}_i^{0.70}]$, then the *uncertainty* for $\alpha = 0.4$ is *overestimated*. Otherwise, the *uncertainty* for $\alpha = 0.4$ is *underestimated*. With this, we formulate the $\alpha-$Coverage as the fraction of the RUL prognostics (out of the N test instances), for which the true RUL lies in the α confidence interval:

$$\alpha\text{-Coverage} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{I}\left(y_i \in \left[\hat{y}_i^{0.5-0.5\alpha}, \hat{y}_i^{0.5-0.5\alpha}\right]\right) \tag{5.7}$$

$$\text{with } \mathcal{I}\left(y_i \in \left[\hat{y}_i^{0.5-0.5\alpha}, \hat{y}_i^{0.5-0.5\alpha}\right]\right) = \begin{cases} 1 & y_i \in \left[\hat{y}_i^{0.5-0.5\alpha}, \hat{y}_i^{0.5-0.5\alpha}\right] \\ 0 & \text{Otherwise} \end{cases}, \tag{5.8}$$

where $\hat{y}_i^{\beta}$ denotes the RUL prognostic belonging to the β^{th} percentile of test instance i. The closer the α-Coverage is to α, the more reliable the RUL prognostics are. For example, in Figure 5.4c, the true RUL does not fall within the 90% confidence interval of the RUL distribution. If we estimate a RUL distribution for ten individual components, we expect that for only one out of these ten components, the true RUL lies outside the 90% confidence interval, as is the case in Figure 5.4c.

Last, if two RUL prognostic methods have the same coverage for a width α, the method that provides tighter confidence intervals is preferred. In other words, a higher sharpness of the RUL distributions is preferred. In this way, the estimated RUL distributions give a more precise picture of the actual RUL. A higher sharpness also leads to a lower CRPS. The tightness of the confidence intervals, or the mean width of the confidence intervals, is defined as [10]:

$$\alpha\text{-Mean width} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{y}_i^{0.5+0.5\alpha} - \hat{y}_i^{0.5-0.5\alpha}\right). \tag{5.9}$$

5.3.4. Reliability Score (RS)

Though the α-Coverage indicates the reliability of the estimated RUL distribution, this reliability is evaluated only relative to a specific α. To conduct a generic, parameter-free reliability analysis of the estimated RUL distribution, we next introduce the Reliability Score (RS). We first introduce the reliability diagram.

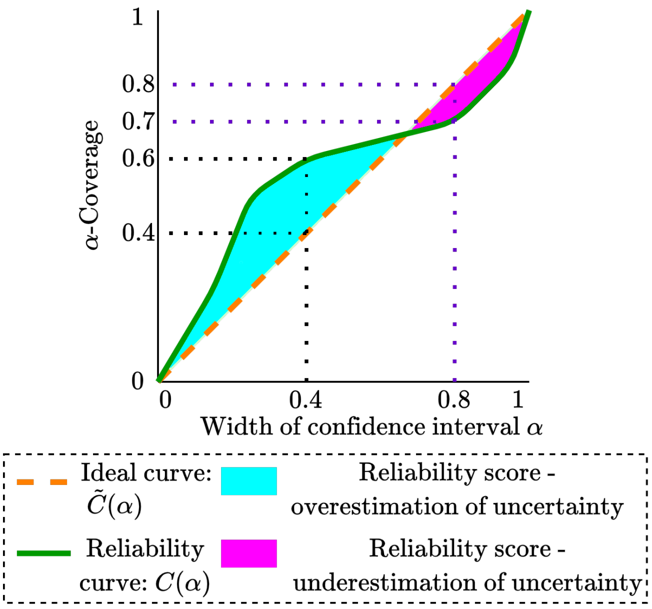


Figure 5.7.: Illustration of the reliability diagram and the Reliability Scores.

For classification problems, a reliability diagram is used as a visual representation of the reliability of the uncertainty associated with the predictions. A reliability diagram is also referred to as a calibration curve. In [12], the reliability diagram is proposed as a RUL prognostic metric. Here, the problem of RUL prognostics is posed as a classification problem with multiple classes. In contrast, in [25], the reliability diagram is defined based on the concept of coverage (see Section 5.3.3). In doing so, a regression problem does not have to be posed as a multi-class classification problem to construct a reliability diagram. The authors of [25] determine a reliability diagram for flight delay estimations. Similarly, we define a reliability curve $C(\alpha)$ based on $\alpha-$Coverage for probabilistic RUL prognostics, i.e., $C(\alpha) = \{\alpha\text{-Coverage}, \alpha \in \{0.00, 0.01, 0.02, \ldots, 1.00\}\}$. The reliability diagram is then a visual representation of this reliability curve. Figure 5.7 gives an illustration of a reliability curve.

The reliability diagram is used to visually inspect whether the uncertainty associated with the RUL prognostics is over- or underestimated. For example, when $\alpha = 0.4$, the ideal coverage would be 0.4 as well. In this case, the actual RUL of 40% of the test instances would fall inside a confidence interval with width $\alpha = 0.4$. However, in the example in Figure 5.7, the 0.4-Coverage is 0.6, i.e., the actual RUL of 60% of the test instances falls inside the confidence interval, instead of 40% of the test instances. The uncertainty of the RUL prognostics is thus overestimated. In contrast, the uncertainty of the RUL prognostics is underestimated at $\alpha = 0.8$ in

Figure 5.7. Here, the actual RUL of only 70% of the test instances falls inside the confidence interval with a width of $\alpha = 0.8$.

In general, for classification problems, the Brier Score [26] is used to quantify the reliability of predictions. However, in our adaption of the reliability diagram, each test instance may fall into multiple confidence intervals. The calculation of the Brier Score is thus not directly applicable. To address this, we define the following Reliability Scores (RS) to quantify the reliability of the RUL prognostics instead:

$$\mathrm{RS}^{\text{under}} = \int_0^1 \mathcal{I}\{C(\alpha) \leq \alpha\}\,(\alpha - C(\alpha))\,d\alpha, \tag{5.10}$$

$$\mathrm{RS}^{\text{over}} = \int_0^1 (1 - \mathcal{I}\{C(\alpha) \leq \alpha\})\,(C(\alpha) - \alpha)\,d\alpha, \tag{5.11}$$

$$\mathrm{RS}^{\text{total}} = \mathrm{RS}^{\text{under}} + \mathrm{RS}^{\text{over}}, \tag{5.12}$$

$$\text{with } \mathcal{I}\{C(\alpha) \leq \alpha\} = \begin{cases} 1, & C(\alpha) \leq \alpha \\ 0, & \text{Otherwise} \end{cases}. \tag{5.13}$$

The $\mathrm{RS}^{\text{over}}$ quantifies the overestimation and $\mathrm{RS}^{\text{under}}$ the underestimation of the *uncertainty* associated with the probabilistic RUL prognostics. Let $\tilde{C}(\alpha) = \{\alpha, \alpha \in \{0.00, 0.01, \ldots, 1.00\}\}$ be the ideal curve, i.e., the curve where the Coverage is exactly the width of the confidence interval α. To quantify the extent to which the uncertainty associated with the probabilistic RUL prognostics is *underestimated*, we calculate the area $\mathrm{RS}^{\text{under}}$ between the ideal curve and the reliability curve $C(\alpha)$ when the reliability curve is *below* the ideal curve (i.e., $C(\alpha) \leq \alpha$, pink area in Figure 5.7). To quantify the extent to which the uncertainty associated with the probabilistic RUL prognostics is *overestimated*, we calculate the area $\mathrm{RS}^{\text{over}}$ between the ideal curve and the reliability curve $C(\alpha)$ when the reliability diagram is *above* the ideal curve (i.e., $C(\alpha) \geq \alpha$, blue area in Figure 5.7). The total RS ($\mathrm{RS}^{\text{total}}$) is then the sum of $\mathrm{RS}^{\text{over}}$ and $\mathrm{RS}^{\text{under}}$.

5.4. RESULTS WITH THE NOVEL METRICS

In this section, we evaluate our metrics for the obtained probabilistic RUL prognostics for the turbofan engines in the C-MAPSS dataset. These probabilistic RUL prognostics are obtained with a CNN with Monte Carlo Dropout (see Section 5.2). Table 5.3 and 5.4 show the corresponding values of the four proposed metrics. The CRPS is lowest for data subset FD003 (6.56), and highest for data subset FD004 (10.09). This is in line with the obtained MAE, which is also lowest for data subset FD003 and highest for data subset FD004. The CRPS thus gives a good overview of the general performance of probabilistic RUL prognostics. Moreover, in contrast with the MAE, the sharpness and accuracy of the estimated RUL distributions are also reflected by the CRPS values.

For data subset FD002, the CRPS^W is lower than the CRPS. This indicates that for this dataset, the RUL is relatively often underestimated. In contrast, for data subset FD003, the CRPS^W is higher than the CRPS. This indicates that for FD003, the RUL is

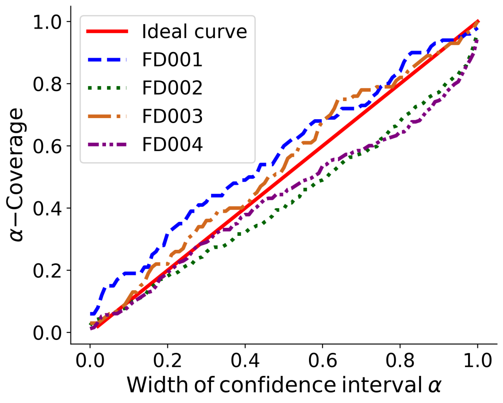


Figure 5.8.: Reliability diagrams - C-MAPSS data subsets.

Test set	MAE^p	CRPS	CRPS^W ($\beta = 1.5$)	RS$^{\text{over}}$	RS$^{\text{under}}$	RS$^{\text{total}}$
FD001	9.22	6.97	7.03	0.073	0.001	0.074
FD002	11.14	8.44	7.80	0.001	0.077	0.078
FD003	9.07	6.56	7.27	0.034	0.001	0.035
FD004	13.44	10.09	10.38	0.001	0.065	0.065

Table 5.3.: Results for the four C-MAPSS data sets - CRPS and Reliability Score.

relatively often overestimated. The weighted CRPS, compared to the standard CRPS, thus gives a good indication on whether the RUL is usually over- or underestimated.

	$\alpha = 0.5$		$\alpha = 0.95$	
Test set	Coverage	Mean width	Coverage	Mean width
FD001	0.60	16.9	0.95	48.0
FD002	0.40	13.2	0.83	38.0
FD003	0.53	15.4	0.93	44.7
FD004	0.44	15.5	0.81	43.8

Table 5.4.: Results for the four C-MAPSS data sets- α-Coverage for several values of α.

The reliability diagram of the four data subsets is shown in Figure 5.8. For data subsets FD001 and FD003, the uncertainty of the RUL prognostics is slightly overestimated. In other words, the prognostics indicate that the RUL lies in an interval with a certain probability. However, these probabilities are too small, relative to the actual number of times the RUL falls within these intervals. For example, let us consider the 0.5-Coverage of data subset FD001. Here, the estimated probability that a test instance falls inside its confidence interval with width 0.5 equals 0.5.

We thus expect that 50% of the test instances fall inside their confidence interval with width 0.5, and 50% fall outside their confidence interval. However, 60% of the test instances fall inside their confidence interval with width 0.5, i.e., the observed probability is 0.6 instead of 0.5 (see Table 5.4). This shows that the uncertainty associated with the RUL estimates is overestimated.

In contrast, for data subsets FD002 and FD004, the uncertainty is underestimated, i.e., the prognostics indicate that the RUL lies in an interval with a certain probability. These probabilities are too high relative to the actual number of times the RUL falls within these intervals. Table 5.3 shows that both the overestimation and the underestimation of the uncertainty associated with the RUL prognostics is well quantified by the Reliability Scores.

Table 5.4 shows the 0.5-Coverage and the 0.95-Coverage. Also this metric indicates that the RUL prognostics for data subsets FD001 and FD003 overestimate the uncertainty associated with the prognostics, while for data subsets FD002 and FD004 the uncertainty associated with the prognostics is underestimated. Moreover, the mean width of the 0.95 confidence interval is large, ranging from 38.0 flights (data subset FD002) to 48.0 flights (data subset FD001). This implies that the sharpness of the RUL distributions is quite low.

5.4.1. RUL prognostics for individual engines

In this section, we analyze our proposed metrics for probabilistic RUL prognostics for four specific engines 53, 4, 86 and 67 of data subset FD001 in Table 5.5 and Table 5.6. The estimated PDF of the RUL of these four engines is shown in Figure 5.4.

Engine number i	Actual RUL y_i	Mean estimated RUL $\bar{y}_i$	$y_i - \bar{y}_i$	Score^p s_i
53	26	29.0	-3.0	0.35
4	82	78.8	3.2	0.28
86	89	113.6	-24.6	10.67
67	77	114.5	-37.5	41.61

Table 5.5.: Performance metrics for engines i =53, i =4, i =86 and i =67 in the test set of FD001 - standard metrics. The metrics, except the score, are in flights.

For engine 53, the actual RUL is very close to the mean estimated RUL. The error is thus only -3.0 flights. Also CRPS$_{53}$, which is the generalization of the absolute error, is only 2.96. However, most of the mass of the estimated distribution of the RUL is on the right of the actual RUL, i.e., the RUL is overestimated (see Figure 5.4a). This is reflected in the relatively high CRPS$_{53}^{W}$ of 3.72. The actual RUL falls both within the $\alpha = 0.5$ and $\alpha = 0.95$ confidence interval, and the widths of these intervals (15 and 45 flights respectively) are relatively small compared to engines 4, 86 and 67.

For engine 4, the mean estimated RUL is close to the actual RUL, with an error of 3.2 flights. Thus CRPS$_4$ is only 3.49. Also, CRPS$_4^W = 2.68$, which is less than CRPS$_4$. This is because most of the mass of the estimated PDF of the RUL is on the left of the actual RUL, i.e., the RUL is underestimated (see Figure 5.4b). The $\alpha = 0.5$ and

Engine number i	CRPS$_i$	CRPS$_i^W$ $\beta = 1.5$	$\mathcal{I}(\alpha)_i$ $\alpha = 0.5$	$\hat{y}_i^{0.75} - \hat{y}_i^{0.25}$	$\mathcal{I}(\alpha)_i$ $\alpha = 0.95$	$\hat{y}_i^{0.975} - \hat{y}_i^{0.025}$
53	2.96	3.72	1	15	1	45
4	3.49	2.68	1	19	1	54
86	17.98	26.96	0	16	1	48
67	30.74	46.11	0	16	0	46

Table 5.6.: Performance metrics for engines i =53, i =4, i =86 and i =67 in the test set of FD001 - new metrics. Here, $\mathcal{I}(\alpha)_i$ equals 1 if the actual RUL y_i is within the α confidence interval of the estimated RUL distribution, and 0 otherwise.

$\alpha = 0.95$ confidence interval both contain the actual RUL, but the width of these intervals (19 and 54 flights respectively) is relatively large compared to the other 3 engines. The low sharpness of this RUL distribution is thus reflected in the large widths of the confidence intervals.

For engines 86 and 67, the mean estimated RUL is far off the actual RUL. This is reflected in the high CRPS values of 17.98 and 30.74, respectively. Moreover, nearly all the mass of the estimated PDF of the RUL of both engines is on the right of the actual RUL, i.e., the RUL is overestimated (see Figures 5.4c and 5.4d). The weighted CRPS metric is thus 26.96 and 46.11, respectively. This is higher than the standard CRPS metric for these two engines. The actual RUL of engine 86 falls within the $\alpha = 0.95$ confidence interval, but the actual RUL of engine 67 does not.

5.5. CONCLUSIONS

In this chapter, we have introduced novel metrics to evaluate the estimated PDF of the RUL of components. The CRPS and CRPS^W metrics evaluate the accuracy and sharpness of the estimated RUL distributions. The α-Coverage and Reliability Scores evaluate the reliability of the RUL prognostics.

We illustrate the four metrics for probabilistic RUL prognostics of the turbofan engines in the C-MAPSS dataset. We obtain these probabilistic RUL prognostics using a CNN with Monte Carlo dropout. The results show the distribution of the RUL of the turbofan engines is well estimated using this method. Moreover, the accuracy, sharpness and reliability of the obtained probabilistic RUL prognostics are shown to be well evaluated by our proposed metrics. Future studies that determine probabilistic RUL prognostics could therefore benefit from evaluating their results using these proposed metrics.

REFERENCES

[1] de Pater, I., & Mitici, M. (2022, July 6-8). Novel metrics to evaluate probabilistic Remaining Useful Life prognostics with applications to turbofan engines. *Proceedings of the 7th European Conference of the Prognostics and Health Management (PHM) Society, 7*, Turin, Italy, Pages: 96–109.

[2] Lee, J., & Mitici, M. (2020). An integrated assessment of safety and efficiency of aircraft maintenance strategies using agent-based modelling and stochastic Petri Nets. *Reliability Engineering & System Safety, 202*, Article number: 107052.

[3] de Pater, I., Reijns, A., & Mitici, M. (2022). Alarm-based predictive maintenance scheduling for aircraft engines with imperfect Remaining Useful Life prognostics. *Reliability Engineering & System Safety, 221*, Article number: 108341.

[4] Li, X., Ding, Q., & Sun, J.-Q. (2018). Remaining Useful Life estimation in prognostics using deep Convolution Neural Networks. *Reliability Engineering & System Safety, 172*, Pages: 1–11.

[5] Xia, J., Feng, Y., Lu, C., Fei, C., & Xue, X. (2021). LSTM-based multi-layer self-attention method for Remaining Useful Life estimation of mechanical systems. *Engineering Failure Analysis, 125*, Article number: 105385.

[6] Lee, J., & Mitici, M. (2022). Multi-objective design of aircraft maintenance using Gaussian process learning and adaptive sampling. *Reliability Engineering & System Safety, 218*, Article number: 108123.

[7] Nguyen, K. T., & Medjaher, K. (2019). A new dynamic predictive maintenance framework using deep learning for failure prognostics. *Reliability Engineering & System Safety, 188*, Pages: 251–262.

[8] Biggio, L., Wieland, A., Chao, M. A., Kastanis, I., & Fink, O. (2021). Uncertainty-aware prognosis via deep Gaussian process. *IEEE Access, 9*, Pages: 123517–123527.

[9] de Pater, I., & Mitici, M. (2021). Predictive maintenance for multi-component systems of repairables with Remaining-Useful-Life prognostics and a limited stock of spare components. *Reliability Engineering & System Safety, 214*, Article number: 107761.

[10] Baraldi, P., Mangili, F., & Zio, E. (2015). A prognostics approach to nuclear component degradation modeling based on Gaussian process regression. *Progress in Nuclear Energy, 78*, Pages: 141–154.

[11] Le Son, K., Fouladirad, M., & Barros, A. (2016). Remaining Useful Lifetime estimation and noisy Gamma deterioration process. *Reliability Engineering & System Safety, 149*, Pages: 76–87.

[12] Saxena, A., Celaya, J., Balaban, E., Goebel, K., Saha, B., Saha, S., & Schwabacher, M. (2008, October 6-9). Metrics for evaluating performance of prognostic

techniques. *International Conference on Prognostics and Health Management*, Denver, Colorado, USA, Pages: 1–17.

[13] Saxena, A., Celaya, J., Saha, B., Saha, S., & Goebel, K. (2009, March 7-14). Evaluating algorithm performance metrics tailored for prognostics. *IEEE Aerospace conference*, Big Sky, Montana, USA, Pages: 1–13.

[14] Lall, P., Lowe, R., & Goebel, K. (2011, April 18-20). Prognostics and health monitoring of electronic systems. *12th International Conference on Thermal, Mechanical & Multi-Physics Simulation and Experiments in Microelectronics and Microsystems*, Linz, Austria, Pages: 1–17.

[15] Saxena, A., & Goebel, K. (2008). *Turbofan engine degradation simulation data set*, NASA Prognostics Data Repository, NASA Ames Research Center, Moffett Field, California, USA.

[16] Ramasso, E., & Saxena, A. (2014, September 29 - October 2). Review and analysis of algorithmic approaches developed for prognostics on CMAPSS dataset. *Proceedings of the Annual Conference of the Prognostics and Health Management (PHM) Society 2014*, *6*, Fort Worth, Texas, USA, Pages: 1–11.

[17] Babu, G. S., Zhao, P., & Li, X.-L. (2016, April 16-19). Deep Convolutional Neural Network based regression approach for estimation of Remaining Useful Life. *Proceedings of the 21st International Conference on Database Systems for Advanced Applications (DASFAA)*, Dallas, Texas, USA, Pages: 214–228.

[18] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[19] Gal, Y., & Ghahramani, Z. (2016, June 19-24). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *Proceedings of The 33rd International Conference on Machine Learning*, *48*, New York, New York, USA, Pages: 1050–1059.

[20] Gal, Y., Hron, J., & Kendall, A. (2017). Concrete dropout. *arXiv preprint arXiv:1705.07832*.

[21] Zoutendijk, M., & Mitici, M. (2021). Probabilistic flight delay predictions using machine learning and applications to the flight-to-gate assignment problem. *Aerospace*, *8*(6), Article number: 152.

[22] Gneiting, T., Raftery, A. E., Westveld III, A. H., & Goldman, T. (2005). Calibrated probabilistic forecasting using ensemble model output statistics and minimum CRPS estimation. *Monthly Weather Review*, *133*(5), Pages: 1098–1118.

[23] Nowotarski, J., & Weron, R. (2018). Recent advances in electricity price forecasting: A review of probabilistic forecasting. *Renewable and Sustainable Energy Reviews*, *81*, Pages: 1548–1568.

[24] Gneiting, T., & Katzfuss, M. (2014). Probabilistic forecasting. *Annual Review of Statistics and Its Application*, *1*, Pages: 125–151.

[25] Vandal, T., Livingston, M., Piho, C., & Zimmerman, S. (2018, October 24-25). Prediction and uncertainty quantification of daily airport flight delays. *Proceedings of the 4th International Conference on Predictive Applications and APIs*, *82*, Boston, Massachusetts, USA, Pages: 45–51.

[26] Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, *78*(1), Pages: 1–3.

6 ALARM-BASED MAINTENANCE SCHEDULING WITH IMPERFECT POINT RUL PROGNOSTICS

In the first two chapters, we develop point Remaining Useful Life (RUL) prognostics, without quantified uncertainty, for aircraft components. However, these point RUL prognostics are imperfect with some errors. These errors make it difficult to integrate point RUL prognostics in the maintenance planning.

In this chapter, we therefore propose a dynamic predictive maintenance scheduling framework for a fleet of aircraft, that integrates the imperfect point RUL prognostics. Based on the evolution of the point RUL prognostics over time, alarms are triggered. After an alarm is triggered, we schedule the maintenance tasks. Here, we use a safety factor when scheduling the maintenance tasks, to account for the errors in the RUL prognostics. The alarms prevent that the maintenance tasks are rescheduled multiple times. We optimize the maintenance schedule with a linear program, while the hyperparameters of this method are optimized with a genetic algorithm.

We illustrate our approach for a fleet of 20 aircraft, each equipped with 2 turbofan aircraft engines. We obtain point RUL prognostics with a Convolution Neural Network.

Parts of this chapter have been published in:

de Pater, I., Reijns, A., & Mitici, M. (2022). Alarm-based predictive maintenance scheduling for aircraft engines with imperfect Remaining Useful Life prognostics. *Reliability Engineering & System Safety, 221*, Article number: 108341

6.1. Introduction

The cost of aircraft maintenance is estimated to be 10.3% of the total airline operating costs, with approximately 3.3 million dollars spent on maintenance per aircraft in 2019 [2]. Striving to reduce these costs, aircraft maintenance is shifting to data-driven, predictive maintenance where on-board sensors are increasingly used to monitor the health condition of the aircraft components. Based on these sensor measurements, dedicated algorithms are developed to estimate the Remaining Useful Life (RUL) of the components. Using RUL prognostics, the aim is to anticipate failures and optimize the deployment of maintenance tasks. One of the main challenges in predictive maintenance is to obtain reliable RUL prognostics and to integrate them in the maintenance planning [3].

Most existing studies focus solely on developing RUL prognostics, using either a model-based or a machine learning approach [4]. Model-based RUL prognostics assume that the degradation of components is characterized by a stochastic process. For instance, in [5, 6], the RUL of aircraft cooling units is estimated using particle filtering with an exponential degradation model and a linear model, respectively. In [7], RUL prognostics for aircraft landing gear brakes are obtained using a linear regression, while a Gamma process characterizes the degradation of the brakes. Machine learning algorithms have been proposed to estimate the RUL of, for instance, aircraft turbofan engines [8–10] and bearings [11, 12]. In [8–10], a Convolutional Neural Network (CNN) is used to estimate the RUL of turbofan engines. To estimate the RUL of rolling element bearings, a CNN with a residual is proposed in [11], while a CNN with multi-scale feature extraction is proposed in [12]. We refer to [4, 13] for an extensive overview of recent studies about RUL prognostics.

Several studies focus on predictive maintenance planning, while the RUL prognostics are based on simple, generic probability distributions. For instance, in [14], the degradation of a railway network is simulated with a mixture of parametric models, and maintenance is planned using a Markov Decision Process. In [15], the degradation of aircraft components is modelled with a stationary Gamma process, while a Large Neighbourhood Search algorithm is proposed for the maintenance planning of a fleet of aircraft.

Few studies develop RUL prognostics and subsequently integrate these prognostics in the maintenance planning [3, 16]. Most of these studies focus on maintenance planning for one (multi-component) system. In [16], multi-class RUL prognostics for aircraft turbofan engines are generated using a Long Short-Term Memory neural network. Based on these prognostics, engine replacements are planned and spare parts are ordered. In [17], prognostics for aircraft airframe cracks are developed using an extended Kalman filter. These prognostics are further used to determine which panels of a single aircraft are maintained, if any. In contrast to these studies, we consider the maintenance of a *fleet* of aircraft.

Even fewer studies develop RUL prognostics and subsequently integrate these prognostics in the maintenance planning for multiple assets/systems. In [18], a particle filtering algorithm is used to determine RUL prognostics for aircraft cooling units. With these prognostics, maintenance for a fleet of aircraft is planned using linear programming, taking into account the availability of spare parts. In [19], the maintenance of multiple aircraft brakes is considered. An aircraft brake is replaced as soon as the estimated RUL falls below a threshold. Multiple objectives, such as minimizing flight delays, minimizing the number of unscheduled maintenance tasks and minimizing the total number of maintenance tasks, are considered. In [20], maintenance is planned for a fleet of vehicles using a multi-objective genetic algorithm. The aim is to minimize the total maintenance costs, the total workload, the expected number of failures and the changes in the maintenance schedule.

In general, these studies show that integrating RUL prognostics into maintenance planning models leads to lower maintenance costs and a more efficient use of spare parts [16, 18]. However, when planning maintenance, the errors (e.g., RMSE, MAE, false negatives, false positives) of the RUL prognostics are not considered. To account for such potential errors when planning maintenance, we propose a system of alarms to initiate maintenance task scheduling, together with a safety margin that adjusts the moment when maintenance tasks are scheduled.

This chapter proposes a dynamic, predictive maintenance planning framework for a fleet of aircraft that integrates RUL prognostics for aircraft components. These prognostics are periodically updated as more measurements become available. Alarms are triggered based on the evolution of the prognostics over time. Triggering an alarm for a component initiates the scheduling of a maintenance task for this component. The ideal time to schedule such a task is determined based on the RUL prognostics and a safety margin, to account for potential errors in the RUL prognostics. Maintenance tasks for a fleet of aircraft are scheduled with an integer linear program, based on these ideal maintenance times. Once a maintenance task for a component is scheduled, we continue to periodically update the RUL prognostics of the component. Based on the evolution of the prognostics for this component over time, maintenance tasks may be rescheduled at a high cost.

The time when alarms are triggered is crucial. Triggering alarms when the estimated RUL is large may result in the initiated maintenance task being re-scheduled several times, as RUL prognostics are updated over time. Triggering alarms when the estimated RUL is small may result in component failures as there may not be enough time and resources left to perform maintenance. Using a genetic algorithm, we optimize the parameters of our alarm policy (the frequency of alarms, the threshold for triggering alarms and the safety margin).

We illustrate our approach for the maintenance planning of a fleet of aircraft, each equipped with two engines. By employing our alarm-based maintenance framework, the costs with engine failures account for only 7.4% of the total maintenance costs. Overall, our framework provides an end-to-end approach for maintenance scheduling of multiple components with imperfect RUL prognostics.

The remainder of this chapter is organized as follows. In Section 6.2, we use a CNN to obtain RUL prognostics for turbofan engines. In Section 6.3, we develop an alarm-based maintenance planning framework that integrates RUL prognostics in the maintenance schedule. In Section 6.4, we illustrate our approach for a fleet of aircraft equipped with turbofan engines. Conclusions are provided in Section 6.5.

6.2. RUL prognostics using a Convolutional Neural Network (CNN)

In this section, we develop RUL prognostics for turbofan engines using Convolutional Neural Networks (CNNs) and the C-MAPSS turbofan aircraft engine degradation dataset [21]. CNNs have successfully been applied to estimate the RUL for turbofan aircraft engines in, for instance, [8–10].

The C-MAPSS dataset consists of simulated data on the degradation of turbofan engines. This data was generated by NASA using the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS). The dataset contains multi-variate temporal data of 21 sensors. There are 4 data subsets, FD001, FD002, FD003 and FD004, each with specific operating and fault conditions. Each subset has one training set where measurements are recorded until the failure of the engine (run-to-failure instances), and one test set. In the test set, the sensor recordings are terminated somewhere before failure, and the aim is to estimate the RUL at that moment. In all cases, each engine has a different level of initial wear. Over time, the condition of an engine degrades as it approaches failure. A description of the 4 data subsets is given in Table 6.1.

	FD001	FD002	FD003	FD004
Training instances	100	260	100	249
Testing instances	100	259	100	248
Operating conditions	1	6	1	6
Fault conditions	1	1	2	2

Table 6.1.: C-MAPSS datasets for turbofan engines [21].

Of the 21 sensors considered, 7 sensors have constant values over time. As such, we select the remaining 14 sensors with non-constant measurements for the input of the CNNs. We normalize the sensor measurements with min-max normalization [8] with respect to the operating condition [9] in each subset as follows:

$$\hat{m}_{ij} = \frac{2\left(m_{ij}^{k} - m_{jk}^{\min}\right)}{m_{jk}^{\max} - m_{jk}^{\min}} - 1, \tag{6.1}$$

with m_{ij}^{k} the sensor measurement of sensor j during flight i, where flight i was performed under operating condition k, while $m_{jk}^{\min}$ and $m_{jk}^{\max}$ denote the minimum and maximum value in the training set of sensor j under operating condition k respectively. Last, $\hat{m}_{ij}$ is the normalized measurement of sensor j during flight i.

6.2.1. Architecture of the CNN

As input for the CNN, we consider multi-dimensional data samples X:

$$X = [x_1, x_2, \ldots, x_N], \tag{6.2}$$

where N is the number of flights included. For each flight $i \in \{1, 2, \ldots, N\}$, x_i contains the sensor measurements obtained during the flight and history of the operating conditions at that flight:

$$x_i = [\hat{m}_{i1}, \hat{m}_{i2}, \ldots, \hat{m}_{iM}, o_{i1}, o_{i2}, \ldots, o_{iO}]. \tag{6.3}$$

Here, $\hat{m}_{ij}$ denotes the normalized measurement of the j^{th} sensor during flight i, M denotes the total number of sensors considered, o_{ir} denotes the history of operating condition r at flight i, and O denotes the number of operating conditions of the considered subset. The history of operating condition r denotes the number of flights, since the first flight of the considered engine up to flight i, an engine has performed in operating condition r [9].

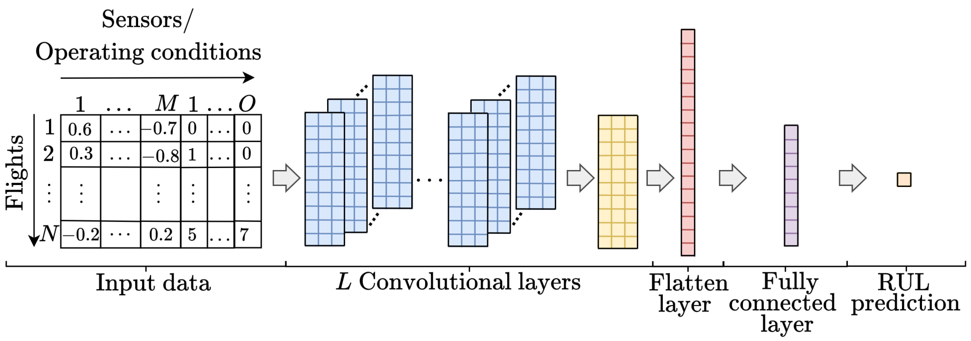


Figure 6.1.: Schematic overview of the CNN.

Figure 6.1 shows the architecture of the proposed CNN. We consider L convolutional layers. The first $L-1$ convolutional layers each have K_f kernels, and thus generate K_f feature maps. Each kernel has a size of $K_s \times 1$. We thus use one-dimensional kernels [8]. The convolutional operation in the l^{th} convolutional layer for the n^{th} kernel k_n^l is [22]:

$$z_n^l = \sigma\left(k_n^l * z^{l-1} + b_n^l\right), \tag{6.4}$$

where z_n^l is the n^{th} feature map of layer l, $*$ is the convolutional operator, z^{l-1} are the feature maps in layer $l-1$, b_n^l is the bias of the n^{th} feature map of layer l, and $\sigma(\cdot)$ is the activation function of the convolutional layer. The L^{th} convolutional layer has one kernel with a size of $K_s^{'} \times 1$. This layer combines all the K_f feature maps of the previous layer into one single feature map.

Using the extracted features of the convolutional layers, the CNN estimates the RUL. The 2-dimensional, final feature map of the last convolutional layer is flattened. In this layer, we apply a drop-out rate ρ to prevent overfitting. The flatten layer is connected with a fully connected layer. Let z^{fl} be the output of the flatten layer, and let w^f be the weights of the fully connected layer. The output z^f of the fully connected layer is then [22]:

$$z^f = \sigma\left(w^f z^{\text{fl}} + b^f\right), \tag{6.5}$$

where b^f denotes the bias of the fully connected layer, and $\sigma(\cdot)$ denotes the activation function of this layer. Last, the final layer with one neuron outputs a RUL prognostic using a linear activation function.

Following a grid-search hyper-parameter tuning, with the hyperparameters of [8] as starting point, we consider $L = 5$ convolutional layers. The first 4 convolutional layers contain $K_f = 10$ kernels, each with a size of $K_s = 10 \times 1$, while the last convolutional layer contains 1 kernel with a size of $K_s^{'} = 3 \times 1$. Same padding is implemented in all convolutional layers to ensure that the feature maps have a constant dimension. In the flatten layer, we apply a drop-out rate of $\rho = 0.5$ during training. Finally, the fully connected layer contains 100 neurons. All layers, except the last layer, apply a hyperbolic tangent (tanh) activation function.

We optimize the weights of the CNN using the Adam optimizer [23] with a batch size of 256 samples, and a maximum of 250 training epochs. The initial learning rate is 0.001, which is multiplied by 0.6 after 10 consecutive epochs without improvement, for a stable convergence of the weights. We use a window size of 30 flights for subsets FD001 and FD003, and of 21 and 19 flights for subsets FD002 and FD004 respectively, i.e., the number of flights available for the shortest test instance in the test sets of FD002 and FD004.

RUL prognostics for aircraft engines

	R_{early}	FD001	FD002	FD003	FD004
Our approach	125	12.22	15.07	12.72	18.10
CNN [8]	125	12.61	22.36	12.64	23.31
MS-DCNN [10]	125	11.44	19.35	11.67	22.22
CNN with pooling [9]	varies	18.45	30.29	19.82	29.16
CNN with pyramid pooling [24]	-	12.64	25.92	12.39	26.84

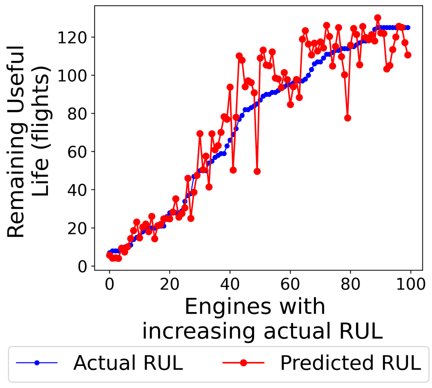
Table 6.2.: RMSE for the RUL prognostics using C-MAPSS and a (variant of) a CNN.

We apply our CNN to each of the 4 test data subsets FD001, FD002, FD003 and FD004. We evaluate the obtained RUL prognostics by means of the Root Mean Square Error (RMSE) metric:

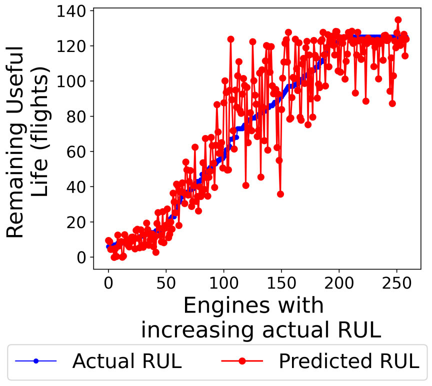
$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{w=1}^{n}(e_w)^2}, \tag{6.6}$$

where n is the number of testing instances in the considered data subset and e_w is the RUL prognostic error (in flights) for an engine w, $e_w = \text{RUL}_w^{\text{actual}} - \text{RUL}_w^{\text{estimated}}$. Moreover, we use a piece-wise linear RUL target function [8, 10, 25] with $R_{\text{early}} = 125$ flights, i.e, when the actual RUL is larger than $R_{\text{early}} = 125$ flights, the target RUL of the neural network is $R_{\text{early}} = 125$ flights.

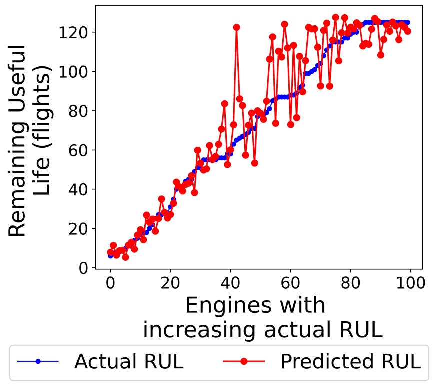
Table 6.2 shows the RMSE for each of the 4 test data subsets of C-MAPSS. The RMSE is highest for subsets FD002 and FD004, where the engines are subject to multiple operating conditions. Figure 6.2 shows the RUL prognostic versus the actual



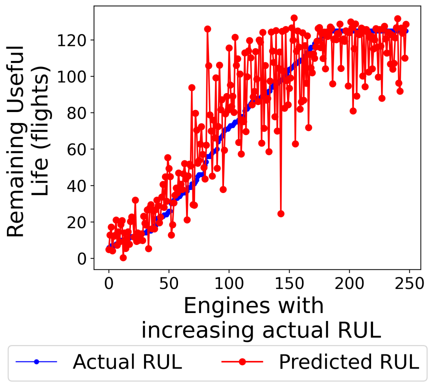
(a) FD001.



(b) FD002.



(c) FD003.



(d) FD004.

Figure 6.2.: RUL prognostics of the engines in the four C-MAPSS data subsets. The engines are sorted in an increasing order of their actual RUL.

RUL for the individual engines in the 4 test data subsets. As expected, the results show that the RUL prognostics are indeed imperfect with non-zero errors.

Table 6.2 also compares the performance of our RUL prognostic model with existing studies that employ CNNs for RUL prognostics as well. The results show that we obtain a lower RMSE for subsets FD002 and FD004 compared to [8–10, 24]. A more advanced CNN in [10] results in a lower RMSE for subsets FD001 and FD003. Overall, our results are comparable to the best existing results for this dataset when using a CNN. Table 6.3 gives an overview of the performance of RUL prognostic models for the C-MAPSS dataset when considering various machine learning algorithms. A more extensive overview of such models can be found in [25, 26]. The lowest RMSE is obtained when using a LSTM neural network with multi-layer self-attention [25], or a hierarchical attention graph convolutional network [27]. Also here, the performance of our RUL prognostic method is comparable with existing methods.

	R_{early}	FD001	FD002	FD003	FD004
Our approach	125	12.22	15.07	12.72	18.10
LSTM-MLSA [25]	125	11.57	14.02	12.13	17.21
CNN-LSTM [28]	125	11.17	-	9.99	-
HAGCN [27]	130	11.93	15.05	11.53	15.74
HDNN [29]	125	13.02	15.24	12.22	18.17
Semi-supervised [30]	130	12.56	22.73	12.10	22.66
CapsNets [31]	125	12.58	16.30	11.71	18.96

Table 6.3.: RMSE for RUL prognostics using C-MAPSS and various machine learning algorithms.

In the next section, we integrate these imperfect RUL prognostics into a predictive maintenance scheduling model for a fleet of aircraft.

6.3. Methodology - Maintenance scheduling with imperfect RUL prognostics

In this section, we propose a generic, alarm-based maintenance planning framework for a fleet of aircraft with imperfect RUL prognostics.

6.3.1. Problem description

Fleet of aircraft with degrading components

Let A denote a fleet of aircraft. Each aircraft $a \in A$ is equipped with a set V_a of identical components. The health of each component degrades over time. A RUL prognostic for each component $v \in V_a$ is obtained every day.

Maintenance slots

Each aircraft $a \in A$ has allocated a set of maintenance slots S_a during which the aircraft is on the ground and maintenance is performed. These slots are scheduled months in advance. During these slots, periodic maintenance tasks and inspections are scheduled in advance as well, as described in the aircraft maintenance manuals [32]. Let d_s denote the day during which a maintenance slot $s \in S_a$ is planned.

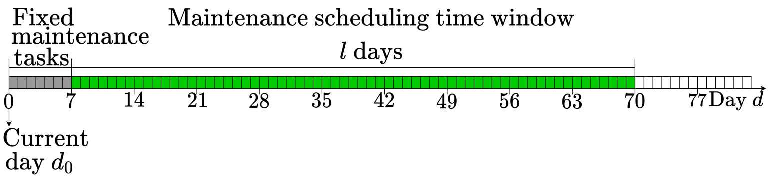
Additional maintenance tasks driven by RUL prognostics

During a maintenance slot, *additional* maintenance tasks may be scheduled based on RUL prognostics, in anticipation of a failure. Performing an additional maintenance task for a component $v \in V_a$ costs c_p. We assume that at most h additional maintenance tasks can be performed every day, due to the limited availability of the maintenance engineers, specialized tools and equipment. This capacity h is uniform across tasks, i.e., independent of the type of maintenance task.

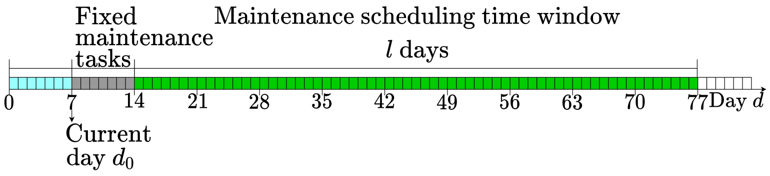
When a component $v \in V_a$ fails, we assume that an *unscheduled* maintenance task for this component is immediately performed at a cost $c_f > c_p$, even if no capacity is available. In this case, the spare maintenance slots reserved by airlines for urgent maintenance tasks are used.

Rolling horizon approach and rescheduling additional maintenance tasks

We assume a discrete-time problem with time steps of 1 day. Additional maintenance tasks are planned using a rolling horizon approach. At current day d_0, we consider a scheduling time window $D_{d_0} = [d_0 + k, d_0 + k + l)$, where k is the minimum number of days required to prepare an additional maintenance task and $k + l$ is the moment in time up to which maintenance slots are known. At day d_0, additional maintenance tasks may be scheduled within this time window D_{d_0} based on the available RUL prognostics. We then advance to the next moment of maintenance scheduling at day $d_0 + \tau$ (new current day) with time window $D_{d_0+\tau} = [d_0 + \tau + k, d_0 + \tau + k + l)$ for which new, updated RUL prognostics are generated. Again, additional maintenance tasks may be scheduled in time window $D_{d_0+\tau}$, based on the updated RUL prognostics. This process is repeated for future time windows.



(a) $d_0 = 0$.



(b) $d_0 = 7$.

Figure 6.3.: Illustration of two sequential time windows of the rolling horizon approach with $\tau = 7$, $k = 7$ and $l = 63$ days.

Figure 6.3 shows an example of a sequence of two maintenance scheduling time windows with $\tau = 7$, $k = 7$ and $l = 63$ days. At current day $d_0 = 0$, we schedule additional maintenance tasks for the time window $[d_0 + k, d_0 + k + l) = [7, 70)$ (Figure 6.3a). We fix all additional tasks scheduled up to day $d_0 + \tau + k = 14$, and advance in time to a new current day $d_0 + \tau = 7$. For this new current day, we now consider the scheduling time window $[14, 77)$ (Figure 6.3b).

A *rescheduling* of an additional maintenance task occurs when this task has been assigned at day d_0 to a slot s in time window D_{d_0}, and the same task is re-assigned at day $d_0+\tau$ to another slot $s' \neq s$ in a subsequent time window $D_{d_0+\tau}$. Rescheduling additional tasks may occur due to updated RUL prognostics or due to exceeding the limit h of additional tasks per day. Since rescheduling a maintenance task often is not desired, it costs $c_r < c_p$.

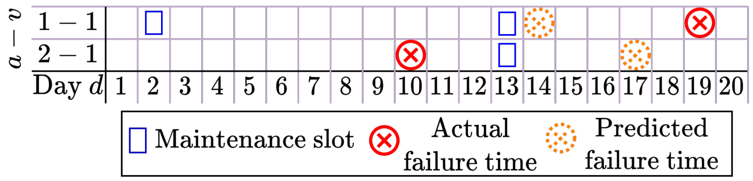


Figure 6.4.: Example - Maintaining 2 aircraft in one time window of the rolling horizon approach, with RUL prognostics.

Figure 6.4 shows an example of maintenance for two aircraft in one time window of the rolling horizon approach, when considering RUL prognostics. We consider the maintenance planning for aircraft 1, component 1 $(1-1)$, and aircraft 2, component 1 $(2-1)$. Our time window consists of 20 days. For aircraft 1, there are two maintenance slots in which we can plan an additional maintenance task for component 1: at day 2, and at day 13. Aircraft 2 has only one maintenance slot, at day 13. Component 1 of aircraft 1 is expected to fail at day 14. However, the actual failure time of this component is day 19. The RUL is thus underestimated. Component 1 of aircraft 2 is expected to fail at day 17, while it actually fails at day 10. The RUL is thus overestimated.

6.3.2. Alarm-based maintenance scheduling

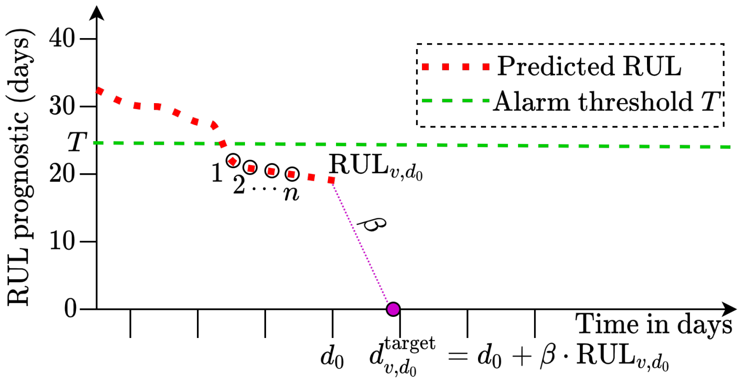


Figure 6.5.: Example - RUL-based alarm for a component v.

We propose to schedule maintenance for a fleet of aircraft using RUL-based alarms. A schematic overview of this framework is given in Figure 6.5. Every day, the RUL prognostic for a component v is updated. In case the estimated RUL falls below an alarm threshold T for n consecutive days (i.e., n days in a row), an alarm is triggered. We refer to this component as an *alarmed* component. At the next maintenance scheduling moment in the rolling horizon approach, an additional maintenance task is scheduled for this component.

We require the RUL prognostic to fall below the alarm threshold T for n *consecutive* days due to the possible non-monotonic trend of the prognostics. As a result, the times at which the RUL prognostic falls below an alarm threshold can be far apart (false positives). Acting on these false positives would lead to unnecessary maintenance and costs. We define n to identify a consistent behavior of degradation in a short period of time instead.

Consider current day d_0 when a RUL prognostic RUL_{v,d_0} for an alarmed component v is available. Ideally, we would schedule maintenance at day $d_0 + \text{RUL}_{v,d_0}$ to minimize the wasted life of the component, while avoiding a failure. However, since we consider imperfect RUL prognostics, we assume a safety factor β, $0 \le \beta \le 1$, such that we aim to schedule maintenance before a *target* day instead:

$$d^{\text{target}}_{v,d_0} = d_0 + \beta \cdot \text{RUL}_{v,d_0}.$$

These parameters T, n and β are optimized in Section 6.3.3.

SCHEDULING MAINTENANCE FOR ALARMED COMPONENTS: ONE TIME WINDOW

Given an alarm threshold T, an n number of times the RUL prognostic falls below T before an alarm is triggered, and a safety factor β, we propose the following maintenance scheduling model for a fleet of aircraft. The objective of the model is to minimize the total maintenance costs. This model is applied for each time window of the rolling horizon approach.

Let V^{alarm} denote the set of alarmed components in the fleet of aircraft at current day d_0, i.e., the set of components for which we should schedule maintenance. Let $S_{a,D_{d_0}} \subseteq S_a$ denote the set of all available maintenance slots for aircraft $a \in A$ such that the slot is within the scheduling time window D_{d_0}, i.e., $d_s \in D_{d_0}$, $\forall s \in S_{a,D_{d_0}}$.

Due to the limited capacity h and the limited number of maintenance slots, it may be that not all alarmed components $v \in V^{\text{alarm}}$ can be maintained in the scheduling time window D_{d_0}. In this case, we assume that a buffer, generic slot is available to maintain the aircraft [18]. Let s^{gen} be a generic slot at day $d_{s^{\text{gen}}} = d_0$. When using this generic slot, a very high cost $c_g >> c_f$ is incurred. This slot does not have capacity constraints. All aircraft may be maintained in this generic slot. The set of slots $S_{a,D_{d_0}}$ available for each aircraft $a \in A$ thus includes this generic slot.

We consider the following integer linear program to schedule additional maintenance tasks at day d_0 in the scheduling time window D_{d_0}:

Decision variable The decision variable is defined as follows:

$$x_{avs} = \begin{cases} 1, & \text{Component } v \in V_a \cap V^{\text{alarm}} \text{ of aircraft } a \in A \text{ is maintained} \\ & \text{in slot } s \in S_{a,D_{d_0}}, \\ 0, & \text{Otherwise,} \end{cases} \tag{6.7}$$

Objective function Let c_{avs} denote the costs of scheduling an additional maintenance task for component $v \in V_a \cap V^{\text{alarm}}$ of aircraft $a \in A$ during slot $s \in S_{a,D_{d_0}}$. We define c_{avs} as:

$$c_{avs} = p^{\text{late}}\left(d_s - d_{v,d_0}^{\text{target}}\right)^+ + p^{\text{early}}\left(d_{v,d_0}^{\text{target}} - d_s\right)^+ + p^{\text{res}} I_v^{\text{res}} + p^{\text{gen}} I_s^{\text{gen}},$$

where p^{late} is a penalty for each day an additional maintenance task is scheduled after the target day $d_{v,d_0}^{\text{target}}$, p^{early} is a penalty for each day an additional maintenance task is scheduled before the target day $d_{v,d_0}^{\text{target}}$ (wasting component life), p^{res} is a penalty for rescheduling an additional maintenance task for a component v to a new slot and $p^{\text{gen}} >> p^{\text{late}}, p^{\text{early}}, p^{\text{res}}$ is the penalty for using the generic slot s^{gen}. The $(\cdot)^+$ operator means that we consider the positive number, i.e., $(x)^+ = \min(0, x)$. Last, I_v^{res} and I_s^{gen} are indicator functions:

$$I_v^{\text{res}} = \begin{cases} 1, & \text{An additional maintenance task for component } v \text{ is rescheduled,} \\ 0, & \text{Otherwise,} \end{cases}$$

$$I_s^{\text{gen}} = \begin{cases} 1, & \text{Slot } s \in S_{a,Dd_0} \text{ is the generic slot } s^{\text{gen}} \\ 0, & \text{Otherwise.} \end{cases}$$

The objective is to minimize the costs of scheduling additional maintenance tasks:

$$\min. \sum_{a \in A} \sum_{v \in V_a : v \in V^{\text{alarm}}} \sum_{s \in S_{a,D_{d_0}}} c_{avs} x_{avs}. \tag{6.8}$$

Constraints We consider the following constraints:

$$\sum_{s \in S_{a,D_{d_0}}} x_{avs} = 1, \qquad \forall a \in A, \forall v \in V_a : v \in V^{\text{alarm}} \tag{6.9}$$

$$\sum_{a \in A} \sum_{v \in V_a \cap V^{\text{alarm}}} \sum_{s \in S_{a,D_{d_0}} \setminus \{s^{\text{gen}}\} : d_s = d} x_{avs} \leq h, \qquad \forall d \in D_{d_0} \tag{6.10}$$

$$x_{avs} \in \{0,1\} \qquad \forall a \in A, \forall v \in V_a \cup V^{\text{alarm}}, \forall s \in S_{a,D_{d_0}} \cup \{s^{\text{gen}}\} \tag{6.11}$$

Constraint (6.9) ensures that an additional maintenance task is scheduled for each alarmed component $v \in V^{\text{alarm}}$. Constraint (6.10) ensures that at most h additional tasks are scheduled every day, aside from the generic slot.

This model assumes known values for T, n and β. We next determine these values.

6.3.3. Optimizing the alarm policy with a genetic algorithm

We now optimize the alarm threshold T, the number of times n that the RUL prognostic falls below T before an alarm is triggered, and the safety factor β of our maintenance planning framework using a genetic algorithm (GA) [33]. The aim is to minimize the long-term maintenance costs.

Agent chromosomes and initialization Let Θ_i be the population of $|\Theta_i| = N$ agents in iteration i of the GA. Let θ_p be the chromosome with the parameters of agent $\theta \in \Theta_i$:

$$\theta_p = \left(T^\theta, n^\theta, \beta^\theta\right), \tag{6.12}$$

with T^θ the alarm threshold, n^θ the number of times the RUL prognostic falls below T before an alarm is triggered, and β^θ the safety factor. These parameters define agent $\theta \in \Theta_i$. We initialize T^θ with a random integer in $[k, l]$, n^θ with a random integer in $[1, 5]$ and β^θ with a random value from $[0.01, 0.02, \dots, 1.00]$ for all agents in the initial generation Θ_0.

Selection of the parents Let θ_f denote the fitness of agent $\theta \in \Theta_i$ of iteration i. We select N parents for the population of the $i+1^{th}$ generation with tournament selection: For each parent, we first randomly select r individual agents from Θ_i. Then, the agent with the highest fitness θ_f of these r agents is selected as a parent.

Reproduction: Crossover and mutation Each pair of 2 parents selected from Θ_i generates 2 new child agents, that become part of the population Θ_{i+1} of generation $i+1$. We use one-point crossover [33] to generate the two new child agents. We mutate each element of the new child agent chromosome with a probability $\tilde{p}$. An element is mutated with random resetting [33].

Termination The GA is terminated after M generations. The agent with the highest fitness across all generations is selected as the final agent.

Monte Carlo simulation to evaluate the fitness of a GA agent

We evaluate the fitness θ_f of an agent θ with parameters $\theta_p = \left(T^\theta, n^\theta, \beta^\theta\right)$ by calculating the expected maintenance costs over a period of 10 years using Monte Carlo simulation. We perform 100 Monte Carlo simulation runs (iterations). For each iteration $i \in \{1, 2, \dots, 100\}$, we generate a maintenance planning over a period of 10 years using the rolling horizon approach (see Section 6.3.1). At the beginning of each scheduling time window D_{d_0} in the rolling horizon approach, we update the RUL prognostics. With these RUL prognostics and the alarm threshold T^θ and n^θ of the agent θ under consideration, we determine the set of alarmed components V^{alarm}. We also determine the target day to maintain each alarmed component using the safety factor β^θ and the updated RUL prognostics. We then assign each alarmed component to exactly one maintenance slot in the scheduling time window D_{d_0}, using the proposed scheduling model (see eqs. (6.7)-(6.10)).

After each iteration i, we calculate the costs $\theta_{c,i}$ of the generated maintenance planning using c_f, the costs of an engine failure, c_p, the costs of an additional maintenance task, c_r, the costs of rescheduling a maintenance task and c_g, the costs of using a generic slot. We define the fitness of agent θ as:

$$\theta_f = \frac{1}{\frac{1}{100}\sum_{i=1}^{100}\theta_{c,i}}, \tag{6.13}$$

i.e., the higher the expected costs, the lower the fitness of the agent.

6.4. Case study and results - Engine maintenance scheduling

We apply our maintenance planning framework in Section 6.3 to the engines in the C-MAPSS data set [21]. Since in the test set of C-MAPSS the sensor measurements terminate at some time before engine failure, RUL prognostics cannot be generated after every flight until failure. Therefore, we apply our maintenance framework for 14% of the training instances of C-MAPSS, which are run-to-failure instances. A similar approach has been taken in [16]. The 14% instances are randomly selected from the C-MAPSS training subsets, i.e., we randomly select 14% of the engines from subset FD001, resulting in 14 engines selected, 14% of the engines from subset FD002, resulting in 37 engines selected, 14% of the engines from subset FD003, resulting in 14 engines selected, and 14% of the engines from subset FD004, resulting in 35 engines selected. In total, we select 100 engines for which we apply our proposed maintenance framework. For these 100 instances, we obtain RUL prognostics using CNNs, as discussed in Section 6.2. For this, we do not use any knowledge about the actual failure times of these engines. The remaining 86% of the training instances of each subset are used to train the CNNs.

Let E denote the set of the selected 100 turbofan engines. These engines have an average lifespan of 204 flights, with a minimum lifespan of 110 flights, and a maximum lifespan of 430 flights.

6.4.1. Imperfect RUL prognostics for turbofan engines

Using the CNN as discussed in Section 6.2, we obtain a RUL prognostic after every flight for each engine in the set E. Figure 6.7 shows the obtained RUL prognostics up to $R_{\text{early}} = 125$ flights before failure for each engine in E. Figure 6.6 shows the RMSE for the obtained prognostics up to 125 flights before failure. The accuracy of the RUL prognostics varies over time and across the 4 subsets of the C-MAPSS dataset.

We evaluate the obtained series of RUL prognostics using the RMSE, the Cumulative Relative Accuracy (CRA) and the Convergence of the RMSE metrics [34]. The CRA$_\lambda$ is defined as follows [34]:

$$\text{CRA}_\lambda = \frac{1}{n}\sum_{w=1}^{n} 1 - \frac{|\text{RUL}_{w,\lambda}^{\text{actual}} - \text{RUL}_{w,\lambda}^{\text{estimateed}}|}{\text{RUL}_{w,\lambda}^{\text{actual}}},$$

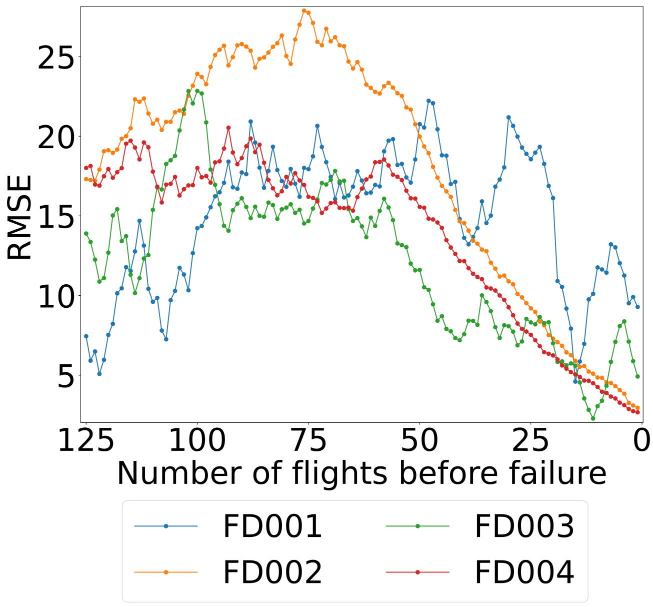


Figure 6.6.: RMSE of the RUL prognostics over time for the engines in E, split over the four C-MAPSS subsets.

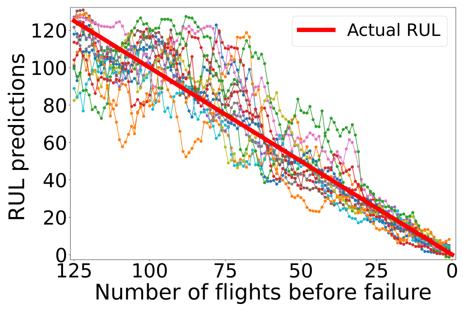
where n is the number of components in the considered data subset, $\text{RUL}^{\text{actual}}_{w,\lambda}$ is the actual RUL of engine w at λ percent of its lifetime (i.e., $\lambda = 0.5$ gives the actual RUL of engine w halfway its lifetime, and $\lambda = 0.9$ gives the actual RUL of engine w at 90% of its lifetime), and $\text{RUL}^{\text{estimated}}_{w,\lambda}$ is the estimated RUL of engine w at λ percent of its lifetime.

The convergence of the RMSE metric [34] quantifies how fast the RMSE metric converges over time to its minimum value, assuming that the RUL prognostics improve over time. The convergence of the RMSE is defined as:

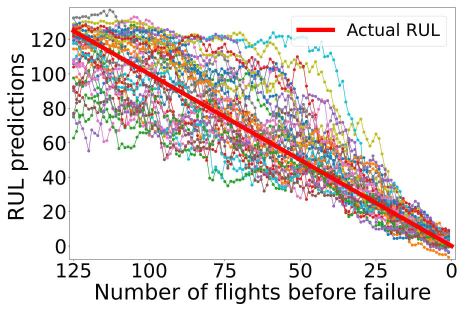
$$\text{Convergence} = \sqrt{\left(x_c - R_{\text{early}}\right)^2 + y_c^2},$$

where (x_c, y_c) is the centroid of the area under the RMSE curve in Figure 6.6. Notice that this curve starts only at $R_{\text{early}} = 125$ flights before failure. The lower the convergence, the faster the RMSE converges.

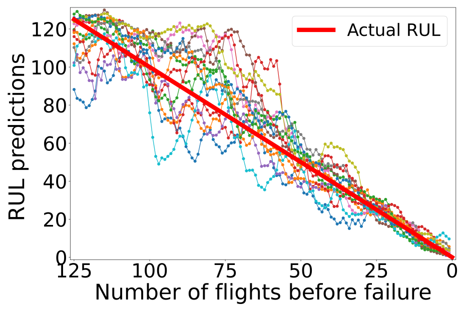
Table 6.4 shows the RMSE, the $\text{CRA}_\lambda, \lambda \in \{0.5, 0.9\}$ and the Convergence of the RMSE, obtained for the engines in the set E. The results show that CRA_λ improves with increasing λ, i.e., the RUL prognostics improve as the engines approach the actual time of failure. The Convergence of the RMSE ranges between 51.00 and 64.64, and the highest Convergence of the RMSE is obtained for data subset FD001.



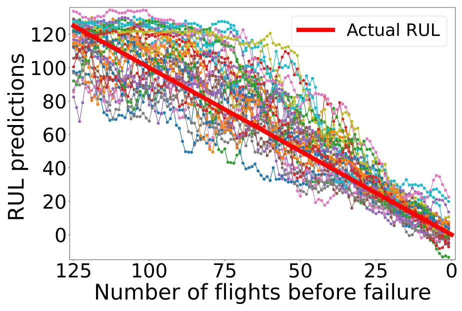
(a) The 14 test engines of data set FD001.



(b) The 37 test engines of data set FD002.



(c) The 14 test engines of data set FD003.



(d) The 35 test engines of data set FD004.

Figure 6.7.: RUL prognostics over time for the engines in E, split over the four C-MAPSS subsets.

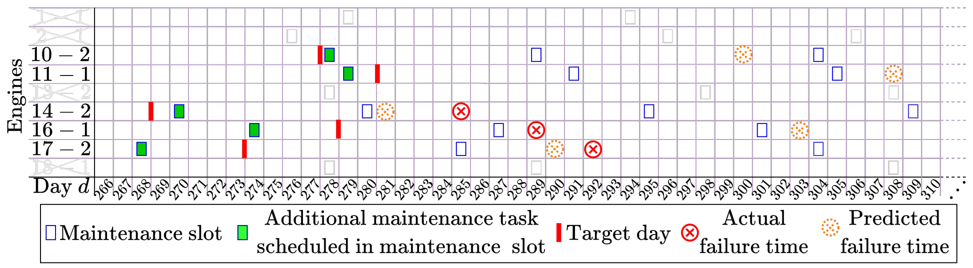
	FD001	FD002	FD003	FD004
RMSE (flight)	14.35	17.98	12.05	15.15
$\text{CRA}_{0.5}$	0.79	0.74	0.79	0.80
$\text{CRA}_{0.9}$	0.93	0.97	0.97	0.98
Convergence (flights)	64.64	51.96	52.07	51.00

Table 6.4.: RMSE, Cumulative Relative Accuracy (CRA), and Convergence of the RMSE, for the RUL prognostics results for the run-to-failure data of the engines in set E.

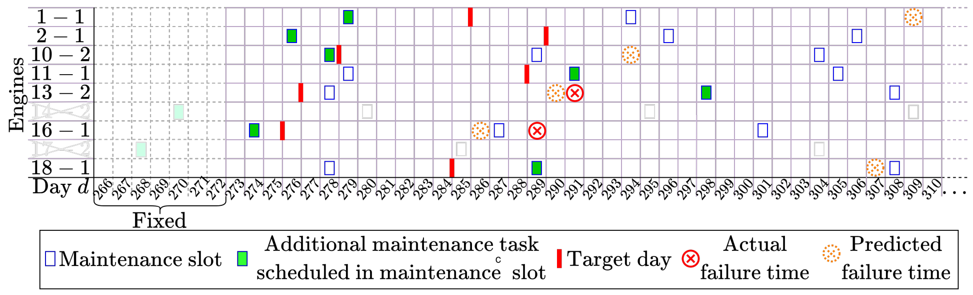
6.4.2. Alarm-based maintenance scheduling for aircraft engines

We consider a fleet of $|A| = 20$ aircraft, each equipped with $|V_a| = 2$ engines. The engines are randomly selected from E. For each aircraft $a \in A$, we label the two engines as $a-1$ (first engine of aircraft a) and $a-2$ (second engine of aircraft a).

For each aircraft, we consider maintenance slots with a frequency of 10 - 20 days, reflecting a realistic maintenance slots frequency [32]. Moreover, an additional maintenance task can be scheduled for only $h = 1$ engine per day, and at least $k = 7$



(a) Maintenance schedule created at day $d_0 = 259$. Here, the actual failure time is day 312 for engine 10-2 and day 327 for engine 11-1.



(b) Maintenance schedule created at day $d_0 = 266$. Here, the actual failure time is day 329 for engine 1-1, day 319 for engine 2-1, day 312 for engine 10-2, day 327 for engine 11-1, and day 314 for engine 18-1. The predicted failure time is day 319 for engine 2-1 and day 317 for engine 11-1.

Figure 6.8.: Maintenance schedule in 2 sequential time windows of the rolling horizon approach, at day $d_0 = 259$ and day $d_0 = 266$.

days (one week) are needed to prepare an additional maintenance task. Also, the available maintenance slots are known up to $k + l = 70$ days (10 weeks) in advance, and the maintenance schedule is updated every $\tau = 7$ days (one week). We assume that each aircraft performs one flight per day.

We assume a cost $c_r = 5000$ for rescheduling an additional maintenance task, $c_p = 10{,}000$ for performing an additional maintenance task, $c_f = 50{,}000$ for an engine failure and $c_g = 10^6$ for using a generic slot. For the objective function of the integer linear program in Section 6.3.2, we consider the maintenance penalties $p^{\text{early}} = 1$ for every day maintenance is scheduled earlier than the target day, $p^{\text{res}} = 100$ for rescheduling an additional maintenance task, $p^{\text{late}} = 1000$ for every day maintenance is scheduled after the target day and $p^{\text{gen}} = 10^6$ for using a generic slot.

With these penalties, we thus consider the target day of a maintenance task as a strict deadline; performing maintenance far before the target day is preferred over performing maintenance just after the target day.

To determine the alarm policy (T, n, β) using the GA in Section 6.3.3, we consider $N = 30$ agents per population, $M = 20$ generations, $r = 5$ participants in each tournament and a mutation probability $\tilde{p} = 1/3$. As result, we obtain the alarm threshold $T = 49$ days, $n = 1$ and the safety factor $\beta = 0.44$.

Figure 6.8 shows the maintenance schedule with this alarm policy for two sequential time windows of the rolling horizon approach, at day $d_0 = 259$ and day $d_0 = 266$. Here, from the total of 40 engines, we only show the alarmed engines. At day d_0, the beginning of scheduling window D_{d_0}, we update the RUL prognostics of the engines. With these updated RUL prognostics, the alarm threshold $T = 49$ days, $n = 1$ and the safety factor $\beta = 0.44$, we then determine the alarmed engines and the corresponding target days. These target days and the available maintenance slots are the input of the integer linear program in Section 6.3.2. An alarmed engine is assigned to exactly one maintenance slot by this integer linear program.

For example, aircraft 17, engine 2 $(17-2)$ has four maintenance slots in Figure 6.8: at days 268, 285, 304 and 322 (not depicted). The estimated failure time of engine 17-2 is at day 290, while the actual failure time is at day 292. A maintenance task for engine 17-2 is scheduled at day 268, well before the target day at day 273.

At day $d_0 = 259$, aircraft 10, 11, 14, 16 and 17 each have one alarmed engine. For each alarmed engine, an additional maintenance task is scheduled in the first maintenance slot available. For the alarmed engines of aircraft 11, 16 and 17, this maintenance slot is before the target day, while for the alarmed engines of aircraft 10 and 14, this maintenance slot is after the target day.

6

At day $d_0 = 266$, the additional maintenance tasks for aircraft 14 and 17 are fixed from the previous time window. However, the additional maintenance tasks for aircraft 10, 11 and 16 may still be rescheduled. Moreover, aircraft 1, 2 and 18 also have an alarmed engine now. For aircraft 10 and 16, the additional maintenance task remains planned at day 278 and day 274 respectively, as in the previous time window. However, the additional maintenance task for aircraft 11 is rescheduled from day 279 to day 291. This reschedulement is because only $h = 1$ additional maintenance task can be planned per day, and at day 279 an additional maintenance task is now scheduled for engine 1-1. Moreover, the maintenance task of engine 13-2 is only planned at day 298, 22 days after its target day and also after its failure time. This is because the only maintenance slot available for this aircraft before day 298 is at day 278, when maintenance for engine 10-2 is already scheduled. At the next maintenance opportunity for engine 10-2, maintenance for engine 18-1 is already scheduled. If we maintain engine 13-2 at day 278, we would thus have to reschedule the maintenance for engine 10-2 to day 304.

Table 6.5 shows the alarmed engines at day $d_0 = 259$ and day $d_0 = 266$. At the moment of the alarm, the estimated RUL is between 39 and 48 days, while the actual RUL is between 30 to 70 days. The RUL prediction error at the moment of the alarm has an error between -22 days (underestimation of the failure time) to 14 days (overestimation of the failure time). The error in the RUL prognostics underlines the need for a safety factor β, with which a target day is determined.

The computational time to optimize the maintenance schedule for the first and second time window using the integer linear program in Section 6.3.2 is 0.012sec and 0.022sec respectively, on a computer with an Intel Core i7 processor (8th generation) at 2.11 GHz and 8Gb RAM. The integer linear program is solved using the optimizer

Engine (a-1, a-2)	Day of alarm	Predicted RUL at alarm (days)	Actual RUL at alarm (days)
1-1	265	46	64
2-1	261	44	58
10-2	259	41	53
11-1	257	48	70
13-2	263	39	28
14-2	243	47	42
16-1	259	44	30
17-2	254	42	38
18-1	262	48	52

Table 6.5.: The day of the alarm, and the estimated and actual RUL at the moment of the alarm, for all engines that are alarmed at day $d_0 = 259$ and day $d_0 = 266$.

Gurobi version 9.0.2 with standard settings, implemented in Python.

ENGINE FAILURES UNDER THE PROPOSED ALARM-BASED MAINTENANCE FRAMEWORK

In this section, we analyze the engine failures that occur during a period of 10 years when applying our maintenance framework for a fleet of $|A| = 20$ aircraft with $|V_a| = 2$ aircraft turbofan engines.

Engine (a-1, a-2)	Day of alarm	Predicted RUL at alarm (days)	Actual RUL at alarm(days)	Actual failure day	Target day d^{target} at alarm
13-1	103	45	21	124	122
11-2	229	48	25	254	250
12-1	227	48	28	255	248
13-2	263	40	28	291	280
6-2	358	45	35	393	377
4-1	1188	37	29	1217	1204
10-1	1859	48	25	1884	1880
1-2	2175	48	28	2203	2196
16-1	2531	44	23	2554	2550
20-1	2683	44	33	2716	2702
17-2	3143	48	28	3171	3164
1-1	3193	46	42	3235	3213

Table 6.6.: Failures occurring in 10 years of operations for a fleet of 20 aircraft using the maintenance framework. The day of the alarm is calculated since the beginning of the simulation, i.e., since day 0.

A total of 12 engine failures occur in the considered time period of 10 years. These failures are described in Table 6.6. At the day of the alarm, the RUL is overestimated by 4 - 24 days: In other words, engine failures mainly occur when the RUL is (greatly) overestimated at the moment of the alarm. Using our proposed safety factor $\beta = 0.44$, the target day is 2 - 22 days before the actual failure time at the day of the alarm. Should additional maintenance tasks for these 12 engines have been scheduled before or at the initial target day, then these engines would thus not have failed.

However, the maintenance tasks are scheduled after the initial target day. For 6 out of the 12 engine failures, this is because no maintenance slot is available before the target day, i.e., there is a lack of maintenance slots. For the other failures, a maintenance slot is available before the target day. However, an additional maintenance task for another engine is already scheduled during the day of this maintenance slot, whereas at most $h = 1$ additional tasks can be scheduled per day. Increasing the number of maintenance slots or the maintenance capacity would thus help to decrease the number of engine failures.

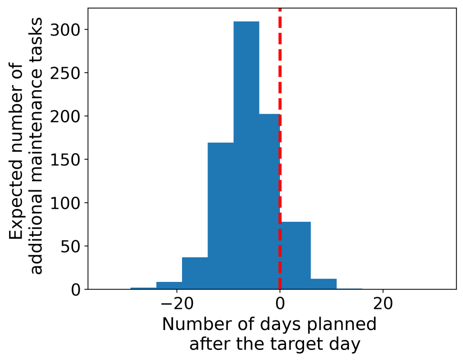
6.4.3. Maintenance with perfect RUL prognostics vs. imperfect RUL prognostics

We evaluate the performance of our proposed maintenance framework for a fleet of $|A| = 20$ aircraft, each equipped with $|V_a| = 2$ engines, over a period of 10 years using Monte Carlo simulation, with 1000 simulation runs. We compare the performance of our framework when considering perfect and imperfect RUL prognostics.

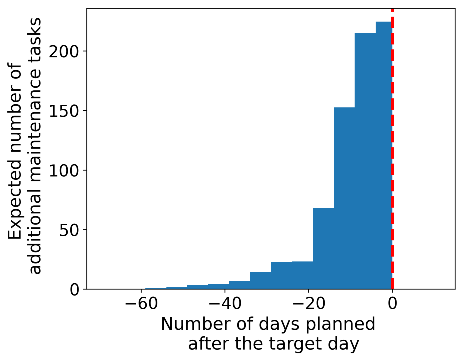
Predictive maintenance planning with perfect RUL prognostics

We apply our maintenance framework in Section 6.3.2 together with perfect RUL prognostics, i.e., the RUL prognostics equal the actual RUL of the engines. As we have perfect RUL prognostics, we set the safety factor $\beta = 1$. Moreover, we do not postpone planning maintenance to obtain more accurate RUL prognostics or avoid reschedulements due to updated RUL prognostics. Instead, we define that an engine becomes alarmed as soon as its RUL prognostic is below an alarm threshold $T = k + l = 70$ days, for $n = 1$ day in a row. Here, $k + l$ is the period of time up to which maintenance slots are known.

Using perfect RUL prognostics, more than 99.9% of the maintenance tasks are scheduled before the target day (see Figure 6.9b). This is possible since an engine becomes alarmed $k + l = 70$ days before its target day. There are thus multiple possible maintenance slots in which the additional maintenance task can be scheduled. The expected number of engine failures is therefore nearly zero (see Table 6.7). In contrast, 11% of the additional maintenance tasks are planned after the target day when considering imperfect RUL prognostics (see Figure 6.9a). This is because an engine becomes alarmed when the estimated RUL equals $T = 49$ days or less. Once a component becomes alarmed, only $\beta \cdot$ estimated RUL $\leq 0.44 \cdot 49 = 21$ days or less are thus available before the target day. There is, therefore, a smaller time window to schedule an additional maintenance task. This leads, in combination with the imperfect RUL prognostics, to a larger expected number of engine failures



(a) Maintenance planning with imperfect RUL prognostics.



(b) Maintenance planning with perfect RUL prognostics.

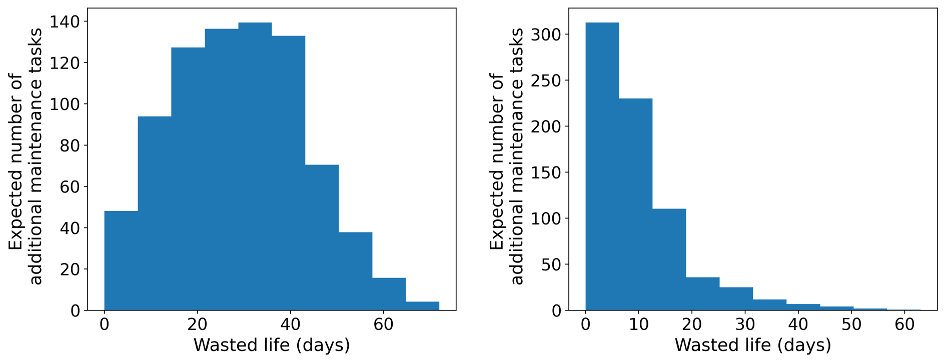
Figure 6.9.: Expected number of days an additional maintenance task is scheduled before (negative number) or after (positive number) the final target day in ten years, using perfect and imperfect RUL prognostics.

and thus to less reliable aircraft (see Table 6.7). We note that the generic slot is never used in our case study, showing that the slots were sufficient to perform the required maintenance tasks.

	Imperfect RUL prognostics	Perfect RUL prognostics
Engine failures	13.61	0.10
Rescheduled maintenance tasks	67.26	2.15
Additional maintenance tasks	819.7	739.0

Table 6.7.: Long-term expected performance in ten years when considering imperfect and perfect RUL prognostics. The results are obtained with 95% confidence intervals that have a maximum width of 0.6 engine failures, 1.4 rescheduled maintenance tasks and 1.2 additional maintenance tasks, respectively.

With imperfect RUL prognostics, the expected number of rescheduled maintenance tasks equals 67.26, while it equals only 2.15 when considering perfect RUL prognostics (Table 6.7). For imperfect RUL prognostics, the number of rescheduled maintenance tasks is higher since, i) the target day changes over time due to updated RUL prognostics and ii) the engines become alarmed $\beta \cdot$estimated RUL $\leq 0.44 \cdot 49 = 21$ days or less before their target day, resulting in a smaller time window to find an optimal maintenance moment for each engine. With imperfect RUL prognostics, more engine life is wasted as well due to the safety factor β (see Figure 6.10). As a consequence, more additional maintenance tasks are performed than when considering perfect RUL prognostics (Table 6.7).



(a) Maintenance planning with imperfect RUL prognostics.

(b) Maintenance planning with perfect RUL prognostics.

Figure 6.10.: Expected engine wasted life, using perfect and imperfect prognostics, in ten years.

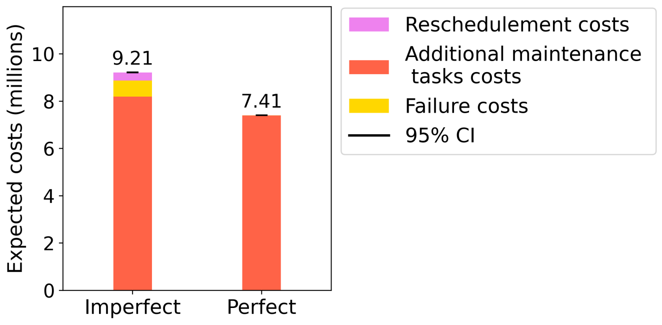


Figure 6.11.: Expected costs over a period of 10 years, using imperfect and perfect RUL prognostics.

The total expected costs decrease by 19.7% when using perfect RUL prognostics instead of imperfect RUL prognostics (see Figure 6.11). In both cases, most of the costs are driven by the costs of performing additional maintenance tasks: 89.7% of the costs come from performing additional maintenance tasks when using imperfect RUL prognostics, while 99.8% of the costs come from performing additional maintenance tasks when using perfect RUL prognostics. When considering imperfect RUL prognostics, 7.4% from the costs are due to engine failures. Also, only 3.7% of the costs are a result of rescheduling maintenance tasks.

$\tilde{p}$	N	r	Iteration i	$\theta_f \cdot 10^7$	Mean cost (millions)
0.1	10	5	12	1.103	9.068
	30	5	24	1.105	9.047
		10	15	1.103	9.067
	50	5	5	1.106	9.043
		10	48	1.105	9.047
$\frac{1}{3}$	10	5	43	1.100	9.092
	30	5	9	1.106	9.043
		10	8	1.106	9.043
	50	5	33	1.106	9.043
		10	6	1.106	9.043
0.5	10	5	22	1.101	9.085
	30	5	6	1.106	9.043
		10	43	1.104	9.058
	50	5	30	1.103	9.067
		10	18	1.105	9.047

Table 6.8.: Sensitivity analysis - hyperparameters of the GA. The best agent is first found in iteration i.

6.4.4. SENSITIVITY ANALYSIS - HYPERPARAMETERS OF THE GENETIC ALGORITHM

We perform a sensitivity analysis to evaluate the influence of the hyperparameters on the GA in Section 6.3.3. We consider the number of agents $N \in \{10, 30, 50\}$, the mutation probability $\tilde{p} \in \{0.1, 1/3, 0.5\}$, the number of participants in the tournament selection $r \in \{5, 10\}$ and $M = 50$ generations. Table 6.8 shows the fitness θ_f, the mean costs of the generated maintenance planning (see eq. (6.13)), and iteration i of the GA when the best agent is found first. The fitness θ_f of the final agents θ ranges from $1.100 \cdot 10^{-7}$ to $1.106 \cdot 10^{-7}$, with a corresponding mean cost per iteration of the Monte Carlo simulation run between 9.043 to 9.092 million, i.e., a difference of 0.5%. The performance of the GA is thus robust: a similar solution is found with all considered combinations of the hyperparameters.

The final agent with the maximum fitness of $1.106 \cdot 10^{-7}$, and the corresponding costs of 9.043 million, is consistently found with a mutation probability of $\tilde{p} = \frac{1}{3}$ and $N = 30$ or $N = 50$ agents. With $\tilde{p} = \frac{1}{3}$, $N = 30$ and $r \in \{5, 10\}$, the best fitness is obtained in just 9 and 8 iterations of the GA, respectively.

6.5. CONCLUSIONS

We have proposed a dynamic maintenance framework for a fleet of aircraft where component RUL prognostics are updated periodically. Maintenance task scheduling is initiated as soon as an alarm is triggered. These alarms are based on the evolution of the RUL prognostics over time. Tasks are scheduled using a rolling horizon approach with time windows. In each time window, an integer linear program specifies the slots in which maintenance is scheduled. The ideal time to schedule a task is determined based on the RUL prognostics and a safety factor, to account for potential errors in the RUL prognostics. The parameters of the maintenance framework are obtained using a genetic algorithm.

We illustrate our maintenance framework for a fleet of 20 aircraft, each equipped with 2 turbofan engines. The RUL prognostics of these turbofan engines are obtained using a CNN. These prognostics are updated every day. The results show that, with our maintenance framework, alarms are triggered early enough to enable the scheduling of additional tasks such that failures are prevented. The total cost savings with failure prevention outweigh the costs with potential task rescheduling due to an early alarm. The results also show that engine failures still occur due to the limited availability of maintenance slots or due to the limited number of maintenance tasks that can be performed per day. The long-term results show that the costs due to engine failures account for only 7.4% of the total maintenance costs. In the ideal case with perfect RUL prognostics, the maintenance costs are 19.7% lower.

The proposed maintenance planning framework is applicable to other aircraft components as well. Of course, the costs considered should be adjusted accordingly. For example, the costs of a component failure c_f and the corresponding penalty p^{late} for planning a maintenance task after the target day may be lowered if a component is non safety-critical. In general, we can apply the proposed maintenance planning framework to condition-monitored assets from other industries as well, e.g., a fleet of trains or a fleet of ships. Additional industry-specific constraints may be considered for the maintenance of these assets. An interesting constraint to consider is to also tune the planning horizon as parameter, depending on the industry-specific possibilities for this planning horizon.

REFERENCES

[1] de Pater, I., Reijns, A., & Mitici, M. (2022). Alarm-based predictive maintenance scheduling for aircraft engines with imperfect Remaining Useful Life prognostics. *Reliability Engineering & System Safety, 221*, Article number: 108341.

[2] Maintenance Cost Technical Group (MCTG). (2020). *Airline maintenance cost executive commentary (FY2019 data), public version* (tech. rep.). International Air Transport Association (IATA).

[3] Hu, Y., Miao, X., Si, Y., Pan, E., & Zio, E. (2021). Prognostics and Health Management: A review from the perspectives of design, development and decision. *Reliability Engineering & System Safety*, *217*, Article number: 108063.

[4] Lei, Y., Li, N., Guo, L., Li, N., Yan, T., & Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing, 104*, Pages: 799–834.

[5] de Pater, I., & Mitici, M. (2021, June 28 - July 2). Model-based Remaining-Useful-Life prognostics for aircraft cooling units. *Proceedings of the European Conference of the Prognostics and Health Management (PHM) Society*, *6*, Virtual, Pages: 1–8.

[6] Mitici, M., & de Pater, I. (2021). Online model-based Remaining-Useful-Life prognostics for aircraft cooling units using time-warping degradation clustering. *Aerospace*, *8*(6), Article number: 168.

[7] Lee, J., & Mitici, M. (2020). An integrated assessment of safety and efficiency of aircraft maintenance strategies using agent-based modelling and stochastic Petri Nets. *Reliability Engineering & System Safety, 202,* Article number: 107052.

[8] Li, X., Ding, Q., & Sun, J.-Q. (2018). Remaining Useful Life estimation in prognostics using deep Convolution Neural Networks. *Reliability Engineering & System Safety, 172,* Pages: 1–11.

[9] Babu, G. S., Zhao, P., & Li, X.-L. (2016, April 16-19). Deep Convolutional Neural Network based regression approach for estimation of Remaining Useful Life. *Proceedings of the 21st International Conference on Database Systems for Advanced Applications (DASFAA)*, Dallas, Texas, USA, Pages: 214–228.

[10] Li, H., Zhao, W., Zhang, Y., & Zio, E. (2020). Remaining Useful Life prediction using multi-scale deep Convolutional Neural Network. *Applied Soft Computing, 89,* Article number: 106113.

[11] Cao, Y., Ding, Y., Jia, M., & Tian, R. (2021). A novel temporal Convolutional Network with residual self-attention mechanism for Remaining Useful Life prediction of rolling bearings. *Reliability Engineering & System Safety, 215,* Article number: 107813.

[12] Li, X., Zhang, W., & Ding, Q. (2019). Deep learning-based Remaining Useful Life estimation of bearings using multi-scale feature extraction. *Reliability Engineering & System Safety*, *182*, Pages: 208–218.

[13] Jimenez, J. J. M., Schwartz, S., Vingerhoeds, R., Grabot, B., & Salaün, M. (2020). Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics. *Journal of Manufacturing Systems*, *56*, Pages: 539–557.

[14] Verbert, K., De Schutter, B., & Babuška, R. (2017). Timely condition-based maintenance planning for multi-component systems. *Reliability Engineering & System Safety*, *159*, Pages: 310–321.

[15] de Pater, I., del Mar Carillo Galera, M., & Mitici, M. (2021, September 19-23). Criticality-based predictive maintenance scheduling for aircraft components with a limited stock of spare components. *Proceedings of the 31st European Safety and Reliability Conference*, Angers, France, Pages: 55–62.

[16] Nguyen, K. T., & Medjaher, K. (2019). A new dynamic predictive maintenance framework using deep learning for failure prognostics. *Reliability Engineering & System Safety*, *188*, Pages: 251–262.

[17] Yiwei, W., Christian, G., Binaud, N., Christian, B., Haftka, R. T., et al. (2017). A cost driven predictive maintenance policy for structural airframe maintenance. *Chinese Journal of Aeronautics*, *30*(3), Pages: 1242–1257.

[18] de Pater, I., & Mitici, M. (2021). Predictive maintenance for multi-component systems of repairables with Remaining-Useful-Life prognostics and a limited stock of spare components. *Reliability Engineering & System Safety*, *214*, Article number: 107761.

[19] Lee, J., & Mitici, M. (2022). Multi-objective design of aircraft maintenance using Gaussian process learning and adaptive sampling. *Reliability Engineering & System Safety*, *218*, Article number: 108123.

[20] Wang, Y., Limmer, S., Van Nguyen, D., Olhofer, M., Bäck, T., & Emmerich, M. (2021). Optimizing the maintenance schedule for a vehicle fleet: A simulation-based case study. *Engineering Optimization*, *54*(7), Pages: 1258–1271.

[21] Saxena, A., & Goebel, K. (2008). *Turbofan engine degradation simulation data set*, NASA Prognostics Data Repository, NASA Ames Research Center, Moffett Field, California, USA.

[22] Wang, B., Lei, Y., Li, N., & Yan, T. (2019). Deep separable Convolutional Network for Remaining Useful Life prediction of machinery. *Mechanical Systems and Signal Processing*, *134*, Article number: 106330.

[23] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

[24] Song, Y., Zhang, Y., Bliek, L., & Xia, T. (2021, September 19-23). A temporal pyramid pooling-based Convolutional Neural Network for Remaining Useful Life prediction. *Proceedings of the 31st European Safety and Reliability Conference*, Angers, France, Pages: 603–609.

[25] Xia, J., Feng, Y., Lu, C., Fei, C., & Xue, X. (2021). LSTM-based multi-layer self-attention method for Remaining Useful Life estimation of mechanical systems. *Engineering Failure Analysis*, *125*, Article number: 105385.

[26] Zhao, Y., & Wang, Y. (2021). Remaining Useful Life prediction for multi-sensor systems using a novel end-to-end deep-learning method. *Measurement*, *182*, Article number: 109685.

[27] Li, T., Zhao, Z., Sun, C., Yan, R., & Chen, X. (2021). Hierarchical attention graph convolutional network to fuse multi-sensor signals for Remaining Useful Life prediction. *Reliability Engineering & System Safety*, *215*, Article number: 107878.

[28] Peng, C., Chen, Y., Chen, Q., Tang, Z., Li, L., & Gui, W. (2021). A Remaining Useful Life prognosis of turbofan engine using temporal and spatial feature fusion. *Sensors*, *21*(2), Article number: 418.

[29] Al-Dulaimi, A., Zabihi, S., Asif, A., & Mohammadi, A. (2019). A multimodal and hybrid deep neural network model for Remaining Useful Life estimation. *Computers in Industry*, *108*, Pages: 186–196.

[30] Ellefsen, A. L., Bjørlykhaug, E., Æsøy, V., Ushakov, S., & Zhang, H. (2019). Remaining Useful Life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering & System Safety*, *183*, Pages: 240–251.

[31] Ruiz-Tagle Palazuelos, A., Droguett, E. L., & Pascual, R. (2020). A novel deep capsule neural network for Remaining Useful Life estimation. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, *234*(1), Pages: 151–167.

[32] Ackert, S. P. (2010). *Basics of aircraft maintenance programs for financiers* (tech. rep.). Aircraft Monitor.

[33] Kramer, O. (2017). Genetic algorithms. In *Genetic algorithm essentials* (Pages: 11–19). Springer.

[34] Saxena, A., Celaya, J., Saha, B., Saha, S., & Goebel, K. (2009, September 27 - October 1). On applying the prognostic performance metrics. *Proceedings of the Annual Conference of the Prognostics and Health Management (PHM) Society 2009*, *1*, San Diego, California, USA, Pages: 1–16.

7 Maintenance scheduling with probabilistic RUL prognostics

In the previous chapter, we show that it is difficult to integrate point Remaining Useful Life (RUL) prognostics without quantified uncertainty in the maintenance planning. In Chapters 4 and 5, we therefore develop probabilistic RUL prognostics, in the form of a Probability Density Function (PDF), instead. In this chapter, we integrate these probabilistic RUL prognostics in the maintenance planning.

We first develop probabilistic RUL prognostics using a Convolutional Neural Network with Monte Carlo dropout (similar to the approach in Chapter 5). We then integrate these probabilistic RUL prognostics in the maintenance planning. First, the optimal maintenance moment for a single component is found. Then, the multi-component maintenance planning is optimized, where the number of components that can be replaced per day is limited. The maintenance planning for a period of ten years is analyzed with a Monte Carlo simulation.

Our approach is illustrated for the maintenance planning of aircraft turbofan engines. The optimal replacement time for the engines is close to the lower bound of the 99% confidence interval of the probabilistic RUL prognostics. Consequently, on average only 0.003 engines fail per ten years in the Monte Carlo simulation.

Parts of this chapter have been published in:

Mitici, M., de Pater, I., Barros, A., & Zeng, Z. (2023). Dynamic predictive maintenance for multiple components using data-driven probabilistic RUL prognostics: The case of turbofan engines. *Reliability Engineering & System Safety, 234,* Article number: 109199

Mitici, M., de Pater, I., Zeng, Z., & Barros, A. (2023, September 3-7). Predictive maintenance planning using renewal reward processes and probabilistic RUL prognostics–Analyzing the influence of accuracy and sharpness of prognostics. *Proceedings of the 33st European Safety and Reliability Conference*, Southampon, UK, Pages: 1034–1041

7.1. INTRODUCTION

Modern systems are monitored by multiple sensors that generate large volumes of data. For example, for a Boeing 787, 20 terabytes of sensor measurements are generated per flight hour [3]. In predictive maintenance, a new maintenance strategy, such sensor measurements are used to estimate the time left until failure, called the Remaining Useful Life (RUL), of components and systems [4]. These RUL prognostics are subsequently used to plan maintenance. It is shown that predictive maintenance, with this integration of RUL prognostics in the maintenance planning, significantly reduces the maintenance costs and the number of failures [5–7]. However, most existing studies on predictive maintenance focus only on developing RUL prognostics, without optimizing the maintenance planning: According to [8], the use of RUL prognostics in the maintenance optimization for complex systems is a relatively underexplored area. On the other hand, many studies optimize the maintenance planning without RUL prognostics, where the probability of failure is based on a generic probability distribution instead [9].

Studies on developing RUL prognostics often input the sensor measurements in a machine learning model, which estimates the RUL. Most studies estimate one point for the RUL, i.e., they develop point RUL prognostics without quantified uncertainty. For example, in [10], point RUL prognostics are developed for batteries using a deep neural network, while point RUL prognostics are developed in [6, 11] for turbofan engines using a Convolutional Neural Network (CNN). Only few studies instead estimate the Probability Density Function (PDF) of the RUL. In [12] and [13], the PDF of the RUL of turbofan engines is estimated using a deep Gaussian process and a neural network with Monte Carlo dropout, respectively.

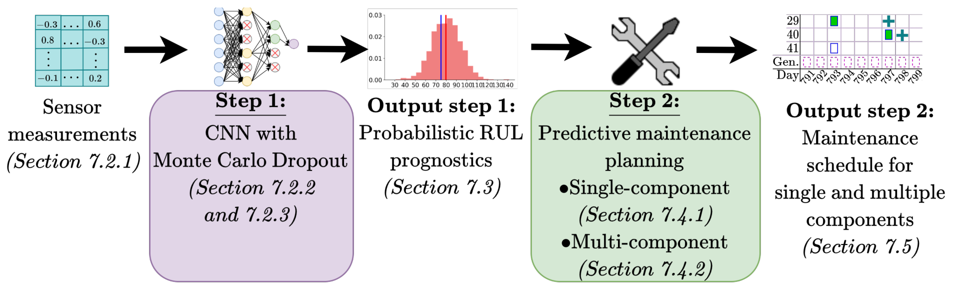

−0.3 ... 0.6
0.8 ... −0.3
−0.1 ... 0.2
Sensor measurements (Section 7.2.1)
Step 1:
CNN with Monte Carlo Dropout (Section 7.2.2 and 7.2.3)
Output step 1:
Probabilistic RUL prognostics (Section 7.3)
Step 2:
Predictive maintenance planning
•Single-component (Section 7.4.1)
•Multi-component (Section 7.4.2)
Output step 2:
Maintenance schedule for single and multiple components (Section 7.5)
29
40
41
Gen.
Day
791 792 793 794 795 796 797 798 799

Figure 7.1.: Overview of the roadmap for predictive maintenance, where sensor measurements are used to develop probabilistic RUL prognostics with a CNN with Monte Carlo dropout. These RUL prognostics are subsequently integrated in the maintenance planning.

Other studies propose advanced maintenance planning models, but assume that the degradation of the components/systems follows a generic distribution with fixed parameters. These studies thus do not develop component-specific RUL prognostics from the sensor measurements. For instance, some studies assume that the degradation of components/systems follows a generic Gamma process [8, 14–16], a Wiener process [17], or a non-homogeneous Poisson process [18]. With these assumptions, the maintenance planning of the components/systems is optimized with the renewal-reward process [16], a Markov decision process [19, 20], or a heuristic [8]. In practice, however, the degradation of components/systems rarely follows such a generic degradation process. By using the sensor measurements to predict the RUL of each specific component, much more detailed information on the failure time of an individual component is obtained.

Few studies develop data-driven RUL prognostics by tracking the specific degradation of a component/system through sensor measurements, and actually integrate these RUL prognostics in the maintenance planning. In [6] and [21], data-driven point RUL prognostics are developed for aircraft engines using a CNN and a recurrent neural network, respectively. These point RUL prognostics are integrated in the maintenance planning using an integer linear program and a threshold-based approach, respectively. With point RUL prognostics, however, there is no insight in the uncertainty of the estimated RUL, which complicates the maintenance planning. Ignoring the uncertainty in the RUL prognostics may lead to many failures. In contrast, we propose a maintenance planning framework that integrates *probabilistic* RUL prognostics into the maintenance planning.

Probabilistic RUL prognostics for aircraft engines are used in [7] to optimize the replacements of the engines with a deep reinforcement learning model. Reinforcement learning, however, is a black-box model, and it is unclear how and why the maintenance decisions are made. In contrast, in this chapter we use the renewal-reward process to optimize the replacements of engines with probabilistic RUL prognostics. With a renewal-reward process, the maintenance decisions are very easy to explain. In [22] the distribution of the RUL for aircraft cooling units is estimated using a physics-based model. Similarly, in [23], the future degradation of a railway track is estimated using a physics-based model. These prognostics are further used to plan the maintenance for the cooling units and railway tracks, respectively. Physic-based models for RUL prognostics, however, require that a physical degradation model is available, which is often not the case for complex systems [24]. In contrast, in this chapter, we therefore generate data-driven probabilistic RUL prognostics for the maintenance planning.

In this chapter, we propose an end-to-end framework for predictive maintenance. In this framework, the sensor measurements are first used to estimate probabilistic RUL prognostics, and these RUL prognostics are subsequently used to optimize the maintenance planning (see Figure 7.1). We develop *probabilistic* RUL prognostics using a CNN with Monte Carlo dropout. These RUL prognostics are subsequently used to determine the optimal maintenance moment for a single component with the renewal-reward process. This is further extended to a multi-component maintenance planning model, where we consider a limited maintenance capacity.

We illustrate our approach with the maintenance planning for aircraft turbofan engines. We simulate the maintenance planning with probabilistic RUL prognostics for several years with a Monte Carlo simulation. With the results of this simulation, we analyse the advantages of using probabilistic RUL prognostics when planning maintenance. For example, an engine is replaced soon if there is a large probability of failure in the near future (i.e., if the estimated RUL is small). In the Monte Carlo simulation, there are therefore only very few engine failures. Last, we compare our maintenance approach with a traditional, time-based maintenance strategy without RUL prognostics. The results show that our approach reduces the maintenance costs by 53% compared to this time-based maintenance strategy. Compared with an ideal case with perfect RUL prognostics, our approach has only slightly more failures.

Overall, the main contributions of this chapter are:

- We obtain reliable probabilistic RUL prognostics, in the form of a PDF, for aircraft turbofan engines using a CNN with Monte Carlo dropout. By estimating the PDF of the RUL, maintenance can be planned while taking the uncertainty associated with the RUL prognostics into account.
- With the renewal-reward process, the optimal maintenance moment is usually found with a generic distribution of the failure time, such as the exponential or Weibull distribution. Instead, we show how a renewal-reward process can be used to plan maintenance based on probabilistic RUL prognostics.
- To find the optimal maintenance moment, we minimize the *expected* costs while considering the uncertainty of the RUL prognostics. Moreover, while most existing studies focus only on single component maintenance planning, we also consider the multi-component maintenance planning, where the limited maintenance capacity is considered.

The remainder of this chapter is structured as follows. In Section 7.2 we develop probabilistic RUL prognostics using a CNN with Monte Carlo dropout. We analyze the accuracy and reliability of these prognostics in Section 7.3. In Section 7.4 we propose an optimization model for the maintenance planning of components, taking into account the estimated PDF of the RUL. In Section 7.5 we illustrate our approach for a set of turbofan engines. We also analyze the performance of our approach relative to a time-based maintenance strategy, and relative to an ideal case with perfect RUL prognostics. Last, conclusions are provided in Section 7.6.

7.2. Methodology - Probabilistic RUL prognostics for turbofan engines

In this section, we estimate the PDF of the RUL of aircraft engines after each flight cycle using a Convolutional Neural Network (CNN) with Monte Carlo dropout.

7.2.1. Description of the dataset

In this study, we consider the aircraft turbofan engines in the C-MAPSS dataset [25]. In this dataset, the degradation of engines is simulated using the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) program of NASA.

The C-MAPSS dataset consists of four subsets: FD001, FD002, FD003 and FD004. In turn, each subset consists of a training and a test set. For each engine in the training set, one measurement per sensor per flight cycle is generated, from the

	FD001	FD002	FD003	FD004
# of training instances	100	260	100	249
# of test instances	100	259	100	248
# of operating conditions	1	6	1	6
# of fault conditions	1	1	2	2

Table 7.1.: The C-MAPSS subsets for aircraft engines [25].

installation of the engine until the failure (i.e., run-to-failure instances). In the test set, one measurement per sensor per flight cycle is generated as well. However, the sensor measurements stop at some time before failure. The goal is to predict the RUL at that moment, i.e., the number of flight cycles until the engine fails. For subset FD002 and FD004, six different flight conditions are present, where each flight is performed under one flight condition. Moreover, in subset FD003 and FD004, two different fault conditions are present (see Table 7.1).

There are 21 sensors in C-MAPSS, of which seven exhibit a constant sensor measurement over time. We therefore only consider the remaining 14 sensors with non-constant measurements. We normalize the sensor measurements in each subset with min-max normalization with respect to the operating condition [26, 27]:

$$\hat{m}_{ij} = \frac{2(m^o_{ij} - m^{\min}_{jo})}{m^{\max}_{jo} - m^{\min}_{jo}} - 1, \tag{7.1}$$

with m^o_{ij} the sensor measurement of sensor j during flight cycle i, where flight cycle i was performed under operating condition o, while $m^{\min}_{jo}$ and $m^{\max}_{jo}$ denote the minimum and maximum value in the training set of sensor j under operating condition o respectively. Last, $\hat{m}_{ij}$ denotes the normalized measurement of sensor j obtained during flight cycle i.

7.2.2. ARCHITECTURE OF THE CONVOLUTIONAL NEURAL NETWORK

Figure 7.2 shows the proposed architecture of the CNN. At flight cycle f of an engine v, we consider data sample X^v_f as input to this CNN:

$$X^v_f = \left[x^v_{f-N}, x^v_{f-N+1}, \ldots, x^v_f\right]. \tag{7.2}$$

Here, N denotes the number of past flight cycles included (i.e., the window size), and x^v_i denotes the normalized sensor measurements of engine v at flight cycle i:

$$x^v_i = \left[\hat{m}^v_{i1}, \hat{m}^v_{i2}, \ldots, \hat{m}^v_{iH}\right], \tag{7.3}$$

with H the total number of considered sensors, and $\hat{m}^v_{ij}$ the normalized sensor measurement of flight cycle i of engine v from sensor j (see eq.(7.1)).

The CNN consists of L convolutional layers (see also Figure 7.2). Each convolutional layer consists of K filters, where each kernel has a size of $1 \times S$,

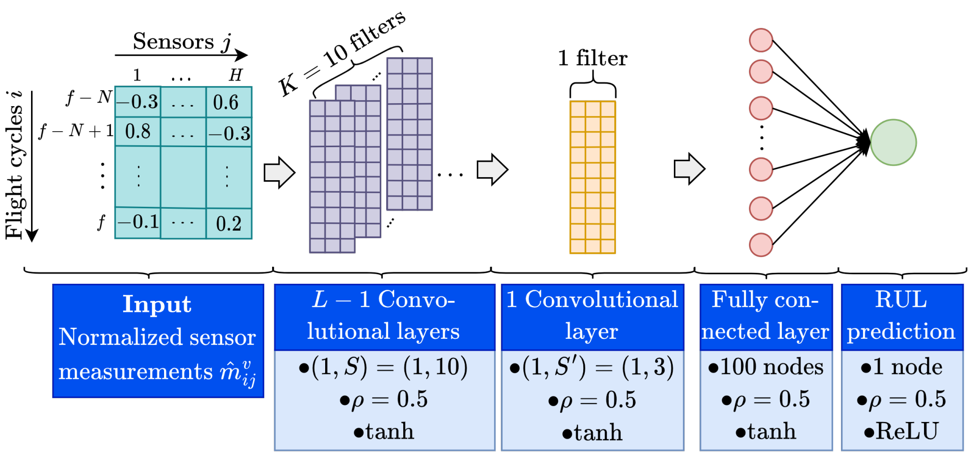


Figure 7.2.: A schematic overview of the considered Convolutional Neural Network.

i.e., we use one-dimensional kernels [26]. The convolutional operation in the l^{th} convolutional layer for the n^{th} filter k^l_n is [28]:

$$z^l_n = \tanh\left(k^l_n * z^{l-1} + b^l_n\right), \tag{7.4}$$

where z^l_n is the n^{th} feature map of layer l, $*$ is the convolutional operator, z^{l-1} are the feature maps in layer $l-1$, b^l_n is the bias of the n^{th} filter of layer l, and $\tanh(\cdot)$ denotes the hyperbolic tangent activation function. Next, we consider a single convolutional layer with one filter, where each kernel has a size of $1 \times S'$. This layer combines all K feature maps in one single feature map. We denote the output of this last convolutional layer by z^L.

Last, we add two fully connected layers to the CNN. These layers predict the RUL based on the extracted features of the last convolutional layer. The output z^f of the first fully connected layer is [28]:

$$z^f = \tanh\left(w^f z^L + b^f\right), \tag{7.5}$$

where b^f is the bias and w^f are the weights of the first fully connected layer. Last, a second fully connected layer with one neuron and the ReLU activation function outputs the final RUL prognostic.

7.2.3. Monte Carlo dropout

We estimate a PDF of the RUL by applying Monte Carlo dropout in the CNN. We apply a dropout rate ρ in each layer of the CNN, except the first layer (to avoid the loss of input information). With Monte Carlo dropout, this dropout rate is not only applied when training the CNN, but also in the testing phase. Specifically, for each test sample, we perform M forward passes through the neural network. During each forward pass, different, randomly selected neurons (ρ percent) are dropped (see

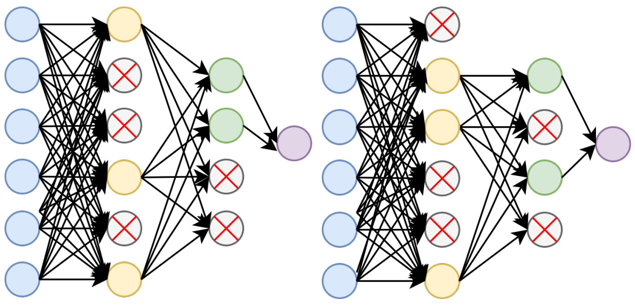


Figure 7.3.: A schematic example of Monte Carlo dropout for a neural network with three fully connected layers, during two passes of a sample through a neural network.

Figure 7.3). Thus, a different RUL prognostic is obtained with each forward pass. In [29], it is shown that a neural network with Monte Carlo dropout approximates a Bayesian neural network representing a deep Gaussian process. Below, we give a short overview of this result and how we apply it to construct a PDF of the RUL.

Let X be the samples with sensor measurements in the training set of CNN, and let Y be the corresponding RUL values. In a Bayesian neural network, the goal is to predict the posterior distribution $p\left(y|x,X,Y\right)$ of the RUL y belonging to a test sample x, given the training samples X and Y:

$$p\left(y|x,X,Y\right)=\int p\left(y|x,\omega\right)p\left(\omega|X,Y\right)d\omega, \tag{7.6}$$

where ω denotes all the weights in the neural network. Here, $p\left(y|x,\omega\right)$ is the probability that the RUL equals y, given test sample x and the weights of the neural network ω. Moreover, $p\left(\omega|X,Y\right)$ is the posterior distribution of the weights, and denotes the probability that the weights are ω, given the training samples X and Y.

It usually is computationally very expensive to analyse the posterior distribution $p\left(\omega|X,Y\right)$ exactly. In variational inference, the posterior distribution $p\left(\omega|X,Y\right)$ is therefore approximated with a distribution $q\left(\omega\right)^*$ instead. Here, we first define a family (i.e., set) Q of possible posterior distributions $q\left(\omega\right)$. The goal is then to find the distribution $q\left(\omega\right)^*\in Q$ that minimizes Kullback-Leibler divergence KL with the true posterior distribution $p\left(\omega|X,Y\right)$ [30]:

$$q\left(\omega\right)^*=\text{arg min}_{q(\omega)\in Q}\left\{KL\left(q\left(\omega\right)|p\left(\omega|X,Y\right)\right)\right\}. \tag{7.7}$$

This is equivalent to finding the distribution $q\left(\omega\right)^*\in Q$ that maximizes the evidence lower bound (ELBO) $\mathcal{L}_{\text{ELBO}}$ [29, 30]:

$$\mathcal{L}_{\text{ELBO}}=\int q\left(\omega\right)\log\left(p\left(Y|X,\omega\right)\right)d\omega-KL\left(q\left(\omega\right)|p\left(\omega\right)\right), \tag{7.8}$$

where $p\left(\omega\right)$ is the prior of the weights. We assume that the prior is the standard multivariate normal distribution. Using $q\left(\omega\right)^*$, we approximate the posterior distribution $p\left(y|x,X,Y\right)$ of the RUL of a test sample by:

$$q\left(y|x\right)=\int p\left(y|x,\omega\right)q\left(\omega\right)^{*}d\omega, \tag{7.9}$$

where $q\left(y|x\right)$ is the approximation of $p\left(y|x,X,Y\right)$.

In [29], the family Q of possible distributions $q(\omega)$ is defined as all Gaussian mixture distributions with two components. The authors of [29] show that this mixture can be approximated by setting $q(\omega)$ in each layer i as:

$$\omega_i=\omega_i^{\text{original}}\cdot\text{diag}\left(\left[\theta_{i,j}\right]_{j=1}^{R_i}\right), \tag{7.10}$$

$$\theta_{ij}\sim\text{Bernoulli}\left(1-\rho\right), j=1,\ldots,R_i \tag{7.11}$$

where ω_i are the weights of layer i, R_i is the number of nodes in layer i, and $\omega_i^{\text{original}}$ are the weights of layer i without dropout. Moreover, $\text{diag}(z)$ denotes the diagonal matrix constructed with a vector z and θ_{ij} is zero when node j of layer i is dropped out, and one if not. Here, we minimize eq. (7.8) by changing the weights ω of the neural network. With this family Q, the authors of [29] obtain the following estimator of minus the evidence lower bound $-\mathcal{L}_{\text{ELBO}}$ [29]:

$$\hat{\mathcal{L}}_{\text{MC}}=\frac{1}{T}\sum_{i=1}^{T}\left(y_i-\hat{y}_i\right)^2+\lambda\sum_{i=1}^{T}||\omega_i||_2^2, \tag{7.12}$$

with T the number of training samples, y_i and $\hat{y}_i$ the actual and estimated RUL of training sample i respectively, and λ a weight decay parameter. This is the same objective as minimized when training a neural network, i.e., to maximize the evidence lower bound we can simply train the neural network with dropout to minimize the standard loss function.

With this result, we approximate the expected value $\hat{y}$ of the RUL of a test sample:

$$\hat{y}=E_{q(y|x)}\left(y\right)=\frac{1}{M}\sum_{j=1}^{M}\hat{y}_j\left(x,\omega^j\right), \tag{7.13}$$

where M is the number of forward passes through the neural network, ω^j are the weights of the neural network belonging to the j^{th} forward pass (i.e., where some neurons are dropped out), and $\hat{y}\left(x,\omega^j\right)$ is the resulting RUL prognostic from the j^{th} forward pass through the neural network. For the PDF of the RUL, we give each individual RUL prognostic $\hat{y}_j\left(x,\omega^j\right)$ a probability of $\frac{1}{M}$.

7.3. RESULTS - PROBABILISTIC RUL PROGNOSTICS FOR AIRCRAFT TURBOFAN ENGINES

In this section, we present the probabilistic RUL prognostics for turbofan engines.

7.3.1. Hyperparameter tuning

The considered hyperparameters of the neural network are in Table 7.2, optimized with as starting point the hyperparameters of the CNN in [6, 26]. In contrast with these papers, the window size equals $N = 30$ for all four data subsets. For data subsets FD002 and FD004, however, some test instances do not have 30 historical flight cycles. For these test instances we apply zero padding [31], i.e., we set all the sensor measurements of the missing flight cycles to zero. This technique is very common in image processing. Moreover, we use a piece-wise linear RUL target function [26, 32, 33] with $R_{\text{early}} = 125$ flight cycles, i.e, the target RUL is $R_{\text{early}} = 125$ flight cycles when the actual RUL is larger than 125 flight cycles.

Hyperparameter	**Value**
Hyperparameters - architecture	
Window-size N	30
Convolutional layers L	5
Number of filters K	10
Kernel size S	10
Kernel size S' last convolutional layer	3
Number of nodes fully connected layer	100
Monte Carlo dropout rate ρ	0.5
Number of passes M	1000
R_{early}	125
Hyperparameters - optimization	
Optimizer	Adam [34]
Number of epochs	250
Training-Validation split	80%-20%
Initial learning rate	0.001
Decrease learning rate when no improvement in validation loss for ... epochs in a row	10
Decrease learning rate by	$\frac{1}{2}$

Table 7.2.: Considered hyperparameters of the CNN.

7.3.2. Mean estimated RUL

Table 7.3 shows the Root Mean Square Error (RMSE) [35], which is calculated with the mean estimated RUL (see eq. (7.13)), for the four data subsets. The RMSE of the mean estimated RUL is higher for subsets FD002 and FD004 than for subsets FD001 and FD003, probably due to the multiple operating modes.

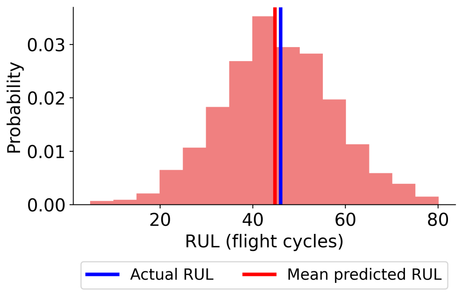
Table 7.3 also shows the results of existing studies employing various machine learning algorithms for the same dataset. The performance of our RUL prognostic method with respect to the mean estimated RUL is comparable to the state-of-the art solutions, especially for data subset FD002 and FD004. Potential contributing factors to the good performance of our approach are that we consider a larger window size N for FD002 and FD004 than most existing studies, and that we normalize the measurements with respect to the operating conditions (see eq. (7.1)).

	FD001	FD002	FD003	FD004
Our approach	12.42	**13.72**	12.16	15.95
CNN [26]	12.61	22.36	12.64	23.31
LSTM-MLSA [33]	11.57	14.02	12.13	17.21
CNN-LSTM [36]	**11.17**	-	**9.99**	-
HAGCN [37]	11.93	15.05	11.53	**15.74**
HDNN [38]	13.02	15.24	12.22	18.17
MPHD-NN [39]	-	14.25	-	16.44

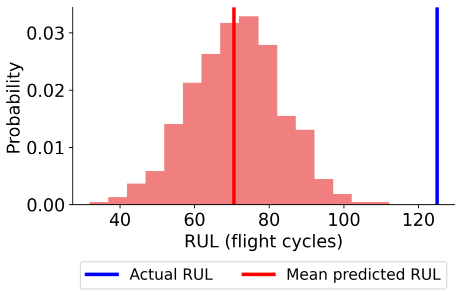
Table 7.3.: RMSE (in flights) for the RUL prognostics using C-MAPSS and various machine learning algorithms. Here, $R_{\text{early}} = 130$ in [37], and $R_{\text{early}} = 125$ in the other considered studies. The best results are denoted in bold.

7.3.3. PDF of the RUL prognostics

Instead of predicting only one number for the RUL, however, we predict the PDF of the RUL. Figure 7.4 shows the PDF of the RUL for two test instances of data subset FD004. For test instance 67 (Figure 7.4a), the mean estimated RUL is close to the actual RUL. The PDF, however, is very wide. For test instance 38 (Figure 7.4b), the mean estimated RUL is far away from the actual RUL. Moreover, the actual RUL falls outside the estimated PDF, even though this PDF is very wide as well.



(a) Test instance 67, data subset FD004 of C-MAPSS.



(b) Test instance 38, data subset FD004 of C-MAPSS.

Figure 7.4.: Histogram of the estimated RUL of two test instances.

The RMSE only evaluates the mean estimated RUL. We therefore use the α-Coverage and the reliability diagram to evaluate the reliability of the PDF of the RUL, i.e., how well the estimated probabilities match with the observed outcomes

[13, 40]. The α-Coverage is defined as [13]:

$$\alpha\text{-Coverage} = \frac{1}{D}\sum_{i=1}^{D}\mathcal{I}^{\alpha}\left(y_i\right), \tag{7.14}$$

$$\text{with } \mathcal{I}^{\alpha}(y_i) = \begin{cases} 1, & y_i \in \left[\hat{y}_i^{0.5-0.5\alpha}, \hat{y}_i^{0.5+0.5\alpha}\right] \\ 0, & \text{Otherwise,} \end{cases}$$

where $\alpha \in [0,1]$ is an user-defined parameter, $\hat{y}_i^k$ is the k^{th} percentile of the estimated RUL distribution for test instance i, and D is the number test instances. $[\hat{y}_i^{0.5-0.5\alpha}, \hat{y}_i^{0.5-0.5\alpha}]$ is thus the α percent confidence interval around the median of the PDF of test instance i. The closer the α-Coverage is to α, the more reliable the RUL prognostics are. Related to this metric is the α-Mean width, which is the mean width in flight cycles of the confidence intervals at α [13, 40]:

$$\alpha\text{-Mean width} = \frac{1}{D}\sum_{i=1}^{D}\left(\hat{y}_i^{0.5+0.5\alpha} - \hat{y}_i^{0.5-0.5\alpha}\right). \tag{7.15}$$

		FD001	FD002	FD003	FD004
$\alpha = 0.50$	α-Coverage	0.54	0.51	0.57	0.52
	α-Mean width	16.3	15.2	16.7	16.8
$\alpha = 0.90$	α-Coverage	0.91	0.85	0.92	0.85
	α-Mean width	39.2	36.1	40.3	40.1
$\alpha = 0.95$	α-Coverage	0.95	0.89	0.97	0.90
	α-Mean width	46.4	42.4	47.6	47.3

Table 7.4.: α-Coverage and α-Mean width (in flight cycles) for the RUL prognostics of the engines in the C-MAPSS datasets.

Table 7.4 shows the α-Coverage for $\alpha \in \{0.5, 0.9, 0.95\}$. As an example, we illustrate the $\alpha = 0.9$ coverage. With $\alpha = 0.9$, we obtain the confidence interval $[\hat{y}_i^{0.05}, \hat{y}_i^{0.95}]$, where $\hat{y}_i^{0.05}$ and $\hat{y}_i^{0.95}$ are the RUL prognostics belonging to the 5%th and the 95%th percentile respectively. If we estimate the RUL and the corresponding 90% confidence interval for test instance i a large number of times, we thus expect that $\alpha = 90\%$ of the resulting confidence intervals contain the actual RUL. To calculate the α-Coverage, we construct this confidence interval with width $\alpha = 0.9$ for all D test instances of each data subset, and count how often the true RUL falls within the confidence interval. It is expected that for $\alpha = 90\%$ of the test instances, the actual RUL y_i falls within the considered confidence interval. For data subset FD003, the actual RUL y_i falls within the confidence interval for 92% of the test instances. This means that the *uncertainty* for data subset FD003 and $\alpha = 0.9$ is slightly *overestimated.* In contrast, for data subset FD002 and FD004, the α-Coverage equals 85%, i.e., the actual RUL y_i of 85% of the test instances falls within the considered confidence interval. This means that the *uncertainty* for data subsets FD002 and FD004 and $\alpha = 0.9$ is *underestimated.*

In general, the α-Coverage is close to α for all subsets and all the considered values for α (see Table 7.4). This means that the estimated probabilities match well with the observed outcomes, and the RUL prognostics are thus reliable. However, the mean widths of the confidence interval are quite large. The uncertainty of the RUL prognostics is thus large, despite the reliability of the RUL prognostics.

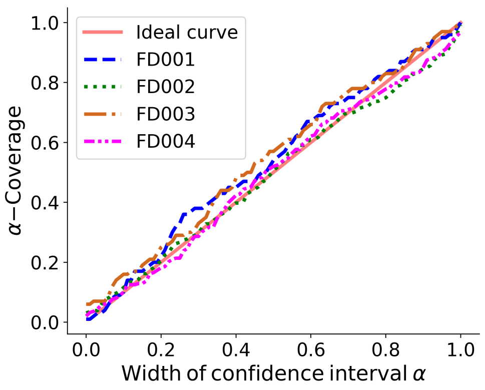


Figure 7.5.: Reliability diagram for the four subsets of C-MAPSS.

Figure 7.5 shows the reliability diagram of the four subsets of C-MAPSS [13]. Here, $C(\alpha)_i = \{\alpha\text{-Coverage}, \alpha \in \{0.00, 0.01, 0.02, \ldots, 1.00\}\}$ is the reliability curve of subset $i \in$ {FD001, FD002, FD003, FD004}. Moreover $\tilde{C}(\alpha) = \{\alpha, \alpha \in \{0.00, 0.01, \ldots, 1.00\}\}$ is the ideal curve, i.e., the curve where the coverage $C(\alpha) = \alpha$. When the reliability curve is beneath the ideal curve for a certain α, then the uncertainty is underestimated at this value for α. In contrast, when the reliability curve is above the ideal curve for a certain α, the uncertainty is overestimated at this value for α. Figure 7.5 shows that the reliability curves of all four data subsets are close to the ideal curve. Thus, the uncertainty of the RUL prognostics is well estimated.

7.4. Methodology - Maintenance scheduling

In this section, we propose a model to find the optimal replacement moment of a component based on the probabilistic RUL prognostics and the expected costs associated with maintenance. For the maintenance planning of a single component, we pose the problem of identifying an optimal replacement time as a renewal-reward process. For the maintenance planning of multiple components, we propose an integer linear program that additionally takes into account the availability of maintenance slots and the capacity of these slots.

7.4.1. Single-component maintenance planning

We find the optimal moment to replace a single component with a renewal-reward process. Let $\{N(t), t \geq 0\}$ be a renewal process where the process regenerates when a component is replaced [41]. Let C_n be the cost incurred during the n^{th} renewal cycle, due to a replacement of the component, and let L_n be the length of the n^{th} cycle, i.e., the time between the n^{th} and the $(n^{\text{th}}-1)$ replacement. In our case, an n^{th} component is thus used for a L_n amount of time. Defining $C(t)$ as the cumulative cost incurred up to time t, the renewal-reward theorem states that the long-term average cost per unit of time ($\lim_{t\to\infty}\frac{C(t)}{t}$) is equivalent to [41]:

$$\lim_{t\to\infty}\frac{C(t)}{t}=\frac{\mathbb{E}[C_1]}{\mathbb{E}[L_1]}.$$

To determine an optimal replacement time that minimizes the long-term average cost per unit of time, we can thus simply minimize the costs incurred during one cycle, over the length of one cycle:

$$\frac{\mathbb{E}(\text{cost incurred during one cycle})}{\mathbb{E}(\text{length of one cycle})}. \tag{7.16}$$

We assume that at some present moment, a component (aircraft engine) has been used for k time steps. At this present moment k, an optimal replacement moment that minimizes eq. (7.16) is determined. If a preventive replacement is scheduled at $k+t_k$ time steps (i.e., in t_k time steps from the present moment), and the component does not fail until time $k+t_k$, then a cost c_{r} is incurred for this preventive replacement. Here, the risk is that the component may fail at some time $k+j, 0\leq j<t_k$. If the component indeed fails after j time steps, than a cost $c_{\mathrm{f}}>c_{\mathrm{r}}$ is incurred and this component is immediately replaced by a new one. The component is thus either i) replaced upon failure at a cost c_{f}, or ii) is preventively replaced at a cost c_{r} after using it for $k+t_k$ time-steps.

The component is continuously monitored by sensors. The sensor measurements of the first k time steps of usage are available. In Section 7.2, $x_i^v, i\in\{1,2,\ldots,k\}$ denotes this series of measurements up to time step k of an engine v. Based on these sensor measurements, the probability that the RUL of the component is i time steps, $i\geq 0$, is estimated using a CNN with Monte Carlo dropout (see Section 7.2). Let $\phi_k(i)$ denote the probability that, after being used for k time steps, the component has a RUL of exactly i time steps, $i\geq 0$.

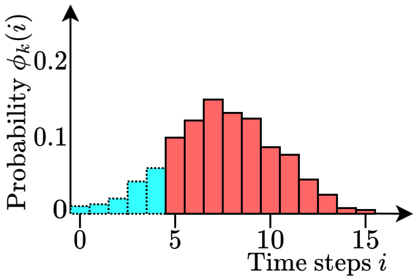


Figure 7.6.: An example of the probability $\phi_k(i)$ for a single component, as estimated at time $k=100$.

Let $C(k,t_k)$ denote the replacement costs of the component and let $L(k,t_k)$ denote the lifetime of the component, given that this component has already been used for k time steps. Here, the component is replaced i) upon failure, or ii) preventively after being used for $k+t_k$ time-steps. We are interested in identifying the optimal value of t_k, i.e., an optimal time to replace the component, which minimizes the long-term average cost per unit of time (see eq. (7.16)):

$$\frac{\mathbb{E}[C(k,t_k)]}{\mathbb{E}[L(k,t_k)]}, \tag{7.17}$$

where the expected replacement cost of the component is:

$$\mathbb{E}[C(k,t_k)] = c_{\mathrm{f}} \sum_{i=0}^{t_k-1} \phi_k(i) + c_{\mathrm{r}}\left(1 - \sum_{i=0}^{t_k-1} \phi_k(i)\right), \tag{7.18}$$

and the expected lifetime of the component is:

$$\mathbb{E}[L(k,t_k)] = k + \sum_{i=0}^{t_k-1} i\cdot\phi_k(i) + t_k\left(1 - \sum_{i=0}^{t_k-1} \phi_k(i)\right). \tag{7.19}$$

Let t_k^* denote the optimal value for t_k, that minimizes eq. (7.17). In general, if there is a high probability that the RUL is zero, then t_k^* is also expected to tend to zero. Conversely, if there is a high probability that the RUL is large, then t_k^* is also expected to be relatively large.

Figure 7.6 shows an example of the probability $\phi_k(i)$ of a component of having a RUL of i time steps, given that it has already been used for $k=100$ time steps. We evaluate the expected costs over the expected lifetime if $t_k=5$, i.e., if we replace the component at day $k+t_k=105$. The probability that the component fails in the next $t_k=5$ days equals $\sum_{i=0}^{4}\phi_k(i)=0.14$ (blue, dotted area in Figure 7.6). The expected cost when replacing the component at day 105 is $c_{\mathrm{f}}\cdot 0.14 + c_{\mathrm{r}}(1-0.14)$. The expected lifetime is $100+0.41+5\cdot(1-0.14)=104.71$.

7.4.2. MULTI-COMPONENT MAINTENANCE PLANNING

We now consider the maintenance planning for multiple components. Let V denote the set of components (aircraft engines). Let d_{p} denote the present day, and d_0^v denote the installation day of a component $v\in V$. Let $k_{\mathrm{p}} = d_{\mathrm{p}} - d_0^v$ denote the usage time of component $v\in V$ at present day d_{p}.

At present day d_{p}, a probabilistic RUL prognostic, i.e., the estimated PDF of the RUL, is available for each component $v\in V$ (see Section 7.2). Let $\phi_{k_{\mathrm{p}}}^v(i)$ denote the estimated probability that the RUL of component $v\in V$ is exactly i flight cycles, after being used for k_{p} flight cycles at the present day d_p. Again, $\phi_{k_{\mathrm{p}}}^v(i)$ is estimated using CNN with Monte Carlo dropout in Section 7.2. We assume that each engine performs one flight cycle per day.

A component can be replaced in dedicated maintenance slots, i.e. days when the component is available for maintenance, and the maintenance facility and required equipment are available. In the case of an aircraft engine, the aircraft is on the ground during these maintenance slots, and the aircraft maintenance hangar and equipment are available [6]. At present day d_p, maintenance slots are known up to l days in advance. Let S^v be the set of maintenance slots available for a component $v \in V$ in the period $[d_\text{p}, d_\text{p} + l)$. The set S^v is specific to component v.

We also consider generic maintenance slots, i.e., a day during which any component $v \in V$ can be replaced, but at an additional high cost c_g. One generic maintenance slot is available per day. Let S_g be the set of generic maintenance slots available in the period $[d_\text{p},\ d_\text{p} + l)$.

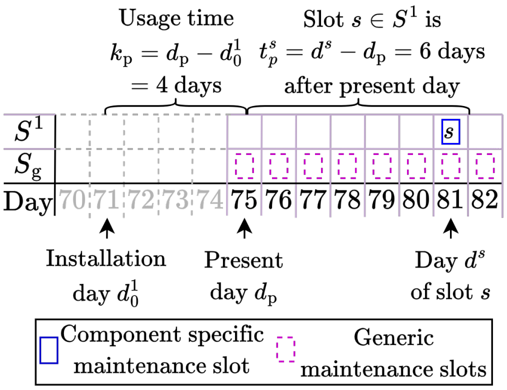
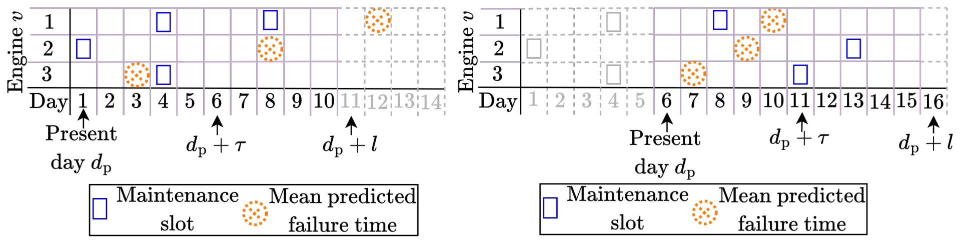


Figure 7.7.: Example of the notation in the multi-component maintenance planning problem for a single engine $v = 1$.

Let d^s be the day belonging to slot $s, s \in S_\text{g} \cup S^v, v \in V$, and let S^d be the set of all slots at day $d \in [d_\text{p}, d_\text{p} + l)$, i.e., all $s \in S_\text{g} \cup S^v, v \in V : d^s = d$. Let $t_\text{p}^s = d^s - d_\text{p}$ denote the number of days a maintenance slot s is available after present day d_p. Last, due to limited resources, at most h components can be replaced per day. Figure 7.7 shows an example of the notation in the multi-component maintenance planning problem for a single engine $v = 1$.

We analyse the maintenance planning for a period of T days using a rolling horizon approach. First, at present day $d_\text{p} = 1$, the maintenance planning is made for the time-window $[d_\text{p}, d_\text{p} + l)$. Here, the RUL prognostics are obtained at present day $d_p = 1$. Then, the replacements planned in the first $\tau, \tau < l$ days of this maintenance planning are executed. It is assumed that components that fail in the first τ days are immediately replaced. Next, we roll to the next time window by updating the present day $d_\text{p} := 1 + \tau$ and by updating the RUL prognostics. With these new RUL prognostics, a maintenance planning is now made for the time-window $[d_\text{p}, d_\text{p} + l)$, i.e., for $[1 + \tau, 1 + \tau + l)$. This process is repeated until the maintenance planning of the full T days is executed.

An example of two iterations of a rolling horizon approach is in Figure 7.8, with $|V| = 3$ engines. At the first present day $d_\text{p} = 1$, we know the maintenance slots of the $l = 10$ subsequent days (day $d_p + l$ is not included). Using the RUL prognostics, we thus make a maintenance planning for the next $l = 10$ days (Figure 7.8a). Then,



(a) First iteration of the rolling horizon approach, present day $d_p = 1$.
(b) Second iteration of the rolling horizon approach, present day $d_p = 6$.

Figure 7.8.: Schematic example of two iterations of the rolling horizon approach, with $|V| = 3$ engines, $\tau = 5$ days, $l = 10$ days, and present days $d_p = 1$ and $d_p = 6$.

we fix the decisions of the first $\tau = 5$ days, update the RUL prognostics and move forward to present day $d_p = 1 + \tau = 6$ (Figure 7.8b). Note that the mean estimated failure time changes between present day $d_p = 1$ and $d_p = 6$, because the RUL prognostics are updated in each new time-window.

MAITNENANCE COSTS

Let c_s^v denote the expected costs over the expected lifetime of the component (see eq. (7.17)) when component v is replaced in slot s, with $d_p \leq d^s < d_p + l$. We calculate c_s^v by dividing the expected costs of this replacement by the expected lifetime, given that component v is replaced i) upon failure or ii) preventively in slot s, whichever comes first. The expected costs c_s^v consist of the expected failure costs, the expected preventive replacement costs, and the expected costs of using the generic slot. Formally,

$$c_s^v = \frac{c_f \sum_{i=0}^{t_p^s - 1} \phi_{k_p}^v(i) + \left(c_r + c_g \mathcal{I}_g(s)\right)\left(1 - \sum_{i=0}^{t_p^s - 1} \phi_{k_p}^v(i)\right)}{(d_p - d_0^v) + \sum_{i=0}^{t_p^s - 1} i\phi_{k_p}^v(i) + t_p^s\left(1 - \sum_{i=0}^{t_s^p - 1} \phi_{k_p}^v(i)\right)}, \tag{7.20}$$

where

$$\mathcal{I}_g(s) = \begin{cases} 1 & s \in S_g \\ 0 & \text{Other}, \end{cases}. \tag{7.21}$$

Let c_{DN}^v denote the expected cost over the expected lifetime of component $v \in V$ within the period $[d_p, d_p + l)$, when no replacement is planned for this component in this period, i.e., when we do nothing (DN). In other words, the replacement of this component is postponed to after day $d_p + l$, and we thus only incur costs in the period $[d_p, d_p + l)$ if the component fails in this period. The costs, *only* considering the period $[d_\mathrm{p}, d_\mathrm{p}+l)$ for both the expected costs and the expected life, are:

$$c^v_{\mathrm{DN}} = \frac{c_\mathrm{f} \sum_{i=0}^{l-1} \phi^v_{k_\mathrm{p}}(i)}{(d_\mathrm{p} - d^v_0) + \sum_{i=0}^{l-1} i\phi^v_{k_\mathrm{p}}(i) + l\left(1 - \sum_{i=0}^{l-1} \phi^v_{k_\mathrm{p}}(i)\right)}. \tag{7.22}$$

With these costs, we propose an Integer Linear Program (ILP) to schedule the component replacements at present day d_p for the time-window $[d_\mathrm{p}, d_\mathrm{p}+l)$.

DECISION VARIABLES

We consider the following decision variable:

$$x_{vs} = \begin{cases} 1, & \text{Component } v \in V \text{ is replaced in slot } s \in S^v \cup S_\mathrm{g} \\ 0, & \text{Otherwise.} \end{cases} \tag{7.23}$$

OBJECTIVE

We aim to minimize the expected costs over the expected lifetime, i.e.,:

$$\text{min.} \sum_{v\in V} \left(\sum_{s\in S^v \cup S_\mathrm{g}} c^v_s x_{vs} + c^v_{\mathrm{DN}} \left(1 - \sum_{s\in S^v \cup S_\mathrm{g}} x_{vs}\right)\right). \tag{7.24}$$

CONSTRAINTS

A component $v \in V$ can be scheduled for replacement at most once in a time-window:

$$\sum_{s\in S^v \cup S_\mathrm{g}} x_{vs} \leq 1, \qquad \forall v \in V \tag{7.25}$$

At most h components can be scheduled for replacement during one day, due to the limited maintenance capacity and limited resources:

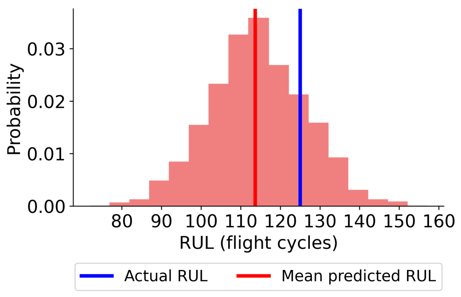
$$\sum_{v\in V} \sum_{s\in S^d} x_{vs} \leq h, \qquad \forall d \in \left[d_\mathrm{p}, d_\mathrm{p}+l\right) \tag{7.26}$$

Last, we consider a binary variable:

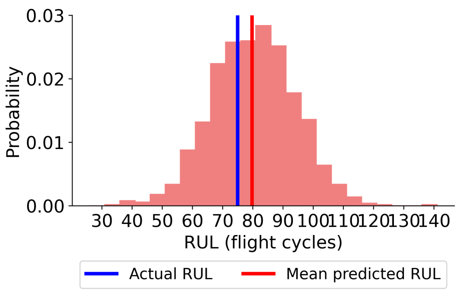
$$x_{vs} \in \{0,1\} \qquad \forall v \in V, \forall s \in \left(S^v \cup S_\mathrm{g}\right). \tag{7.27}$$

7.5. RESULTS - MAINTENANCE PLANNING FOR TURBOFAN ENGINES

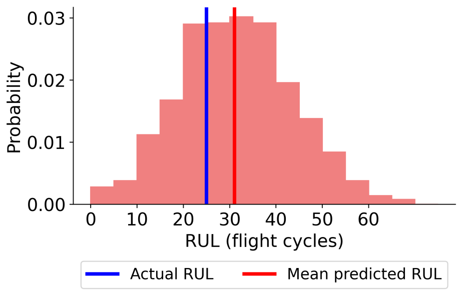
In this section, we illustrate our maintenance planning methodology proposed in Section 7.4 for the aircraft turbofan engines.



(a) Actual RUL = 125 flight cycles.



(b) Actual RUL = 75 flight cycles.



(c) Actual RUL = 25 flight cycles.

Figure 7.9.: Predicted PDF of the RUL of engine 2 of subset FD001, C-MAPSS. This is the first engine selected from subset FD001 for maintenance planning.

7.5.1. Probabilistic RUL prognostics for the maintenance planning

In Section 7.3 we have presented the RUL prognostics for the engines in the C-MAPSS test sets. For these test engines, the measurements stop at some moment before failure, i.e., these are not complete series until the moment of failure. To analyse the predictive maintenance planning, however, we need complete series of sensor measurements up to the moment of failure, i.e., we need run-to-failure instances. The engines in the C-MAPSS training set have such complete run-to-failure series of measurements. Thus, we use the engines in the C-MAPSS training set for maintenance planning. We first randomly select 80% of the engines of each training set (568 engines in total) [6, 42] to train the CNNs (see Section 7.2). For the remaining 20% of engines (a total of 141 engines), we estimate the PDF of the RUL after each flight using the trained CNNs and Monte Carlo dropout. The RUL prognostics of these 141 engines are then used to analyze the maintenance planning model proposed in Section 7.4.

Figure 7.9 shows the obtained PDF of the RUL of engine 2 of subset FD001 when the actual RUL is 125, 75 and 25 flight cycles. Engine 2 is the first engine randomly selected from FD001 for maintenance planning. The PDF of the RUL of this engine

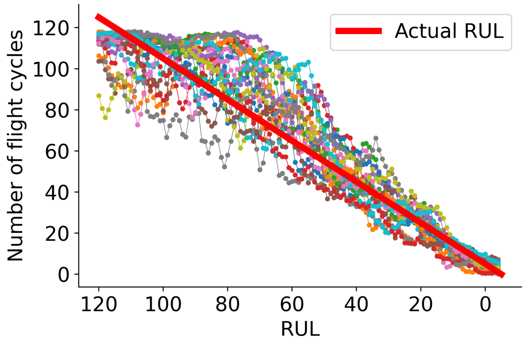


Figure 7.10.: The mean estimated RUL for the last 125 flight cycles for the engines selected for maintenance planning from subset FD001, C-MAPSS.

is centered around the actual RUL for all three time moments. However, the mean estimated RUL is closer to the actual RUL when the actual RUL is 75 or 25 flight cycles, than when the actual RUL is 125 flight cycles.

Figure 7.10 shows the mean estimated RUL for the last 125 flight cycles before failure for all engines selected for maintenance planning from FD001. Here, each colored line shows the RUL prognostics belonging to one engine. After each flight, the mean estimated RUL is updated for each engine. For all engines, the mean estimated RUL slightly underestimates the actual RUL when the actual RUL is 125 flight cycles. For a few engines, moreover, the mean estimated RUL deviates substantially from the actual RUL when the actual RUL is still large. However, the mean estimated RUL converges to the actual RUL when an engine becomes closer to failure.

		FD001	FD002	FD003	FD004
	RMSE	13.06	15.15	13.58	15.93
$\alpha = 0.50$	α-Coverage	0.50	0.47	0.60	0.60
	α-Mean width	16.3	15.5	16.9	16.5
$\alpha = 0.90$	α-Coverage	0.90	0.82	0.89	0.88
	α-Mean width	39.2	37.5	40.7	39.9
$\alpha = 0.95$	α-Coverage	0.94	0.88	0.93	0.91
	α-Mean width	46.4	44.4	48.1	47.2

Table 7.5.: The RMSE (with the mean estimated RULs), α-Coverage and α-Mean width for the RUL prognostics of the engines selected for maintenance planning from the C-MAPSS training sets.

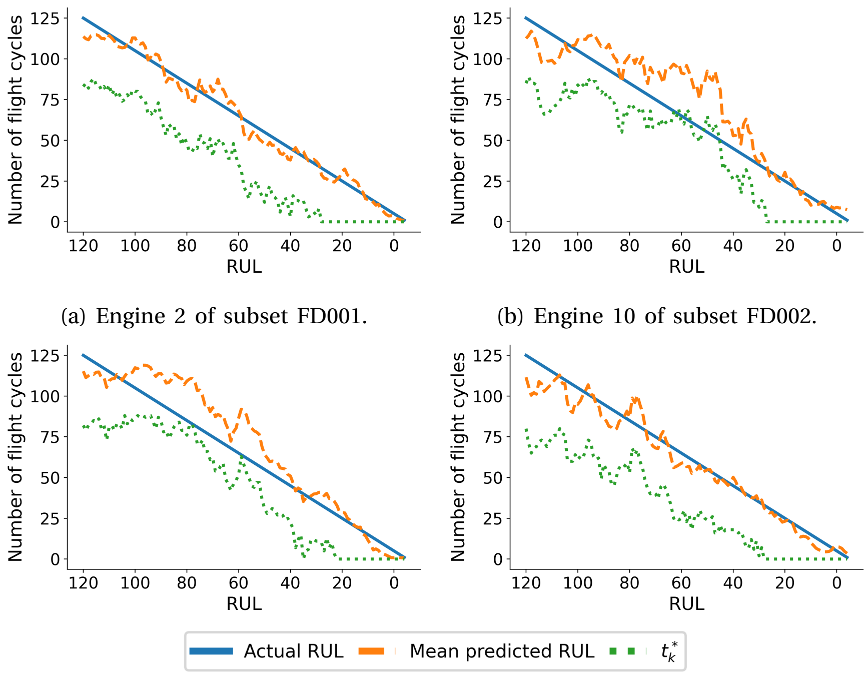
Table 7.5 shows the metrics of the RUL prognostics with all the 141 engines selected for maintenance planning. The RMSE is higher than for the test instances in Section 7.2, while the α-Coverage diverges more from α. This is as expected, since there are less failure instances available to train the CNN.

7.5.2. SINGLE-ENGINE REPLACEMENT PLANNING

Actual RUL	k	Mean estimated RUL	99% CI of the RUL	t_k^*	$k+t_k^*$
		Engine 2 - subset FD001			
		True lifetime = 287 flight cycles			
125	162	113.7	[83,144]	83	245
100	187	99.3	[66, 128]	66	253
75	212	79.7	[40,114]	40	252
50	237	45.5	[14,75]	13	250
25	262	31	[0,64]	0	262
		Engine 10 of subset FD002			
		True lifetime = 184 flight cycles			
125	59	112.5	[82, 141]	85	144
100	84	114.2	[85, 146]	85	169
75	109	91.6	[55, 127]	58	167
50	134	81.1	[49, 114]	48	182
25	159	30.4	[0,60]	0	159
		Engine 2 of subset FD003			
		True lifetime = 253 flight cycles			
125	128	115.3	[83, 146]	82	210
100	153	118.4	[89,151]	88	241
75	178	92.9	[67,125]	62	240
50	203	59.3	[27,92]	27	230
25	228	28.4	[0,57]	0	227
		Engine 3 of subset FD004			
		True lifetime = 307 flight cycles			
125	182	111.5	[81, 144]	80	262
100	207	101.2	[67,135]	72	279
75	232	68.1	[40,100]	40	272
50	257	50.4	[23,79]	21	278
25	282	23.2	[0,53]	0	282

Table 7.6.: The actual RUL, the number of cycles the engine has already been in use k, the mean estimated RUL, the 99% confidence interval (CI) of the estimated RUL, the optimal number of flight cycles to use the engine before preventive replacement t_k^*, and the optimal replacement moment of the engine $k+t_k^*$. The first engines randomly selected for maintenance planning from FD001, FD002, FD003, FD004 are chosen for illustration.

In this section, we discuss the optimal moment of replacement $k+t_k^*$ from Section 7.4.1, eq. (7.17). We consider a preventive replacement cost $c_r = 10$ and a failure cost $c_f = 50$. Table 7.6 shows for four engines the optimal moment for replacement $k+t_k^*$, at five moments during the life of each engine. Engine 2, 10, 2, and 3 is the first randomly selected engine for maintenance planning from C-MAPPS subset FD001, FD002, FD003, and FD004, respectively. As an example, consider engine 2 of subset



(a) Engine 2 of subset FD001.

(b) Engine 10 of subset FD002.

(c) Engine 2 of subset FD003.

(d) Engine 3 of subset FD004.

Figure 7.11.: Actual RUL, mean estimated RUL and t_k^*. Engine 2, 10, 2 and 3 are the first randomly selected engines for maintenance planning from FD001, FD002, FD003 and FD004, respectively.

FD001 when the actual RUL is 125 flight cycles. At this moment, the engine has been used for $k = 162$ flight cycles. The mean estimated RUL is 113.7 flight cycles, and the 99% confidence interval of the estimated RUL is $[83, 144]$. Given the current usage of 162 flight cycles, it is optimal to replace this engine after an additional of $t_k^* = 83$ flight cycles, i.e., at the lower bound of the 99% confidence interval of the estimated RUL. The optimal moment of a preventive replacement of this engine is thus $k + t_k^* = 162 + 83 = 245$ flight cycles.

The optimal moment of replacement $k + t_k^*$ varies over time, since the RUL prognostics are updated after each flight cycle. In Table 7.6, t_k^* is close to the lower bound of the 99% confidence interval of the RUL prognostic. Here, t_k^* is thus smaller if the RUL prognostics are more uncertain, i.e., if the confidence intervals are wider. On the other hand, the more certain the RUL prognostics are, the closer t_k^* is to the mean estimated RUL. Figure 7.11 further illustrates the optimal number of flight cycles t_k^* for the engines in Table 7.6. For all four engines, t_k^* follows the same trend as the mean estimated RUL. Moreover, when the actual RUL is 25 flight cycles or less, it is optimal to immediately perform a preventive replacement for all four engines, i.e., $t_k^* = 0$. This is because, at this moment, we estimate with a quite high probability that the RUL of the engine is 0 flight cycles (see the 99% confidence intervals of the RUL in Table 7.6).

For engine 10 of subset FD002 (Figure 7.11b), the mean estimated RUL is larger than the actual RUL, when the actual RUL is around 60 flight cycles. Based on these prognostics, equation (7.17) is minimized when the engine is replaced after the failure time. For example, the mean estimated RUL is 93 flight cycles, with a 100% confidence interval of $[62, 123]$, while the actual RUL is only 55 flight cycles. The estimated probability that the engine fails on or before the actual RUL is thus zero. The optimal replacement moment, based on the RUL prognostics, is therefore $t_k^* = 65$ flight cycles, i.e., after the failure time of the engine. However, due to the updating of the RUL prognostics, it is again optimal to replace engine 10 of FD002 before the failure time around 40 flight cycles before failure onwards.

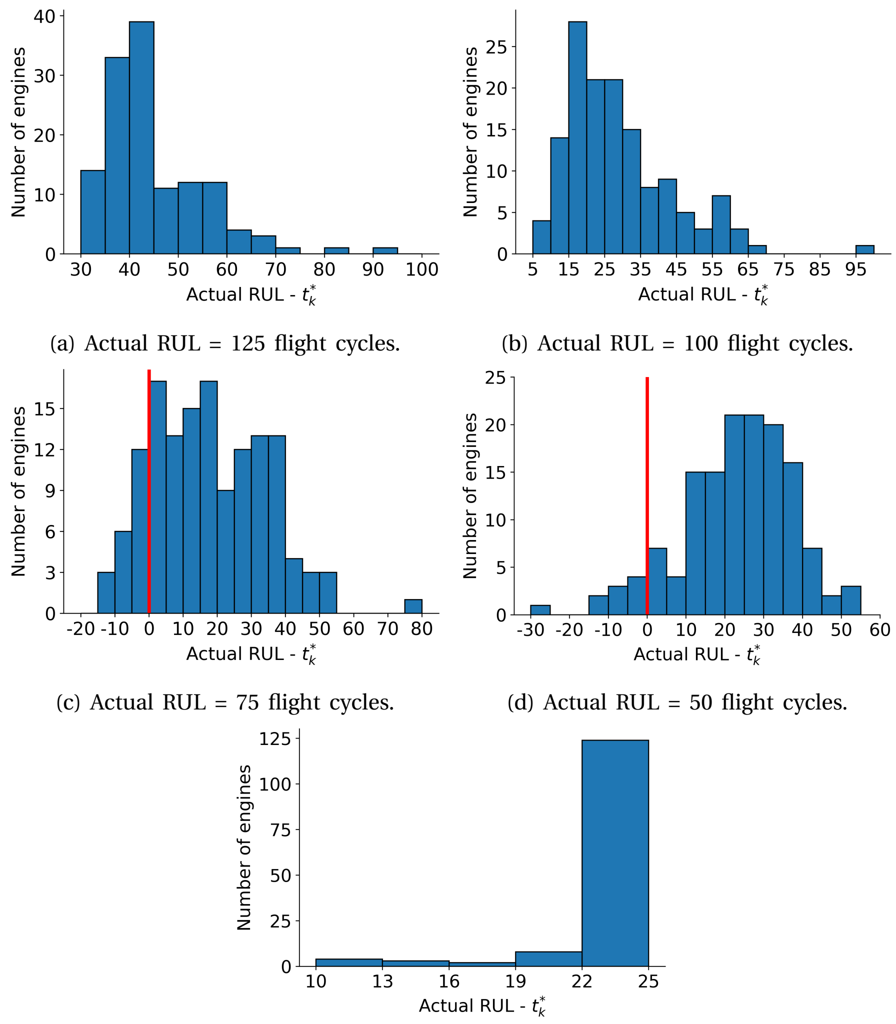
Figure 7.12 shows the actual $\text{RUL} - t_k^*$ for all 141 engines considered for maintenance planning, at five moments during the engines' life. As an example, for engine 2 of subset FD001 in Table 7.6, it is optimal to replace the engine after $t_k^* = 83$ flight cycles, while the actual RUL is 125 flight cycles. The actual $\text{RUL} - t_k^* = 125 - 83 = 42$ flight cycles, i.e., the optimal moment of replacement is 42 flight cycles before engine failure.

When the actual RUL is 125 cycles or when the actual RUL is 100 cycles, then the optimal moment of replacement is always before the engine fails. When the actual RUL is 125 cycles, it is optimal to replace each engine at least 30 cycles before its failure. This is because the RUL is slightly underestimated when the actual RUL is 125 cycles (see Figure 7.10), and the lower bounds of the 95% and 99% confidence intervals are always much smaller than 125 flight cycles. When the actual RUL is 75 or 50 flight cycles, however, the RUL prognostics sometimes overestimate the actual RUL: For some engines, the estimated probability that the RUL is equal to or smaller than the actual RUL is even zero. The optimal moment of replacement for some engines therefore falls after the engine failure date when the actual RUL is 75 or 50 flight cycles. When the actual RUL is 25 flight cycles (Figure 7.12e), however, it is optimal to immediately perform a preventive replacement for most engines, i.e., $t_k^* = 0$. Moreover, 25 flight cycles before failure, the optimal maintenance moment is at least 10 days before the engine fails.

7.5.3. MULTI-ENGINE REPLACEMENT PLANNING

In this section, we plan maintenance for multiple engines using the methodology in Section 7.4.2. We consider a fleet of $|V| = 50$ engines, which are randomly selected from the 141 engines considered for maintenance. Maintenance slots are randomly sampled with a frequency of one per 10-20 days [6]. We assume that at most $h = 1$ engine per day can be maintained, and that maintenance slots are known $l = 50$ days ahead. We consider a preventive replacement cost $c_\text{r} = 10$, a cost of using a generic slot $c_\text{g} = 10$ and a cost of a failure are $c_\text{f} = 50$. As mentioned before, we also assume that each engine performs one flight cycle per day.

We analyse the maintenance planning of the engines for a period of $T = 10 \times 365$ days (i.e., ten years). We plan maintenance $l = 50$ days ahead, where we fix the maintenance during the first $\tau = 10$ days. Last, we assume that a failed engine is



(a) Actual RUL = 125 flight cycles.

(b) Actual RUL = 100 flight cycles.

(c) Actual RUL = 75 flight cycles.

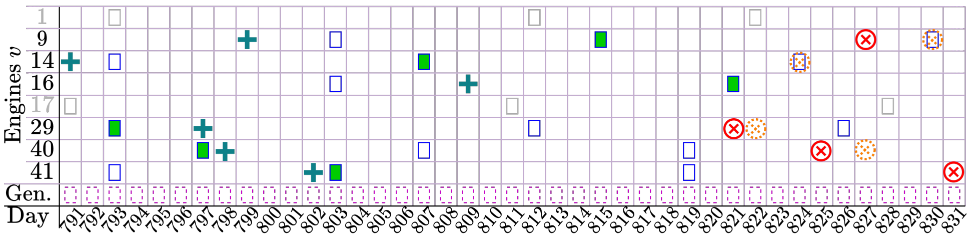
(d) Actual RUL = 50 flight cycles.

(e) Actual RUL = 25 flight cycles.

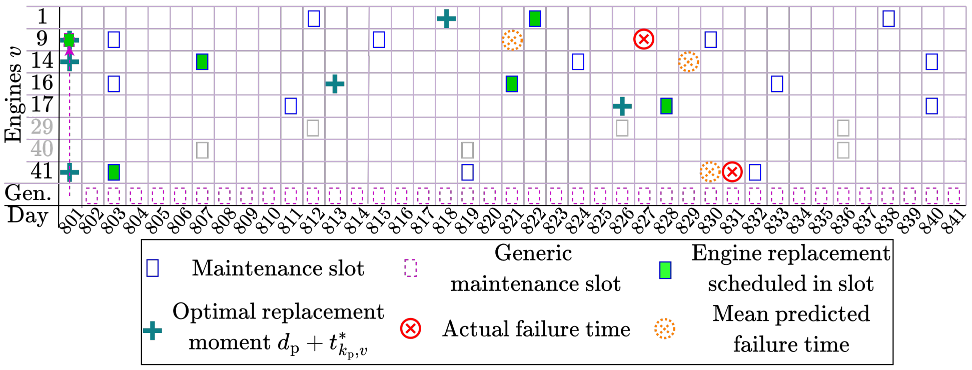
Figure 7.12.: Histogram of (actual RUL - t_k^*), for all engines selected for maintenance planning from the C-MAPSS datasets.

immediately replaced with a new engine. We implement the ILP in Python using Gurobi, on a computer with an Intel Core i7 processor at 2.11 GHz and 8Gb RAM. It requires 109 seconds to create a maintenance planning for a period of ten years.

Figure 7.13 shows the resulting maintenance planning at present days $d_\mathrm{p} = 791$ and $d_\mathrm{p} = 801$. At day 791, a replacement is planned for 6 engines, namely engines 9, 14, 16, 29, 40 and 41. For each engine with a planned replacement, we show the available maintenance slots (blue squares), the actual failure time (red cross in a



(a) Present day $d_{\mathrm{p}} = 791$. Not depicted is that engine 14 fails at day 843, that engine 16 fails at day 854, while the mean estimated failure time is at day 845, and that the mean estimated failure time of engine 41 is at day 836.



(b) Present day $d_{\mathrm{p}} = 801$. Not depicted is that engine 1 fails at day 846, while the mean estimated failure time is at day 850, that engine 14 fails at day 843, that engine 16 fails at day 854, while the mean estimated failure time is at day 844, and that engine 17 fails at day 845, while the mean estimated failure time is at day 852.

Figure 7.13.: Maintenance planning at present day $d_{\mathrm{p}} = 791$ and day $d_{\mathrm{p}} = 801$. Only the engines for which a replacement is planned during these two present days are depicted.

circle) and the mean estimated failure time (orange cross in a dotted circle). For an engine $v \in V$ at present day d_{p}, we use $t^*_{k_{\mathrm{p}},v}$ to denote the optimal t^*_k from eq. (7.17) in Section 7.4.1. Here, $d_{\mathrm{p}} + t^*_{k_{\mathrm{p}},v}$ thus denotes the optimal replacement moment from the renewal-reward process with a single engine (blue cross).

For engine 29, 40 and 41, the engine replacement is planned close to the optimal moment from the renewal-reward process. For engines 29 and 40, the replacement is planned 4 and 1 day(s) before this optimal moment, respectively. For engine 41, the replacement is planned 1 day after this optimal moment. For engine 9, 14 and 16, however, the replacement is planned 16, 16 and 12 days after the optimal moment from the single-engine problem, respectively. For all these three engines, an earlier maintenance slot closer to the single-engine optimal replacement moment is available. However, engine 29 or engine 41 is already replaced at the days of these maintenance slots. Since we assume that at most one engine per day can be maintained, we cannot maintain engine 9, 14 and 16 at these days as well.

Using the rolling horizon approach, the maintenance decisions of the first $\tau = 10$ days of the maintenance planning are executed. Engine 29 and engine 40 are replaced at day 793 and day 797 respectively, and no engine failure occurs. The RUL prognostics are updated, and a new maintenance planning is then made at day 801.

At day 801, the replacements for engine 14, 16 and 41 remain planned in the same maintenance slots. The updated prognostics of engine 9, however, indicate that this engine could fail very soon: At present day 791, the first available maintenance slot that fulfilled the capacity constraint was at day 815. The probability that engine 9 would fail before day 815 was estimated at 7.7%. The costs of using a generic slot (10) thus exceeded the expected failure costs ($c_f \cdot 0.077 = 50 \cdot 0.077 = 7.7$). The replacement was therefore scheduled at day 815. At present day 801, however, the estimated probability that engine 9 fails before day 815 is 26.5%. The expected failure costs ($50 \cdot 0.265 = 13.25$) are thus higher than the cost of using a generic slot. A generic slot (purple, dotted square) is therefore used to replace this engine immediately at day 801. Additionally, a replacement is planned for engine 1 at day 822, four days after the optimal moment for a single engine, and for engine 17 at day 829, two days after the optimal moment for a single engine.

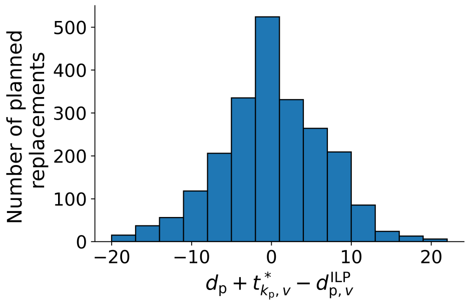


Figure 7.14.: The optimum replacement time $d_p + t^*_{k_p,v}$ for single engine replacement (see eq. (7.17)) minus the scheduled replacement time $d^{\text{ILP}}_{p,v}$ in the multi-engine replacement problem.

Let $d^{\text{ILP}}_{p,v}$ be the day of a scheduled replacement of engine $v \in V$, as planned at present day d_p using the ILP for multi-engine maintenance planning. Figure 7.14 gives a histogram of the optimal moment of replacement $d_p + t^*_{k_p,v}$ for the case of a single engine, minus the scheduled replacement time $d^{\text{ILP}}_{p,v}$ from the multi-engine problem. For example, in Figure 7.13a, it is optimal at present day $d_p = 791$ to replace engine $v = 41$ at day 802, i.e. $d_p + t^*_{k_p,v} = 791 + 11 = 802$. However, it is planned to replace engine $v = 41$ at day $d^{\text{ILP}}_{p,v} = 803$ instead. The replacement of engine 41 is thus scheduled $802 - 803 = -1$, i.e., one day day after the optimal moment.

Figure 7.14 shows that the results obtained for the maintenance of multiple engines are similar to the results for single-engine maintenance: Most of the time, the optimal replacement moment of the single-engine problem is close to the optimal replacement moment of the multi-engine problem. The optimal replacement time obtained from the multi-engine problem, however, can also deviate up to 20 days from the optimal replacement time obtained from the single-engine problem due to the limited availability of slots and the limit of h replacements per day.

7.5.4. Long-term performance of different maintenance strategies

In this section, we compare three maintenance strategies: i) the proposed maintenance strategy with probabilistic RUL prognostics (Section 7.4), ii) a maintenance strategy with perfect RUL prognostics, and iii) a time-based maintenance strategy without RUL prognostics. The long-term performance of these maintenance strategies is analyzed by means of a Monte Carlo simulation.

Perfect RUL prognostics For this maintenance strategy, we assume that we exactly know the failure time of the engine without any uncertainty, i.e., that we have perfect RUL prognostics. For this strategy, we use the same ILP as in Section 7.4.2. However, we now input the actual RUL in eq. (7.20) and eq. (7.22), instead of the estimated probabilistic RUL prognostics.

Time-based maintenance For this strategy, we determine the probability of the failure of an engine based on its current usage. In Section 7.5.1, we selected 568 engines to train the CNN, while the remaining 141 engines were used for maintenance planning. For the time-based maintenance strategy, we use the 568 engines to determine a *generic* histogram of the lifetime of engines (see Figure 7.15).

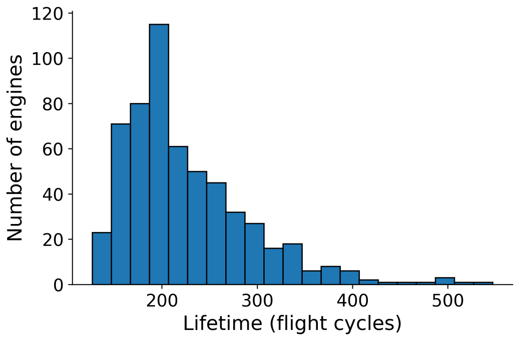


Figure 7.15.: Histogram of the lifetime of the 568 historical failure instances selected for training the CNN in Section 7.5.1.

For any of the 141 engines selected for maintenance planning, the estimated probability that the lifetime of this engine is i flight cycles at the moment of installation, is the proportion of the 568 engines for which the lifetime is i flight cycles. For example, 2 out of the 568 engines have a lifetime of 231 flight cycles. When planning maintenance, we therefore estimate that, at the moment of installation, the probability that the lifetime of an engine is 231 flight cycles is $\frac{2}{568}$. Let $\tilde{\phi}(i)$ denote the probability that the lifetime of an engine is exactly i flight cycles. In the considered example, $\tilde{\phi}(231) = \frac{2}{568} \approx 0.0035$.

Consider that any engine of the 141 engines has been used for k flight cycles. The lifetime L of this engine is thus larger than or equal to k cycles, i.e., $L \geq k$. Given that $L \geq k$, the conditional probability $\tilde{\phi}(k+i|L \geq k)$ that the lifetime $L = k+i$ flight cycles is solely based on the histogram in Figure 7.15:

$$\tilde{\phi}(k+i|L \geq k) = \frac{\tilde{\phi}(L = k+i \cap L \geq k)}{\tilde{\phi}(L \geq k)} \tag{7.28}$$

$$= \frac{\tilde{\phi}(k+i)}{1 - \sum_{i=0}^{k-1} \tilde{\phi}(i)} \tag{7.29}$$

The difference between Section 7.5.1 and the time-based maintenance strategy is that under the time-based maintenance strategy, the probability of failure for *all* 141 engines is estimated to be the *same, given an usage period of k cycles.* The sensor measurements are not considered in calculating the probability of failures.

As another example, assume that an engine has been used for $k = 220$ flight cycles. The probability $1 - \sum_{i=0}^{219} \tilde{\phi}(i)$ that the lifetime of an engine is 220 flight cycles or larger is 0.4296. The conditional probability that the lifetime L of this engine equals 231 flight cycles is thus $\tilde{\phi}(231|L \geq 220) = \frac{0.0035}{0.4296} = 0.008$. Using these conditional probabilities $\tilde{\phi}(k+i|L \geq k)$, we plan the single-component and multi-component replacements with the same method as before.

Time-based single-component replacement

Now that $\tilde{\phi}(k+i|L \geq k)$ is determined based on the histogram in Figure 7.15, eq. (7.18) with the expected costs and (7.19) with the expected lifetime become:

$$\mathbb{E}[C(k,t_k)] = c_\mathrm{f} \sum_{i=0}^{t_k-1} \tilde{\phi}(k+i|L \geq k) + c_\mathrm{r}\left(1 - \sum_{i=0}^{t_k-1} \tilde{\phi}(k+i|L \geq k)\right),$$

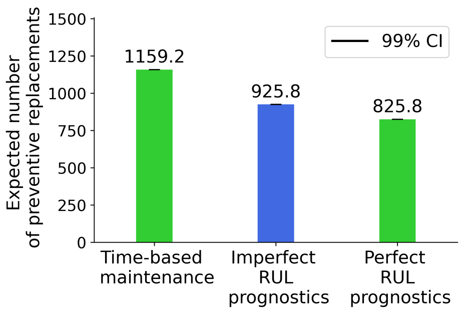
$$\mathbb{E}[L(k,t_k)] = k + \sum_{i=0}^{t_k-1} i \cdot \tilde{\phi}(k+i|L \geq k) + t_k\left(1 - \sum_{i=0}^{t_k-1} \tilde{\phi}(k+i|L \geq k)\right).$$

Time-based multi-component replacement

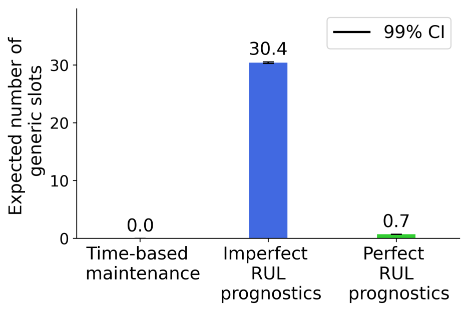
We again plan the multi-component replacement using the ILP in Section 7.4.2. Let $k_\mathrm{p}^v = d_\mathrm{p} - d_0^v$ be the number of days engine $v \in V$ is in use at present day d_p.

Now that $\tilde{\phi}(k+i|L \geq k)$ is determined based on the histogram in Figure 7.15, the expected costs of planning to replace an engine v in slot s (eq. (7.20)) become:

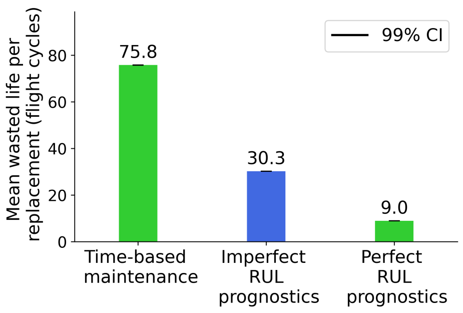
$$c_s^v = \frac{c_\mathrm{f} \sum_{i=0}^{t_\mathrm{p}^s-1} \tilde{\phi}(k_\mathrm{p}^v + i|L \geq k_\mathrm{p}^v) + \left(c_\mathrm{r} + c_\mathrm{g}\mathcal{I}_\mathrm{g}(s)\right) \cdot \left(1 - \sum_{i=0}^{t_\mathrm{p}^s-1} \tilde{\phi}(k_\mathrm{p}^v + i|L \geq k_\mathrm{p}^v)\right)}{k_\mathrm{p}^v + \sum_{i=0}^{t_\mathrm{p}^s-1} i \cdot \tilde{\phi}(k_\mathrm{p}^v + i|L \geq k_\mathrm{p}^v) + t_\mathrm{p}^s\left(1 - \sum_{i=0}^{t_\mathrm{p}^s-1} \tilde{\phi}(k_\mathrm{p}^v + i|L \geq k_\mathrm{p}^v)\right)}. \tag{7.30}$$



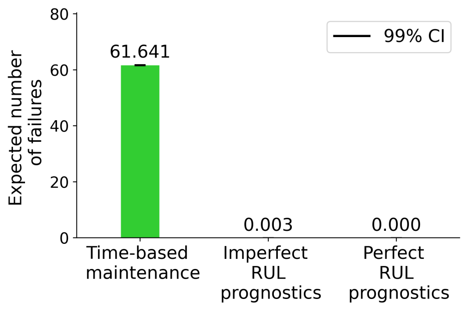
(a) Expected number of preventive engine replacements.



(b) Expected number of used generic slots.



(c) Mean wasted life per replacement.



(d) Expected number of failures.

Figure 7.16.: The expected number of engine replacements, used generic slots and failures per ten years, and the mean wasted life per replacement, for i) time-based maintenance, ii) maintenance with probabilistic, imperfect RUL prognostics and iii) maintenance with perfect RUL prognostics. The 99% confidence interval of the mean (CI) is also given.

while the expected costs of doing nothing become:

$$c_{\mathrm{DN}}^{v} = \frac{c_{\mathrm{f}} \sum_{i=0}^{l-1} \tilde{\phi}(k_{\mathrm{p}}^{v} + i | L \geq k_{\mathrm{p}}^{v})}{k_{\mathrm{p}}^{v} + \sum_{i=0}^{l-1} i \cdot \tilde{\phi}(k_{\mathrm{p}}^{v} + i | L \geq k_{\mathrm{p}}^{v}) + l\left(1 - \sum_{i=0}^{l-1} \tilde{\phi}(k_{\mathrm{p}}^{v} + i | L \geq k_{\mathrm{p}}^{v})\right)}.$$

Long-term results We evaluate the long-term performance of the three maintenance strategies by performing a Monte Carlo simulation with 10,000 simulation runs, where each run lasts 10 years (see Figure 7.16).

The expected number of replacements decreases by 32% when considering imperfect RUL prognostics instead of time-based maintenance. When using perfect instead of imperfect RUL prognostics, the expected number of replacements further decreases by 11% (see Figure 7.16a). This difference is because the mean wasted life per replacement is highest (75.8 flights) when using time-based maintenance, and lowest when considering perfect RUL prognostics (9.0 flights, see Figure 7.16c).

However, when using imperfect RUL prognostics, wasting the life of the engines prevents many engine failures: Only 26 engines fail in the 10000 Monte Carlo simulations of ten years. When using perfect RUL prognostics, the exact failure time is known, and no engine thus fails in any simulation. With time-based maintenance, on average 62 engines fail per ten years, even though an engine is replaced on average 75.8 flight cycles before failure (see Figure 7.16d).

Last, we expect to use 30.4 generic slots when considering imperfect RUL prognostics, while we expect to use less than one generic slot with perfect RUL prognostics or time-based maintenance (see Figure 7.16b). This is because imperfect RUL prognostics are updated over time. The estimated failure time of an engine may therefore suddenly decrease when new sensor measurements become available. In this case, it is sometimes more cost efficient to replace an engine in a generic slot than to risk a failure. In contrast, the prognostics for perfect prognostics and time-based maintenance do not suddenly change.

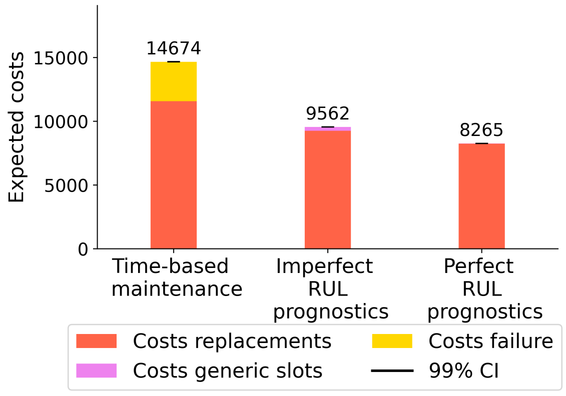


Figure 7.17.: The expected costs over a period of ten year, for i) time-based maintenance, ii) maintenance with probabilistic, imperfect RUL prognostics and iii) maintenance with perfect RUL prognostics. The 99% confidence interval of the mean (CI) is also given.

Figure 7.17 shows the expected maintenance cost per ten years for the three maintenance strategies. For all three strategies, most costs come from replacing the engines. When considering time-based maintenance, moreover, the expected costs of engine failures are 3082. In contrast, these costs are negligible when considering perfect or imperfect RUL prognostics. With imperfect RUL prognostics, however, the expected costs of using generic slots are 304, while these costs are negligible for the other two strategies. Overall, the costs decrease with 53% when considering imperfect RUL prognostics instead of time-based maintenance. Moreover, the maintenance costs further decrease by 14% when considering perfect RUL prognostics.

7.6. CONCLUSIONS

This chapter proposes an end-to-end framework for predictive maintenance for systems, where the sensor measurements are used to generate probabilistic RUL prognostics, which are integrated in the maintenance planning for single and multiple components/systems. Our proposed approach is illustrated with the C-MAPSS turbofan engines. Probabilistic RUL prognostics (the PDF of the RUL) are obtained using a CNN with Monte Carlo dropout. The resulting probabilistic prognostics are shown to be reliable, with a high α-Coverage for all values of α.

Using these probabilistic RUL prognostics and the renewal-reward process, we determine an optimal moment to replace an engine. The optimal moment of replacement is shown to be close to the lower bound of the 99% confidence interval of the RUL prognostics. The results also show that more uncertain the RUL prognostics are, i.e., the wider the confidence intervals are, the earlier a component is preventively replaced to avoid a failure.

We further plan the replacements of multiple engines using an integer linear programming model that integrates the RUL prognostics. Here, the planning is further constrained by the availability of maintenance slots and the limited number of replacements that can be scheduled per day. In the long-run, we show that our approach leads to low expected number of engine failures. Compared with the ideal case when the true RUL is known exactly in advance (ideal RUL prognostics), our approach leads to only 11% more engine replacements and a cost increase of only 14%. We also analyze the long-term performance of our approach with the long-term performance of a traditional, time-based maintenance strategy. The results show that by using our probabilistic RUL prognostics, the expected number of engine replacements decreases with 32% per year. Moreover, the expected number of failure decreases from over 61 to 0.003 in ten years.

As future work, we aim to further analyze the impact of the costs on the maintenance planning results, and to plan maintenance for different types of (aircraft) systems and components, such as multi-component aircraft systems.

REFERENCES

[1] Mitici, M., de Pater, I., Barros, A., & Zeng, Z. (2023). Dynamic predictive maintenance for multiple components using data-driven probabilistic RUL prognostics: The case of turbofan engines. *Reliability Engineering & System Safety, 234,* Article number: 109199.

[2] Mitici, M., de Pater, I., Zeng, Z., & Barros, A. (2023, September 3-7). Predictive maintenance planning using renewal reward processes and probabilistic RUL prognostics–Analyzing the influence of accuracy and sharpness of prognostics. *Proceedings of the 33st European Safety and Reliability Conference,* Southampon, UK, Pages: 1034–1041.

[3] Badea, V. E., Zamfiroiu, A., & Boncea, R. (2018). Big data in the aerospace industry. *Informatica Economica, 22*(1), Pages: 17–24.

[4] Hu, Y., Miao, X., Si, Y., Pan, E., & Zio, E. (2022). Prognostics and Health Management: A review from the perspectives of design, development and decision. *Reliability Engineering & System Safety, 217,* Article number: 108063.

[5] Alaswad, S., & Xiang, Y. (2017). A review on condition-based maintenance optimization models for stochastically deteriorating system. *Reliability Engineering & System Safety, 157,* Pages: 54–63.

[6] de Pater, I., Reijns, A., & Mitici, M. (2022). Alarm-based predictive maintenance scheduling for aircraft engines with imperfect Remaining Useful Life prognostics. *Reliability Engineering & System Safety, 221,* Article number: 108341.

[7] Lee, J., & Mitici, M. (2022). Deep reinforcement learning for predictive aircraft maintenance using probabilistic Remaining-Useful-Life prognostics. *Reliability Engineering & System Safety, 230,* Article number: 108908.

[8] Shi, Y., Zhu, W., Xiang, Y., & Feng, Q. (2020). Condition-based maintenance optimization for multi-component systems subject to a system reliability requirement. *Reliability Engineering & System Safety, 202,* Article number: 107042.

[9] Keizer, M. C. O., Flapper, S. D. P., & Teunter, R. H. (2017). Condition-based maintenance policies for systems with multiple dependent components: A review. *European Journal of Operational Research, 261*(2), Pages: 405–420.

[10] Ren, L., Zhao, L., Hong, S., Zhao, S., Wang, H., & Zhang, L. (2018). Remaining Useful Life prediction for lithium-ion battery: A deep learning approach. *IEEE Access, 6,* Pages: 50587–50598.

[11] de Pater, I., & Mitici, M. (2023). Developing health indicators and RUL prognostics for systems with few failure instances and varying operating conditions using a LSTM autoencoder. *Engineering Applications of Artificial Intelligence, 117,* Article number: 105582.

[12] Biggio, L., Wieland, A., Chao, M. A., Kastanis, I., & Fink, O. (2021). Uncertainty-aware prognosis via deep Gaussian process. *IEEE Access, 9*, Pages: 123517–123527.

[13] de Pater, I., & Mitici, M. (2022, July 6-8). Novel metrics to evaluate probabilistic Remaining Useful Life prognostics with applications to turbofan engines. *Proceedings of the 7th European Conference of the Prognostics and Health Management (PHM) Society, 7*, Turin, Italy, Pages: 96–109.

[14] Lee, J., & Mitici, M. (2020). An integrated assessment of safety and efficiency of aircraft maintenance strategies using agent-based modelling and stochastic Petri Nets. *Reliability Engineering & System Safety, 202*, Article number: 107052.

[15] Do, P., Assaf, R., Scarf, P., & Iung, B. (2019). Modelling and application of condition-based maintenance for a two-component system with stochastic and economic dependencies. *Reliability Engineering & System Safety, 182*, Pages: 86–97.

[16] Van Horenbeek, A., & Pintelon, L. (2013). A dynamic predictive maintenance policy for complex multi-component systems. *Reliability Engineering & System Safety, 120*, Pages: 39–50.

[17] Zhang, Z., Si, X., Hu, C., & Lei, Y. (2018). Degradation data analysis and Remaining Useful Life estimation: A review on Wiener-process-based methods. *European Journal of Operational Research, 271*(3), Pages: 775–796.

[18] Caballé, N. C., Castro, I. T., Pérez, C. J., & Lanza-Gutiérrez, J. M. (2015). A condition-based maintenance of a dependent degradation-threshold-shock model in a system with multiple degradation processes. *Reliability Engineering & System Safety, 134*, Pages: 98–109.

[19] Schöbi, R., & Chatzi, E. N. (2016). Maintenance planning using continuous-state Partially Observable Markov Decision Processes and non-linear action models. *Structure and Infrastructure Engineering, 12*(8), Pages: 977–994.

[20] Andriotis, C., & Papakonstantinou, K. (2019). Managing engineering systems with large state and action spaces through deep reinforcement learning. *Reliability Engineering & System Safety, 191*, Article number: 106483.

[21] Chen, C., Zhu, Z. H., Shi, J., Lu, N., & Jiang, B. (2021). Dynamic predictive maintenance scheduling using deep learning ensemble for system health prognostics. *IEEE Sensors Journal, 21*(23), Pages: 26878–26891.

[22] de Pater, I., & Mitici, M. (2021). Predictive maintenance for multi-component systems of repairables with Remaining-Useful-Life prognostics and a limited stock of spare components. *Reliability Engineering & System Safety, 214*, Article number: 107761.

[23] Consilvio, A., Di Febbraro, A., & Sacco, N. (2020). A rolling-horizon approach for predictive maintenance planning to reduce the risk of rail service disruptions. *IEEE Transactions on Reliability, 70*(3), Pages: 875–886.

[24] Chao, M. A., Kulkarni, C., Goebel, K., & Fink, O. (2022). Fusing physics-based and deep learning models for prognostics. *Reliability Engineering & System Safety, 217*, Article number: 107961.

[25] Saxena, A., & Goebel, K. (2008). *Turbofan engine degradation simulation data set*, NASA Prognostics Data Repository, NASA Ames Research Center, Moffett Field, California, USA.

[26] Li, X., Ding, Q., & Sun, J.-Q. (2018). Remaining Useful Life estimation in prognostics using deep Convolution Neural Networks. *Reliability Engineering & System Safety*, *172*, Pages: 1–11.

[27] Babu, G. S., Zhao, P., & Li, X.-L. (2016, April 16-19). Deep Convolutional Neural Network based regression approach for estimation of Remaining Useful Life. *Proceedings of the 21st International Conference on Database Systems for Advanced Applications (DASFAA)*, Dallas, Texas, USA, Pages: 214–228.

[28] Wang, B., Lei, Y., Li, N., & Yan, T. (2019). Deep separable Convolutional Network for Remaining Useful Life prediction of machinery. *Mechanical Systems and Signal Processing*, *134*, Article number: 106330.

[29] Gal, Y., & Ghahramani, Z. (2016, June 19-24). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *Proceedings of The 33rd International Conference on Machine Learning*, *48*, New York, New York, USA, Pages: 1050–1059.

[30] Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, *112*(518), Pages: 859–877.

[31] Hashemi, M. (2019). Enlarging smaller images before inputting into Convolutional Neural Network: Zero-padding vs. interpolation. *Journal of Big Data*, *6*(1), Pages: 1–13.

[32] Li, H., Zhao, W., Zhang, Y., & Zio, E. (2020). Remaining Useful Life prediction using multi-scale deep Convolutional Neural Network. *Applied Soft Computing*, *89*, Article number: 106113.

[33] Xia, J., Feng, Y., Lu, C., Fei, C., & Xue, X. (2021). LSTM-based multi-layer self-attention method for Remaining Useful Life estimation of mechanical systems. *Engineering Failure Analysis*, *125*, Article number: 105385.

[34] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[35] Lei, Y., Li, N., Guo, L., Li, N., Yan, T., & Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing*, *104*, Pages: 799–834.

[36] Peng, C., Chen, Y., Chen, Q., Tang, Z., Li, L., & Gui, W. (2021). A Remaining Useful Life prognosis of turbofan engine using temporal and spatial feature fusion. *Sensors*, *21*(2), Article number: 418.

[37] Li, T., Zhao, Z., Sun, C., Yan, R., & Chen, X. (2021). Hierarchical attention graph convolutional network to fuse multi-sensor signals for Remaining Useful Life prediction. *Reliability Engineering & System Safety*, *215*, Article number: 107878.

[38] Al-Dulaimi, A., Zabihi, S., Asif, A., & Mohammadi, A. (2019). A multimodal and hybrid deep neural network model for Remaining Useful Life estimation. *Computers in Industry*, *108*, Pages: 186–196.

[39] Al-Dulaimi, A., Asif, A., & Mohammadi, A. (2020, June 8-10). Multipath parallel hybrid deep neural networks framework for Remaining Useful Life estimation.

IEEE International Conference on Prognostics and Health Management (ICPHM), Detroit, Michigan, USA, Pages: 1–7.

[40] Baraldi, P., Mangili, F., & Zio, E. (2015). A prognostics approach to nuclear component degradation modeling based on Gaussian process regression. *Progress in Nuclear Energy, 78*, Pages: 141–154.

[41] Tijms, H. C. (1986). *Stochastic modelling and analysis: A computational approach*. John Wiley & Sons, Inc.

[42] Nguyen, K. T., & Medjaher, K. (2019). A new dynamic predictive maintenance framework using deep learning for failure prognostics. *Reliability Engineering & System Safety, 188*, Pages: 251–262.

8 MAINTENANCE SCHEDULING WITH PROBABILISTIC RUL PROGNOSTICS AND A LIMITED STOCK OF SPARES

In the previous chapter, we scheduled maintenance for aircraft systems using probabilistic Remaining Useful Life (RUL) prognostics. Here, we did not consider the availability of spare parts, or the individual components and redundancies in the aircraft system. In this chapter, we therefore schedule maintenance for multi-component aircraft systems with a limited number of spare components.

First, we develop probabilistic RUL prognostics, in the form of a Probability Density Function (PDF), using a model-based approach (similar to the approach in Chapter 4). We then schedule maintenance for multiple multi-component systems with these probabilistic RUL prognostics, using an integer linear program and a rolling horizon approach. The maintenance of the multiple systems is linked through the availability of the spare components and the shared maintenance opportunities.

We illustrate our approach for a fleet of aircraft, each equipped with a cooling system consisting of four cooling units. Our predictive maintenance strategy reduces the maintenance costs with 48% relative to a corrective maintenance strategy and with 30% relative to a preventive maintenance strategy.

Parts of this chapter have been published in:

de Pater, I., & Mitici, M. (2021). Predictive maintenance for multi-component systems of repairables with Remaining-Useful-Life prognostics and a limited stock of spare components. *Reliability Engineering & System Safety, 214*, Article number: 107761

de Pater, I., del Mar Carillo Galera, M., & Mitici, M. (2021, September 19-23). Criticality-based predictive maintenance scheduling for aircraft components with a limited stock of spare components. *Proceedings of the 31st European Safety and Reliability Conference*, Angers, France, Pages: 55–62

8.1. Introduction

Aircraft maintenance is key for safe and efficient airline operations, with airlines spending approximately 9% of their total operation costs on Maintenance, Repair and Overhaul, which, in 2018, was estimated to be 69 billion dollars [3]. Striving for cost savings, aircraft maintenance is currently shifting from corrective or preventive maintenance towards predictive maintenance. For predictive maintenance, sensors are continuously monitoring the health of components and systems, algorithms are generating Remaining Useful Life (RUL) prognostics, and maintenance is performed based on these prognostics in anticipation of failures [4].

One of the main challenges of predictive maintenance is to obtain RUL prognostics for systems and components. RUL prognostics support a high exploitation time of the systems and components, while limiting Aircraft-On-Ground events due to unexpected failures. Equally challenging is to integrate RUL prognostics into the aircraft maintenance schedule, while the entire complexity of this process is taken into account: The management of spare components, the availability of maintenance time slots and system reliability requirements.

Most studies focus solely on developing RUL prognostics using either a model-based, a data-driven or a hybrid approach [5]. Model-based approaches are proposed in, for instance, [6, 7]. In [6] a two-factor state-space model of the degradation is used to develop RUL prognostics, with an application to rolling element bearings. In [7], particle filtering is combined with a support vector data description to obtain RUL prognostics for engines. In this chapter, we also propose a model-based approach to obtain RUL prognostics for the cooling units (CUs) in wide-body aircraft. However, our focus does not lie on developing RUL prognostics only, but also on proposing a maintenance scheduling model that integrates such prognostics.

For predictive maintenance scheduling, threshold-based maintenance policies are frequently used [8], i.e., as soon as the degradation of a component exceeds a threshold, a maintenance action is planned [9–15]. The degradation thresholds are determined using Monte Carlo simulation [10, 15, 16], semi-regenerative processes [9, 13], Bayesian networks [11], or heuristics [12, 14].

Other studies use Markov Decision Processes (MDPs) [17, 18] and Partially Observable Markov Decision Processes (POMDPs) [8, 19–21] to optimize the maintenance schedule. In [18], an MDP is formulated for the maintenance optimization of a system subject to both failures due to gradual deterioration and to abrupt, sudden failures. In [19], POMDPs are proposed to model the predictive maintenance schedule, with a focus on civil engineering structures. This methodology is further applied to obtain an optimal maintenance schedule for concrete structures in [20]. Also in [21], a POMDP formulation is proposed to schedule maintenance for civil structures. One of the challenges for using (PO)MDPs is the large computational time [17, 19]. To address this issue, in [19], a point-based algorithm is used to solve the POMDP, while in [8], deep reinforcement learning is applied to optimize the maintenance of steel bridge structures.

Only a few studies, however, develop RUL prognostics models and integrate them in the maintenance schedule. In [22], the RUL of a rolling element bearing is estimated with a feed forward Neural Network. Based on these prognostics, maintenance is planned using a search algorithm. In [23], an exponential model is developed to estimate the RUL of a rolling element bearing, and maintenance is planned just before the estimated failure time. In [24], an exponential model is also used to estimate the RUL of a rolling element bearing. With this, optimal maintenance moments and ordering times of spare components are determined. In [25], the RUL of an aircraft component is estimated using a Short Long-Term Memory Neural Network. This is used to determine optimal times to order new spare parts and plan maintenance as well. In [26], an extended Kalman filter is developed to estimate the crack growth in an airframe of an aircraft. Using these prognostics, maintenance for the airframe is planned. However, all these studies consider the maintenance scheduling for only one system, while in this study we consider the maintenance scheduling for multiple multi-component systems.

In [27], RUL prognostics for an aircraft hydraulic system, consisting of multiple sub-systems (i.e., a multi-component system), are developed using a Kalman filter. With this, a maintenance schedule for a single aircraft is proposed using an exhaustive search strategy. In contrast, we plan maintenance for a fleet of multiple aircraft, i.e., for multiple multi-component systems, that are linked through the availability of spare components and through the shared maintenance opportunities.

Last, but not least, the consideration of spare parts for predictive and condition-based maintenance (with or without integrated RUL prognostics) is crucial. One cannot execute a maintenance replacement without having a spare component to perform the replacement with. Many studies determine an optimal component replacement time and assume that a spare component is always available at these times [22, 23, 28]. Other studies determine optimal times to order one-time-use, non-repairable components [12, 24, 25]. For aircraft, however, many components are repairables, i.e., a failed component is sent to a repair shop to be repaired (overhauling [29]). Ordering repairable components is either expensive and/or it takes a long time to receive these components from the manufacturer. In general, the airlines repair and reuse components or, if really necessary, lease a new component. The lease is ended as soon as an own spare component is repaired. Our approach proposes a predictive maintenance scheduling model for repairables. To the best of our knowledge, this is the first study that considers predictive maintenance scheduling for repairable components of multi-component systems [29]. While this is relevant for aircraft maintenance, a similar approach can also be used for the maintenance scheduling of repairable components for other systems and domains.

In this chapter we propose a rolling horizon maintenance scheduling model for multiple multi-component systems of repairable components. This rolling horizon maintenance scheduling model integrates i) model-based RUL prognostics for the components, ii) the availability of spare components and, iii) available maintenance time slots when an aircraft could be maintained (see Figure 8.1). Moreover, the scheduling model incorporates a reliability constraint for each multi-component system. The RUL prognostics are generated using a model-based approach with a particle filtering algorithm. Over time, as more sensor data become available, these prognostics are updated. The updated RUL prognostics are then used in each time window of the rolling horizon maintenance scheduling model to decide

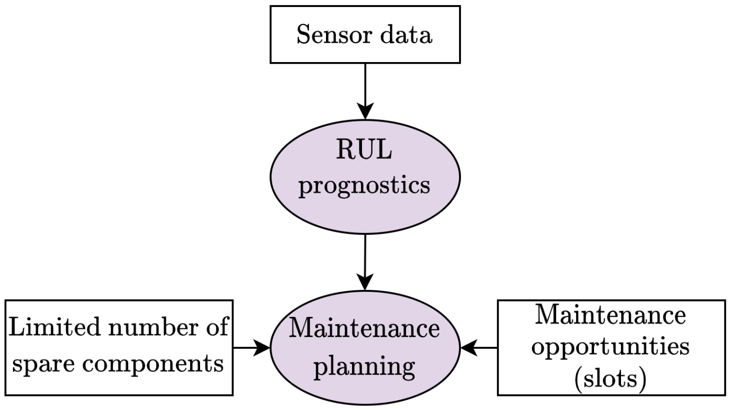

Sensor data
RUL prognostics
Limited number of spare components
Maintenance planning
Maintenance opportunities (slots)

Figure 8.1.: An integrated maintenance scheduling approach with Remaining Useful Life prognostics for components, the management of spare components and fixed maintenance opportunities.

which components to replace. A linear integer program is proposed to solve the maintenance scheduling problem in each time window.

To illustrate our approach, a case study with a fleet of 13 wide-body aircraft, each equipped with a multi-component system of cooling units (CUs), is considered. An optimal maintenance schedule for the CU replacements in a fleet of aircraft is obtained using a rolling horizon approach. The performance of this maintenance schedule in terms of maintenance costs, number of replacements and number of system failures is analyzed. Last, the long-term performance of our prognostic-based maintenance scheduling model is compared against a corrective and a preventive maintenance strategy. The results show that our model outperforms these two strategies with respect to maintenance costs and the number of Aircraft-On-Ground events.

The main contributions of this chapter are as follows:

- An integrated rolling horizon maintenance scheduling model for a *fleet* of aircraft, each equipped with a system of multiple *repairable* components, is developed. This maintenance schedule integrates model-based Remaining Useful Life prognostics and considers the management of a *limited stock of spare* repairable components.

- A realistic maintenance setting is considered, where aircraft maintenance can only be performed during pre-defined time slots, during which the aircraft is on ground and can undergo maintenance.

- The overhauling of repairable components is considered, i.e. a limited total number of spare components is assumed to be available. Upon failure, a component is sent to a repair-shop. Once repaired, the component is returned to the pool of spares components. The overhauling of repairable components has been identified as a research gap in [29].

- A predictive maintenance scheduling model is developed for *multiple multi-component* systems. The maintenance of multiple systems is linked through the availability of spare repairable components and through the shared maintenance opportunities.

The remainder of this chapter is structured as follows. In Section 8.2 we provide the problem description and introduce the main notation. We then develop model-based RUL prognostics for aircraft cooling units in Section 8.3. In Section 8.4 we develop an integrated maintenance scheduling model for a fleet of aircraft, each equipped with a multi-component system of repairable components. This model integrates the RUL prognostics, the management of a limited stock of spare components, and the available maintenance slots. In Section 8.5 we illustrate our model for a fleet of wide-body aircraft, each equipped with a multi-component system of cooling units. The performance of our prognostics-based maintenance scheduling model is compared against a corrective and a preventive maintenance strategy in Section 8.6. Last, Section 8.7 provides conclusions and recommendations for future research.

8.2. Problem description

We consider a discrete time model, where every day d there are decisions made regarding the maintenance schedule of the aircraft. These decisions are based on the Remaining Useful Life prognostics of the aircraft components, the available spare components and the available time slots in which maintenance can be performed.

8.2.1. Multi-component aircraft system

Let A denote a fleet of aircraft. Each aircraft has a system of multiple, identical repairable components. Let C_a, $a \in A$, denote the set of components of this system in aircraft $a \in A$. Each component is assumed to fail independently of the other components. When a component fails, it is replaced with an as-good-as-new one. A replacement can also be triggered by the Remaining Useful Life prognostic of this component, in anticipation of a failure. The installation day of the as-good-as-new component is denoted by d_{ac}^{install}, $a \in A$, $c \in C_a$. At the same time, the removed component is sent for repair.

The aircraft is said to be in an *Aircraft-On-Ground* (AOG) condition and can no longer fly, if this multi-component system fails. A system is considered to be failed when the number of failed components exceeds the number of minimum allowed component failures, as specified by the Minimum Equipment List (MEL) [30].

8.2.2. Maintenance slots

A maintenance slot is a time interval during which maintenance on an aircraft can be performed [26, 27]. Over time, there is a sequence of slots S. Each slot $s \in S$ has a capacity m_s specifying the number of aircraft that can be simultaneously maintained during this slot. There is no limit on the number of components that can be replaced per aircraft within a maintenance slot. For a specific aircraft $a \in A$, the set $S_a \subseteq S$ specifies the slots in which aircraft a can be maintained. A slot s starts during day d_s. The cost of maintaining an aircraft in slot s is c_s.

8.2.3. Repairable components

We plan maintenance for repairable components, i.e., after removal the component undergoes a repair process such that it can be used again instead of being discarded [29, 31]. When a component fails, it is removed from the aircraft while a spare, as-good-as-new component from the stock is installed instead. The faulty component is repaired. This repair takes Δ days. Once repaired, the component is added to the stock. We assume that a component is in an as-good-as-new condition once repaired. There is a limited amount of spare components (limited stock). A component is leased from an external supplier if there are no spares in stock when a component is replaced. We assume that a leased component is immediately available for installment. In the case study, we consider the repairable aircraft cooling units.

There is a fixed cost c^{Lf} for leasing a component. Additionally, a cost c^{Ld} is incurred for every day the component is leased. Last, c^{fix} denotes the cost of repairing a component that is not failed but for which the RUL prognostic indicates a failure in the near-future. If, however, the component is already failed at the time of replacement, then a cost $c^{\text{fix}} + c^{\text{ex}}$ is incurred to repair the component. It is thus more costly to replace a failed component than a non-failed component with a estimated failure in the near-future.

8.2.4. Probabilistic RUL prognostics

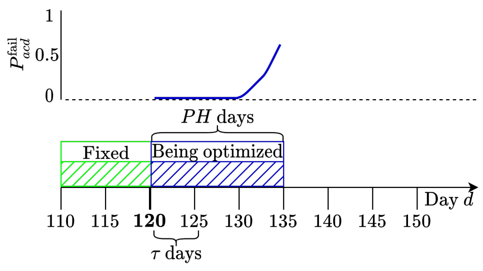
8

Each component $c \in C_a$ of aircraft a is monitored by sensors. Based on the available sensor measurements, at current day d_0, a RUL prognostic for each component is made. Based on these RUL prognostics, we determine P^{fail}_{acd}, the probability that component c of aircraft a fails by the beginning of day $d > d_0$. The RUL prognostic model and P^{fail}_{acd} are discussed in detail in Section 8.3.

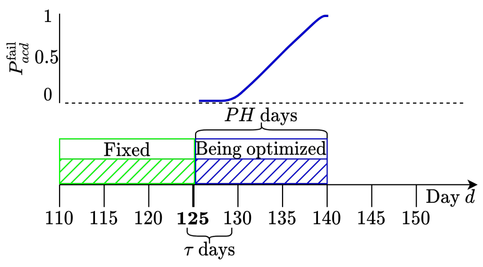
Based on P^{fail}_{acd}, the probability of a system failure at the beginning of day $d > d_0$, or equivalently, the probability of the aircraft being in an AOG-condition at the beginning of day d, denoted by P^{AOG}_{ad}, is determined.

8.2.5. Maintenance scheduling objective

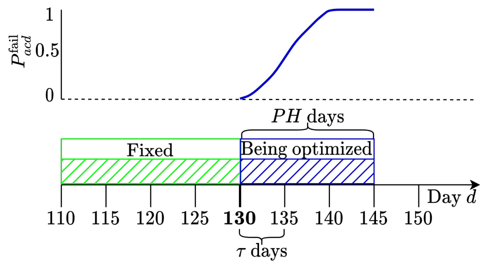
Taking into account i) the maintenance slots available for each aircraft to undergo maintenance, ii) the RUL prognostic of each aircraft component and iii) the available spare components, we are interested in assigning the aircraft to maintenance slots, such that the total cost of the maintenance schedule is minimized. Furthermore, for each aircraft assigned to a maintenance slot it is determined which component/components of this aircraft are replaced.



(a) $d_0 = 120$.



(b) $d_0 = 125$.



(c) $d_0 = 130$.

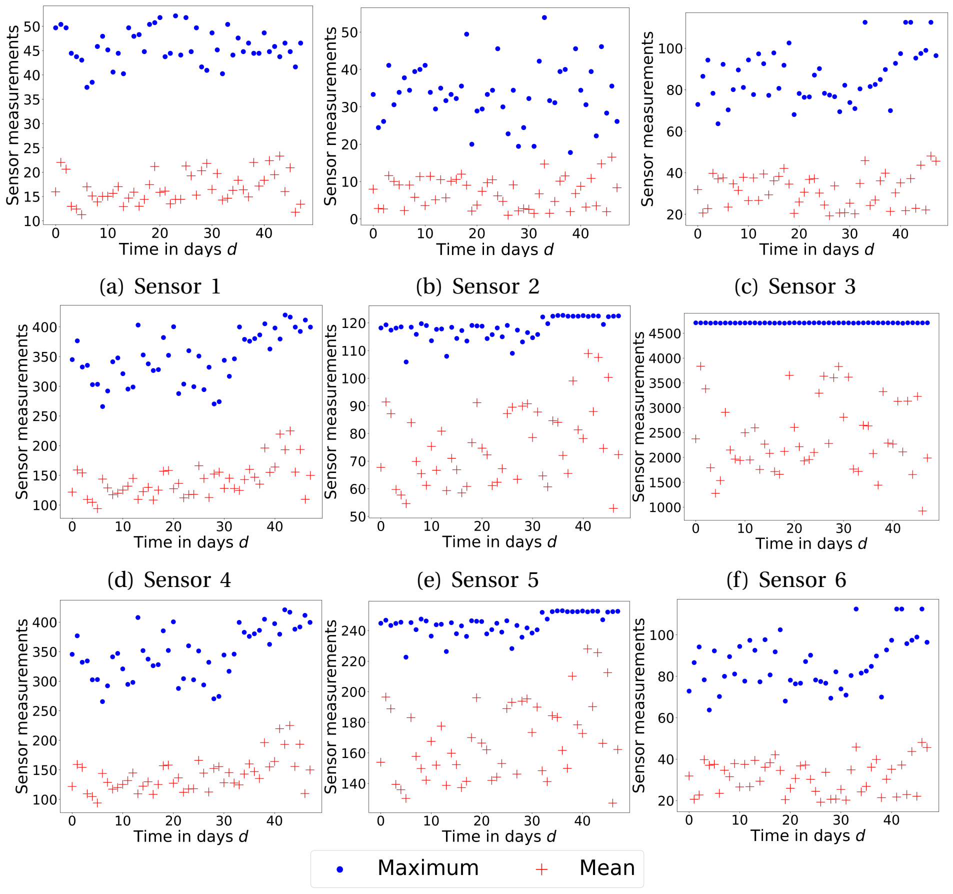
Figure 8.2.: Illustration of the rolling horizon approach and the update of the prognostic information, $\tau = 5$ days, $PH = 15$ days.

8.2.6. ROLLING HORIZON MAINTENANCE SCHEDULING

We determine a maintenance schedule using a rolling horizon approach [14, 32, 33]. In each iteration of the rolling horizon approach, we optimize the maintenance schedule for a time window of PH days, that starts at day d_0. At the beginning of this time window, we have: i) all the maintenance slots available during this time window, given by the set S, ii) the RUL prognostics for each component and for each day $d \in [d_0, d_0 + PH)$ (i.e., P^{fail}_{acd} is specified for each day d within the time window, and for each component $c \in C_a$ of each aircraft $a \in A$), and iii) the number of spare components initially available at the beginning of each day $d \in [d_0, d_0 + PH]$, denoted by S^{begin}_d. If initially, components are leased at the beginning of day d, then S^{begin}_d is negative. For the first time window, a maintenance schedule is created. The decisions of the first τ days of this maintenance schedule are then fixed, and the time window is moved forward τ days. Here, $\tau \leq PH$. Next, a new maintenance schedule is created for this slided time window. This is iterated for several successive time windows, until the end time of the maintenance schedule is reached.

An example of the rolling horizon approach is given in Figure 8.2. Here, there are three iterations of the rolling horizon procedure, with a time window of $PH = 15$ days that moves forward $\tau = 5$ days each iteration. The first iteration (Figure 8.2a) starts at day $d_0 = 120$. All decisions regarding the maintenance schedule before day $d_0 = 120$ are fixed, while the maintenance schedule between day $d_0 = 120$ and day $d_0 + PH = 135$ is under optimization. Then, the decisions of the first $\tau = 5$ days of this maintenance schedule are fixed and the time window is moved $\tau = 5$ days forwards. In the next iteration (Figure 8.2b), the maintenance schedule is optimized between day $d_0 = 125$ and day $d_0 + PH = 140$. This is repeated for the last iteration as well (Figure 8.2c). Also, at the beginning of each iteration the RUL prognostics are updated. This is illustrated for a component $c \in C_a$ of an aircraft $a \in A$.

8.3. Probabilistic RUL prognostics for aircraft cooling units



(a) Sensor 1 (b) Sensor 2 (c) Sensor 3

(d) Sensor 4 (e) Sensor 5 (f) Sensor 6

(g) Sensor 7 (h) Sensor 8 (i) Sensor 9

Figure 8.3.: Mean and maximum sensor measurement per day for one CU for all nine available sensors. This CU fails at day 48.

In this section, we develop model-based Remaining Useful Life prognostics for aircraft cooling units from the sensor measurements.

8.3.1. AIRCRAFT COOLING UNITS (CUS)

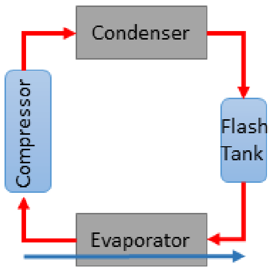


Figure 8.4.: Schematic overview of a cooling unit.

All considered aircraft are equipped with 4 identical cooling units (CUs). The CUs are part of the cooling system. Figure 8.4 shows a schematic overview of one CU, consisting of a condenser, a flash tank, an evaporator and a compressor. Figure 8.5 shows a schematic overview of the cooling system in an aircraft, where there are 4 CUs that are integrated with a Pump, Galley cooling units and Air Heat Exchangers. We assume that each aircraft performs one flight per day.

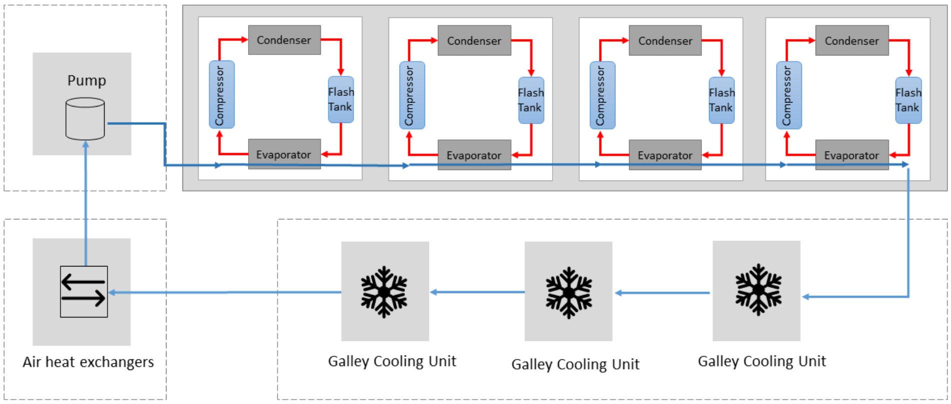


Figure 8.5.: Schematic overview of the cooling system.

8.3.2. HEALTH INDICATOR FOR CUS

As the CU (the aircraft) is increasingly used over time, the filter gets clogged, accelerating the compressor wear, which ultimately leads to a failure. We consider nine sensors monitoring the CUs. Figure 8.3 shows the mean and maximum sensor measurement per day until failure for one CU and for each of the nine available sensors. For the purpose of our analysis, the data sets are anonymized.

Let δ_d denote the flight time during the d^{th} day when this CU is in use, i.e., δ_d is the number of valid sensor measurements larger than a threshold $\varphi = 0$. Let $y^s_{d,b}$ denote the b^{th} valid sensor measurement during day d for this CU, generated by a sensor s. We normalize the measurements during day d as follows:

$$\tilde{y}^s_{d,b} = \frac{y^s_{d,b}}{\max_s - \max_{b \in 1,\ldots,\delta_d}\left(y^s_{d,b}\right)}, \tag{8.1}$$

with $\max_s$ the overall maximum measurement generated by sensor s.

We then define the health indicator m^i_d of CU i at day d as follows, with $n > 1$:

$$m_d = \frac{1}{n}\sum_{j=d-n}^{d}\frac{1}{\delta_j}\sum_{b=1}^{\delta_j}\tilde{y}^s_{j,b}. \tag{8.2}$$

Our health indicator combines the increasing maximum sensor measurement towards failure (see Figure 8.3) and the increasing mean sensor measurement towards failure (see again Figure 8.3), while it is at the same time independent of the length of the flights during a day d. For our analysis, we select for the health indicator the sensor with the largest correlation coefficient with the time to failure [34, 35], which in our case is sensor 8 with a correlation coefficient of 0.77. Figure 8.6 shows the health indicator obtained 30 days before failure for 5 CUs. For all CUs, the increase in the health indicator accelerates towards failure.

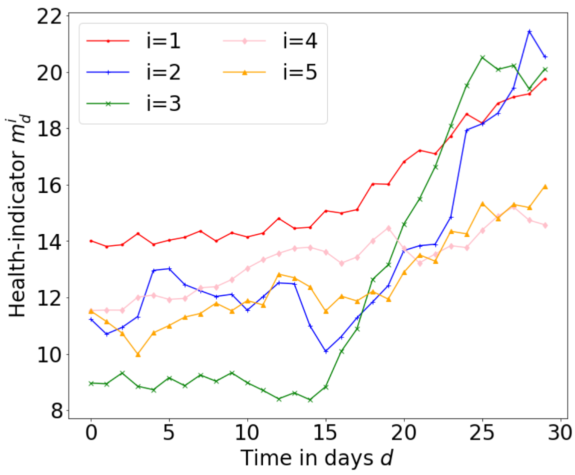


Figure 8.6.: The health indicator m^i_d for 5 CUs i 30 days before failure.

8.3.3. Methodology - RUL prognostics for CUs

Based on the health indicator m_d, we now determine the RUL prognostics for each of these components. There are two phases for the health indicator. In the first phase, this component is only monitored and the health indicator m_d is recorded every day d.

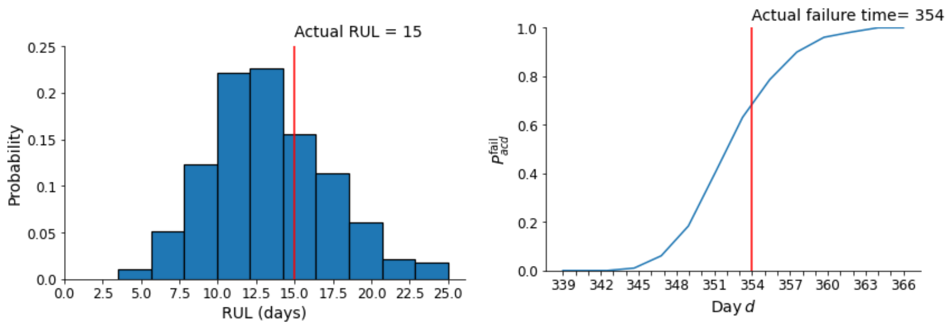
As soon as the health indicator reaches a prognostics threshold T^P, i.e., as soon as $m_d > T^P$, a second phase begins where a prognostic for the RUL of this component is determined. In this second phase, we consider the true degradation level of this component, denoted by x_d, and the health indicator m_d at day d as follows:

$$x_d = x_{d-1} + \alpha_d \lambda_d \exp^{\lambda_d d}, \tag{8.3}$$

$$m_d = x_d + \nu_d, \tag{8.4}$$

where $\alpha_d \sim N(\mu_\alpha, \sigma_\alpha^2), \lambda_d \sim N(\mu_\lambda, \sigma_\lambda^2)$, and $\nu_d \sim N(0, \sigma_\nu^2)$ are model parameters.

The exponential functional form in eq. (8.3) is assumed since the cumulative damage in a component has an effect on the degradation rate of the component [36]. An exponential degradation model is a good approximation for non-linear degradation processes such as corrosion, bearing degradation and the deterioration of LED lighting [37–41]. The CU can also be seen as subject to accelerated wear due to increasing filter clogging.



(a) The estimated PDF of the RUL.

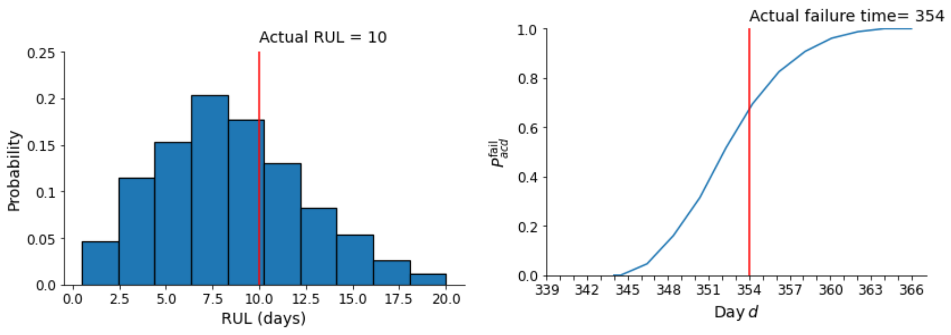
(b) The probability of failure P_{acd}^{fail}.

Figure 8.7.: The RUL prognostics for CU c of aircraft a, estimated at day 339 since the start of the monitoring phase. The actual RUL is 15 days, and actual failure time is at day 354.

Next, we estimate the RUL of this component using a particle filtering algorithm (see, for instance, [42]). We consider the recorded health indicator values $m_{d'}$ for this component up to the current day d, i.e., $d' < d$. Based on these indices, we estimate the RUL of this component as follows. We initialize x_0 with the measured health levels prior to the second phase. We consider n initial particles $\left(x_0^{(i)}, \alpha_0^{(i)}, \lambda_0^{(i)})\right), i \in \{1, 2, \ldots, \ n\}$, each with initial weight $1/n$. Then, new particles $\left(x_d^{(i)}, \alpha_d^{(i)}, \lambda_d^{(i)}\right)$ are obtained as follows:

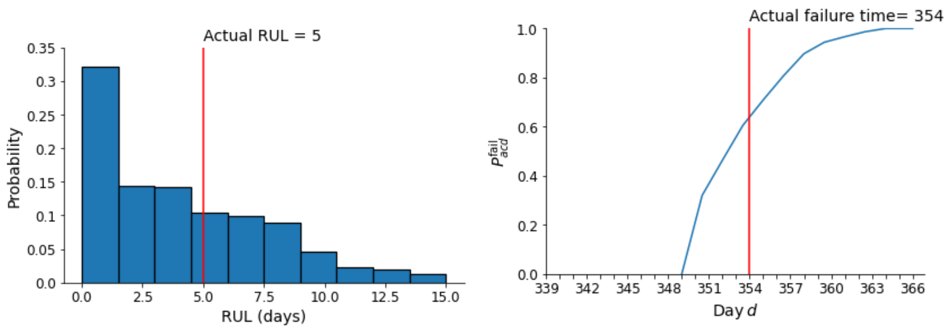
$$x_d^{(i)} = x_{d-1}^{(i)} + \alpha_d^{(i)} \lambda_d^{(i)} \exp(\lambda_d^{(i)} d), \tag{8.5}$$

where $\alpha_d^{(i)}$ and $\lambda_d^{(i)}$ are realizations of the random variables α_d and λ_d, respectively.



(a) The estimated PDF of the RUL. (b) The probability of failure P^{fail}_{acd}.

Figure 8.8.: The RUL prognostics for CU c of aircraft a, estimated at day 344 since the start of the monitoring phase. The actual RUL is 10 days, and actual failure time is at day 354.



(a) The estimated PDF of the RUL. (b) The probability of failure P^{fail}_{acd}.

Figure 8.9.: The RUL prognostics for CU c of aircraft a, estimated at day 349 since the start of the monitoring phase. The actual RUL is 5 days, and actual failure time is at day 354.

As d increases, and new measurements are available, the weights of the particles are updated and normalized with:

$$p\left(m_d|x_d^{(i)}\right) = \frac{1}{\sqrt{2\pi}\sigma_v}\exp\left(-\frac{1}{2}\left(\frac{m_d - x_d^{(i)}}{\sigma_v}\right)^2\right). \tag{8.6}$$

Now, given the weights of the particles, these particles are re-sampled [16] and, again, their weights are updated as $1/n$. Last, the RUL z_d of this component is estimated at the current day d based on the re-sampled particles and the measurements up to and including day d, where the RUL z_d is defined as:

$$\text{RUL} = \inf\{z_d : x_(d + z_d) \geq D|x_0, x_1, \ldots, x_d\}, \tag{8.7}$$

where D is a pre-defined failure threshold, $x_0, x_1, \ldots, x_d$ are the estimated degradation levels of this component at days $0, 1, \ldots, d$, respectively, and $x_(d + z_d)$ is the estimated degradation level at time $d + z_d$. We use eq. (8.7) to predict the RUL z_d^i of each individual particle i in the particle filtering algorithm as follows:

$$z_d^i = \inf\left\{ z_d^i : x_{d+z_d^i}^{(i)} \geq D | x_0^{(i)}, x_1^{(i)}, \ldots, x_d^{(i)} \right\}. \tag{8.8}$$

Here, $x_0^{(i)}, x_1^{(i)}, \ldots, x_d^{(i)}$ are the estimated degradation levels of particle i at days $0, 1, \ldots, d$, and $x_{d+z_d}^{(i)}$ is the estimated degradation level of particle i at day $d + z_d$.

Last, the probability that the RUL equals z_d at current day d is approximated by:

$$p\left(\text{RUL} = z_d | m_0, m_1, \ldots, m_d\right) = \sum_{i=1}^{n} w_d^{(i)} \mathcal{D}\left(z_d - z_d^i\right), \tag{8.9}$$

where w_d^i is the weight of the ith particle, and $\mathcal{D}(\cdot)$ is an indicator function:

$$\mathcal{D}(y) = \begin{cases} 1 & y = 0, \\ 0 & y \neq 0. \end{cases} \tag{8.10}$$

From eq. (8.9), which provides the PDF of the RUL obtained at current day d for a component $c \in C_a$ of aircraft $a \in A$, we obtain the probability $P_{acd^*}^{\text{fail}}$ that this component c of aircraft a fails before some future day $d^* > d$ as follows:

$$P_{acd^*}^{\text{fail}} = P\left(\text{RUL} \leq (d^* - d)\right). \tag{8.11}$$

Thus, given a current day d, eq. (8.11) determines the probability of failure before a day $d^* > d$ for a specific CU. If, however, the CU is in the first, monitoring-only phase, than we assume that $P_{acd^*}^{\text{fail}} = 0.001$.

8

8.3.4. RESULTS - RUL PROGNOSTICS FOR CUS

Following the methodology in Section 8.3.3, we determine the RUL prognostics for CUs using 1000 particles, $\sigma_v = 0.01$, $n = 10$ days, $\mathcal{D} = 22$ and $T^P = 11$. Furthermore, we determine μ_α, μ_λ, σ_α^2 and σ_λ^2 using Maximum Likelihood Estimation of α and λ on the log transformation of eq. (8.3) on the available data [43]. Figures 8.7, 8.8 and 8.9 show the PDF of the RUL and the distribution of P_{acd}^{fail} of the same CU c of an aircraft a estimated at day 339 (15 days before failure), day 344 (10 days before failure) and at day 349 (5 days before failure) since the start of the monitoring phase. The RUL estimation is precise, i.e., the actual RUL always falls within the probability distribution of the estimated RUL, while the uncertainty in the prognostic is low. For all prognostic horizons, the actual RUL is slightly underestimated. For this CU, it takes on average 14.4 seconds to estimate the RUL distribution using a computer with an Intel Core i7 processor at 2.11 GHz and 8Gb RAM.

8.4. Methodology - Predictive maintenance scheduling model

Using the prognostics obtained in Section 8.3 and information about the availability of the maintenance slots and spare components, we introduce a linear integer program to plan the maintenance of multiple aircraft systems of repairable components. This model is applied, using a rolling horizon approach, for a scheduling time window of PH days $[d_0,\dots,d_0+PH)$ (see Section 8.2.6 and Figure 8.2). We first introduce some additional notation and definitions.

Definition 2 *An aircraft is said to be* critical *when the probability that this aircraft is in an* AOG-condition *at the end of the scheduling time window* $[d_0,\dots,d_0+PH)$ *exceeds a reliability threshold* r*, i.e.,* $P^{AOG}_{a(d_0+PH)} \geq r$.

Let $A_r \subseteq A$ denote the set of critical aircraft at the beginning of the scheduling time window $[d_0,\dots,d_0+PH)$.

Let G_a denote the set of all possible subsets of the components of aircraft $a \in A_r$ that can be replaced in the scheduling time window $[d_0,\dots,d_0+PH)$, such that $P^{\text{AOG}}_{a(d_0+PH)} < r$. We assume that once a component is replaced in a scheduling time window, then this component cannot fail anymore in the same time window. The set G_a depends on the configuration of the multi-component system. To illustrate G_a, we discuss an example of a system where the components are linked in series, i.e., if one component fails, the whole system fails. Let critical aircraft a have a system consisting of 4 components in series, $C_a = \{1,2,3,4\}$. Let the probability of failure for component $k \in \{1,2\}$ by day d_0+PH be $P^{\text{fail}}_{ak(d_0+PH)} > r$. Let the probability of failure for component $k \in \{3,4\}$ by day d_0+PH be $P^{\text{fail}}_{ak(d_0+PH)} << r$. Then, at least component 1 and 2 must be replaced to ensure that $P^{\text{AOG}}_{a(d_0+PH)} < r$. The set G_a of component subsets that can be replaced to avoid the aircraft being in an AOG-condition is thus:

$$G_a = \big\{\{1,2\},\{1,2,3\},\{1,2,4\},\{1,2,3,4\}\big\}$$

We now introduce the decision variables, objective function and constraints of the predictive maintenance scheduling model with limited spare components.

Decision variables

We consider the following main decision variable.

$$X_{acs} = \begin{cases} 1, & \text{Component } c \in C_a \text{ of aircraft } a \in A \text{ is replaced in} \\ & \text{maintenance slot } s \in S_a, \\ 0, & \text{Otherwise.} \end{cases}$$

We also consider the following three auxiliary variables which i) keep track of the maintenance schedule for an entire aircraft, ii) keep track of the number of leased components at the end of a day, and iii) keep track of the number of newly leased components during a day. First,

$$Y_{as} = \begin{cases} 1, & \text{Aircraft } a \in A \text{ is assigned to slot } s \in S_a, \\ 0, & \text{Otherwise.} \end{cases}$$

Here, the auxiliary variable Y_{as} is defined by the decision variables X_{acs} as follows:

$$Y_{as} \geq X_{acs}, \qquad \forall a \in A, \forall c \in C_a, \forall s \in S_a \tag{8.12}$$

$$\tag{8.13}$$

where eq. (8.12) ensures that when a component $c \in C_a$ of aircraft $a \in A$ is replaced in maintenance slot $s \in S_a$, the entire aircraft is assigned to maintenance slot s.

Second, we define the number of leased spare parts at the end of day $d \in [d_0, \dots, d_0 + PH + \Delta)$ as:

$$L_d = \max \left\{ 0, \sum_{a \in A} \sum_{c \in C_a} \sum_{\substack{s \in S_a: \\ d_s \leq d < d_s + \Delta}} X_{acs} - S_d^{\text{begin}} \right\} \qquad \forall d \in [d_0, ..., d_0 + PH + \Delta), \tag{8.14}$$

where S_d^{begin} is the number of spare components initially available at the beginning of day d (see Section 8.2.6). Eq. (8.14) defines the number of leased spare components to be the number of components in repair at the beginning of day d, minus the number of initially available spare components. If a component is replaced on day $d \in [d_0, \dots, d_0 + PH)$, then this component is in repair until day $d + \Delta$.

Third, we define L_d^{new} to be the number of newly leased spare parts during day $d \in [d_0, ..., d_0 + PH + \Delta)$. The following two constraints apply for L_d^{new}:

$$L_d^{\text{new}} = \max \{0, L_d - L_{d-1}\} \qquad \forall d \in [d_0 + 1, ..., d_0 + PH + \Delta) \tag{8.15}$$

$$L_{d_0}^{\text{new}} = \max \left\{ 0, L_{d_0} - \max \left\{ 0, S_{d_0 - 1}^{\text{begin}} \right\} \right\}. \tag{8.16}$$

Equations (8.14), (8.15) and (8.16) are linearized exactly with the use of binary dummy variables, following [44, Chapter 4.5].

OBJECTIVE FUNCTION

We consider the following objective function that minimizes the total costs with the maintenance of multiple aircraft systems.

$$\min \sum_{a \in A} \sum_{c \in C_a} \left(\left(\sum_{s \in S_a} X_{acs} \frac{c^{\text{fix}} + P_{acd_s}^{\text{fail}} \cdot c^{\text{ex}}}{d_s - d_{ac}^{\text{install}}} \right) + \left(\left(1 - \sum_{s \in S_a} X_{acs} \right) \frac{c^{\text{fix}} + P_{ac(d_0 + PH)}^{\text{fail}} \cdot c^{\text{ex}}}{d_0 + PH - d_{ac}^{\text{install}}} \right) \right)$$
$$+ \sum_{a \in A} \sum_{s \in S_a} Y_{as} \cdot c_s + \sum_{d = d_0}^{d_0 + PH + \Delta - 1} \left(L_d \cdot c^{\text{Ld}} + L_d^{\text{new}} \cdot c^{\text{Lf}} \right). \tag{8.17}$$

The first term of eq. (8.17) represents the expected cost of replacing a component. This cost is incurred either when the replacement is planned within the scheduling time window $[d_0, \dots, d_0 + PH)$, or later when the decision to replace is postponed to the beginning of the next scheduling time window. In the first case, a fixed repair cost c^{fix} is incurred, plus a cost c^{ex} when the component is actually failed at the moment of replacement. This cost is normalized with the number of days the component is in use $d_s - d_{ac}^{\text{install}}$, i.e., it is preferred to use the component as long as possible. In the second case, we consider the cost of postponing the replacement, which contains the same costs c^{fix} and c^{ex}, relative to the earliest possible replacement time when the decision is postponed. Overall, the first term of eq. (8.17) trades-off between replacing a component in the current time window (which gives a lower exploitation time of the component, but also a lower probability of failure) or postponing the replacement to a later time window (which gives a higher exploitation time of the component, but also a higher probability of failure).

The second term of eq. (8.17) represents the costs of assigning an entire aircraft to a maintenance slot, while the last term represents the cost of leasing spare components for an entire fleet of aircraft.

Constraints

Additionally to constraints (8.12), (8.14), (8.15), (8.16) that define the auxiliary variables, we consider the following constraints:

$$\sum_{s \in S_a} Y_{as} \leq 1 \qquad \forall a \in A \quad (8.18)$$

$$\sum_{a \in A} Y_{as} \leq m_s \qquad \forall s \in S \quad (8.19)$$

$$\exists g \in G_a: \sum_{c \in g} \sum_{\substack{s \in S_a: \\ d_s < d_a^r}} X_{acs} \geq |g| \qquad \forall a \in A_r$$

$$\text{where } d_a^r = \text{argmin}_{d \in \{d_0+1, \dots, d_0+PH\}} \{P_{ad}^{\text{AOG}} | P_{ad}^{\text{AOG}} \geq r\} \quad (8.20)$$

$$X_{acs} \in \{0,1\} \qquad \forall a \in A, \forall s \in S_a, \forall c \in C_a \quad (8.21)$$

$$Y_{ac} \in \{0,1\} \qquad \forall a \in A, \forall s \in S_a \quad (8.22)$$

$$L_d, L_d^{\text{new}} \in \mathbb{N}^+ \qquad \forall d \in \{d_0, \dots, d_0 + PH + \Delta\} \quad (8.23)$$

Constraint (8.18) ensures that each aircraft is assigned to at most one maintenance slot within the scheduling time window. Constraint (8.19) ensures that the number of aircraft assigned to a maintenance slot s does not exceed the slot's capacity m_s. Constraint (8.20) ensures that the probability that an aircraft is in an AOG-condition does not exceed a reliability threshold r within the scheduling time window. To prevent that an aircraft $a \in A_r$ is in an AOG-condition, a subset of the components must be replaced before d_a^r, where d_a^r is the first day d within the time window $[d_0 + 1, \dots, d_0 + PH)$ when $P_{ad_a^r}^{\text{AOG}} \geq r$. To ensure that $P_{a(d_0+PH)}^{\text{AOG}} < r$, i.e., that the probability of an AOG-condition for aircraft $a \in A_r$ does not exceed the reliability threshold, all the components in at least one subset $g \in G_a$ have to be replaced, i.e., all $|g|$ components of the subset g are replaced. This constraint is linearized exactly with the use of binary dummy variables, following [44, Chapter 3.6]. Finally, Constraints (8.21), (8.22) and (8.23) define the domains of the decision variables.

8.5. RESULTS - PREDICTIVE MAINTENANCE SCHEDULING FOR COOLING UNITS

In this section, we illustrate the maintenance scheduling model (see Section 8.4) for a fleet of $|A| = 13$ homogeneous, wide-body aircraft. Each aircraft is equipped with $N = 4$ identical cooling units (CUs) in the cooling system, as introduced in Section 8.3. First, we discuss the cooling units system and its k-out-of-N system's configuration in Section 8.5.1. In Section 8.5.2 we illustrate the maintenance scheduling model for this multi-component, k-out-of-N system. Last, in Section 8.5.3 the computational time of the model is discussed for different sizes of aircraft fleet.

8.5.1. k-OUT-OF-N SYSTEM OF CUS

Each aircraft is equipped with $N = 4$ cooling units (CUs), which are linked in a k-out-of-N system. Here, the Minimum Equipment List (MEL) requires that $k = 2$ [30]. An aircraft is thus allowed to fly (i.e., not in an AOG-condition) if at least $k+1 = 3$ or more CUs are operational. However, if exactly $k = 2$ CUs are operational, then the aircraft is still allowed to fly for a maximum of $V = 10$ days [30]. Otherwise, the aircraft is in an *Aircraft-On-Ground* condition, which is defined as follows:

Definition 3 *An aircraft is in an* Aircraft-On-Ground *(AOG) condition as soon as i) $(N-k)+1$ or more components fail, or ii) $(N-k)$ components have been failed for more than V days.*

The probability P^{AOG}_{ad} that an aircraft $a \in A$ with a k-out-of-N system is in an AOG-condition at the beginning of day d, is as follows:

$$\begin{aligned} P^{\text{AOG}}_{ad} =& P\big(i \in \{(Nf-k)+1,\dots,N\} \text{ components fail before the beginning of day } d, \\ & \text{or exactly } (N-k) \text{ components fail before the beginning of day } d-V\big) \end{aligned}$$

For the case of the cooling system with $N = 4$, $k = 2$ and $V = 10$, the probability of an aircraft being in an AOG-condition at day d is:

$$\begin{aligned} P^{\text{AOG}}_{ad} =& \prod_{i=1}^{4} P^{\text{fail}}_{aid} + \sum_{i=1}^{4}\left(1-P^{\text{fail}}_{aid}\right)\prod_{\substack{l=1\\ l\neq i}}^{4} P^{\text{fail}}_{ald} \\ &+ \sum_{i=1}^{3}\sum_{j=i+1}^{4} P^{\text{fail}}_{ai(d-10)}P^{\text{fail}}_{aj(d-10)}\prod_{\substack{l=1\\ l\notin\{i,j\}}}^{4}\left(1-P^{\text{fail}}_{ald}\right). \end{aligned} \tag{8.24}$$

In Section 8.4, we define that the set G_a contains all subsets of C_a that could be replaced to ensure that $P^{\text{AOG}}_{a(d_0+PH)} < r$, i.e., the set of components that could be replaced to avoid having the aircraft in an AOG-condition. To illustrate G_a for the cooling system, we discuss the following example. Let a critical aircraft $a \in A_r$ (see Definition 2) have $N = 4$ CUs, i.e., $C_a = \{1,2,3,4\}$. Furthermore, let $r = 0.01$ and $PH = 15$ days. Then $P^{\text{AOG}}_{a(d_0+15)}$ is the sum of i) the probability that three of four components fail by day $d_0 + 15$, and ii) the probability that two components fail by day $d_0 + 15 - 10$ and no components fail between day $d_0 + 15 - 10$ and day $d_0 + 15$ (see eq. (8.24)). Moreover, let the probabilities that components 1, 2 3 and 4 fail by day $d_0 + 15$ be $P^{\text{fail}}_{a1(d_0+15)} = 1$, $P^{\text{fail}}_{a2(d_0+15)} = 0.05$, $P^{\text{fail}}_{a3(d_0+15)} = 0.05$ and $P^{\text{fail}}_{a4(d_0+15)} = 0.001$. Last, let the probabilities that components 1, 2, 3 and 4 fail by day $d_0 + 15 - 10$ be $P^{\text{fail}}_{a1(d_0+5)} = 1$, $P^{\text{fail}}_{a2(d_0+5)} = 0.02$, $P^{\text{fail}}_{a3(d_0+5)} = 0.02$ and $P^{\text{fail}}_{a3(d_0+5)} = 0.001$.

In this example, the set of replaced components must include at least component $\{1\}$, or components $\{2,3\}$ to ensure that $P^{\text{AOG}}_{a(d_0+15)} < 0.01$. Thus, the set of component subsets that can be replaced to solve the aircraft criticality (see Definition 2) is:

$$G_a = \{\{1\},\{2,3\},\{1,2\},\{1,3\},\{1,4\},\{1,2,3\},\{1,2,4\},\{1,3,4\},\{2,3,4\},\{1,2,3,4\}\}.$$

8.5.2. MAINTENANCE SCHEDULING

Costs	
c^{fix}	10^4
c^{ex}	$5 \cdot 10^3$
c^{Lf}	$4 \cdot 10^4$
c^{Ld}	10^3
Rolling horizon parameters	
PH	15 days
τ	5 days
CU-related parameters	
N	4 CUs
k	2 CUs
Δ	4 weeks
V	10 days
S_0^{begin}	3 CUs
Reliability-related parameters	
r	0.01

Table 8.1.: Parameter values for the maintenance scheduling model in Section 8.4.

In this section, we illustrate the maintenance scheduling model (see Section 8.4) for a fleet of $|A| = 13$ homogeneous, wide-body aircraft. The initial stock of spare CUs for this fleet of 13 aircraft at day 0 is $S_0^{\text{begin}} = 3$. Moreover, the first $\tau = 5$ days of each maintenance schedule in the rolling horizon approach are fixed. In general, various values for τ can be considered. The other parameter values for our proposed maintenance scheduling model are given in Table 8.1.

In practice, it is assumed that there are two types of maintenance slots for the aircraft: i) aircraft-specific slots, which are dedicated to one specific aircraft, and ii) generic slots, which can be used by all aircraft. We assume that at most two aircraft can be maintained at the same time in a generic slot, i.e., $m_s^{\text{generic}} = 2$. In extreme cases, when there are very few aircraft-specific slots or a large number of aircraft, this capacity could be increased. One generic slot is available every day. Last, we assume that the cost c_s of a maintenance slot s is $c_s^{\text{generic}} = 10^4$ for a generic slot and $c_s^{\text{specific}} = 1$ for an aircraft-specific slot. For our analysis, we use historical aircraft-specific slots that have been used in practice by the fleet of 13 aircraft. On average, an aircraft has 35 of these aircraft-specific maintenance slots per year.

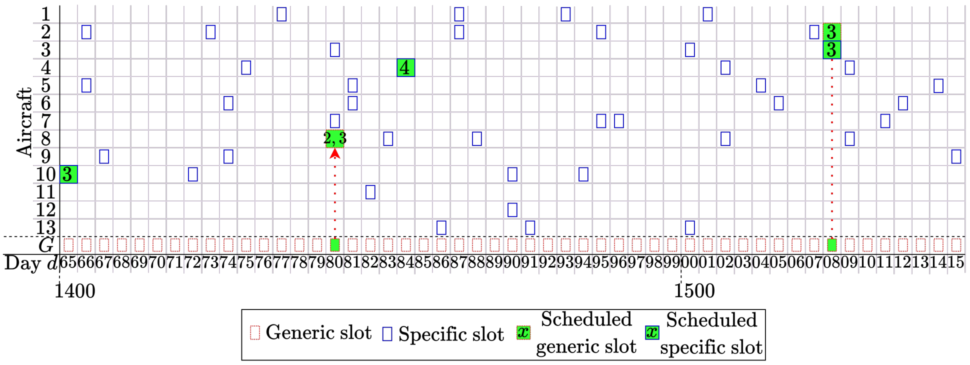


Figure 8.10.: Maintenance schedule for 50 days, from day 1465 to 1515 for a fleet of 13 wide-body aircraft.

Figure 8.10 shows the final maintenance schedule of the fleet of 13 aircraft for a period of 50 days, using a rolling horizon approach with scheduling time windows of $PH = 15$ days, of which each time the first $\tau = 5$ days are fixed. In this period, 6 CUs are replaced, 1 CU is leased and the total maintenance costs of the CUs is 137.203. These results are obtained in 3.3 seconds with the Gurobi solver version 9.0.2 with standard settings (branch-and-cut algorithm), implemented in Python, using an Intel Core i7 processor at 2.11 GHz and 8Gb RAM. The model is initialized with a random installation time from the uniform distribution for each CU, between 80 and 200 days before the start of the maintenance schedule ($d_{ac}^{\text{install}} \sim U(80, 200)$).

$a \in A$	$c \in C_a$	**Day of slot s_d**	**Failed at replacement?**	**Actual RUL**
10	3	1465	No	9 days
8	2	1480	Yes	-
8	3	1480	No	11 days
4	4	1484	No	6 days
2	3	1508	No	9 days
3	3	1508	Yes	-

Table 8.2.: The replaced components in the maintenance schedule in Figure 8.10.

In Figure 8.10, the aircraft-specific maintenance slots available for each aircraft during the 50 days period are depicted. There is also a generic slot available every day. Aircraft $a \in \{3,4,10\}$ are assigned to an aircraft-specific maintenance slot, while aircraft $a \in \{2,8\}$ are assigned to generic slots. Regarding the aircraft-specific slots, aircraft 3 is planned to be maintained during day 1508, aircraft 4 during day 1484 and aircraft 10 during day 1465. Regarding the generic slots, aircraft 2 is assigned to a generic slot at day 1508 and aircraft 8 during day 1480.

The components that are replaced in the maintenance schedule of 50 days are given in Table 8.2. Aircraft 8 is assigned to a maintenance slot at day 1480, during which two components, CU 2 and 3, are replaced. For the other aircraft, only one component per maintenance slot is replaced. Out of the 6 replacements, 4 components are replaced before they fail (66%). On average 8.75 days of the RUL are wasted when a component is replaced before its failure time. During the 50 days considered, there is one new component leased at day 1484 (i.e. $L^{\text{new}}_{1484} = 1$). This component is leased until day 1492 (i.e. $L_d = 1\,\forall d \in [1484,\ldots,1492]$ while $L_d = 0\,\forall d \in [1465,\ldots,1483] \cup [1493,\ldots,1515]$).

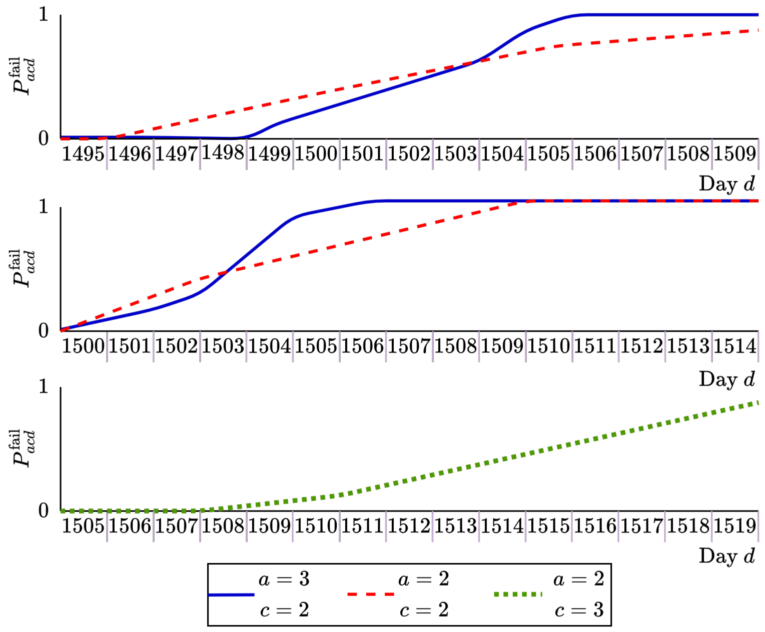


Figure 8.11.: The prognostics at the beginning of scheduling time windows $[1495,1510)$,$[1500,1515)$, $[1505,1520)$.

To illustrate the dynamic character of our rolling horizon approach, Figures 8.11 and 8.12 show three rolling time windows, which correspond to the last several days in Figure 8.10. Figure 8.11 shows the prognostics at the beginning of each time window. Only the CUs that have not failed yet, but that are in the second phase of the prognostics at the beginning of the time window, are shown. These prognostics are used as input in the maintenance scheduling model in Figure 8.12. Also here, we only show the aircraft with some CUs that might fail in the future.

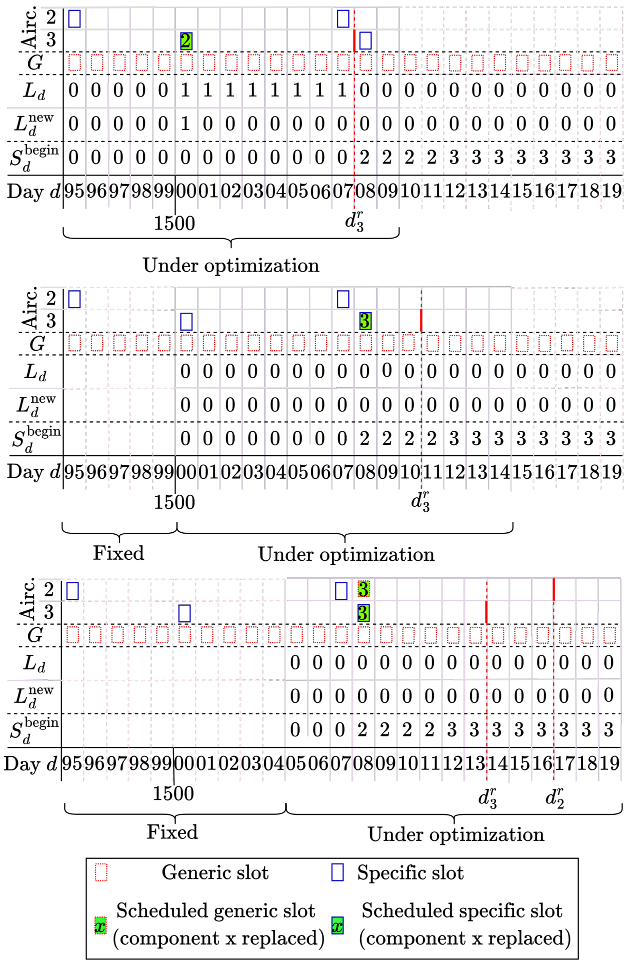


Figure 8.12.: The maintenance schedule of three iterations of the rolling horizon approach for time windows [1495, 1510),[1500, 1515) and [1505, 1520).

At the beginning of time window [1495, 1510), two CUs are in the second phase of the prognostics: CU 2 of aircraft 3 and CU 2 of aircraft 2 (see Figure 8.11). CU 3 of aircraft 3 is already failed. This aircraft is therefore critical, and some components have to be replaced before day 1508. In contrast, all CUs of aircraft 2 are still functional, and this aircraft is therefore not critical. For this time window [1495, 1510), there are no spare CUs available until day 1508. Aircraft 2 has no generic slots after or on day 1508, and the replacement of CU 2 of aircraft 2 is therefore not scheduled. However, a replacement of a CU of aircraft 3 has to be scheduled before day 1508, i.e., before a spare CU becomes available, due to the required reliability of each aircraft. The replacement of CU 2, with a estimated near-future failure, is therefore scheduled in the aircraft-specific slot at day 1500 (see Figure 8.12), and it is planned to lease a CU. The maintenance schedule of the first five days, [1495, 1499], is fixed. Since there is no maintenance planned in the first five days, no maintenance is thus executed and no CUs are leased.

In the next time window, [1500, 1515), CU 2 of aircraft 3 and CU 2 of aircraft 2 are not failed yet (see Figure 8.11). With the updated prognostics for CU 2 of aircraft 3, some components have to be replaced before day 1511 in this time window, instead of before day 1508. Aircraft 3 is therefore scheduled to be repaired in the generic slot during day 1508, when a spare CU becomes available. Both CU 2 and CU 3 of aircraft 3 are failed by this day, and one of them (CU 3) is selected for replacement in a specific slot. As before, the first five days of this maintenance schedule, [1500, 1504], are now fixed.

In the third time window, [1505, 1520), both CU 2 of aircraft 2 and CU 2 of aircraft 3 have failed. However, CU 3 of aircraft 2 is now in the second phase of the prognostics (i.e., estimated to fail in the near-future) as well. The aircraft is therefore critical; Some components have to be replaced before day 1517. An aircraft-specific slot for aircraft 2 is available on day 1507. However, no spare CU is available then. Since using a generic slot is much cheaper than leasing a spare CU, the replacement of CU 3 of aircraft 2 is scheduled in a generic slot at day 1508. The maintenance actions planned from day 1505 to day 1509 are fixed, which means that the maintenance planned on day 1508 (see Figure 8.10) is now fixed.

8.5.3. Computation time vs size of aircraft fleet

Table 8.3 shows the total computational time required to obtain a maintenance schedule for 60 months for different aircraft fleet sizes. Here, the number of spare CUs and the capacity of the generic slots is proportional to the fleet size. We also include the average computation time required to solve the maintenance scheduling problem for one time window (15 days). These computation times are obtained using a computer with an Intel Core i7 processor at 2.11 GHz and 8Gb RAM. For an aircraft fleet as large as 140 aircraft, a total of 1239 seconds are needed to obtain a maintenance schedule for 60 months, with an average computation time of 1.22 seconds to solve the problem for one time window of 15 days.

	Size of fleet of aircraft				
	13	30	60	90	120
Time (sec.) - 60 months schedule	71	179	482	752	1239
Time (sec.) - one time window of 15 days	0.04	0.14	0.44	0.73	1.22

Table 8.3.: Total computational time in seconds for the maintenance scheduling for various aircraft fleet sizes.

8.6. PREDICTIVE MAINTENANCE VS. CORRECTIVE AND PREVENTIVE MAINTENANCE

In this section, we compare our proposed predictive maintenance model with limited spare components (see Section 8.4) with a corrective and a preventive maintenance strategy (see [45, 46]), for the k-out-of-N systems. For these two maintenance strategies, we also consider a limited amount of spare components and fixed maintenance slots. Corrective and preventive maintenance strategies are often used in the practice of aircraft maintenance [16, 47, 48].

CORRECTIVE MAINTENANCE (CM) FOR k-OUT-OF-N SYSTEMS OF REPAIRABLES WITH LIMITED SPARES

We consider a corrective maintenance (CM) strategy where the system is maintained only when $k = 2$ or more components of the system are failed (see also Definition 3). We plan the aircraft maintenance in the following order of priority: First, the maintenance for all aircraft already in an AOG-condition (see Definition 3) is planned. An aircraft in an AOG-condition is assigned to the earliest available maintenance slot. When there are $f \geq k$ failed components in the aircraft, at least $f - 1$ failed components are replaced in this maintenance slot. If there are not enough spare components, then extra components are leased so that all $f - 1$ failed components can be replaced.

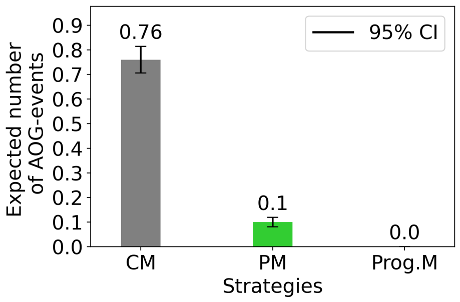
Second, all aircraft with $k = 2$ failed components that are not yet in an AOG-condition (see Definition 3), are assigned to maintenance slots. Such an aircraft is maintained in the earliest available aircraft-specific slot, as long as this does not lead to an AOG-condition. Otherwise, the aircraft is maintained in the earliest available maintenance slot, irrespective of the type of slot. At least 1 failed component is replaced. If there are not enough spare components, then extra components are leased.

Last, all remaining failed components in the two types of aircraft above are replaced as well, as long as there are enough spare components.

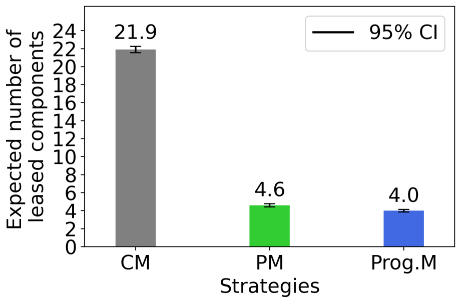
PREVENTIVE MAINTENANCE (PM) FOR k-OUT-OF-N SYSTEMS OF REPAIRABLES WITH LIMITED SPARES

We also consider a preventive maintenance (PM) strategy where the system is maintained to prevent a system failure. To prevent that the entire system fails, i.e., at least $k+1$ components are failed, or k components are failed for more than V days, we replace components as soon as they fail, provided spare components are available. First, the aircraft for which the system has $k = 2$ or more failed components are maintained as in the CM strategy. Then, the failed components in the remaining aircraft are replaced as well. These aircraft can only be assigned to aircraft-specific slots. Furthermore, no spare components can be leased to replace these failed components.

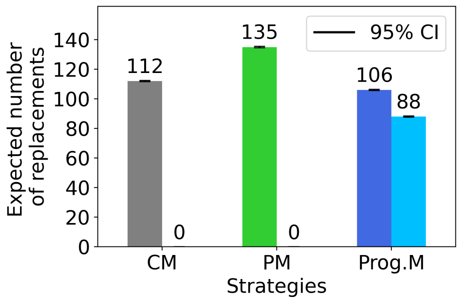
We analyze CM, PM and the prognostics-based maintenance scheduling model for a fleet of 13 aircraft for a period of 60 months using Monte Carlo simulation



(a) The number of AOG-events.



(b) The number of leased components.



(c) The total number of replacements (denoted by T), and the number of replacements of non-failed components (denoted by NF).

Figure 8.13.: The expected long-term performance of PM, CM and prognostics-based maintenance model ($Prog.M$) for a period of 60 months and a fleet of 13 wide-body aircraft, including 95% confidence intervals (CI).

with a 1000 simulation runs. All parameters and costs are the same as in Table 8.1. Figure 8.13 shows the performance of CM, PM and our proposed prognostics-based maintenance scheduling model. Table 8.4 gives 95% confidence intervals. Figure 8.13a shows the expected number of times an aircraft is in an AOG-condition (see Definition 3) for the three strategies. This is called an AOG-event. The results show that the CM strategy leads to the highest number of expected AOG-events.

Figure 8.13b shows the expected number of leased spare components per strategy. The most spare components are leased for the CM strategy. Both the PM strategy and the prognostic maintenance scheduling model need relatively few spare components. The total number of replacements T, and the number of replacements of non-failed components NF, is given in Figure 8.13c. For the CM and PM strategies, by definition, only failed components are replaced. The number of total replacements is highest for the PM strategy, because components are replaced as soon as they fail (provided that there are enough spare components). In contrast, for

	95% CI- AOG events	95% CI- Leases	95% CI- Replacements	95% CI- Total costs (million)
CM	[0.71, 0.82]	[21.6, 22.3]	[112.2, 113.1] (T)	[3.05, 3.10]
PM	[0.08, 0.11]	[4.43, 4.78]	[134.7, 135.6] (T)	[2.26, 2.29]
Pred.M	0.0	[3.90, 4.19]	[105.1, 106.0] (T) [87.2, 88.0] (NF)	[1.57, 1.60]

Table 8.4.: 95% confidence interval (CI) of the long-term performance of PM, CM and Pred.M (predictive maintenance), where T is the total number of replacements, and NF is the total number of replacements of non-failed components.

the *CM* strategy, failed components are replaced only when there are at least $k = 2$ failed components in a system. For the prognostic-based maintenance scheduling, the total number of replacements is the lowest because components that fail are not necessarily immediately replaced. When the probability of an AOG-condition for an aircraft exceeds the reliability threshold r, it is often more beneficial to replace the component(s) that have a failure estimated in the near-future, thus saving repair costs. Here, for on average 88 out of the 106 replacements, the components are not failed at the time of replacement.

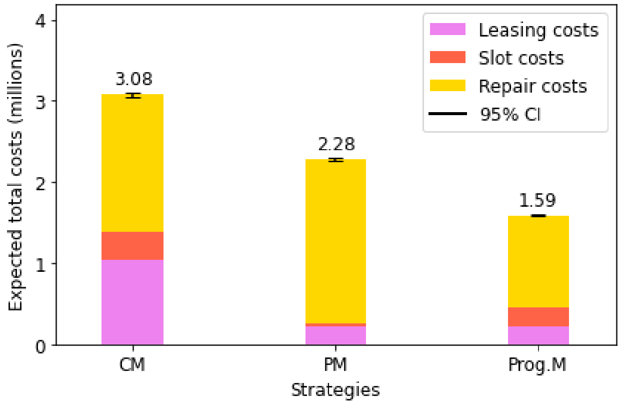


Figure 8.14.: The expected long-term maintenance costs of *PM*, *CM* and prognostics-based maintenance model (*Prog.M*) for a period of 60 months and a fleet of 13 wide-body aircraft, including 95% confidence intervals (CI).

Last, the total expected maintenance costs are given in Figure 8.14. For all strategies, the repair costs constitute the largest fraction of the expected total costs, while the expected slot costs constitute the smallest fraction of the expected total costs. The expected total costs are the highest for the *CM* strategy, while the prognostic maintenance schedule has the lowest expected total costs.

Overall, the results of our case study show that the prognostics-based maintenance scheduling model is most beneficial, with the lowest expected maintenance costs and the lowest expected number of number of AOG-events.

8.7. Conclusion

An integrated approach to develop RUL prognostics from sensor data, and to subsequently optimize the maintenance schedule, is proposed for a fleet of aircraft, each equipped with a multi-component system of repairable components. RUL prognostics are updated over time with new sensor measurements. The maintenance schedule takes the RUL prognostics into account to schedule component replacements in a rolling horizon fashion. As a case study, a fleet of wide-body aircraft, each equipped with a system of cooling units, is considered. First, model-based RUL prognostics are developed for these aircraft cooling units. Second, these RUL prognostics are integrated into a rolling horizon maintenance scheduling model. The scheduling model also considers a limited stock of spare components, as well as the available maintenance slots. Moreover, a reliability constraint is imposed on each considered system.

The results show that by integrating prognostics into the maintenance schedule, components are replaced in anticipation of failure without wasting their useful life. Our proposed prognostics-based predictive maintenance scheduling model reduces the costs by 48% relative to a corrective maintenance strategy and by 30% relative to a preventive maintenance strategy. Overall, our approach shows how RUL prognostics could be integrated in the maintenance schedule, and illustrates the potential costs savings with predictive maintenance.

As future work, we plan to further extend our maintenance scheduling model. We plan to consider other aircraft systems and components for the maintenance scheduling as well, and to compare the predictive maintenance strategy with several other corrective and preventive maintenance strategies, using a larger range of performance indicators. Last, we plan to relax the assumption that a repaired CU is “as-good-as-new”, and instead consider imperfect repairs. With such extensions, we aim to obtain an increasingly closer-to-implementation prognostics-driven maintenance scheduling model.

REFERENCES

[1] de Pater, I., & Mitici, M. (2021). Predictive maintenance for multi-component systems of repairables with Remaining-Useful-Life prognostics and a limited stock of spare components. *Reliability Engineering & System Safety, 214*, Article number: 107761.

[2] de Pater, I., del Mar Carillo Galera, M., & Mitici, M. (2021, September 19-23). Criticality-based predictive maintenance scheduling for aircraft components with a limited stock of spare components. *Proceedings of the 31st European Safety and Reliability Conference*, Angers, France, Pages: 55–62.

[3] Maintenance Cost Technical Group (MCTG). (2020). *Airline maintenance cost executive commentary (FY2019 data), public version* (tech. rep.). International Air Transport Association (IATA).

[4] Daily, J., & Peterson, J. (2017). Predictive maintenance: How big data analysis can improve maintenance. In *Supply Chain Integration Challenges in Commercial Aerospace* (Pages: 267–278). Springer.

[5] Atamuradov, V., Medjaher, K., Dersin, P., Lamoureux, B., & Zerhouni, N. (2017). Prognostics and Health Management for maintenance practitioners - Review, implementation and tools evaluation. *International Journal of Prognostics and Health Management, 8*(3), Pages: 1–31.

[6] Li, N., Gebraeel, N., Lei, Y., Bian, L., & Si, X. (2019). Remaining Useful Life prediction of machinery under time-varying operating conditions based on a two-factor state-space model. *Reliability Engineering & System Safety, 186*, Pages: 88–100.

[7] Jiao, R., Peng, K., Dong, J., & Zhang, C. (2020). Fault monitoring and Remaining Useful Life prediction framework for multiple fault modes in prognostics. *Reliability Engineering & System Safety, 203*, Article number: 107028.

[8] Andriotis, C., & Papakonstantinou, K. (2019). Managing engineering systems with large state and action spaces through deep reinforcement learning. *Reliability Engineering & System Safety, 191*, Article number: 106483.

[9] Huynh, K. T., Grall, A., & Bérenguer, C. (2018). A parametric predictive maintenance decision-making framework considering improved system health prognosis precision. *IEEE Transactions on Reliability, 68*(1), Pages: 375–396.

[10] Nguyen, K.-A., Do, P., & Grall, A. (2014). Condition-based maintenance for multi-component systems using importance measure and predictive information. *International Journal of Systems Science: Operations & Logistics, 1*(4), Pages: 228–245.

[11] Nielsen, J. S., & Sørensen, J. D. (2018). Computational framework for risk-based planning of inspections, maintenance and condition monitoring using discrete

Bayesian networks. *Structure and Infrastructure Engineering, 14*(8), Pages: 1082–1094.

[12] Wang, L., Chu, J., & Mao, W. (2009). A condition-based replacement and spare provisioning policy for deteriorating systems with uncertain deterioration to failure. *European Journal of Operational Research, 194*(1), Pages: 184–205.

[13] Huynh, K. T., Barros, A., & Bérenguer, C. (2014). Multi-level decision-making for the predictive maintenance of k-out-of-N: F deteriorating systems. *IEEE Transactions on Reliability, 64*(1), Pages: 94–117.

[14] Shi, Y., Zhu, W., Xiang, Y., & Feng, Q. (2020). Condition-based maintenance optimization for multi-component systems subject to a system reliability requirement. *Reliability Engineering & System Safety, 202*, Article number: 107042.

[15] Mercier, S., & Pham, H. H. (2012). A preventive maintenance policy for a continuously monitored system with correlated wear indicators. *European Journal of Operational Research, 222*(2), Pages: 263–272.

[16] Lee, J., & Mitici, M. (2020). An integrated assessment of safety and efficiency of aircraft maintenance strategies using agent-based modelling and stochastic Petri Nets. *Reliability Engineering & System Safety, 202*, Article number: 107052.

[17] Zhang, Z., Wu, S., Li, B., & Lee, S. (2015). (n, N) Type maintenance policy for multi-component systems with failure interactions. *International Journal of Systems Science, 46*(6), Pages: 1051–1064.

[18] Chen, D., & Trivedi, K. S. (2005). Optimization for condition-based maintenance with semi-Markov decision process. *Reliability Engineering & System Safety, 90*(1), Pages: 25–29.

[19] Papakonstantinou, K. G., & Shinozuka, M. (2014a). Planning structural inspection and maintenance policies via dynamic programming and Markov processes. Part I: Theory. *Reliability Engineering & System Safety, 130*, Pages: 202–213.

[20] Papakonstantinou, K. G., & Shinozuka, M. (2014b). Planning structural inspection and maintenance policies via dynamic programming and Markov processes. Part II: POMDP implementation. *Reliability Engineering & System Safety, 130*, Pages: 214–224.

[21] Schöbi, R., & Chatzi, E. N. (2016). Maintenance planning using continuous-state Partially Observable Markov Decision Processes and non-linear action models. *Structure and Infrastructure Engineering, 12*(8), Pages: 977–994.

[22] Wu, S.-j., Gebraeel, N., Lawley, M. A., & Yih, Y. (2007). A neural network integrated decision support system for condition-based optimal predictive maintenance policy. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 37*(2), Pages: 226–236.

[23] Kaiser, K. A., & Gebraeel, N. Z. (2009). Predictive maintenance management using sensor-based degradation models. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 39*(4), Pages: 840–849.

[24] Elwany, A. H., & Gebraeel, N. Z. (2008). Sensor-driven prognostic models for equipment replacement and spare parts inventory. *IIE Transactions, 40*(7), Pages: 629–639.

[25] Nguyen, K. T., & Medjaher, K. (2019). A new dynamic predictive maintenance framework using deep learning for failure prognostics. *Reliability Engineering & System Safety, 188*, Pages: 251–262.

[26] Yiwei, W., Christian, G., Binaud, N., Christian, B., Haftka, R. T., et al. (2017). A cost driven predictive maintenance policy for structural airframe maintenance. *Chinese Journal of Aeronautics*, *30*(3), Pages: 1242–1257.

[27] Vianna, W. O. L., & Yoneyama, T. (2017). Predictive maintenance optimization for aircraft redundant systems subjected to multiple wear profiles. *IEEE Systems Journal*, *12*(2), Pages: 1170–1181.

[28] Van Horenbeek, A., & Pintelon, L. (2013). A dynamic predictive maintenance policy for complex multi-component systems. *Reliability Engineering & System Safety, 120*, Pages: 39–50.

[29] Keizer, M. C. O., Flapper, S. D. P., & Teunter, R. H. (2017). Condition-based maintenance policies for systems with multiple dependent components: A review. *European Journal of Operational Research*, *261*(2), Pages: 405–420.

[30] *Easy access rules for the Master Minimum Equipment List (CS-MMEL) (Initial issue)* (tech. rep.). (2018). European Aviation Safety Agency (EASA).

[31] de Smidt-Destombes, K. S., van der Heijden, M. C., & van Harten, A. (2009). Joint optimisation of spare part inventory, maintenance frequency and repair capacity for k-out-of-N systems. *International Journal of Production Economics*, *118*(1), Pages: 260–268.

[32] Liang, Z., & Parlikad, A. K. (2020). Predictive group maintenance for multi-system multi-component networks. *Reliability Engineering & System Safety, 195*, Article number: 106704.

[33] Bouvard, K., Artus, S., Bérenguer, C., & Cocquempot, V. (2011). Condition-based dynamic maintenance operations planning & grouping. Application to commercial heavy vehicles. *Reliability Engineering & System Safety*, *96*(6), Pages: 601–610.

[34] Lei, Y., Li, N., Guo, L., Li, N., Yan, T., & Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing, 104*, Pages: 799–834.

[35] Yang, F., Habibullah, M. S., Zhang, T., Xu, Z., Lim, P., & Nadarajan, S. (2016). Health index-based prognostics for Remaining Useful Life predictions in electrical machines. *IEEE Transactions on Industrial Electronics*, *63*(4), Pages: 2633–2644.

[36] Si, X.-S., Wang, W., Chen, M.-Y., Hu, C.-H., & Zhou, D.-H. (2013). A degradation path-dependent approach for Remaining Useful Life estimation with an exact and closed-form solution. *European Journal of Operational Research*, *226*(1), Pages: 53–66.

[37] Tseng, S.-T., Tang, J., & Ku, I.-H. (2003). Determination of burn-in parameters and residual life for highly reliable products. *Naval Research Logistics (NRL)*, *50*(1), Pages: 1–14.

[38] Park, C., & Padgett, W. J. (2006). Stochastic degradation models with several accelerating variables. *IEEE Transactions on Reliability*, *55*(2), Pages: 379–390.

[39] Gebraeel, N. Z., Lawley, M. A., Li, R., & Ryan, J. K. (2005). Residual-life distributions from component degradation signals: A Bayesian approach. *IIE Transactions*, *37*(6), Pages: 543–557.

[40] Elwany, A. H., Gebraeel, N. Z., & Maillart, L. M. (2011). Structured replacement policies for components with complex degradation processes and dedicated sensors. *Operations Research*, *59*(3), Pages: 684–695.

[41] Chen, N., & Tsui, K. L. (2013). Condition monitoring and Remaining Useful Life prediction using degradation signals: Revisited. *IIE Transactions*, *45*(9), Pages: 939–952.

[42] Djuric, P. M., Kotecha, J. H., Zhang, J., Huang, Y., Ghirmai, T., Bugallo, M. F., & Miguez, J. (2003). Particle filtering. *IEEE Signal Processing Magazine*, *20*(5), Pages: 19–38.

[43] Lei, Y., Li, N., Gontarz, S., Lin, J., Radkowski, S., & Dybala, J. (2016). A model-based method for Remaining Useful Life prediction of machinery. *IEEE Transactions on Reliability*, *65*(3), Pages: 1314–1326.

[44] MirHassani, S., & Hooshmand, F. (2019). *Methods and models in mathematical programming*. Springer.

[45] Mital, A., Desai, A., Subramanian, A., & Mital, A. (2014). *Product development: A structured approach to consumer product development, design, and manufacture* (2nd ed.). Elsevier.

[46] Ahmad, R., & Kamaruddin, S. (2012). An overview of time-based and condition-based maintenance in industrial application. *Computers & Industrial Engineering*, *63*(1), Pages: 135–149.

[47] Ackert, S. P. (2010). *Basics of aircraft maintenance programs for financiers* (tech. rep.). Aircraft Monitor.

[48] Ren, H., Chen, X., & Chen, Y. (2017). *Reliability based aircraft maintenance optimization and applications*. Academic Press.

9

A STOCHASTIC PROGRAM FOR MAINTENANCE SCHEDULING UNDER ENDOGENOUS UNCERTAINTY WITH PROBABILISTIC RUL PROGNOSTICS

In Chapters 6, 7 and 8, we first create an initial maintenance planning, and update this planning over time with a rolling horizon approach. However, these future updates are not considered yet when creating the initial planning. In this chapter, we instead jointly optimize the initial maintenance planning and the future updates.

We formulate the maintenance planning problem as a multi-stage stochastic program. This stochastic program contains endogenous uncertainty, since the RUL of a system changes after maintenance. We solve the stochastic program with the nested Benders decomposition algorithm. To reduce the computational time, we propose a new clustering algorithm that is integrated in the nested Benders decomposition algorithm.

We illustrate our approach with aircraft engines, where we plan maintenance for up to five engines and for a planning horizon of four weeks (28 days/stages). We follow Chapter 7 to make the probabilistic RUL prognostics for these engines. By solving the stochastic program, we lower the expected costs by up to 0.89% *compared to the upper bound solution. Moreover, our clustering algorithm reduces the computational time. With five engines, we execute the same number of iterations 25 times faster with the clustering algorithm, than without the clustering algorithm.*

Parts of this chapter are under review for publication:

de Pater, I., & Mitci, M. (2023). Predictive maintenance planning under endogenous uncertainty using stochastic programming with a novel clustering algorithm integrated in the nested Benders decomposition. *Under review at Mathematical Programming Series B.*

9.1. INTRODUCTION

Maintenance for complex systems is expensive. The cost of aircraft maintenance account for 10% of the total airline operating costs [1]. For offshore wind turbines, the maintenance and operation cost account for 25% – 30% of the total life cycle costs [2]. Data-driven predictive maintenance aims to reduce these costs, while increasing the availability of the systems. Nowadays, there are many sensors that monitor the health condition of complex systems [3]. For predictive maintenance, these sensor measurements are used to estimate a Probability Density Function (PDF) of the Remaining Useful Life (RUL, time left until failure) of a system. These predictions, called “probabilistic RUL prognostics”, are subsequently used to plan maintenance [3]. In this chapter, we formulate the predictive maintenance planning problem with probabilistic RUL prognostics as a multi-stage stochastic program.

Stochastic programming has been considered for maintenance planning under different types of uncertainty: Uncertain weather for the maintenance of offshore wind farms [4], uncertain power demand for the maintenance of nuclear power plants [5] and uncertainty in the maintenance duration [6]. In these papers, the uncertainty is assumed to be “exogenous”, i.e., the uncertainty is not influenced by the decisions. In contrast, we consider the uncertainty of the estimated RUL. The PDF of the RUL is dependent on the maintenance decisions, since the probability of failure decreases after maintaining a system. This is called “endogenous” uncertainty.

In [7, 8], a distinction is made between endogenous uncertainty type 1 and type 2. For endogenous uncertainty type 1, the decisions change the underlying probability distribution of the uncertain parameters. This type of uncertainty occurs for instance in reliable transportation network design [9], where the unknown capacity after a disruption depends on the protection decisions, and highway strengthening [10], where the unknown failure probability after a disaster depends on the investment decisions. With endogenous uncertainty type 1, the stochastic programs usually become non-linear [9, 10], which complicates the solution methods.

Endogenous uncertainty type 1 occurs when the condition-based maintenance planning is combined with the operation planning [11–13]. In these studies, the degradation of a system is modelled with a generic degradation process. The decisions on the operation planning influence this degradation process, which in turn influences the estimated RUL/failure time. In [11–13], the uncertain RUL is integrated in the stochastic program by making the probability of failure a variable, which gives a non-linear program. In contrast, in this chapter, we estimate the PDF of the RUL based on the sensor measurements of the system, without considering the operational planning. We therefore do not have type 1 endogenous uncertainty.

For endogenous uncertainty type 2, the decisions change the timing of when the outcome of the uncertain parameters is observed, and thus the shape of the scenario tree [7, 8]. Type 2 endogenous uncertainty occurs for instance in gas field development [7] (where the uncertain gas reserve is only revealed after deciding to exploit a gas field) and clinical trial planning for new drugs [14] (where the unknown effectiveness of a drug is only revealed after deciding to perform a drug trial). A stochastic program with this type of endogenous uncertainty is often formulated as a linear program. Here, non-anticipativity constraints enforce that the outcome of two different scenarios is the same when the uncertain parameter has not been observed yet. Whether these constraints are active depends on the decisions [7, 8].

Endogenous uncertainty type 2 in condition-based maintenance planning is considered in [12, 15, 16]. In our problem, we also have endogenous uncertainty type 2: When we do not replace a component before it fails, we observe the RUL/failure time when the component fails. But when we replace a component before it fails, we never observe the RUL/failure time. The authors of [15, 16] optimize the condition-based maintenance planning for a multi-component system under this type of endogenous uncertainty. Here, the probability of failure is estimated after inspections, and the multi-stage problem is approximated by a two-stage problem with a rolling horizon approach. In contrast with [15, 16], however, we plan predictive maintenance for multiple systems (each consisting of a single component). The maintenance for multiple systems is linked over time by the limited capacity and the availability of spare components. We therefore solve the full multi-stage stochastic program instead of approximating it with a two-stage problem.

In this chapter, we optimize the predictive maintenance planning for multiple systems using a multi-stage stochastic linear program. First, we obtain probabilistic RUL prognostics for these systems based on the sensor measurements with a Convolutional Neural Network. We then optimize the maintenance planning with these RUL prognostics, by jointly optimizing the initial maintenance planning at the first day and the updates at later days. Due to a limited capacity and a limited number of spare components, only a limited number of systems can be replaced at the same time. In this study, we provide the following contributions:

- We formulate the predictive maintenance planning problem for multiple systems as a multi-stage stochastic integer linear program. We estimate the PDF of the RUL of these systems with a Convolutional Neural Network with Monte Carlo dropout. Moreover, we formulate the stochastic program such that the constraint matrix is totally unimodular. With this, we obtain an integer optimal solution when solving the relaxation of the integer stochastic program.

- We propose a new dynamic clustering algorithm, based on the endogenous uncertainty of our problem. We integrate this new algorithm in the nested Benders decomposition algorithm [17, 18], that we use to solve the multi-stage stochastic program. The clustering algorithm decreases the computational time for solving the problem to optimality.

- We apply our method to a realistic case study for aircraft engines, where we consider up to 5 engines and a planning horizon of 4 weeks (28 days/stages). Considering 5 engines, we perform the same number of iterations 25 times faster with our clustering algorithm, than without the clustering algorithm.

In the remainder of this work, we first introduce the considered predictive maintenance planning problem in Section 9.2. We then formulate the corresponding multi-stage stochastic program in Section 9.3, and propose the solution method with our new clustering algorithm in Section 9.4. Last, we apply this method to a case study with aircraft engines in Section 9.5.

9.2. Problem formulation - predictive maintenance scheduling

We consider the maintenance planning for a set of systems E. Let $d = 0$ denote the current day. We optimize the maintenance planning over the next D days.

9.2.1. Data-driven RUL prognostics

We estimate the Probability Density Function (PDF) of the Remaining Useful Life (RUL) for each system $e \in E$, using the sensor measurements of this system. The RUL of system e is the number of days left until failure for this system. Let p_{ed} denote the probability that the RUL of system $e \in E$ is d days, i.e., the probability that system e fails at day $d, 0 \le d \le D$. We also calculate the conditional probability p'_{ed} that the RUL of system $e \in E$ is d days, given that it has not failed before day d (i.e., the hazard rate):

$$p'_{ed} = \frac{p_{ed}}{1 - \sum_{h=0}^{d-1} p_{eh}}. \tag{9.1}$$

In Section 9.5.1, we define a Convolutional Neural Network that is used to estimate these failure probabilities.

The estimated PDF of the RUL of system e is only valid if no maintenance is performed on this system. As soon as a system is maintained, we assume that the probability of failure becomes zero until day D, i.e., we have endogenous uncertainty.

9.2.2. Constraints for the maintenance scheduling

We assume that the maintenance task for a system $e \in E$ consists of replacing this system, i.e., the system is removed, and a spare system is immediately installed instead. The removed, faulty system is maintained in an external facility. This takes d^s days, i.e., the repair lead time is d^s days. After maintenance, the system can be reused. Until it is reused, it is added to the stock of spare systems. Let s be the number of initially available spare systems in stock. Let d^g denote the number of days it takes to replace a system $e \in E$. We assume that the replacement for each system $e \in E$ begins at the start of every day $d \in [0, 1, \dots, D]$, and that at most g systems can be replaced simultaneously. A similar assumption is made in [19, 20].

9.2.3. Maintenance costs

Let c_r denote the costs of replacing, and subsequently maintaining, a system $e \in E$, let c_f denote the cost incurred when a system $e \in E$ fails, and let c_l denote the costs per day a system remains failed (i.e., the system is failed, but its replacement has not started yet). We minimize the expected costs over the expected lifetime of the systems. Let I_e denote the number of days system $e \in E$ has been used before the current day 0. The total lifetime of a system e is $d + I_e$ if a system e fails at, or is replaced at, day $d \in [0, 1, \dots, D]$.

An initial maintenance planning is made at the current day $d = 0$. This planning can be updated over time. For instance, we can reschedule a replacement to an earlier day if a system fails. Since constantly changing the maintenance planning is undesirable, we add a small penalty if we change the maintenance planning at a later day $d > 0$. Specifically, we incur a cost $c_i > 0$ for inserting a replacement in the maintenance planning, and a cost $c_k > 0$ for cancelling a planned replacement.

9.3. MULTI-STAGE STOCHASTIC INTEGER LINEAR PROGRAM FOR PREDICTIVE MAINTENANCE SCHEDULING

We pose the optimization of the predictive maintenance planning, including possible updates at future days, as a multi-stage stochastic integer linear program as follows.

9.3.1. SCENARIO TREE

We consider a discrete-time multi-stage stochastic program. At the beginning of each day $d \in [0,1,\dots,D]$, we observe if system $e \in E$ has failed or not. We assume that at the beginning of day $d = 0$, at the root node, no system has failed yet. With this, we construct a scenario tree. Figure 9.1 shows an example of a scenario tree with two systems $E = \{1,2\}$ and three days $d \in [0,1,2]$.

Each possible outcome in the scenario tree is represented by a node. Let N denote the set of nodes in the multi-stage scenario tree. Let $N_d \subseteq N$ denote the set of nodes in the scenario tree that belong to day d. We denote the day of a node $n \in N$ by $d(n)$. At day $d = 0$, there is one root node, denoted by 0. Let $E^n_{\text{fail}} \subseteq E$ denote the set of systems that have already failed before node n. Let $E^n_{\text{new}} \subseteq E$ be the set of systems that fail at the beginning of day $d(n)$ at node n. For each system $e \in E^n_{\text{fail}}$, we also have the day $d^{\text{fail}}_e(n)$ system e failed in the considered node n. For instance, for node 7 in Figure 9.1, $E^7_{\text{new}} = \{2\}$, while $E^7_{\text{fail}} = \{1\}$ with $d^{\text{fail}}_1(7) = 1$. We assume that a system fails at most once in the considered planning horizon.

For each node $n \in N$, we denote the direct successor nodes by $S(n) \subseteq N_{d(n)+1}$. The leaf nodes, i.e., the nodes at day D, do not have any successors. For a successor node $m \in S(n)$ of node n, the following six conditions S1-S6 hold:

S1 $d(m) = d(n) + 1$

S2 $E^m_{\text{fail}} = E^n_{\text{fail}} \cup E^n_{\text{new}}$

S3 $\forall e \in E^n_{\text{fail}}, d^{\text{fail}}_e(m) = d^{\text{fail}}_e(n)$

S4 $\forall e \in E^n_{\text{new}}, d^{\text{fail}}_e(m) = d(n)$

S5 $\nexists e \in E^m_{\text{new}} : e \in E^m_{\text{fail}}$.

S6 The probability q_{nm} to go from node n to node m is strictly larger than 0.

The probability q_{nm} to go from node n to node m (where node m fulfills conditions S1 - S5) is:

$$q_{nm} = \prod_{e \in E^m_{\text{new}}} p'_{ed(m)} \prod_{e \in E \setminus \left(E^m_{\text{new}} \cup E^m_{\text{fail}}\right)} \left(1 - p'_{ed(m)}\right). \tag{9.2}$$

Let $a(n) \in N_{d(n)-1}$ denote the direct ancestor node of a node $n \in N \setminus \{0\}$.

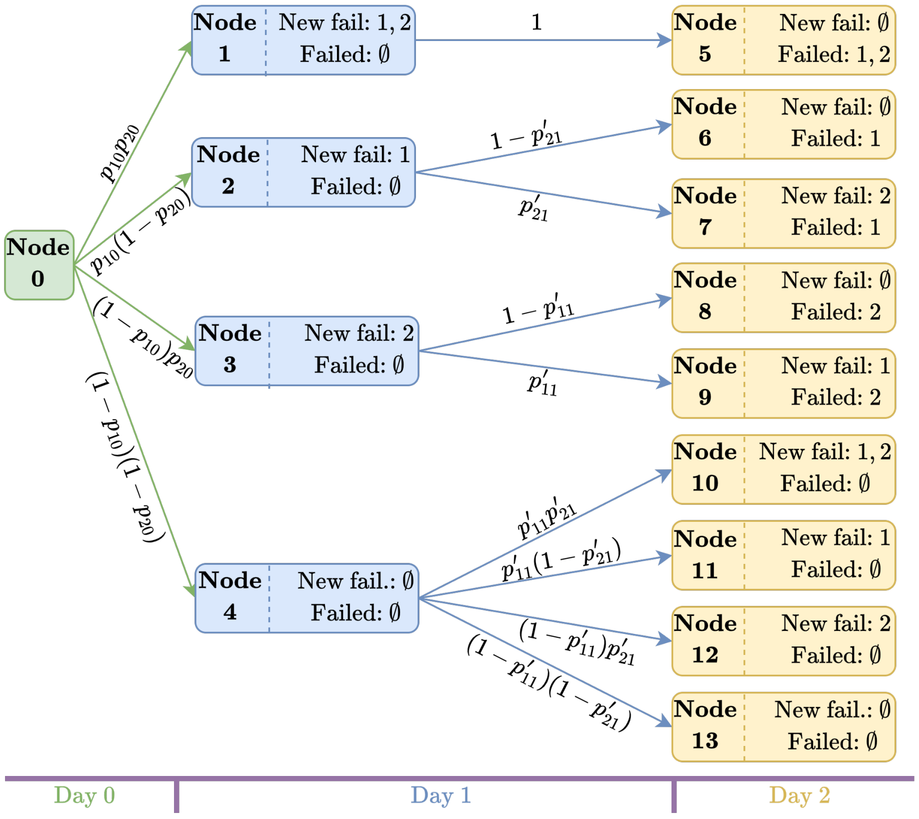

Node 0
Node 1 New fail: 1, 2 Failed: ∅
Node 2 New fail: 1 Failed: ∅
Node 3 New fail: 2 Failed: ∅
Node 4 New fail.: ∅ Failed: ∅
Node 5 New fail: ∅ Failed: 1, 2
Node 6 New fail: ∅ Failed: 1
Node 7 New fail: 2 Failed: 1
Node 8 New fail: ∅ Failed: 2
Node 9 New fail: 1 Failed: 2
Node 10 New fail: 1, 2 Failed: ∅
Node 11 New fail: 1 Failed: ∅
Node 12 New fail: 2 Failed: ∅
Node 13 New fail.: ∅ Failed: ∅
$p_{10}p_{20}$
$p_{10}(1-p_{20})$
$(1-p_{10})p_{20}$
$(1-p_{10})(1-p_{20})$
1
$1-p'_{21}$
p'_{21}
$1-p'_{11}$
p'_{11}
$p'_{11}p'_{21}$
$p'_{11}(1-p'_{21})$
$(1-p'_{11})p'_{21}$
$(1-p'_{11})(1-p'_{21})$
Day 0
Day 1
Day 2

Figure 9.1.: Example of a scenario tree for two systems $E = \{1,2\}$, and with three days $d \in [0,1,2]$. On the arcs are the probabilities to go from one node to the other node.

9.3.2. Model formulation

Variables

The main decision variable x^n_{ed} denotes, at node n, at which day $d \geq d(n)$ we plan to replace system e. This variable is defined for each node $n \in N$, each system $e \in E$ and each day $d \in [d(n), d(n)+1, \dots, D]$:

$$x^n_{ed} = \begin{cases} 1, & \text{It is planned at node } n \text{ that system } e \text{ will be replaced at day } d, \\ 0, & \text{Otherwise.} \end{cases} \tag{9.3}$$

At each node $n \in N \setminus \{0\}$, we introduce auxiliary variables that cancel planned replacements or that insert new replacements. Both variables are defined for each node $n \in N \setminus \{0\}$, each system $e \in E$, and each day $d \in [d(n), d(n)+1, \dots, D]$:

$$i_{ed}^{n} = \begin{cases} 1, & \text{At node } n\text{, a replacement for system } e \text{ is planned at day } d\text{, i.e.,} \\ & x_{ed}^{n} = 1\text{. This replacement was not planned for system } e \text{ at day } d \\ & \text{at the ancestor node } a(n)\text{, i.e., } x_{ed}^{a(n)} = 0. \\ 0, & \text{Otherwise.} \end{cases} \tag{9.4}$$

$$k_{ed}^{n} = \begin{cases} 1, & \text{The replacement of system } e\text{, planned at day } d \text{ at the ancestor} \\ & \text{node } a(n) \text{ (i.e., } x_{ed}^{a(n)} = 1\text{), is cancelled at node } n\text{, i.e., } x_{ed}^{n} = 0. \\ 0, & \text{Otherwise.} \end{cases} \tag{9.5}$$

We also define an auxiliary variable indicating whether system e has already been replaced at node n. This variable is defined for each node $n \in N$ and each system $e \in E$ as follows:

$$r_{e}^{n} = \begin{cases} 1, & \text{System } e \text{ has been replaced in the past at node } n. \\ 0, & \text{Otherwise.} \end{cases} \tag{9.6}$$

Last, we define for each node $n \in N$, the number of spare systems in stock at day $d \in [d(n), d(n)+1, \dots, D]$:

$$v_{d}^{n} = \text{Number of available spares in stock at node } n \text{ at day } d, \tag{9.7}$$

where v_{d}^{n} is an integer variable. Similarly, we define for each node $n \in N$ the left-over capacity at day $d \in [d(n), d(n)+1, \dots, D]$, i.e., the number of replacements that can still be made per day:

$$w_{d}^{n} = \text{Remaining capacity at node } n \text{ at day } d, \tag{9.8}$$

where w_{d}^{n} is an integer variable as well.

MODEL FORMULATION AT THE ROOT NODE

For the root node 0, we consider the following integer linear program:

$$\begin{aligned} \text{Min. } & v^{0} && && (9.9) \\ \text{s.t. } & r_{e}^{0} = 0 && \forall e \in E && (9.10) \\ & v_{d}^{0} + \sum_{e \in E} x_{e0}^{0} = s && \forall d \in [0, 1, \dots, d^{s}-1] && (9.11) \\ & v_{d}^{0} = s && \forall d \in [d^{s}, d^{s}+1, \dots, D] && (9.12) \\ & \sum_{e \in E} \sum_{d'=\max(0, d-d^{s}+1)}^{d} x_{ed'}^{0} \le s && \forall d \in [0, 1, \dots, D] && (9.13) \\ & w_{d}^{0} + \sum_{e \in E} x_{e0}^{0} = g && \forall d \in [0, 1, \dots, d^{g}-1] && (9.14) \\ & w_{d}^{0} = g && \forall d \in [d^{g}, d^{g}+1, \dots, D] && (9.15) \end{aligned}$$

$$\sum_{e \in E} \sum_{d'=\max(0,d-d^g+1)}^{d} x^0_{ed'} \leq g \qquad \forall d \in [0,1,\dots,D]. \tag{9.16}$$

$$v^0_d \in \{0,1,\dots,s\}, w^0_d \in \{0,1,\dots,g\} \qquad \forall d \in [0,\dots,D] \tag{9.17}$$

$$x^0_{ed} \in \{0,1\} \qquad \forall e \in E, \forall d \in [0,\dots,D] \tag{9.18}$$

v^0 denotes the objective function, which is discussed below. Eq. (9.10) defines whether or not a system has already been replaced. We assume that at the root node, no system is replaced yet.

Eq. (9.11) and eq. (9.12) define the number of spares available in stock at all future days. Here, only the planned replacements at the current day 0 are considered, since all replacements planned at future days may still be cancelled. If we replace a system e at day d, then it comes back in stock as spare on day $d + d^s$. The number of spares is thus s after d^s days (eq. (9.12)), and s minus the number of systems we replace at day 0 before day d^s (eq. (9.11)). Last, eq. (9.13) ensures that we cannot plan more replacements then there are spare systems in stock, i.e., we can schedule at most s replacements per d^s days.

Eq. (9.14) and (9.15) define the remaining capacity. Again, only the planned replacements at current day 0 are taken into account. Each replacement takes d^g days. The remaining capacity is therefore g after d^g days (eq. (9.15)), and g minus the number of systems replaced at day 0 before day d^g (eq. (9.14)). Eq. (9.16) ensures that we cannot exceed the available capacity, i.e., we can schedule at most g replacements per d^g days.

Objective function Let y^n denote the set of decision variables of the linear program at a node $n \in N$. For any node $n \in N \setminus \{0\}$, the objective function is a function of the decision variables $y^{a(n)}$ at the ancestor node $a(n)$. Let $v^n(y^{a(n)})$ denote the objective function of node $n \in N \setminus \{0\}$. The objective function v^0 at the root node is then:

$$v^0 = \sum_{e \in E} \frac{c_r}{0 + I_e} x^0_{e0} + \sum_{m \in S(0)} q_{0m} v^m(y^0). \tag{9.19}$$

Eq. (9.19) consists of the replacement costs and the expected future costs. We only add the replacement costs c_r for replacements that are planned at day 0, since replacements planned at future days can still be cancelled. We scale these replacement costs by the lifetime of the system, i.e., we minimize the costs per day a system has been used.

Model formulation at the other nodes

For any node $n \in N \setminus \{0\}$, we consider the following integer linear program:

$$\text{Min. } v^n\left(y^{a(n)}\right) \tag{9.20}$$

$$\text{s.t. } x^n_{ed} - i^n_{ed} \leq x^{a(n)}_{ed} \qquad \forall e \in E, \forall d \in [d(n),\dots,D] \tag{9.21}$$

$$x^{a(n)}_{ed} - k^n_{ed} \leq x^n_{ed} \qquad \forall e \in E, \forall d \in [d(n),\dots,D] \tag{9.22}$$

$$r^n_e = r^{a(n)}_e + x^{a(n)}_{ed(a(n))} \qquad \forall e \in E \tag{9.23}$$

$$x^n_{ed} + \left(r^{a(n)}_e + x^{a(n)}_{ed(a(n))}\right) \leq 1 \qquad \forall e \in E, \forall d \in [d(n), \ldots, D] \quad (9.24)$$

$$v^n_d + \sum_{e \in E} x^n_{ed(n)} = v^{a(n)}_d \qquad \forall d \in [d(n), \ldots, \min(d(n) + d^s - 1, D)] \quad (9.25)$$

$$v^n_d = s \qquad \forall d \in [d(n) + d^s, \ldots, D] \quad (9.26)$$

$$\sum_{e \in E} \sum_{d'=\max(d(n), d-d^s+1)}^{d} x^n_{ed'} \leq v^{a(n)}_d \qquad \forall d \in [d(n), \ldots, D] \quad (9.27)$$

$$w^n_d + \sum_{e \in E} x^n_{ed(n)} = w^{a(n)}_d \qquad \forall d \in [d(n), \ldots, \min(d(n) + d^g - 1, D)] \quad (9.28)$$

$$w^n_d = g \qquad \forall d \in [d(n) + d^g, \ldots, D] \quad (9.29)$$

$$\sum_{e \in E} \sum_{d'=\max(d(n), d-d^g+1)}^{d} x^n_{ed'} \leq w^{a(n)}_d \qquad \forall d \in [d(n), \ldots, D]. \quad (9.30)$$

$$v^n_d \in \{0, 1, \ldots, s\}, w^n_d \in \{0, 1, \ldots, g\} \qquad \forall d \in [d(n), \ldots, D] \quad (9.31)$$

$$x^n_{ed}, i^n_{ed}, k^n_{ed} \in \{0, 1\} \qquad \forall e \in E, \forall d \in [d(n), \ldots, D] \quad (9.32)$$

$$r^n_e \in \{0, 1\} \qquad \forall e \in E. \quad (9.33)$$

$v^n\left(y^{a(n)}\right)$ denotes the objective function, which is discussed below. Eq. (9.21) and (9.22) are new, compared to the model at the root node, and define the variable for inserting new and cancelling previously planned replacements, respectively. Here, we enforce $i^n_{ed} = 1$ if we plan to replace system e at day d ($x^n_{ed} = 1$), while we did not plan this at the ancestor node $a(n)$ ($x^{a(n)}_{ed} = 0$). Similarly, $k^n_{ed} = 1$ if we plan at the ancestor node $a(n)$ to replace system e at day d ($x^{a(n)}_{ed} = 1$), while we do not plan this anymore at node n ($x^n_{ed} = 0$). Since the cost c_i and c_k associated with inserting and cancelling a replacement are strictly positive, i^n_{ed} is zero in the optimal solution if no new replacement is inserted, while k^n_{ed} is zero in the optimal solution if we do not cancel a planned replacement.

Eq. (9.23) defines the variable r^n_e. A system e is already replaced at node n ($r^n_e = 1$) if: i) it is already replaced at the ancestor node $a(n)$ ($r^{a(n)}_e = 1$), or ii) if it is replaced at day $d(n) - 1$ at the ancestor node $a(n)$ ($x^{a(n)}_{ed(a(n))} = 1$). Eq. (9.24) ensures that each system is replaced at most once: All future planned replacements are cancelled as soon as a system is replaced.

Eqs. (9.25) and (9.26) define the number of available spare systems. These constraints are similar to the constraints on the spares at the root node. However, the number of available spares now depends on the number of available spares $v^{a(n)}_d$ at the ancestor node $a(n)$. Constraint (9.27) defines, as before, that we cannot plan more replacements than the number of available spare systems. Similarly, eq. (9.28) and (9.29) define the remaining capacity, which depends on the remaining capacity $w^{a(n)}_d$ at the ancestor node $a(n)$. Eq. (9.30) ensures that we cannot plan more replacements than the available capacity.

Objective function: The objective function $v^n(y^{a(n)})$ for node n is:

$$v^n\left(y^{a(n)}\right) = \sum_{e \in E^n_{\text{new}}} \left(1 - r^n_e\right) \frac{c_f}{d(n) + I_e} \quad (9.34)$$

$$+ \sum_{e\in E^n_{\text{new}}\cup E^n_{\text{fail}}} \left(1-r^n_e - x^n_{ed(n)}\right)\frac{c_l}{d^{\text{fail}}_e(n)+I_e} + x^n_{ed(n)}\frac{c_r}{d^{\text{fail}}_e(n)+I_e}$$

$$+ \sum_{e\in E\setminus(E^n_{\text{fail}}\cup E^n_{\text{new}})} x^n_{ed(n)}\frac{c_r}{d(n)+I_e} + \sum_{e\in E}\sum_{d=d(n)}^{D}\left(c_i i^n_{ed}+c_k k^n_{ed}\right)$$

$$+ \sum_{m\in S(n)} q_{nm} v^m\left(y^n\right).$$

The first term of this objective function contains the cost of new failures. Here, we do not incur any failure cost if system e is already replaced before node n ($r^n_e = 1$). The second term of the objective contains the cost of i) systems that remain failed for an extra day, without being replaced, and ii) the cost of replacing failed systems. The third term contains the cost of replacing systems that have not failed yet. As before, we only add the cost of systems that are replaced at day $d(n)$, since all replacements planned at future days can still be cancelled. We scale these three cost terms by the number of days a system has been used.

The fourth term of the objective contains the cost of cancelling and inserting replacements. If system e has not failed and is not replaced yet, we do not know the number of days the system e will be used. These are therefore the only costs that we do not scale by the number of days a system is used. The last term contains the expected future costs of node n. This term is not included in the objective function of the leaf nodes at day D.

A shorter (schematic) way to denote the objective function $v^n(y^{a(n)})$ at node n is:

$$v^n\left(y^{a(n)}\right) = C^n + \left(c^n\right)^T y^n + Q^n\left(y^n\right), \tag{9.35}$$

where T denotes the transpose. The first part C^n is the constant term, while $Q^n(y^n)$ denotes the future expected costs:

$$C^n = \sum_{e\in E^n_{\text{new}}} \frac{c_f}{d(n)+I_e} + \sum_{e\in E^n_{\text{new}}\cup E^n_{\text{fail}}} \frac{c_l}{d^{\text{fail}}_e(n)+I_e}, \tag{9.36}$$

$$Q^n\left(y^n\right) = \sum_{m\in S(n)} q_{nm} v^m\left(y^n\right). \tag{9.37}$$

Last, $(c^n)^T y^n$ contains all remaining terms of the objective function i.e., the cost terms (in the vector c^n) multiplied by the decision variables y^n. The objective of a leaf node $n \in N_D$ consists of the first two terms ($C^n + (c^n)^T y^n$) only.

The uncertainty in this problem is only in the objective function of the linear programs [21], which depends on the probabilistic RUL of the systems. In contrast, the constraint matrix and right-hand side of the linear programs only depend on the decisions at the ancestor node, and not on any random parameters.

9.3.3. Endogenous uncertainty and non-anticipativity

The uncertainty in our problem is endogenous (type 2), since the decision to replace a system or not affects the PDF of the RUL. Before replacing a system, the probability of failure is estimated based on the sensor measurements. After replacing a system, we assume that the probability of failure is zero at all future days.

For instance, assume that we replace system 1 at day 0 in Figure 9.1, and that we observe at day 1 that system 2 fails. We then do not know if we are in node 1 (system 2 fails, system 1 would have failed if we had not replaced it) or in node 3 (only system 2 fails), i.e., we do not observe the RUL of system 1. The optimal solution at node 1 and node 3 should therefore not depend on the unobserved RUL of the replaced system 1. With endogenous uncertainty type 2, this is usually enforced with non-anticipativity constraints [7, 8]. However, in our problem, this is not necessary. We instead proof the following theorem in Appendix 9.A:

Theorem 1: For the considered problem, the optimal solution at a node n does not depend on the (unobserved) RUL of the replaced systems.

9.3.4. Totally unimodular constraint matrix

The linear program in each node contains integer variables. If the constraint matrix of an integer program is totally unimodular, and if the right-hand side coefficients are integral, then every extreme point of the feasible region is integral [22]. The optimal solution of the relaxation of the integer linear program (found with the simplex method), is thus integer. In Appendix 9.B, we proof the following theorem:

Theorem 2: For the considered problem, it holds that:

- The constraint matrix of the linear program at any node $n \in N$ is totally unimodular
- In the optimal solution of the stochastic program, the right-hand side of each linear program is integer.

When solving the linear relaxation of the integer program (with the simplex method), the optimal solution will thus be integer.

To ensure that the constraint matrix is totally unimodular, we allow that multiple replacements are planned for a single system. However, in constraint (9.24), we do restrict that at most one replacement is executed per system. In general, it is not desirable to plan multiple replacements and subsequently cancel them. To address this, we set the cancellation costs c_k larger than the inserting costs c_i ($c_k > c_i$), such that it is not optimal to plan multiple replacements.

9.4. Nested Benders decomposition and a novel clustering algorithm

Our problem is sufficiently expensive (the objective value of any node $n \in N$ cannot go to $-\infty$) and complete (the objective value of any node $n \in N$ cannot go to ∞). We thus solve the relaxation of the multi-stage stochastic integer program using nested Benders decomposition [17, 18], also called the nested L-shaped method. We first shortly introduce this method in the context of our problem, and we then propose our new dynamic clustering algorithm, to reduce the computational time.

9.4.1. Nested Benders decomposition

The difficult term in a stochastic program are the expected future costs $Q^n(y^n)$. In the nested Benders decomposition, $Q^n(y^n)$ is therefore replaced by a variable θ^n for each node $n \in N \setminus N_D$. We let the variable θ^n converge to $Q^n(y^n)$ by iteratively adding constraints on θ^n, called the "optimality cuts", to the linear program. To start, we add the constraint $\theta^n \geq 0$ to each node $n \in N \setminus N_D$, where 0 is a trivial lower bound for the expected future costs. At each node $n \in N$, we always have a feasible solution: We can always cancel all planned replacements and plan no new replacements. It is thus not necessary to derive feasibility cuts.

We move through the scenario tree using the "Fast-Forward-Fast-Backward" (FFFB) heuristic [18]. We start in the "forward mode" (with the initial constraint that $\theta^n \geq 0$ to each node $n \in N \setminus N_D$) and solve the linear program of the root node, of each node at day 1, etc., until we have solved the linear program of each leaf node at day D. Here, we take all optimality cuts added to the linear program, which is only the trivial constraint $\theta^n \geq 0$ in the first iteration, into account. We then switch to the "backward mode", and use the solution of the leaf nodes to derive a new optimality cut for each node at day $D-1$, and solve the linear program at each node at day $D-1$ with this new optimality cut. This continues until we have added a new optimality cut to the root node. We then switch again to the forward mode and repeat the procedure, until we have found an optimal solution. Each switch in direction is counted as a new iteration.

Let $\hat{\pi}^{m,i}$ denote the optimal dual variables of the linear program at a successor node $m \in S(n)$ of node n in iteration i of the nested Benders decomposition. The optimality cut that we add to node n is:

$$\theta^n \geq \sum_{m \in S(n)} q_{nm} \left(C^m + \left(\hat{\pi}^{m,i}\right)^T \left(h^m - T^m y^n\right) \right). \tag{9.38}$$

It is only useful to add this cut if the optimal value of θ^n at iteration $i-1$, $\hat{\theta}^{n,i-1}$, does not fulfill this constraint yet. Here, we use an optimality tolerance of ϵ, where ϵ is a very small number. We thus only add the cut if $\hat{\theta}^{n,i-1} + \epsilon$ is strictly smaller than the value of the cut in iteration $i-1$. We find an optimal solution (given the optimality tolerance), and thus terminate the algorithm, if we move backward from the leaf nodes to the root node without adding any optimality cuts.

By adding the optimality cuts to the linear programs, the constraint matrices are not totally unimodular anymore. The solutions we find at each node during the algorithm are thus not necessary integer. However, with this algorithm, we approximate the optimal solution. If there is one unique optimal solution, this optimal solution is integer [23], with a tolerance of ϵ. If there are multiple optimal solutions, then the obtained optimal solution is not necessarily integer [23]. In this case, we can solve our problem once with integer constraints. However, we expect that this is highly unlikely for the considered problem, and we indeed obtain only integer solutions in the case study.

Multicut strategy

The optimality cut in eq. (9.38) is derived with the "unicut" strategy, where only one optimality cut at the time is added to a node n. Another strategy to derive optimality cuts is the multicut strategy [24]. With this strategy, we add one variable θ^n_m to the linear program at node n for each successor node $m \in S(n)$. The objective at node n becomes to minimize $C^n + (c^n)^T y^n + \sum_{m \in S(n)} \theta^n_m$. We add an optimality cut on θ^n_m to node n for each successor node $m \in S(n)$ when going backward:

$$\theta^n_m \geq q_{nm}\left(C^m + (\hat{\pi}^{m,i})^T (h^m - T^m y^n)\right). \tag{9.39}$$

As before, we only add this cut if it is not yet fulfilled in the current solution, with an optimality tolerance of ϵ.

9.4.2. A dynamic clustering algorithm under endogenous uncertainty

The number of nodes in the scenario tree grows exponentially with the number of systems and days. In this section, we therefore use the endogenous uncertainty of our problem to reduce the number of linear programs we have to solve in each iteration i of the nested Benders decomposition.

For instance, assume that in iteration i of the nested Benders decomposition, we replace system 1 at the root node in the example in Figure 9.2. In Appendix 9.A, we show that then, the optimal solution of node 1 and 3 (or any other node) does not depend on the unobserved RUL of system 1. Moreover, node 1 and 3 have the same ancestor node (the root node), and are therefore indistinguishable [8] in this example. The optimal solutions are therefore equal, given the independence of the optimal solution on the unobserved RUL of system 1. If the optimal solutions of node 1 and 3 are the same, we suspect that the linear program is the same as well. Also the optimal solutions of node 5, 8 and 9 do not depend on the unobserved RUL of system 1. Since the optimal solutions of the ancestor nodes 1 and 3 are equal, we expect that the optimal solutions of node 5,8 and 9 are equal as well. If the linear programs of node 1 and 3 (or node 5, 8 and 9) are indeed the same in iteration i, we can group node 1 and 3 (or node 5, 8 and 9) together in one cluster in iteration i. Then, we need to solve only one linear program per cluster.

Definition of a cluster

We cluster nodes in the scenario tree in each iteration i of the nested Benders decomposition, based on the replacement decisions at iteration i. Let $E^{n,i}_{\text{rep}}$ denote the set of systems that are already replaced at node n in iteration i, i.e., $r^n_e = 1\ \forall e \in E^{n,i}_{\text{rep}}$, while $r^n_e \neq 1\ \forall e \in E \setminus E^{n,i}_{\text{rep}}$. A cluster $\alpha^i \subseteq N$ at iteration i is a set of nodes, where any node $n, m \in \alpha^i$ fulfill the following five conditions C1-C5:

- C1 The days of the nodes are the same: $d(n) = d(m)$.
- C2 The set of replaced systems at iteration i is the same: $E^{n,i}_{\text{rep}} = E^{m,i}_{\text{rep}}$.
- C3 Each replaced system is replaced at the same day in iteration i.

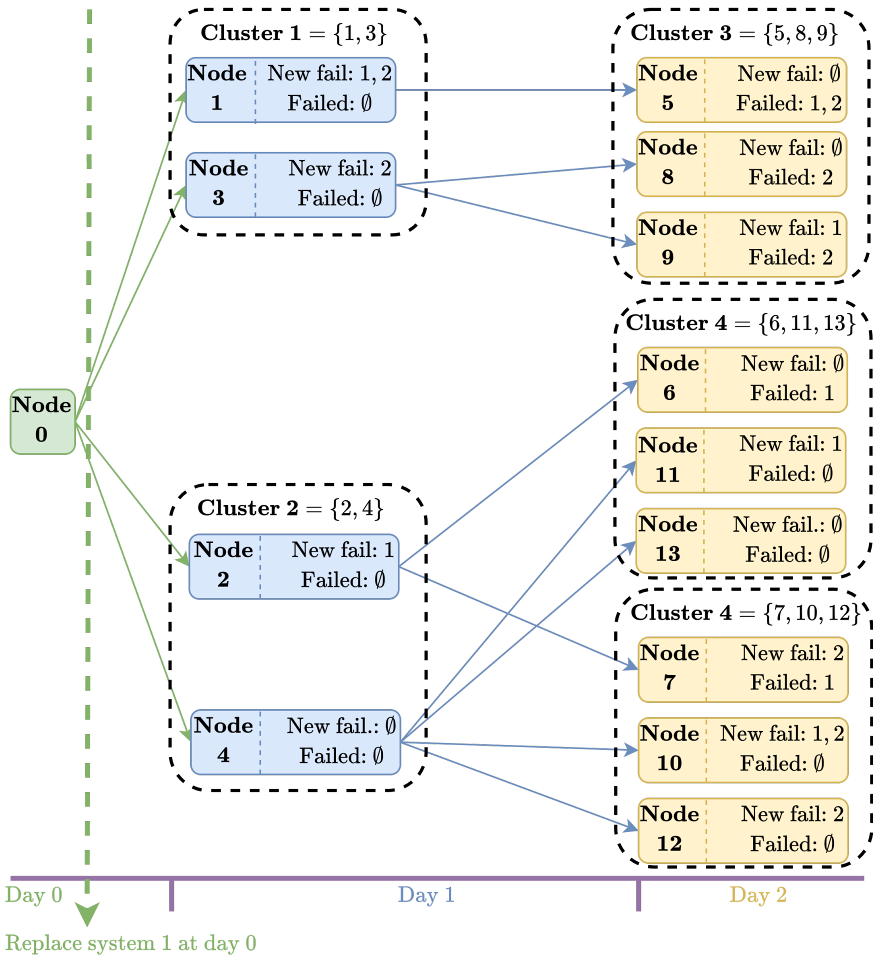

Cluster 1 = {1, 3}
Node 1 New fail: 1, 2 Failed: ∅
Node 3 New fail: 2 Failed: ∅
Cluster 3 = {5, 8, 9}
Node 5 New fail: ∅ Failed: 1, 2
Node 8 New fail: ∅ Failed: 2
Node 9 New fail: 1 Failed: 2
Cluster 4 = {6, 11, 13}
Node 6 New fail: ∅ Failed: 1
Node 11 New fail: 1 Failed: ∅
Node 13 New fail.: ∅ Failed: ∅
Node 0
Cluster 2 = {2, 4}
Node 2 New fail: 1 Failed: ∅
Node 4 New fail.: ∅ Failed: ∅
Cluster 4 = {7, 10, 12}
Node 7 New fail: 2 Failed: 1
Node 10 New fail: 1, 2 Failed: ∅
Node 12 New fail: 2 Failed: ∅
Day 0
Day 1
Day 2
Replace system 1 at day 0

Figure 9.2.: Example of clustering in a scenario tree with two systems $E = \{1,2\}$ and three days $d \in \{0,1,2\}$, where system 1 is replaced at day $d = 0$.

C4 If a replaced system $e \in E^{n,i}_{\text{rep}}$ has, at node n, failed before or at the day it is replaced in iteration i, then i) $e \in E^{m}_{\text{fail}}$ and ii) $d^{\text{fail}}_{e}(n) = d^{\text{fail}}_{e}(m)$. In other words, if the RUL of a replaced system is observed in one node, then it is also observed in the other node and it is the same.

C5 For each system e that is not yet replaced at node n and m in iteration i, the failure history is the same. For each system $e \in E \setminus E^{n,i}_{\text{rep}}$, one of the following three conditions holds:

C5a System e fails at day $d(n)$ in both node n and node m, i.e., $e \in E^{n}_{\text{new}}$ if and only if $e \in E^{m}_{\text{new}}$. This implies that $E^{n}_{\text{new}} \setminus E^{n,i}_{\text{rep}} = E^{m}_{\text{new}} \setminus E^{m,i}_{\text{rep}}$. Let $E^{\alpha^i}_{\text{new}} = E^{z}_{\text{new}} \setminus E^{z,i}_{\text{rep}}$, with z any node in α^i.

C5b System e has already failed at node n and node m, with the same day of failure, i.e., $e \in E^n_{\text{fail}}$ if and only if $e \in E^m_{\text{fail}}$, with $d^{\text{fail}}_e(n) = d^{\text{fail}}_e(m)$. This implies that $E^n_{\text{fail}} \setminus E^{n,i}_{\text{rep}} = E^m_{\text{fail}} \setminus E^{m,i}_{\text{rep}}$.

C5c System e is still working at node n and node m, i.e., if $e \notin E^n_{\text{fail}} \cup E^n_{\text{new}}$ if and only if $e \notin E^m_{\text{fail}} \cup E^m_{\text{new}}$. This implies that $(E \setminus E^{n,i}_{\text{rep}}) \setminus (E^n_{\text{fail}} \cup E^n_{\text{new}}) = (E \setminus E^{m,i}_{\text{rep}}) \setminus (E^m_{\text{fail}} \cup E^m_{\text{new}})$. Let $E^{\alpha^i}_{\text{work}} = (E \setminus E^{z,i}_{\text{rep}}) \setminus (E^z_{\text{fail}} \cup E^z_{\text{new}})$ be the set of working, non-replaced systems of cluster α^i, with z any node in α^i.

In other words, the RUL of a non-replaced system is either i) not observed in both nodes, or ii) is observed in both nodes and is the same.

Two nodes in the same cluster are indistinguishable in iteration i (for the definition of indistinguishable, see [8]). Using this, we proof the following theorem in Appendix 9.C:

Theorem 3: The integer linear program of two nodes n, m in the same cluster α^i in iteration i of the nested Benders decomposition, is the same during this iteration.

Nested Benders decomposition with the clustering algorithm

Assume that we are at the root node at iteration i of the nested Benders decomposition, and that we move forward through the tree. The solution of the linear relaxation of the integer program at the root node, or at any other node, is not necessarily integer due to the optimality cuts. However, we cluster based on the set of replaced systems $E^{n,i}_{\text{rep}}$, where a system e is only in $E^{n,i}_{\text{rep}}$ if $r^n_e = 1$ at iteration i. With non-integer replacement decisions, the set $E^{n,i}_{\text{rep}}$ is empty, and each cluster exists of a just single node.

To prevent this, we group the integer programs of the nodes belonging to day 0 up to day δ of the planning horizon together in one large integer program. This large integer program is the large-scale deterministic equivalent (LDE) of our problem up to day δ. Such grouping is often performed in multi-stage stochastic programs [18]. When we are at the root node, in the forward mode, we first solve this large integer program up to day δ. Here, we do not consider the linear relaxation, i.e., we impose that the decision variables are integer. The Benders decomposition algorithm was originally developed for solving first-stage (master) integer programs, so this algorithm still works with this restriction [17].

Because we now impose that the solution has to be integer, the constraint matrix of the integer program up to day δ does not have to be totally unimodular anymore. Since we set $c_k > c_i$, it is optimal to plan at most one replacement per system. To guide our program to the optimal solution, we add the valid inequality that at most one replacement per system is planned at each node to the large integer program belonging to day 0 up to day δ:

$$\sum_{d=d(n)}^{D} x^n_{ed} \leq 1 \qquad \forall e \in E, \forall d \in [0, 1, \ldots, \delta], \forall n \in N_d. \tag{9.40}$$

Based on the integer solution from day 0 up to day δ, we cluster all nodes after day δ. Here, we put two nodes in the same cluster if they fulfill conditions C1 - C5.

For instance, in the example in Figure 9.2, we consider $\delta = 0$. The optimal solution at the root node is to replace system 1 at day 0. With this replacement, we create two clusters at day 1, and 3 clusters at day 2.

Clustering algorithm and the unicut strategy After solving the integer program up to day δ and the clustering, we move forward to day $\delta+1$. Within one cluster α^i at day $\delta+1$, the linear program, including the expected future costs, of all nodes is the same (only in iteration i). In the nested Benders decomposition, the expected future costs at node n are approximated by θ^n. The optimality cuts at node n then give lower bounds on θ^n. Since the expected future costs are the same for any node $n \in \alpha^i$, these lower bounds are valid for the expected future costs of all nodes $n \in \alpha^i$ (again, only in iteration i). We thus form one linear program for cluster α^i, which is the same as the linear program of any node $n \in \alpha^i$. We replace the expected future costs in this linear program of cluster α^i by θ, and add all optimality cuts of all nodes $n \in \alpha^i$ as constraints on θ. An additional advantage of this is that sharing cuts may accelerate the convergence of the nested Benders decomposition [25].

After solving one linear program for each cluster at day $\delta+1$, we solve one linear program per cluster at day $\delta+2$, etc., until we reach the last day D. We again solve the linear relaxation of the integer program for any day $d > \delta$. Due to the optimality cuts, the solutions are not necessarily integer. The clusters are thus only made based on the integer solution up to day δ.

After reaching day D, we continue with iteration $i+1$ and move backwards through the scenario tree. For a cluster α^i at day $d < D$, a “successor cluster” α'^i is a cluster at day $d+1$, for which there exists a least one node $n' \in \alpha'^i$ that is a successor node of any node $n \in \alpha^i$ (see the proof in Appendix 9.C, eq. (9.47)). For each cluster at day $D-1$, we derive an optimality cut on θ using the optimal solution of each successor cluster. To derive this cut, we use the probability to go from any node in cluster α^i to any node in successor cluster α'^i (see eq. (9.63) in Appendix 9.C). The optimality cut gives a lower bound on the expected future costs in iteration i at cluster α^i. As these expected future costs are the same for all nodes $n \in \alpha^i$ (in iteration i), this cut provides a valid lower bound on θ^n for any node $n \in \alpha^i$. We add this cut to all nodes $n \in \alpha^i$ as constraint on θ^n. We go backward until we have added optimality cuts to the large integer program belonging to day 0 up to day δ. We then continue with iteration $i+2$, moving forward again. Note that the clustering is dynamic, i.e, the solution of the integer program from day 0 up to day δ, and thus the clusters, may be different in iteration $i+2$ than in iteration i.

Clustering algorithm and the multicut strategy The clustering is different with the multicut strategy than with the unicut strategy. In the multicut strategy, we add one variable θ_m for each successor node $m \in S(n)$, for all nodes $n \in \alpha^i$, to the linear program of a cluster α^i. In the objective function of cluster α^i, we add $\tilde{q}_{n\alpha^i}\theta_m$ for each node $m \in S(n)$ and each node $n \in \alpha^i$. Here, $\tilde{q}_{n\alpha^i}$ denotes the probability that we are at node n if we are in cluster α^i in iteration i. When we go backward through the scenario tree, we do not consider the clusters anymore. Instead, we derive an optimality cut for θ_m^n for each node n as before (eq. (9.39)).

9.5. Case study and results - Maintenance scheduling of aircraft engines

In this section we illustrate the stochastic program introduced in Section 9.3 to plan the replacement of aircraft engines. We first estimate the Remaining Useful Life (RUL) of these engines.

9.5.1. Probabilistic RUL prognostics for aircraft engines

We apply the proposed methodology to the predictive maintenance planning for the aircraft engines in the C-MAPSS dataset [26, 27]. In this section, we shortly discuss how we estimate the PDF of the RUL of these engines (i.e., the probabilistic RUL prognostics) following [19].

	FD001	FD002	FD003	FD004
Training instances	100	260	100	249
Testing instances	100	259	100	248
Operating conditions	1	6	1	6
Fault modes	1	1	2	2

Table 9.1.: C-MAPSS datasets for turbofan engines from [26, 27].

The C-MAPSS dataset contains the simulated measurements of 21 sensors. For each engine, one measurement per sensor per flight is available. The data of the C-MAPSS dataset is divided into four subsets with different operating conditions and fault modes, named FD001, FD002, FD003 and FD004 (see Table 9.1). Each subset is in turn divided into a training and a test set. Each test set contains engines for which the measurements stop at some point before failure. We estimate the RUL of the engine at this point.

To estimate the RUL, we use the Convolutional Neural Network (CNN) introduced in [19], consisting of 5 convolutional layers and 2 fully connected layers. We use the same architecture, the same sensor selection, the same hyperparameters and the same optimization tools as in [19] (see Section 3.1 in [19]). In the test phase, we estimate the PDF of the RUL using Monte Carlo dropout [19, 28], with a dropout rate of 0.5 in the first fully connected layer.

Table 9.2 shows the the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) [3] with the mean estimated RUL of the test engines. The MAE is between 8.6 to 11.1 flights and the RMSE is between 12.1 to 15.6 flights. This is comparable to the prognostics in state-of-the-art papers (see [19]).

	FD001	FD002	FD003	FD004
RMSE (flights)	12.2	13.8	12.1	15.6
MAE (flights)	9.0	9.8	8.6	11.1

Table 9.2.: Metrics of the mean estimated RUL for the test engines of the four subsets of C-MAPSS.

The obtained RUL prognostic is in flights and not in days. We assume that each aircraft performs two flights per day. Let f_d^1 and f_d^2 denote the first and second flight of a single aircraft during day d. The probability p_{ed} that the RUL of engine e equals d days, is the probability that the RUL equals f_d^1 flights plus the probability that the RUL equals f_d^2 flights.

9.5.2. Maintenance scheduling - Description of the parameters

Parameter	Description	Value
	Problem parameters	
D	Last day of the maintenance planning	27
d^g	Number of days it takes to replace an engine	3
g	Maintenance capacity	1
d^s	Number of days it takes to externally repair an engine	15
s	Initial number of spare engines	2 or 3
	Costs (euros)	
c_r	Cost of replacing an engine	20,000
c_f	Cost of an engine failure (once)	20,000
c_l	Cost per day an engine remains failed	100,000
c_i	Cost of inserting a new replacement in the planning	2
c_k	Cost of cancelling a planned replacement	1
	Parameters of the solution method	
ϵ	Optimality tolerance	10^{-5}
δ	Last day we group at the root node	9

Table 9.3.: Overview of the considered parameters.

An overview of the considered parameters is in Table 9.3. We construct a maintenance planning for four weeks, from day 0 up to day $D = 27$ (i.e., 28 days/stages). Depending on the number of engines in the case study, we consider an initial stock of $s = 2$ or $s = 3$ spare engines.

We consider a cost of $c_r = 20,000$ for replacing an engine, and a cost of $c_f = 20,000$ for an engine failure. Once an engine has failed, the aircraft cannot be used anymore until it undergoes maintenance. This may lead to flight delays and flight cancellations, which are very expensive. We therefore consider a cost of $c_l = 100,000$ for each day an engine remains failed.

We also consider a small penalty of $c_i = 1$ and $c_k = 2$ for inserting or cancelling an engine replacement, respectively. We set $c_k > c_i$, to ensure that at most one replacement per engine is planned in the optimal solution. Recall that we scale the other costs by the lifetime of the engine. For instance, if an engine is replaced after 200 days, we add a cost of $c_r/200 = 100$ per day to the objective. In this example, replacing an engine is per day 100 times as expensive as inserting a replacement, and 50 times as expensive as cancelling a replacement.

Table 9.4 shows an overview of the engines for which we plan maintenance. The estimated PDFs of the RUL of the engines are in Figure 9.3. To ensure a diverse set

Engine name	Subset	Number of engine	Install day I_e	$p_{ed} > 0$ for the first time at day $d = \dots$	Mean estimated RUL (days)	Width of PDF of RUL (days)
1-24	FD001	24	-93	4	9	9
2-39	FD002	39	-75	16	22	13
3-32	FD003	32	-60	21	27	17
4-70	FD004	70	-95	13	18	13
1-49	FD001	49	-152	3	7	10

Table 9.4.: Overview of the engines that we consider in the maintenance planning.

of RUL prognostics, we have iteratively selected the engines from another subset. Moreover, we have chosen the engines such that it is optimal to replace them somewhere in the next four weeks. Last, the engines are selected such that the mean estimated RUL and the width of the PDF vary.

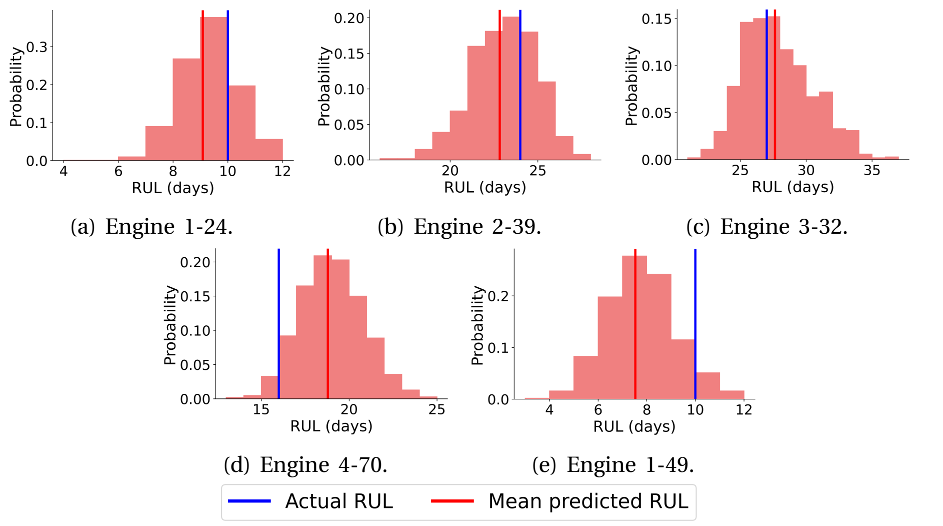


Figure 9.3.: PDF of the estimated RUL for the five considered engines.

9.5.3. Different solution strategies

We analyze the computational time of four different solution methods. We combine each solution method, except the LDE, with the unicut and multicut strategy. We implement each solution method in Python using Gurobi version 10, on a computer with 8GB of RAM memory and 4 Intel i7 (8th generation) CPU cores.

- **LDE:** For the large-scale deterministic equivalent (LDE) of the problem, we add all constraints and objective functions of all nodes to a single, very large, integer program (the LDE) and solve it.

- **Benders:** This method uses the nested Benders decomposition method (see Section 9.4.1).
- **Cluster:** This method uses the nested Benders decomposition method, combined with the clustering algorithm (see Section 9.4.2).
- **Group:** This method uses the nested Benders decomposition, where we also add the integer programs of the nodes from day 0 up to day δ together (as in the clustering algorithm, Section 9.4.2). However, we do *not* consider any clusters.

LOWER BOUND AND UPPER BOUND

As lower bound on the objective value of the stochastic program, we use the solution of the wait-and-see (WS) problem [17]. For the WS problem, we assume that we know at day 0 the true RUL with absolute certainty. Let LB denote the lower bound, i.e., the value of the optimal solution of the WS problem. Let z denote the corresponding optimal objective value of the multi-stage stochastic program. We define the Expected Value of Perfect Information (EVPI) [17] as percentage:

$$\text{EVPI} = 100 \cdot \frac{z - \text{LB}}{\text{LB}}. \tag{9.41}$$

The EVPI states how much costs (in percent) we would save by exactly knowing the true RUL without any uncertainty.

The expected value of the expected value (EV) problem (the EEV) is often used as upper bound of a stochastic program. In the EV problem, we replace the random RUL by the mean estimated RUL. The optimal solution would be to replace each engine just before the mean estimated RUL (if possible). The engines thus have a large probability of failure before being replaced, and the EEV is therefore high.

Instead, we propose a problem-specific method to find a tighter upper bound. Our problem is stochastic since we may change the maintenance planning (by cancelling or inserting replacements) after day 0. For the upper bound problem, we assume that we can only make an initial maintenance planning at day 0, and that this planning cannot be changed anymore. We thus do not update this maintenance planning at later days, even if an engine fails. The upper bound UB is then the expected value of the optimal solution of this upper bound problem. We define the Value of the Stochastic Solution (VSS) [17] as percentage:

$$\text{VSS} = 100 \cdot \frac{\text{UB} - z}{z}. \tag{9.42}$$

The VSS shows how much costs (in percent) we expect to save by solving the stochastic program instead of implementing the upper bound solution.

Before we solve the stochastic program, we first analyse the gap between the upper and lower bound (in percent):

$$\Delta(\text{UB} - \text{LB}) = 100 \cdot \frac{\text{UB} - \text{LB}}{\text{LB}}. \tag{9.43}$$

This gives an upper bound on both the VSS and the EVPI, i.e., VSS $\leq \Delta(\mathrm{UB}-\mathrm{LB})$ and EVPI$\leq \Delta(\mathrm{UB}-\mathrm{LB})$ [29]. It is only worthwhile to solve the stochastic program if the VSS is large. If $\Delta(\mathrm{UB}-\mathrm{LB})$ is small, then the VSS is also small, and we can simply solve the upper bound problem instead of the stochastic program.

9.5.4. Numerical results: Single engine maintenance scheduling

In this section, we analyse the results when planning maintenance for each single engine in Table 9.4. We thus report the optimal maintenance planning when considering only one single engine, i.e., the restrictions on the capacity and the number of spares are irrelevant.

Single engine maintenance scheduling - Upper and lower bound

Engine	# of nodes	Bounds			
name	in scenario tree	LB	UB	UB - LB	$\Delta(\mathrm{UB}-\mathrm{LB})$
1-24	208	198.38	205.52	7.14	3.60%
2-39	106	206.60	220.22	13.62	6.59%
3-32	56	134.73	247.41	112.68	83.63%
4-70	145	177.79	185.56	7.77	4.37%
1-49	233	126.37	129.29	2.92	2.31%

Table 9.5.: Overview of the size of the scenario tree and the lower bound (LB) and upper bound (UB) for the single engine maintenance planning.

Table 9.5 gives the LB and the UB for the single engine maintenance planning. The gap $\Delta(\mathrm{UB}-\mathrm{LB})$ is between 2.31% and 6.59% for engine 1-24, 2-39, 4-70 and 1-49. For engine 3-32, the gap is even 83.63%. This is because the mean estimated RUL of this engine is large. For some cases in the WS problem, the engine does not fail in the next four weeks, and is thus not replaced. This lowers the cost considerably. However, even a gap of 2.31% in the costs is quite large. We thus continue with solving the stochastic program.

Single engine maintenance scheduling - stochastic solution

Table 9.6 gives the results for the single engine stochastic maintenance planning. The VSS is relatively small, between 0.00% (engine 1-49) to 0.89% (engine 2-39). Table 9.6 also shows the planned day of replacement, as planned at the root node (day 0). In the stochastic solution, we plan to replace most engines one day later than in the upper bound solution (except for engine 1-49). This is as expected: In the stochastic program, we can immediately reschedule the replacement when an engine fails. If an engine fails in the upper bound problem, however, it remains failed until the day the replacement was planned at the root node. This incurs a high cost c_l per day. It is thus beneficial to replace engines earlier in the upper

	Objective value			Planning at root node (day 0)	
Engine name	Optimal obj. value z	VSS	EVPI	Day of replacement - UB solution	Day of replacement -stochastic solution
1-24	204.46	0.52%	3.06%	5	6
2-39	218.28	0.89%	5.65%	16	17
3-32	247.09	0.13%	83.40%	21	22
4-70	184.78	0.42%	3.93%	13	14
1-49	129.29	0.00%	2.31%	3	3

Table 9.6.: The stochastic solution for the single engine maintenance plannings: The optimal objective value, the VSS and the EVPI, and the day of replacement, as planned at day 0 (root node), in the upper bound solution and the stochastic solution.

bound solution. Here, the exception is engine 1-49, where the stochastic solution and the upper bound solution are the same.

9.5.5. Numerical Results: Multi-engine maintenance scheduling

In this section, we analyse the multi-engine maintenance planning. Most airlines operate a fleet of a few dozen to a few hundred aircraft. However, since engines rarely fail, we expect that only a handful of engines in the fleet may fail within the *same* next four weeks. We do not consider the engines without a chance of failure in the next four weeks in the maintenance optimization, since it is optimal not to plan any replacement for these engines. Instead, for the multi-engine maintenance planning, we only consider engines for which it is optimal to replace them somewhere in the next four weeks. We consider the following four cases:

- **2 engines:** We make a maintenance planing for engine 1-24 and engine 2-39 in Table 9.4. We consider $s = 2$ spares in the initial stock.
- **3 engines:** We make a maintenance planing for engine 1-24, 2-39 and engine 3-32 in Table 9.4. We consider $s = 2$ spares in the initial stock.
- **4 engines:** We make a maintenance planning for engine 1-24, 2-39, 3-32 and engine 4-70 in Table 9.4. We consider $s = 2$ spares in the initial stock.
- **5 engines:** We make a maintenance planning for engine 1-24, 2-39, 3-32, 4-70 and engine 1-49 in Table 9.4. To ensure that we can replace all engines, we consider $s = 3$ spares in the initial stock.

Multi-engine maintenance scheduling - Upper and lower bound

Table 9.7 shows the LB and UB for the different multi-engine maintenance planning cases. As expected, the number of nodes grows exponentially with the number of

Total number of engines	# of nodes in scenario tree	Bounds			
		LB	UB	UB − LB	Δ(UB − LB)
2	894	404.97	425.74	20.77	5.13%
3	3.666	540.78	673.15	132.37	24.48%
4	44.580	730.19	876.42	146.23	20.03%
5	445.653	847.29	988.83	141.54	16.71%

Table 9.7.: Overview of the size of the scenario tree and the lower bound (LB) and upper bound (UB) for the multi-engine maintenance planning.

considered engines. The gap between the upper and lower bound (Δ(UB − LB)) is quite large, between 5.13% (2 engines) to 24.48% (3 engines). We therefore continue with solving the stochastic program.

Multi-engine maintenance scheduling - Stochastic solution

Total number of engines	Objective value			Planning at root node (day 0)									
				Day of replacement -UB solution					Day of replacement -stochastic solution				
	Optimal obj. value z	VSS	EVPI	1-24	2-39	3-32	4-70	1-49	1-24	2-39	3-32	4-70	1-49
2	422.74	0.71%	4.39%	5	16	n.a.	n.a.	n.a.	6	17	n.a.	n.a.	n.a.
3	669.83	0.50%	23.86%	5	16	21	n.a.	n.a.	6	17	22	n.a.	n.a.
4	876.11	0.04%	19.98%	2	17	21	6	n.a.	2	17	22	6	n.a.
5	986.11	0.28%	16.38%	5	17	21	13	2	6	17	22	13	1

Table 9.8.: The stochastic solution for the multi-engine maintenance planning: The optimal objective value, the VSS and the EVPI, and the day of replacement, as planned at day 0 (root node), in the upper bound solution and the stochastic solution.

Table 9.8 shows the stochastic results for the multi-engine maintenance planning. The VSS is relatively small, between 0.04% (four engines) to 0.71% (two engines). The VSS with four engines is small since we only plan the replacement of engine 3-32 at a different day in the stochastic solution than in the upper bound solution (at the root node). In contrast, the EVPI is very large. With perfect RUL prognostics, the expected costs would decrease by 4.39% (2 engines) to 23.86% (3 engines).

Figure 9.4 illustrates the initial maintenance planning at the root node with five engines. For the stochastic solution, we plan to replace engine 1-24, 2-39 and 3-32 at the same day as in Table 9.6, i.e., at the optimal replacement day if we only consider a single engine. Engine 1-49 is now replaced at day 1, instead of at day 3 (the optimal replacement day if we only consider a single engine). In this way, the engine becomes available as spare at day 16. Engine 4-70 is replaced at day 13, instead of at day 14 (the optimal replacement day if we only consider a single engine). In this way, the capacity to replace a new engine becomes available at day

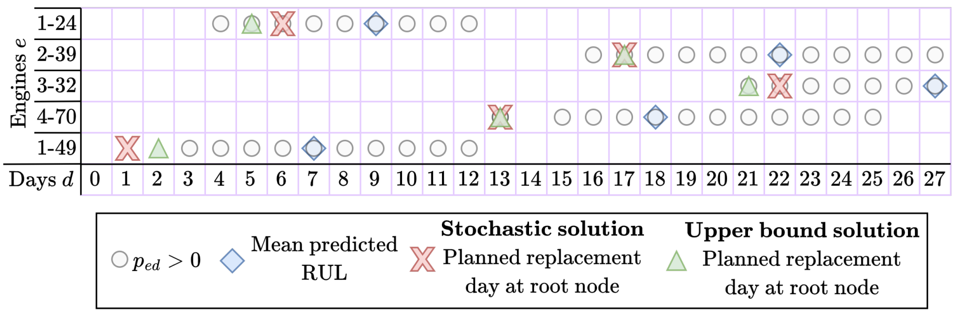


Figure 9.4.: Illustration of the optimal stochastic solution and the optimal upper bound solution for the case with five engines.

16. This capacity and the spare is used to replace engine 2-39 at day 16, if it fails at that day, and at day 17 otherwise.

Computational time for the multi-engine maintenance scheduling

Total number of engines	Method	LDE	Benders	Benders	Group	Group	Cluster	Cluster
	Unicut/ Multicut	n.a.	Unicut	Multicut	Unicut	Multicut	Unicut	Multicut
2	Time	0.4	0.7	0.6	**0.1**	**0.1**	**0.1**	**0.1**
	# of it.	n.a.	56	48	10	10	10	14
3	Time	5.4	6.7	5.8	1.9	1.1	0.8	**0.7**
	# of it.	n.a.	126	100	50	28	44	32
4	Time	> 180	> 180	> 180	36.9	46.6	**9.8**	16.0
	# of it.	n.a.	> 249	> 210	70	72	86	74
5	Time	> 180	> 180	> 180	> 180	> 180	**77.5**	> 180
	# of it.	n.a.	> 21	> 16	> 24	> 20	102	> 51

Table 9.9.: Computational time (in minutes) and number of iterations for solving the stochastic program to optimality for i) various solution methods and ii) the unicut and multicut strategy. We stop solving the problem after three hours (denoted by > 180), where we also give the number of iterations in three hours (denoted by > ...). The shortest time per instance is denoted in bold.

Table 9.9 shows the computational time for solving the stochastic program to optimality, using various solution methods. Here, we consider a maximum computational time of three hours. As expected, the computational time grows exponentially with the number of engines. With 2 or 3 engines, all methods solve the stochastic program to optimality in less than 10 minutes.

With 4 engines, the computational time of the different methods diverges. Setting up the LDE of the stochastic program with four engines already takes more than 3 hours. And the plain nested Benders decomposition algorithm converges very slowly: Even though we perform 249 (unicut strategy) and 210 (multicut strategy) iterations, we do not find the optimal solution within three hours. In contrast, with the group and cluster method, we only need between 70 to 86 iterations to find an optimal solution, and we converge to the optimal solution in less than one hour. Here, the cluster method is 3.8 (unicut strategy) and 2.9 (multicut strategy) times faster than the grouping method. The cluster method with the unicut strategy is the fastest method, and finds the optimal solution within 10 minutes.

With 5 engines, we even only find the optimal solution with the cluster method combined with the unicut strategy. Here, we find the optimal solution after 77 minutes and 102 iterations. With the cluster method and the multicut strategy, we only perform 51 iterations in the full three hours. This is because with the multicut strategy, we do not consider the clusters when moving backwards through the tree. With the grouping method, we have even performed only 24 (unicut) and 20 (multicut) iterations after three hours. With the clustering algorithm and the unicut strategy, we already perform the first 24 iterations in 7.1 minutes, which is is 25 times faster than the grouping method. This shows the benefits of our clustering algorithm, with the unicut strategy, when solving a multi-stage stochastic program.

When considering five engines (or more), in the maintenance optimization, the size of the scenario tree and the computational time becomes very large. We therefore advice to use the upper bound solution with five or more engines. For our case study, the optimality gap with the upper bound solution is only 0.28% when considering five engines (and up to 0.89% over all case studies) while we find the upper bound solution in less than a minute. However, since engines rarely fail, we expect that a situation where five engines or more have a probability of failure in the next four weeks rarely occurs.

9.6. CONCLUSIONS

In this chapter, we formulate the predictive maintenance planning problem for multiple systems with probabilistic RUL prognostics as a multi-stage stochastic integer linear program. We formulate the stochastic program such that the constraint matrix is totally unimodular. We can therefore solve the relaxation of the integer linear program, and still obtain an integer optimal solution. Moreover, our problem has endogenous uncertainty type 2. Instead of using non-anticipativity constraints to handle this endogenous uncertainty, we formulate our problem such that the optimal solution does not depend on the (possibly unobserved) random RUL of the replaced systems. Last, we propose a new clustering algorithm, based on the endogenous uncertainty in our problem. We integrate this clustering algorithm in the nested Benders decomposition algorithm to accelerate this solution method.

We apply our method to a case study with aircraft engines, where we plan maintenance for the next four weeks (28 days/stages) and for up to 5 engines. We also consider a problem-specific upper bound solution, where we assume that we cannot update the maintenance schedule over time. By solving the stochastic program, we lower the expected costs by up to 0.89%, compared to the upper bound solution. Moreover, our new clustering algorithm indeed decreases the computational time. With 5 engines, we only find the optimal solution within the time limit of three hours with the new clustering algorithm (combined with the unicut strategy). Moreover, with 5 engines, we solve the same number of iterations 25 times faster with the clustering algorithm, than without the clustering algorithm. However, due to the exponential grow of the computational time, we advice to use the upper bound solution when considering 5 or more engines.

As future work we plan to extend the considered problem by integrating more aspects of predictive maintenance into the optimization. For instance, we plan to optimize the maintenance planning with multiple types of maintenance tasks instead of only replacements, such as inspections and repairs. We also aim to consider imperfect replacements/maintenance, where the component still has a probability of failure after being maintained.

Appendix 9.A. Proof of theorem 1

Let E^n_{rep} be the set of systems that are already replaced at node n, i.e., $r^n_e = 1 \, \forall e \in E^n_{\text{rep}}$. This set thus depends on our maintenance decisions made before node n. If a system is replaced before it fails, we do not observe the random RUL. We show that the optimal solution of node n does not depend on the (unobserved) RUL of the replaced systems.

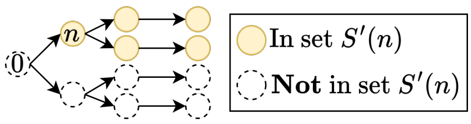


Figure 9.5.: A schematic example of the set $S'(n)$ with the direct and indirect successor nodes of a node n.

Let $S'(n) \subseteq N$ be the set with node n and with all direct and indirect successor nodes of node n, i.e., $S'(n)$ contains node n, the successor nodes $S(n)$ of node n, the successor nodes $S(m)$ for each successor node $m \in S(n)$, etc. (see Figure 9.5). By the definition of r^z_e (constraint (9.23)), the set of replaced systems at a node n, E^n_{rep}, is a subset of the replaced systems at node $z \in S'(n)$, i.e., $E^n_{\text{rep}} \subseteq E^z_{\text{rep}}$. It therefore holds that for any node $z \in S'(n)$, $r^z_e = 1$ (constraint (9.23)) and $x^z_{ed(z)} = 0$ (constraint (9.24)) for all $e \in E^n_{\text{rep}}$. By filling in these values in the first two terms $C^z + (c^z)^T y^z$ of the objective of any node $z \in S'(n)$ (eq. (9.35)), we obtain:

$$
\begin{aligned}
C^z + (c^z)^T y^z = & \sum_{e \in E^z_{\text{new}} \setminus E^n_{\text{rep}}} \left(1 - r^z_e\right) \frac{c_f}{d(z) + I_e} \\
& + \sum_{e \in \left(E^z_{\text{new}} \cup E^z_{\text{fail}}\right) \setminus E^n_{\text{rep}}} \left(1 - r^z_e - x^n_{ed(z)}\right) \frac{c_l}{d^{\text{fail}}_e(z) + I_e} + x^z_{ed(z)} \frac{c_r}{d^{\text{fail}}_e(z) + I_e} \\
& + \sum_{e \in E \setminus E^n_{\text{rep}} \setminus \left(E^z_{\text{fail}} \cup E^z_{\text{new}}\right)} x^z_{ed(z)} \frac{c_r}{d(z) + I_e} + \sum_{e \in E} \sum_{d = d(z)}^{D} c_i i^z_{ed} + c_k k^z_{ed}.
\end{aligned} \tag{9.44}
$$

In eq. (9.44), the only term with the replaced systems are the costs of inserting and cancelling replacements. These costs do not depend on the random RUL of the system. $C^z + (c^z)^T y^z$ of node $z \in S'(n)$ is thus completely independent of the RUL of the replaced systems E^n_{rep} at node n.

We now show by induction that the optimal solution in node n is independent on the, possibly unobserved, RUL of the replaced systems at node n:

Base step for a leaf node in $S'(n)$ For a leaf node $z \in S'(n)$ (i.e., $d(z) = D$), the objective function consists only of the terms $C^z + (c^z)^T y^z$. The constraint matrix and right-hand side of all nodes $m \in N_d$ is the same for all days $d \in [0, 1, \ldots, D]$. The constraint matrix and right-hand side are thus also completely independent of the RUL of the systems (see Section 9.3.2). The optimal solution at node z is therefore independent of the RUL of the replaced systems at node n.

Induction step Assume that for all nodes $z \in S'(n)$ with $d(z) \geq d$, $d \in [d(n)+1,\ldots,D]$, the optimal solution at node z is independent of the RUL of the replaced systems at node n.

We now consider a node $m \in S'(n)$ with $d(m) = d-1$. The last term of the objective function at node m, $Q^m(y^m)$ (see eq. (9.37)), are the expected future costs. By the induction step, the value of the optimal solution $v^z\left(y^m\right)$ of a successor node $z \in S(m)$ is independent of the RUL of the replaced systems at node n. The objective function $C^m + (c^m)^T y^m + Q^m(y^m)$ (see eq. (9.35)) is thus independent of the RUL of the replaced systems at node n. The constraint matrix and right-hand side of any node $n \in N$ are also independent of the RUL of the systems (see the base step). The optimal solution of the linear program at node $m \in S'(n)$, including node n itself, is thus independent of the (unobserved) RUL of the replaced systems at node n.

Appendix 9.B. Proof of theorem 2

Appendix 9.B.1. Totally unimodular constraint matrix

In this appendix, we show that the constraint matrix at any node $n \in N$ is totally unimodular using the consecutive ones property [30]. A matrix fulfills the consecutive ones property if i) all entries are either 0 or +1 and ii) for each row, the ones appear consecutively. Moreover, when adding a unit vector as row/column to a totally unimodular matrix, and when multiplying a row/column in a totally unimodular matrix with minus one, the matrix remains totally unimodular [22].

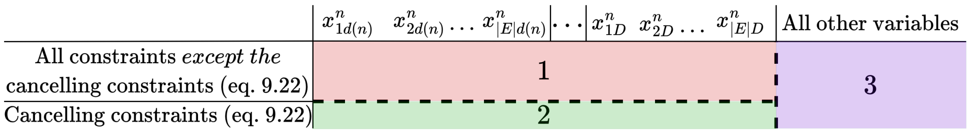

$x^n_{1d(n)}$ $x^n_{2d(n)}$ … $x^n_{|E|d(n)}$ … x^n_{1D} x^n_{2D} … $x^n_{|E|D}$
All other variables
All constraints *except the* cancelling constraints (eq. 9.22)
Cancelling constraints (eq. 9.22)
1
2
3

Figure 9.6.: Overview of the constraint matrix for any node $n \in N \setminus \{0\}$.

Figure 9.6 shows a schematic overview of the constraint matrix for any node $n \in N \setminus \{0\}$. We divide this constraint matrix in 3 blocks:

- **Block 1:** Block 1 in Figure 9.6 contains the coefficients belonging to the x^n_{ed} variables, and to all constraints except the cancelling constraints (eq. (9.22)). The first column contains the coefficients of $x^n_{1d(n)}$, where 1 denotes the first considered system, until the E^{th} column contains the coefficients of $x^n_{|E|d(n)}$, where $|E|$ denotes the last system. We then continue with the coefficients of $x^n_{1(d(n)+1)}$ in the next column, until the last column of block 1 contains the coefficients of $x^n_{|E|D}$ (see Figure 9.6). With this ordering of the variables, the matrix in block 1 fulfills the consecutive ones property [30], and is thus totally unimodular:
  - In constraint (9.30) and (9.27), we iteratively sum the x^n_{ed} variables over all systems and over a subset of consecutive days. In constraint (9.28)

and eq. (9.25), we sum the $x^n_{ed(n)}$ variables over all systems. Due to the ordering of the variables, the ones in the rows of these constraint appear consecutive.

  – Constraints eq. (9.21) and eq. (9.24) contain only one x^n_{ed} variable per constraint, each time with a coefficient of one. Constraints eq. (9.23), (9.26) and (9.29) do not contain the variables x^n_{ed}. The rows of these constraints in block 1 thus contain a single one, and only zeroes beside, or only zeroes.

- **Block 2:** The rows in block 2 contain the coefficients for the x^n_{ed} variables of the cancelling constraints (eq. (9.22)). Each row, with the coefficients of the x^n_{ed} variables, of a cancelling constraint contains a single -1, and only zeroes beside. The matrix in block 1 and 2 is thus totally unimodular.

- **Block 3:** Block 3 contains the columns with the coefficients belonging to all other variables. There is no specific ordering of these variables. Each variable, except the x^n_{ed} variables, is present in exactly one constraint, with a coefficient of either -1 or 1. The column with the coefficients belonging to any variable (except the x^n_{ed} variables) thus contains a single -1 or 1 element, and only zeroes beside. The matrix in block 1, 2 and 3 is thus totally unimodular.

Our constraint matrix is thus totally unimodular for any node $n \in N \setminus \{0\}$. The constraint matrix of the root node is a submatrix of the constraint matrix in Figure 9.6, without the cancellation and inserting variables, and without constraints (9.21), (9.22) and (9.24). The constraint matrix at the root node is thus also totally unimodular.

APPENDIX 9.B.2. INTEGER RIGHT-HAND SIDE WITH INDUCTION

We now show by induction that in the optimal solution (found with the simplex method), the right-hand side of the integer program at each node $n \in N$ is integer:

Base step for the root node The right-hand side of the linear program of the root node consists of 0 (eq. (9.10)), s, (eq. (9.11), (9.12), and (9.13)), and g, (eq. (9.14), (9.15) and (9.16)). The right-hand side of the linear program of the root node is thus integer, while the constraint matrix is totally unimodular. The optimal solution (when using the simplex method) of the relaxation of the integer program is thus integer.

Induction step Assume that for any day $d' \le d$, with $d \in [0, 1, \ldots, D-1]$, the right-hand side of the integer program at any node $n \in N_{d'}$ is integer. The optimal solution of such a node n is then also integer (when using the simplex method).

Consider a node $m \in N$ at a day $d(m) = d+1$. The right-hand side of the linear program consists of integers (s, g and 1) , together with a linear combination (with coefficients of -1 and 1 only) of the variables of the ancestor node. By the induction step, the optimal solution of the ancestor node is integer. This means that the right-hand side of the linear program of node m is integer, and that the optimal solution (when using the simplex method) is integer as well.

Appendix 9.C. Proof of theorem 3.

In this appendix, we prove that the linear program of two nodes in the same cluster is the same. This proof consists of two parts. An overview of the proof, with the corresponding sections, is in Figure 9.7.

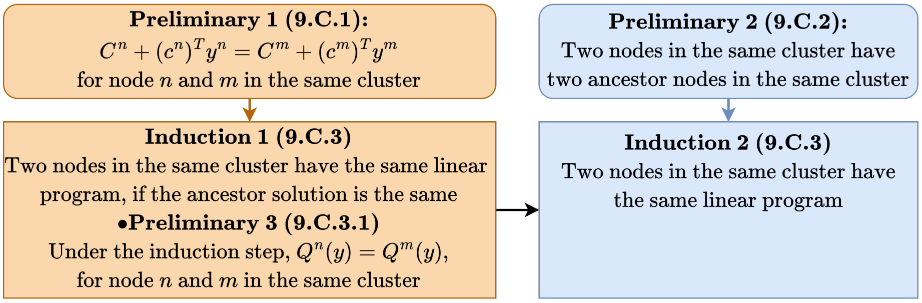

Preliminary 1 (9.C.1):
$C^n + (c^n)^T y^n = C^m + (c^m)^T y^m$
for node n and m in the same cluster
Preliminary 2 (9.C.2):
Two nodes in the same cluster have two ancestor nodes in the same cluster
Induction 1 (9.C.3)
Two nodes in the same cluster have the same linear program, if the ancestor solution is the same
•**Preliminary 3 (9.C.3.1)**
Under the induction step, $Q^n(y) = Q^m(y)$, for node n and m in the same cluster
Induction 2 (9.C.3)
Two nodes in the same cluster have the same linear program

Figure 9.7.: Overview of the proof that the linear program of two nodes in the same cluster is the same

Appendix 9.C.1. Preliminary 1: $C^n + (c^n)^T y = C^m + (c^m)^T y$ for two nodes n and m in the same cluster.

Consider nodes $n, m \in N \setminus \{0\}$ that are both in the same cluster α^i at iteration i of the nested Benders decomposition. In this section, we prove that $C^n + (c^n)^T y^n = C^m + (c^m)^T y^m$ (see the schematic objective in Section 9.3.2). To make the notation between the linear program of node n and m consistent, we rename the variables by removing the dependency on the node, i.e., $x_{ed} = x^n_{ed}$, $y = y^n$, etc.

We use the rewritten form of $C^n + (c^n)^T y^n$ in eq. (9.44) (Appendix 9.A). Here, we replace i) $d(n)$ by $d(m)$ (C1), ii) $E^n_{\text{new}} \setminus E^{n,i}_{\text{rep}}$ by $E^m_{\text{new}} \setminus E^{m,i}_{\text{rep}}$ (C5a), iii) $\left(E^n_{\text{new}} \cup E^n_{\text{fail}}\right) \setminus E^{n,i}_{\text{rep}}$ by $\left(E^m_{\text{new}} \cup E^m_{\text{fail}}\right) \setminus E^{m,i}_{\text{rep}}$ (C5a, C5b), where we replace $d^{\text{fail}}_e(n)$ by $d^{\text{fail}}_e(m)$ for each system in $E^n_{\text{fail}} \setminus E^{n,i}_{\text{rep}}$ (C5b), and iv) $(E \setminus E^{n,i}_{\text{rep}}) \setminus (E^n_{\text{fail}} \cup E^n_{\text{new}})$ by $(E \setminus E^{m,i}_{\text{rep}}) \setminus (E^m_{\text{fail}} \cup E^m_{\text{new}})$ (C5c). With this, we obtain:

$$\begin{aligned} C^n + (c^n)^T y = & \sum_{e \in E^m_{\text{new}} \setminus E^{m,i}_{\text{rep}}} (1 - r_e) \frac{c_f}{d(m) + I_e} \\ & + \sum_{e \in \left(E^m_{\text{new}} \cup E^m_{\text{fail}}\right) \setminus E^{m,i}_{\text{rep}}} \left(1 - r_e - x_{ed(m)}\right) \frac{c_l}{d^{\text{fail}}_e(m) + I_e} + x_{ed(m)} \frac{c_r}{d^{\text{fail}}_e(m) + I_e} \\ & + \sum_{e \in (E \setminus E^{m,i}_{\text{rep}}) \setminus (E^m_{\text{fail}} \cup E^m_{\text{new}})} x_{ed(m)} \frac{c_r}{d(m) + I_e} + \sum_{e \in E} \sum_{d = d(m)}^{D} c_i i_{ed} + c_k k_{ed} \end{aligned} \tag{9.45}$$

$$= C^m + (c^m)^T y. \tag{9.46}$$

where we again use the rewritten form of $C^m + (c^m)^T y^m$ in eq. (9.44) (Appendix 9.A) to go to eq. (9.46).

APPENDIX 9.C.2. PRELIMINARY 2: TWO NODES IN THE SAME CLUSTER HAVE TWO ANCESTOR NODES IN THE SAME CLUSTER

We consider two nodes $n, m \in N$ that are in the same cluster α^i in iteration i of the nested Benders decomposition. We now prove by contradiction that the ancestor nodes $a(n)$ and $a(m)$ are also in the same cluster at iteration i.

Assume that node n and m are in the same cluster, while $a(n)$ and $a(m)$ are not in the same cluster at iteration i. Then, node $a(n)$ and $a(m)$ violate at least one of the cluster conditions. We show that in this case, node n and m also violate at least one cluster condition, which is a contradiction:

1. **a(n) and a(m) violate condition C1.** This means that $d(a(n)) \neq d(a(m))$. Then $d(n) = d(a(n)) + 1 \neq d(a(m)) + 1 = d(m)$ (S1). Node n and m thus violate condition C1.

2. **a(n) and a(m) violate condition C2.** This means that $E_{\text{rep}}^{a(n),i} \neq E_{\text{rep}}^{a(m),i}$. Without loss of generality (w.l.o.g.), we assume that $\exists e \in E_{\text{rep}}^{a(n),i} : e \notin E_{\text{rep}}^{a(m),i}$, i.e., $r_e^{a(n)} = 1, r_e^{a(m)} = 0$. Then $r_e^n = 1$ (constraint (9.23)), and $e \in E_{\text{rep}}^{n,i}$ (by definition). There are two options for system e at node $a(m)$:
   - $x_{ed(a(m))}^{a(m)} = 0$. By constraint (9.23), $r_e^m = 0$, and $e \notin E_{\text{rep}}^{m,i}$ (by definition). Thus, $E_{\text{rep}}^{n,i} \neq E_{\text{rep}}^{m,i}$, and node n and m violate condition C2.
   - $x_{ed(a(m))}^{a(m)} = 1$. By constraint 9.23, $r_e^m = 1$ and $e \in E_{\text{rep}}^{m,i}$ (by definition). However, for node m, system e is replaced at day $d(a(m)) = d(a(n))$ (S1), while for node n, system e is replaced before day $d(a(n))$. Node n and m thus violate condition C3.

3. **a(n) and a(m) violate condition C3.** This means that $\exists e \in E_{\text{rep}}^{a(n),i}$, such that the day of replacement of this system is not the same in node $a(n)$ and $a(m)$. The day of replacement is fixed, since each system can only be replaced once (constraint (9.24)). Node n and m thus also violate condition C3.

4. **a(n) and a(m) violate condition C4.** We assume w.l.o.g. that $\exists e \in E_{\text{rep}}^{a(n),i}$ such that $e \in E_{\text{fail}}^{a(n)}$ and has, at node $a(n)$, failed before or at the day it is replaced (in iteration i), that violates condition C4. For node n, it thus holds that $e \in E_{\text{fail}}^n$ (S2), with $d_e^{\text{fail}}(n) = d_e^{\text{fail}}(a(n)) < d(a(n))$ (S3, S4), and also for node n, this system failed before or at the day it is replaced. For system e at node $a(m)$, there are three options:
   - $e \notin E_{\text{fail}}^{a(m)}$, and $e \notin E_{\text{new}}^{a(m)}$. Then, $e \notin E_{\text{fail}}^m$ (S2).
   - $e \notin E_{\text{fail}}^{a(m)}$, but $e \in E_{\text{new}}^{a(m)}$. Then, $e \in E_{\text{fail}}^m$ (S2) with $d_e^{\text{fail}}(m) = d(a(m)) = d(a(n)) > d_e^{\text{fail}}(n)$ (S1, S4).
   - $e \in E_{\text{fail}}^{a(m)}$, but $d_e^{\text{fail}}(a(n)) \neq d_e^{\text{fail}}(a(m))$. Then, $d_e^{\text{fail}}(n) = d_e^{\text{fail}}(a(n))) \neq d_e^{\text{fail}}(a(m))) = d_e^{\text{fail}}(m)$ (S3).

Thus, node n and m also violate condition C4.

5. **a(n) and a(m) violate condition C5.** We assume w.l.o.g. that $\exists e \in E \setminus E^{a(n),i}_{\text{rep}} : e \in E \setminus E^{a(m),i}_{\text{rep}}$, for which condition C5 does not hold. Then, there are three options:

a) We do not replace system e in node $a(n)$ or node $a(m)$, i.e., $x^{a(n)}_{ed(a(n))} \neq 1$, $x^{a(m)}_{ed(a(m))} \neq 1$. Then, $e \notin E^{n,i}_{\text{rep}}, e \notin E^{m,i}_{\text{rep}}$ (by definition). At least one of the following holds:

- *$a(n)$ and $a(m)$ violate condition C5a.* We assume w.l.o.g. that $e \in E^{a(n)}_{\text{new}}, e \notin E^{a(m)}_{\text{new}}$. Then, $e \in E^{n}_{\text{fail}}$ (S2) with $d^{\text{fail}}_{e}(n) = d(a(n)) = d(a(m))$ (S1, S4). There are two options for system e at node $a(m)$:
  - $e \in E^{a(m)}_{\text{fail}}$. Then, $e \in E^{m}_{\text{fail}}$ (S2), with $d^{\text{fail}}_{e}(m) = d^{\text{fail}}_{e}(a(m)) < d(a(m)) = d^{\text{fail}}_{e}(n)$ (S3, S4).
  - $e \notin E^{a(m)}_{\text{fail}}$. Then, $e \notin E^{m}_{\text{fail}}$ (S2).
- *$a(n)$ and $a(m)$ violate condition C5b.* We assume w.l.o.g. that $e \in E^{a(n)}_{\text{fail}}$. Then, $e \in E^{n}_{\text{fail}}$ (S2) with $d^{\text{fail}}_{e}(n) = d^{\text{fail}}_{e}(a(n)) < d(a(n))$ (S3, S4). There are three options for system e at node $a(m)$:
  - $e \notin E^{a(m)}_{\text{fail}}, e \notin E^{a(m)}_{\text{new}}$. Then, $e \notin E^{m}_{\text{fail}}$ (S2).
  - $e \notin E^{a(m)}_{\text{fail}}$ but $e \in E^{a(m)}_{\text{new}}$. Then, $e \in E^{m}_{\text{fail}}$ with $d^{\text{fail}}_{e}(m) = d(a(m)) = d(a(n)) > d^{\text{fail}}_{e}(n)$ (S2, S4).
  - $e \in E^{a(m)}_{\text{fail}}$, but $d^{\text{fail}}_{e}(a(m)) \neq d^{\text{fail}}_{e}(a(n))$. Then, $e \in E^{m}_{\text{fail}}$ (S2), but $d^{\text{fail}}_{e}(m) = d^{\text{fail}}_{e}(a(m)) \neq d^{\text{fail}}_{e}(a(n)) = d^{\text{fail}}_{e}(n)$ (S3).
- *$a(n)$ and $a(m)$ violate condition C5c.* We assume w.l.o.g. that $e \in E^{a(n)}_{\text{fail}} \cup E^{a(n)}_{\text{new}}$ and that $e \notin E^{a(m)}_{\text{fail}} \cup E^{a(m)}_{\text{new}}$. Then, $e \in E^{n}_{\text{fail}}$, while $e \notin E^{m}_{\text{fail}}$ (S2).

In all these cases, node n and m violate condition *C5b*.

b) We assume w.l.o.g. that we replace system e at node $a(n)$ ($x^{a(n)}_{ed(a(n))} = 1$), but not at node $a(m)$ ($x^{a(m)}_{ed(a(m))} \neq 1$). The, $r^{n}_{e} = 1$, while $r^{m}_{e} = 0$ (constraint 9.23), and $e \in E^{n,i}_{\text{rep}}$ but $e \notin E^{m,i}_{\text{rep}}$ (by definition). Then, node n and m violate condition C2.

c) We replace system e at both node $a(n)$ and at node $a(m)$, i.e., $x^{a(n)}_{ed(a(n))} = 1$, $x^{a(m)}_{ed(a(m))} = 1$. Then, $e \in E^{n,i}_{\text{rep}}, e \in E^{m,i}_{\text{rep}}$ (by definition). At least one of the following holds:

- *$a(n)$ and $a(m)$ violate condition C5a.* We assume w.l.o.g. that $e \in E^{a(n)}_{\text{new}}$, $e \notin E^{a(m)}_{\text{new}}$. Then, $e \in E^{n}_{\text{fail}}$ (S2), with $d^{\text{fail}}_{e}(n) = d(a(n)) = d(a(m))$ (S1, S4). For node n, the failure day of system e equals the day it is replaced, and condition C4 thus has to hold. There are two options for system e at node $a(m)$:
  - $e \in E^{a(m)}_{\text{fail}}$. Then, $e \in E^{m}_{\text{fail}}$ (S2) with $d^{\text{fail}}_{e}(m) = d^{\text{fail}}_{e}(a(m)) < d(a(m)) = d^{\text{fail}}_{e}(n)$ (S3, S4).
  - $e \notin E^{a(m)}_{\text{fail}}$. Then, $e \notin E^{m}_{\text{fail}}$ (S2).

- *$a(n)$ and $a(m)$ violate condition C5b.* We assume w.l.o.g. that $e \in E_{\text{fail}}^{a(n)}$. Then, $e \in E_{\text{fail}}^{n}$ ((S2) with $d_e^{\text{fail}}(n) = d_e^{\text{fail}}(a(n)) < d(a(n))$ (S3, S4). For node n, the failure day of system e is before the day it is replaced, and condition C4 thus has to hold. There are three options for system e at node $a(m)$:
  - $e \notin E_{\text{fail}}^{a(m)}$, $e \notin E_{\text{new}}^{a(m)}$. Thus, $e \notin E_{\text{fail}}^{m}$ (S2).
  - $e \notin E_{\text{fail}}^{a(m)}$, but $e \in E_{\text{new}}^{a(m)}$. Then, $e \in E_{\text{fail}}^{m}$ with $d_e^{\text{fail}}(m) = d(a(m)) = d(a(n)) > d_e^{\text{fail}}(n)$ (S2, S4).
  - $e \in E_{\text{fail}}^{a(m)}$, but $d_e^{\text{fail}}(a(m)) \neq d_e^{\text{fail}}(a(n))$. Then, $e \in E_{\text{fail}}^{m}$ (S2) but $d_e^{\text{fail}}(m) = d_e^{\text{fail}}(a(m)) \neq d_e^{\text{fail}}(a(n)) = d_e^{\text{fail}}(n)$ (S3).
- *$a(n)$ and $a(m)$ violate condition C5c.* We assume w.l.o.g. that $e \in E_{\text{fail}}^{a(n)} \cup E_{\text{new}}^{a(n)}$ and that $e \notin E_{\text{fail}}^{a(m)} \cup E_{\text{new}}^{a(m)}$. Then, $e \in E_{\text{fail}}^{n}$ (S2), with $d_e^{\text{fail}}(n) \leq d(a(n))$ (S3, S4). For node n, system e failed before or at the day it is replaced, and condition C4 thus has to hold. However, $e \notin E_{\text{fail}}^{m}$ (S2).

In all cases, node n and m thus violate condition C4.

Concluding, if node n, m are in the same cluster at iteration i, then the ancestor nodes $a(n)$ and $a(m)$ are also in the same cluster at iteration i.

APPENDIX 9.C.3. INDUCTION 1: TWO NODES n, m IN THE SAME CLUSTER HAVE THE SAME LINEAR PROGRAM - IF THE SOLUTION OF THE ANCESTOR NODES $a(n)$ AND $a(m)$ IS THE SAME

Consider two nodes $n, m \in N$ that are in the same cluster α^i at iteration i of the nested Benders decomposition. Let $\hat{y}^{i,z}$ denote the solution of the linear program at node z in iteration i. We assume that the solution of the ancestor nodes of node n and m is equal in iteration i, i.e., $\hat{y}^{i,a(n)} = \hat{y}^{i,a(m)}$. Under this assumption, we prove that the linear program of node n and m are equal in iteration i.

The schematic version of the constraints of the linear program of a node $z \in N \setminus \{0\}$ is $W^z y^z = h^z - T^z y^{a(z)}$. The constraint matrix W^z, the recourse matrix T^z and the right-hand side coefficients h^z are the same for all nodes $z \in N_d$, for any day $d \in [0, 1, \ldots, D]$ (see Section 9.3.2).

Base case at day D We consider two leaf nodes $n, m \in N_D$ that are in the same cluster α^i at iteration i.

- Since $d(n) = d(m)$, the constraint matrix and right-hand side of the linear program at node n and m are equal if $\hat{y}^{i,a(n)} = \hat{y}^{i,a(m)}$.
- The schematic objective of a leaf node z is $C^z + (c^z)^T y^z$ (see eq. (9.35)). In Appendix 9.C.1 (preliminary 1), we show that $C^n + (c^n)^T y = C^m + (c^m)^T y$ for two nodes n, m in the same cluster. Leaf nodes n and m thus have the same objective function.

At iteration i, the linear programs of leaf node n and m are thus equal, given that $\hat{y}^{i,a(n)} = \hat{y}^{i,a(m)}$.

Induction step Assume that the statement holds for any day $d' \geq d$, with $d \in [2,3,\dots,D]$: For two nodes $n',m' \in N_{d'}$, that are in the same cluster at iteration i, the linear programs are equal if $\hat{y}^{i,a(n')} = \hat{y}^{i,a(m')}$. We prove that the linear programs of two nodes $n,m \in N_{d-1}$ at day $d-1$, that are in the same cluster α^i, are equal if $\hat{y}^{i,a(n)} = \hat{y}^{i,a(m)}$:

- Since $d(n) = d(m)$, the constraint matrix and right-hand side coefficients of node n and m are equal if $\hat{y}^{i,a(n)} = \hat{y}^{i,a(m)}$.
- In Appendix 9.C.1 (preliminary 1), we show that $C^n + (c^n)^T y = C^m + (c^m)^T y$ for two nodes in the same cluster. In Appendix 9.C.3 (preliminary 3, see below), we show that, with the induction step, the expected future costs are also equal for node n and m, i.e., $\sum_{z\in S(n)} q_{nz} v^z\left(y^n\right) = \sum_{z\in S(m)} q_{mz} v^z\left(y^m\right)$. Node n and m thus have the same objective function.

At iteration i, the linear programs of nodes n and m are thus equal, given that $\hat{y}^{i,a(n)} = \hat{y}^{i,a(m)}$.

Preliminary 3: Expected future costs are the same for two nodes in the same cluster

In this part, we prove that the future expected costs for two nodes n and m are equal, if node n and m are in the same cluster α^i (given the induction step, and in iteration i). Let $\alpha^i(z)$ denote the cluster of a node $z \in N$ in iteration i. Let $S^i(\alpha^i)$ denote the set with all successor clusters of a cluster α^i at iteration i:

$$S^i(\alpha^i) = \left\{\alpha'^i : \exists z \in \alpha^i, \exists g \in S(z) : \alpha^i(g) = \alpha'^i\right\}. \tag{9.47}$$

With the induction step, the objective function of two nodes z,g in the same cluster is the same, if $d(g) = d(z) \geq d$. Let $v^{\alpha^i}(y)$ denote the objective function of any node z in cluster α^i ($d(z) \geq d$), with y as input. With this, we rewrite the future expected costs at node n:

$$\begin{aligned}
\sum_{z\in S(n)} v^z(y) q_{nz} &= \sum_{\alpha'^i \in S^i(\alpha^i)} \sum_{z\in\alpha'^i : z\in S(n)} v^z(y) q_{nz} && (9.48)\\
&= \sum_{\alpha'^i \in S^i(\alpha^i)} v^{\alpha'^i}(y) \sum_{z\in\alpha'^i : z\in S(n)} \prod_{e\in E^z_{\text{new}}} p'_{ed(z)} \prod_{e\in E\setminus\left(E^z_{\text{new}}\cup E^z_{\text{fail}}\right)} \left(1-p'_{ed(z)}\right) && (9.49)\\
&= \sum_{\alpha'^i \in S^i(\alpha^i)} v^{\alpha'^i}(y) \sum_{z\in\alpha'^i : z\in S(n)} \prod_{e\in E^z_{\text{new}}\setminus E^{z,i}_{\text{rep}}} p'_{ed(z)} \prod_{e\in\left(E\setminus E^{z,i}_{\text{rep}}\right)\setminus\left(E^z_{\text{new}}\cup E^z_{\text{fail}}\right)} \left(1-p'_{ed(z)}\right)\\
&\quad \prod_{e\in E^z_{\text{new}}\cap E^{z,i}_{\text{rep}}} p'_{ed(z)} \prod_{e\in E^{z,i}_{\text{rep}}\setminus\left(E^z_{\text{new}}\cup E^z_{\text{fail}}\right)} \left(1-p'_{ed(z)}\right) && (9.50)\\
&= \sum_{\alpha'^i \in S^i(\alpha^i)} v^{\alpha'^i}(y) \prod_{e\in E^{\alpha'^i}_{\text{new}}} p'_{ed(z)} \prod_{e\in E^{\alpha'^i}_{\text{work}}} \left(1-p'_{ed(z)}\right) && (9.51)\\
&\quad \sum_{z\in\alpha'^i : z\in S(n)} \prod_{e\in E^z_{\text{new}}\cap E^{z,i}_{\text{rep}}} p'_{ed(z)} \prod_{e\in E^{z,i}_{\text{rep}}\setminus(E^z_{\text{new}}\cup E^z_{\text{fail}})} \left(1-p'_{ed(z)}\right).
\end{aligned}$$

Here, we go to eq. (9.48) by using that all clusters with any successor node of node n, are included in $S^i(\alpha^i)$ (see eq. (9.47)). We go from eq. (9.48) to eq. (9.49) by using the definition of q_{nz} (see eq. (9.2)), and by using the induction step to rewrite $v^z(y)$. In eq. (9.50), we split the products in two parts, over the replaced and the non-replaced systems. In eq. (9.51), we use condition C5a and condition C5c of a cluster to rewrite the future expected costs.

We now analyse the last term in eq. (9.51), which we name $\beta^{\alpha'^i,n}$:

$$\beta^{\alpha'^i,n} = \sum_{z\in\alpha'^i:z\in S(n)} \prod_{e\in E^z_{\text{new}}\cap E^{z,i}_{\text{rep}}} p'_{ed(z)} \prod_{e\in E^{z,i}_{\text{rep}}\setminus(E^z_{\text{new}}\cup E^z_{\text{fail}})} \left(1-p'_{ed(z)}\right). \tag{9.52}$$

Let $E^{n,i}_{\text{new rep}}$ denote the set of systems that are replaced at day $d(n)$ at node n in iteration i, i.e., $x^n_{ed(n)}=1$ for all $e\in E^{n,i}_{\text{new rep}}$. For any successor node $z\in S(n)$, it follows from the definition of r^z_e (constraint (9.23)) and from the definition of $E^{z,i}_{\text{rep}}$ that:

$$E^{z,i}_{\text{rep}} = E^{n,i}_{\text{rep}} \cup E^{n,i}_{\text{new rep}} \tag{9.53}$$

We denote this set by $E^{n,i}_{\text{suc rep}}$. For any successor node $z\in S(n)$, it holds that $E^z_{\text{fail}} = E^n_{\text{new}}\cup E^n_{\text{fail}}$ (S2). Let $E^n_{\text{suc fail}} = E^n_{\text{new}}\cup E^n_{\text{fail}}$ denote the set of already failed systems at any successor node $z\in S(n)$. Last, we define $A^{n,i} = E^{n,i}_{\text{suc rep}}\setminus E^n_{\text{suc fail}}$ as the set of replaced systems that have not yet failed at node n (in iteration i). According to condition S5 of a successor node, there cannot be a system in $e\in E^z_{\text{new}}$ $(z\in S(n))$ that is already in E^z_{fail}, i.e., $E^z_{\text{new}} = E^z_{\text{new}}\setminus E^z_{\text{fail}}$. From here, we derive that:

$$E^z_{\text{new}}\cap E^{n,i}_{\text{suc rep}} = (E^z_{\text{new}}\setminus E^n_{\text{suc fail}})\cap E^{n,i}_{\text{suc rep}} \tag{9.54}$$

$$= E^z_{\text{new}}\cap (E^{n,i}_{\text{suc rep}}\setminus E^n_{\text{suc fail}}) \tag{9.55}$$

$$= E^z_{\text{new}}\cap A^{n,i}. \tag{9.56}$$

With this, we rewrite $\beta^{\alpha^i,n}$:

$$\beta^{\alpha'^i,n} = \sum_{z\in\alpha'^i:z\in S(n)} \prod_{e\in E^z_{\text{new}}\cap E^{n,i}_{\text{suc rep}}} p'_{ed(z)} \prod_{e\in E^{n,i}_{\text{suc rep}}\setminus E^n_{\text{suc fail}}\setminus E^z_{\text{new}}} \left(1-p'_{ed(z)}\right) \tag{9.57}$$

$$= \sum_{z\in\alpha'^i:z\in S(n)} \prod_{e\in E^z_{\text{new}}\cap A^{n,i}} p'_{ed(z)} \prod_{e\in A^{n,i}\setminus E^z_{\text{new}}} \left(1-p'_{ed(z)}\right). \tag{9.58}$$

Two successor nodes $z,g\in S(n)$ fulfill condition C1 to condition C4 of a cluster automatically. Any two successor nodes $z,g\in S(n)$ for which the non-replaced systems fulfill condition C5, are therefore in the same cluster. This is thus independent of which of the replaced systems in $A^{n,i}$ fail at node z/g. In other words, the successor cluster $\alpha'^i\in S^i(\alpha^i)$ (where $n\in\alpha^i$) contains one successor node of node n for each possible combination of new failures from the replaced systems in the set $A^{i,n}$ (if the probability of this combination is strictly larger than zero).

Each successor node may thus contain $l = 0$ up to $l = |A^{i,n}|$ new failures of replaced systems (that did not fail yet at node n). The probability that l replaced systems, out of the $|A^{i,n}|$ replaced systems, fail at day $d(n)+1$ is given by:

$$\sum_{E_{\text{new}} \in F_l} \prod_{e \in E_{\text{new}}} p'_{e,d(n)+1} \prod_{e \in A^{i,n} \setminus E_{\text{new}}} \left(1 - p'_{e,d(n)+1}\right), \tag{9.59}$$

where F^l denotes the set with all subsets of $A^{i,n}$ of length l. This equals the probability of having l out of $|A^{i,n}|$ successes in the Poisson binomial distribution. With this, we rewrite $\beta^{\alpha'^i,n}$:

$$\begin{aligned}
\beta^{\alpha'^i,n} &= \sum_{l=0}^{|A^{i,n}|} \sum_{E_{\text{new}} \in F_l} \prod_{e \in E_{\text{new}}} p'_{e,d(n)+1} \prod_{e \in A^{i,n} \setminus E_{\text{new}}} \left(1 - p'_{e,d(n)+1}\right) && (9.60)\\
&= \prod_{e \in A^{i,n}} \left(p'_{e,d(n)+1} + \left(1 - p'_{e,d(n)+1}\right)\right) && (9.61)\\
&= 1 && (9.62)
\end{aligned}$$

Here, we follow eq. 2 in [31] in going from eq. (9.60) to eq. (9.61). This is also intuitive: $\beta^{\alpha'^i,n}$ equals summing the PDF of the Poisson binomial distribution over all possible outcomes, i.e., over all possible number of successes l, which is 1. We use this in eq. (9.51):

$$\sum_{z \in S(n)} v^z(y) q_{nz} = \sum_{\alpha'^i \in S^i(\alpha^i)} v^{\alpha'^i}(y) \prod_{e \in E^{\alpha'^i}_{\text{new}}} p'_{ed(\alpha'^i)} \prod_{e \in E^{\alpha'^i}_{\text{work}}} \left(1 - p'_{ed(\alpha'^i)}\right), \tag{9.63}$$

where $d(\alpha'^i)$ denotes the day of any node in cluster α'^i. Note that $\prod_{e \in E^{\alpha'^i}_{\text{new}}} p'_{ed(\alpha'^i)} \prod_{e \in E^{\alpha'^i}_{\text{work}}} \left(1 - p'_{ed(\alpha'^i)}\right)$ is the probability to go from any node in cluster α^i to any node in cluster α'^i. The expected future cost function for a node n in cluster α^i thus only depends on the cluster α^i. The expected future cost functions are therefore the same for any two nodes $n, m \in \alpha^i$ (with the induction step).

Appendix 9.C.4. Induction 2: Two nodes in the same cluster have the same linear program

We consider two nodes $n, m \in N$ that are in the same cluster α^i at iteration i of the nested Benders decomposition. In this section, we prove by induction that the solution of the ancestor nodes $a(n)$ and $a(m)$ is equal, i.e., $\hat{y}^{i,a(n)} = \hat{y}^{i,a(m)}$. Given the first induction (Appendix 9.C.3), the linear programs of node n and m are thus equal.

Base case at day $d = 1$ We consider two nodes $n, m \in N_1$ at day $d = 1$, that are in the same cluster α^i at iteration i. By definition, the root node is the ancestor of node n and m. Node n and m thus have the same ancestor solution, i.e., $\hat{y}^{i,a(n)} = \hat{y}^{i,0} = \hat{y}^{i,a(m)}$. Given the first induction (Appendix 9.C.3), the linear programs of node n and m are thus equal in iteration i.

Induction step Assume that the statement holds for any day $d' \leq d$, with $d \in [1, 2, \dots, D-1]$: For two nodes $n', m' \in N_{d'}$ in the same cluster at iteration i, the linear programs are equal in iteration i.

We now consider two nodes $n, m \in N_{d+1}$, at day $d+1$, that are in the same cluster α^i at iteration i. The ancestor nodes $a(n)$ and $a(m)$ of node n and m are also in the same cluster α'^i in iteration i (preliminary 2, Appendix 9.C.2). Since $d(a(n)) = d(a(m)) = d$, the linear programs of the ancestor nodes $a(n)$ and $a(m)$ are equal (given the induction step). There might be multiple optimal solutions to the linear program of node $a(n)$ and $a(m)$. However, since the linear programs of all nodes in cluster α'^i are equal (induction step), we simply solve the linear program belonging of a single node in cluster α'^i. We use this solution as the optimal solution to all nodes in cluster α'^i, and guarantee that $\hat{y}^{i,a(n)} = \hat{y}^{i,a(m)}$. By the first induction (Appendix 9.C.3), the linear programs of node n and m are thus equal in iteration i.

REFERENCES

[1] Maintenance Cost Technical Group (MCTG). (2020). *Airline maintenance cost executive commentary (FY2019 data), public version* (tech. rep.). International Air Transport Association (IATA).

[2] Schouten, T. N., Dekker, R., Hekimoğlu, M., & Eruguz, A. S. (2022). Maintenance optimization for a single wind turbine component under time-varying costs. *European Journal of Operational Research*, *300*(3), Pages: 979–991.

[3] Lei, Y., Li, N., Guo, L., Li, N., Yan, T., & Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing*, *104*, Pages: 799–834.

[4] Papadopoulos, P., Coit, D. W., & Aziz Ezzat, A. (2022). Stochos: Stochastic opportunistic maintenance scheduling for offshore wind farms. *IISE Transactions*, Pages: 1–15.

[5] Lusby, R., Muller, L. F., & Petersen, B. (2013). A solution approach based on Benders decomposition for the preventive maintenance scheduling problem of a stochastic large-scale energy system. *Journal of Scheduling*, *16*, Pages: 605–628.

[6] Amaran, S., Zhang, T., Sahinidis, N. V., Sharda, B., & Bury, S. J. (2016). Medium-term maintenance turnaround planning under uncertainty for integrated chemical sites. *Computers & Chemical Engineering*, *84*, Pages: 422–433.

[7] Goel, V., & Grossmann, I. E. (2004). A stochastic programming approach to planning of offshore gas field developments under uncertainty in reserves. *Computers & Chemical Engineering*, *28*(8), Pages: 1409–1429.

[8] Goel, V., & Grossmann, I. E. (2006). A class of stochastic programs with decision dependent uncertainty. *Mathematical programming*, *108*, Pages: 355–394.

[9] Bhuiyan, T. H., Medal, H. R., & Harun, S. (2020). A stochastic programming model with endogenous and exogenous uncertainty for reliable network design under random disruption. *European Journal of Operational Research*, *285*(2), Pages: 670–694.

[10] Peeta, S., Salman, F. S., Gunnec, D., & Viswanath, K. (2010). Pre-disaster investment decisions for strengthening a highway network. *Computers & Operations Research*, *37*(10), Pages: 1708–1719.

[11] Leo, E., & Engell, S. (2022). Condition-based maintenance optimization via stochastic programming with endogenous uncertainty. *Computers & Chemical Engineering*, *156*, Article number: 107550.

[12] Leo, E., & Engell, S. (2023). Handling Type-I and Type-II endogenous uncertainties in simultaneous production planning and condition-based maintenance optimization in continuous production. *Computers & Chemical Engineering*, *174*, Article number: 108227.

[13] Basciftci, B., Ahmed, S., & Gebraeel, N. (2020). Data-driven maintenance and operations scheduling in power systems under decision-dependent uncertainty. *IISE transactions*, *52*(6), Pages: 589–602.

[14] Colvin, M., & Maravelias, C. T. (2008). A stochastic programming approach for clinical trial planning in new drug development. *Computers & Chemical Engineering*, *32*(11), Pages: 2626–2642.

[15] Zhu, Z., & Xiang, Y. (2021). Condition-based maintenance for multi-component systems: Modeling, structural properties, and algorithms. *IISE transactions*, *53*(1), Pages: 88–100.

[16] Zhu, Z., Xiang, Y., & Zeng, B. (2021). Multicomponent maintenance optimization: A stochastic programming approach. *INFORMS Journal on Computing*, *33*(3), Pages: 898–914.

[17] Haneveld, W. K. K., Van der Vlerk, M. H., & Romeijnders, W. (2019). *Stochastic programming: Modeling decision problems under uncertainty*. Springer Nature.

[18] Murphy, J. (2013). Benders, nested Benders and stochastic programming: An intuitive introduction. *arXiv preprint arXiv:1312.3158*.

[19] Mitici, M., de Pater, I., Barros, A., & Zeng, Z. (2023). Dynamic predictive maintenance for multiple components using data-driven probabilistic RUL prognostics: The case of turbofan engines. *Reliability Engineering & System Safety*, *234*, Article number: 109199.

[20] de Pater, I., Reijns, A., & Mitici, M. (2022). Alarm-based predictive maintenance scheduling for aircraft engines with imperfect Remaining Useful Life prognostics. *Reliability Engineering & System Safety*, *221*, Article number: 108341.

[21] Garstka, S. J. (1973). Stochastic programs with recourse: Random recourse costs only. *Management Science*, *19*(7), Pages: 747–750.

[22] Wolsey, L. A., & Nemhauser, G. L. (1988). *Integer and combinatorial optimization*. John Wiley & Sons.

[23] Kong, N., Schaefer, A. J., & Ahmed, S. (2013). Totally unimodular stochastic programs. *Mathematical Programming*, *138*, Pages: 1–13.

[24] Birge, J. R., & Louveaux, F. V. (1988). A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, *34*(3), Pages: 384–392.

[25] Infanger, G., & Morton, D. P. (1996). Cut sharing for multistage stochastic linear programs with interstage dependency. *Mathematical Programming*, *75*(2), Pages: 241–256.

[26] Saxena, A., & Goebel, K. (2008). *Turbofan engine degradation simulation data set*, NASA Prognostics Data Repository, NASA Ames Research Center, Moffett Field, California, USA.

[27] Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008, October 6-9). Damage propagation modeling for aircraft engine run-to-failure simulation. *International Conference on Prognostics and Health Management*, Denver, Colorado, USA, Pages: 1–9.

[28] Gal, Y., & Ghahramani, Z. (2016, June 19-24). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *Proceedings*

of The 33rd International Conference on Machine Learning, 48, New York, New York, USA, Pages: 1050–1059.

[29] Birge, J. R., & Louveaux, F. (2011). *Introduction to stochastic programming* (2nd ed.). Springer Science & Business Media.

[30] Fulkerson, D., & Gross, O. (1965). Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, *15*(3), Pages: 835–855.

[31] Wang, Y. H. (1993). On the number of successes in independent trials. *Statistica Sinica*, *3*(2), Pages: 295–312.

10 CONCLUSION

This thesis presents an overarching predictive aircraft maintenance framework. This framework describes, first, how to estimate the RUL of aircraft systems from the sensor measurements (Chapters 2 and 3), second, how to quantify the uncertainty of these RUL prognostics (Chapters 4 and 5), and last, how to use these RUL prognostics to optimize the aircraft maintenance planning (Chapters 6, 7, 8 and 9). Here, the main findings are presented (Section 10.1), the final conclusions are drawn (Section 10.2), and future research directions are specified (Section 10.3).

10.1. REVIEW OF THE PREDICTIVE MAINTENANCE CHALLENGES

In the Introduction (Chapter 1), three challenges of predictive aircraft maintenance were stated. How these challenges were addressed in this thesis is discussed below.

10.1.1. REVIEW OF CHALLENGE 1

The first challenge concerns the development of accurate point RUL prognostics for aircraft systems. Accurately estimating the RUL of aircraft systems is challenging because, first, there is a lack of failure data and, second, aircraft are operated under highly-varying conditions. In Chapter 2, these challenges were addressed by using an unsupervised learning method (a Long Short-Term Memory autoencoder) to create a health indicator for aircraft systems. This health indicator was then used to estimate the RUL with a similarity-based matching method. This method was applied to estimate the RUL of aircraft engines, with only six available failure instances. Even with so few failure instances, the RUL prognostics are accurate with a Root Mean Square Error (RMSE) of 2.67 flights only. Moreover, we handle the varying operating conditions by integrating these operating conditions at several places in the autoencoder. This improves the correlation between the health indicator and the operating time (trendability) by 45%.

Training a RUL prognostic neural network can be very time consuming. In Chapter 3, this training time was significantly reduced through developing a new initialization method for the weights in the last layer of a neural network. To initialize the weights, the initial loss was minimized by solving a constrained linear regression problem. In the case study, this new initialization strategy was applied in a neural network that estimates the RUL of aircraft engines. With the new initialization method, it is shown that 34% less epochs are needed to reach the same validation loss as the best benchmark initialization method (Kaiming initialization [1]). Here, an epoch is one iteration of training the neural network, where all training data are used once to fine-tune the weights. The new initialization strategy was also applied to an image classification neural network. The new initialization strategy, combined with transfer learning, gives the highest accuracy on the test set of the considered classification problem.

Overall, Chapter 2 provides a generic data-driven approach to estimate the RUL of aircraft systems, which can be applied to any (aircraft) system with high-frequency sensor measurements for which at least a few failure instances are available. With the method in Chapter 3, the training of the neural network in Chapter 2, or of other RUL prognostic neural networks, can be accelerated. Together, these two chapters provide an approach for the fast development of accurate RUL prognostics for aircraft systems.

10.1.2. REVIEW OF CHALLENGE 2

The second challenge is to improve the uncertainty quantification of the RUL prognostics, developing *probabilistic RUL prognostics*. In Chapter 4, the Probability Density Function (PDF) of the RUL of aircraft cooling units was estimated. First, several potential health indicators for the cooling units were evaluated. With the best health-indicator, the moment the cooling unit becomes unhealthy was subsequently identified with Chebyshev's inequality. Next, a dynamic time-warping clustering method was used to determine the degradation model of an unhealthy cooling unit, and the PDF of the RUL was estimated with this degradation model and particle filtering. In the case study, all cooling units are diagnosed as unhealthy between 40 to 2 flights before failure. Particle filtering already provides accurate RUL prognostics at this moment, with a RMSE of only 4.04 flights. Last, the inner workings of a filtering method are explainable, whereas black-box models such as neural networks are not.

While many metrics exist to evaluate *point RUL prognostics* (i.e., RUL prognostics without quantified uncertainty), there is a lack of metrics to evaluate probabilistic RUL prognostics. In Chapter 5, four new metrics were proposed to evaluate probabilistic RUL prognostics in the form of a PDF. The first two metrics, the Continuous Ranked Probability Score (CRPS) and the weighted CRPS, evaluate the accuracy and sharpness of a single probabilistic RUL prognostic. The other two metrics, the $\alpha-$Coverage and Reliability Score, evaluate the reliability of multiple probabilistic RUL prognostics. In the case study, a PDF of the RUL of aircraft engines was estimated with a Convolutional Neural Network (CNN) with Monte Carlo dropout. The new metrics shows that the obtained probabilistic RUL prognostics are reliable and accurate. However, they also indicate that the sharpness of the probabilistic RUL prognostics can still be improved.

Together, Chapters 4 and 5 provide an approach to develop and evaluate probabilistic RUL prognostics for aircraft systems. The model-based approach in Chapter 4 can be applied to aircraft systems for which a reliable health indicator is available, especially if an aircraft system exhibits different degradation trends. With the metrics in Chapter 5, the trustworthiness of probabilistic RUL prognostics in the form of a PDF can be evaluated before employing them in the maintenance planning.

10.1.3. REVIEW OF CHALLENGE 3

The last step, and third challenge, in predictive maintenance is to optimize the maintenance schedule for a fleet of aircraft with these RUL prognostics.

In Chapter 6, the aircraft maintenance schedule was optimized with imperfect point RUL prognostics. Maintenance was only planned once the RUL prognostic of a system fell below an alarm threshold several flights in a row, i.e., when an alarm was triggered, to avoid rescheduling maintenance tasks. The target maintenance date for an alarmed aircraft system was determined based on the RUL prognostic and a safety factor, to avoid failures due to errors in the RUL prognostic. The hyperparameters (the safety factor and the alarm threshold) of this optimization method were determined with a genetic algorithm. With this method, the maintenance schedule for aircraft engines was optimized, where the RUL was estimated with a CNN. The maintenance schedule was analysed with a Monte Carlo simulation. Due to the safety factor, on average only 1.6% of the maintenance tasks take place after the engine failed, while, due to the alarm threshold, on average only 8.2% of the maintenance tasks is rescheduled.

Probabilistic RUL prognostics were instead used to optimize the aircraft maintenance schedule in Chapter 7. First, the optimal maintenance moment for a single aircraft system was determined using the renewal-reward process. With a linear program, maintenance was subsequently scheduled for a fleet of aircraft. The probabilistic RUL prognostics allow a trade-off between maintaining a system now, avoiding a possible failure, versus postponing the maintenance, thus extending the lifetime. In the case study with aircraft engines, the PDF of the RUL was estimated with a CNN with Monte Carlo dropout. The risk of failure soon outweighs the benefits of extending the lifetime of an engine: The optimal maintenance moment for the engines is usually close to the lower bound of the 99% confidence interval of the estimated RUL. Due to these early optimal maintenance times, on average only 0.003 engines fail in ten years with a fleet of 50 aircraft, as estimated with a Monte Carlo simulation of the long-term maintenance schedule.

In Chapter 8, this problem was extended by including a limited number of spare components, and by considering a multi-component aircraft system with redundant components. For each component within the system, the RUL was estimated using a particle filtering algorithm, followed by an optimization of the maintenance planning with a linear program. In the case study, this method was applied to schedule maintenance for aircraft cooling systems, each equipped with four cooling units. The long-term maintenance schedule was analysed with a Monte Carlo simulation, which showed that on average 83% of the maintenance tasks take place before the failure of the cooling unit.

Last, the aircraft maintenance scheduling problem was formulated as a multi-stage stochastic program in Chapter 9. By solving this stochastic program with the nested Benders decomposition algorithm, the initial maintenance planning and the future updates were jointly optimized. A new clustering algorithm was proposed to accelerate the solution method, based on the endogenous (decision-dependent) uncertainty in the maintenance planning problem. In the case study, maintenance was planned for up to five aircraft engines with a planning horizon of four weeks (28 days/stages). Solving the stochastic program instead of the benchmark method (that does not consider future updates) reduces the expected costs by up to 0.89%. Moreover, with five engines, the same number of iterations of the nested Benders decomposition algorithm is executed

25 times faster with the clustering algorithm, than without the clustering algorithm. The computational time still grows exponentially with the number of engines, however, potentially limiting the implementation of this approach.

Concluding, Chapters 6, 7, 8 and 9 show how both point- and probabilistic RUL prognostics can be used to optimize the maintenance planning, while considering several maintenance aspects such as the fixed maintenance opportunities and the spare parts. By analyzing the long-term maintenance planning, these chapters also demonstrate the effect of predictive aircraft maintenance on the maintenance costs, the reliability of the aircraft and on the maintenance efficiency.

10.2. MAIN CONCLUSIONS

The overall conclusions from this thesis are summarized here.

1. Predictive aircraft maintenance increases the aircraft reliability and the maintenance efficiency, while reducing maintenance costs.

In Chapters 7 and 8, a predictive maintenance strategy (i.e., a maintenance strategy with RUL prognostics) was compared with a preventive maintenance strategy (i.e., a maintenance strategy without RUL prognostics). The predictive maintenance strategy results in more reliable aircraft, with less failures, while the maintenance becomes more efficient, with less maintenance tasks. In Chapter 7, the expected number of engine failures in ten years decreases from 61.6 with preventive maintenance to 0.003 with predictive maintenance, while the expected number of maintenance tasks decreases from 1159.2 to 925.8. In Chapter 8, the expected number of Aircraft On Ground events in five years is very low (not more than 0.1) with both preventive and predictive maintenance, while the expected number of maintenance tasks decreases from 135 to 106 with predictive maintenance. Overall, the maintenance costs decrease with 53% (Chapter 7) and 30% (Chapter 8) in the predictive maintenance strategy.

2. Better RUL prognostics will further increase the aircraft reliability and the maintenance efficiency, while further reducing maintenance costs.

In Chapters 6 and 7, a predictive maintenance strategy with the imperfect RUL prognostics from the RUL prognostic models was compared with the ideal case of maintenance with perfect RUL prognostics without any errors or uncertainty. In both chapters, the number of failures and the number of maintenance tasks decrease when considering perfect RUL prognostics. In Chapter 6, the expected number of engine failures in five years decreases from 13.61 to 0.10 with perfect RUL prognostics, while the expected number of maintenance tasks decreases from 819.7 to 739.0. In Chapter 7, the expected number of failures in ten years decreases from 0.003 to less than 0.001 with perfect RUL prognostics, while the expected number of maintenance tasks decreases from 925.8 to 825.8. Last, the maintenance costs decrease by 19.5% (Chapter 6) and 14% (Chapter 7) with the perfect RUL prognostics. From this, it is concluded that efforts to further improve the RUL prognostic models are worthwhile.

3. When only a limited number of failure instances is available, it is best to first develop a health indicator and only then estimate the RUL.

Usually, only very limited failure data are available for aircraft systems. It is therefore often not possible to train an accurate supervised learning model that directly estimates the RUL from the sensor measurements. Instead, more accurate RUL prognostics can be obtained by first developing a health indicator for the system, and then estimating the RUL with this health indicator. Whereas for some systems, a standard physical formula can be used to create a health indicator (see Chapter 4), other systems instead require training an unsupervised learning model with the unlabelled data samples from non-degraded systems, to develop a health indicator (see Chapter 2). The developed health indicator can subsequently be used to estimate the RUL, for instance using a filtering method (Chapter 4) or a similarity-based matching method (Chapter 2). In the case study in Chapter 2, the RMSE of the RUL prognostics made with the health indicator is indeed 19% lower than the RMSE of the RUL prognostics made with a supervised learning neural network. Moreover, compared with black-box models such as neural networks, RUL prognostics based on health indicators are explainable to human operators responsible for the maintenance tactics and strategies.

4. Quantifying the uncertainty of the RUL prognostics significantly improves predictive aircraft maintenance.

Throughout this thesis, both point RUL prognostics without quantified uncertainty (Chapters 2, 3 and 6) and probabilistic RUL prognostics (Chapters 4, 5, 7, 8 and 9) were considered. The maintenance planning was optimized with point RUL prognostics in Chapter 6. However, the method proposed in this chapter only works if many failure instances are available, which is often not the case for aircraft systems. Otherwise, the hyperparameters of this method cannot be optimized. In contrast, with probabilistic RUL prognostics, there is a clear, quantifiable trade-off between the risks and benefits of postponing maintenance for a system. This trade-off facilitates the maintenance planning. Moreover, the trustworthiness of probabilistic RUL prognostics can be easily evaluated with the metrics introduced in Chapter 5.

5. To improve predictive maintenance, it is important to include, adapt and combine mathematical and physical methods from different fields.

Throughout this thesis, a variety of mathematical methods from different fields were applied to predictive maintenance. In Chapter 2, a neural network from the field of natural language processing, developed for translating texts, was used to create a health indicator. In Chapter 3, an econometric method was applied to initialize the weights in the last layer of a neural network. In Chapter 4, the health indicators were clustered with dynamic time-warping, a method used in automatic speech recognition [2]. For the predictive maintenance planning, RUL prognostics from machine learning models were combined with optimization methods from operations research (Chapters 6, 7, 8 and 9). And last, in Chapters 4 and 8, the Root Mean Square, a formula used in physics to describe the energy content of a signal [3], was employed to develop a health indicator for the aircraft cooling units. Including, adapting and combining different mathematical and physical methods improves the predictive maintenance practice.

10.3. RECOMMENDATIONS FOR FUTURE RESEARCH

Many open challenges still exist that complicate the implementation of predictive aircraft maintenance in current-day practice. Below, some of these challenges are discussed and recommendations for future research are provided.

10.3.1. RECOMMENDATIONS REGARDING THE RUL PROGNOSTICS

1. Availability and quality of aircraft condition-monitoring data.

Modern aircraft health monitoring systems constantly measure the condition of aircraft components, generating huge amounts of sensor data [4]. It is extremely time-consuming and challenging to safely store, clean and document these huge amounts of data. First, it is expensive to store terabytes of data [5], especially since this storage should happen in a safe way, guaranteeing that the data are not corrupted [6]. Second, the data are often not clean, with many instances of missing or "wrong" data due to broken sensors [5]. Third, the written description of maintenance actions often varies per technician and can be hard to understand for others [7]. Last, it is sometimes unclear who actually owns the aircraft sensor data [8]. To obtain accurate RUL prognostic models, all these issues above should be addressed, and the whole aircraft data management process should be streamlined and improved.

2. No data about failure instances.

An important assumption in this thesis is that some data of failure instances always exist. However, for some aircraft systems, no failure data are available at all, for instance for new systems or for safety-critical systems [7]. For several aircraft components, usually part of non safety-critical systems with redundant components, it might be unclear when and even *if* the component failed [5]. For these components, it is unknown whether a data sample comes from a healthy, unhealthy or failed component, i.e., the sample cannot be labelled. Another challenge is therefore to develop a RUL prognostic model for systems without any failure instances, using only unlabelled data samples.

3. Explainability and validation of the RUL prognostic model.

The European Union Aviation Safety Agency (EASA) has to approve the RUL prognostic models of an airline. For this, it is essential to validate and verify the model on a validation and verification data set [6]. However, it is not possible to make a large validation and verification set if very few (or none) failure instances are available. Second, a RUL prognostic model has to be explainable for maintenance operators to be approved [6], which is difficult when using black-box models such as neural networks. Future research should therefore focus on developing RUL prognostic models that can be approved by EASA, i.e., for which a validation and verification procedure with limited failure data is developed and, second, that are explainable.

10.3.2. RECOMMENDATIONS REGARDING THE MAINTENANCE SCHEDULING

1. Maintenance scheduling for the full aircraft.

In this thesis, maintenance is planned for a single aircraft system, in multiple aircraft, based on the RUL prognostics. However, maintenance often needs to be planned for all systems in an aircraft combined. Whereas for some systems, RUL prognostics are available, for other systems, time-based deadlines for both inspections and maintenance tasks are required [9]. Optimizing the maintenance planning for a fleet of aircraft, when considering all systems in each aircraft, remains a daunting challenge.

2. Integration of the predictive maintenance planning and the flight scheduling.
The aircraft maintenance planning is intertwined with the flight scheduling, i.e., the assignment of aircraft to flights. Since all flights need an aircraft assigned to it, maintenance cannot be planned for too many aircraft at the same time. Moreover, in the flight schedule, it should be taken into account that an aircraft is at the right maintenance location at the planned maintenance moment. The flight scheduling problem also has constraints itself. For instance, each flight should get an aircraft with the right capacity (not too few/too many seats). Jointly optimizing the maintenance schedule with RUL prognostics and the flight schedule is therefore a challenging research direction.

3. Uncertainty in the probabilistic RUL prognostics.
An implicit assumption in this thesis is that the uncertainty quantification of the probabilistic RUL prognostics is "correct", i.e., that the estimated PDF of the RUL completely coincides with the true PDF of the RUL. For instance, if it is estimated with probability zero that the RUL of a system is 25 flights, it is assumed that this system indeed cannot fail at 25 flights. However, as indicated by the metrics in Chapter 5, the estimated PDFs of the RUL are not always correct. The estimated PDF should, if the estimation method is unbiased, converge to the true PDF when more labelled data samples become available. However, since typically only a limited number of labelled data samples is available, the uncertainty in the estimated PDF remains. Optimizing the predictive maintenance planning while taking this uncertainty in the estimated PDF of the RUL into account is therefore still an open problem.

Overall, this thesis provides a generic framework for predictive aircraft maintenance, that describes how to estimate the RUL with quantified uncertainty for aircraft systems, and how to use these RUL prognostics to optimize the aircraft maintenance planning. With these recommendations, this framework can be extended to improve, enhance and expand the current-day predictive aircraft maintenance process.

REFERENCES

[1] He, K., Zhang, X., Ren, S., & Sun, J. (2015, December 7-13). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *Proceedings of the IEEE International Conference on Computer Vision (ICCV),* Santiago, Chile, Pages: 1026–1034.

[2] Müller, M. (2007). *Information retrieval for music and motion*. Springer.

[3] Zhu, J., Nostrand, T., Spiegel, C., & Morton, B. (2014, September 29 - October 2). Survey of condition indicators for condition monitoring systems. *Proceedings of the Annual Conference of the Prognostics and Health Management (PHM) Society, 6*, Fort Worth, Texas, USA, Pages: 1–13.

[4] Badea, V. E., Zamfiroiu, A., & Boncea, R. (2018). Big data in the aerospace industry. *Informatica Economica, 22*(1), Pages: 17–24.

[5] Verhagen, W. J., Santos, B. F., Freeman, F., van Kessel, P., Zarouchas, D., Loutas, T., Yeun, R. C., & Heiets, I. (2023). Condition-based maintenance in aviation: Challenges and opportunities. *Aerospace, 10*(9), Article number: 762.

[6] *EASA concept paper: First usable guidance for level 1 machine learning applications* (1st ed., tech. rep.). (2023). European Union Aviation Safety Agency (EASA).

[7] Fink, O., Wang, Q., Svensen, M., Dersin, P., Lee, W.-J., & Ducoffe, M. (2020). Potential, challenges and future directions for deep learning in Prognostics and Health Management applications. *Engineering Applications of Artificial Intelligence, 92*, Article number: 103678.

[8] *From aircraft health monitoring to aircraft health management* (tech. rep.). (2022). International Air Transport Association (IATA).

[9] Tseremoglou, I., & Santos, B. F. (2024). Condition-based maintenance scheduling of an aircraft fleet under partial observability: A deep reinforcement learning approach. *Reliability Engineering & System Safety, 241*, Article number: 109582.

CURRICULUM VITÆ

Ingeborg de Pater was born on October 18, 1995 in Utrecht, the Netherlands. In 2014, she started her bachelor in Econometrics and Operations Research at the Erasmus University Rotterdam, the Netherlands. As part of her bachelor, she joined an exchange program to the City University of Hong Kong, she did a minor in Modern Physics at the Delft University of Technology, followed some philosophy courses and participated in the Bachelor Honours Research Class. Besides her studies, she worked as a student-assistant, teaching tutorials for various courses to first-year bachelor students.

After finishing her bachelor (Cum Laude), she continued with her master in Econometrics and Management Science at the Erasmus University Rotterdam, where she followed the track in Operations Research and Quantitative Logistics. She completed her master by writing a thesis during an internship at the Dutch Railways (NS) on rebalancing the rolling stock in the railway network. During her study, she was an active member of a student association, where she participated in various committees, and of the debating society, organizing a debating summer school and participating in numerous debating tournaments. During her master, she was the treasurer of this debating society.

After obtaining her master degree (Cum Laude), Ingeborg started in 2020 as a PhD candidate in the Air Transport and Operations group at the Aerospace Engineering faculty of the Delft University of Technology. Her research, as described in this thesis, was on predictive aircraft maintenance, where she estimated the Remaining Useful Life (RUL) of aircraft systems with quantified uncertainty, and optimized the aircraft maintenance schedule with these RUL prognostics. At the end of her PhD, she was also a visiting student at Computer Science faculty of Utrecht University, the Netherlands. She presented her work at various conferences during her PhD, and won several awards for her work.

In her free time, she enjoys cooking, trying recipes from different cuisines, baking, hiking (or, in the Netherlands, walking), visiting the zoo and museums, reading, mostly fiction books, and occasionally playing recorder.

List of Publications

Journal papers

8. de Pater, I., & Mitci, M. (2023). Predictive maintenance planning under endogenous uncertainty using stochastic programming with a novel clustering algorithm integrated in the nested Benders decomposition. *Under review at Mathematical Programming Series B.*

7. Landau, D. H. M. C., de Pater, I., Mitici, M., & Saurabh, N. (2023). A federated learning framework preserving data-privacy for airlines that collaboratively generate Remaining Useful Life prognostics. *Under review at Future Generation Computer Systems.*

6. de Pater, I., & Mitici, M. (2023a). A mathematical framework for improved weight initialization of neural networks using Lagrange multipliers. *Neural Networks*, *166*, Pages: 579–594.

5. Mitici, M., de Pater, I., Barros, A., & Zeng, Z. (2023). Dynamic predictive maintenance for multiple components using data-driven probabilistic RUL prognostics: The case of turbofan engines. *Reliability Engineering & System Safety*, *234*, Article number: 109199.

4. de Pater, I., & Mitici, M. (2023b). Developing health indicators and RUL prognostics for systems with few failure instances and varying operating conditions using a LSTM autoencoder. *Engineering Applications of Artificial Intelligence*, *117*, Article number: 105582.

3. de Pater, I., Reijns, A., & Mitici, M. (2022). Alarm-based predictive maintenance scheduling for aircraft engines with imperfect Remaining Useful Life prognostics. *Reliability Engineering & System Safety*, *221*, Article number: 108341.

2. de Pater, I., & Mitici, M. (2021). Predictive maintenance for multi-component systems of repairables with Remaining-Useful-Life prognostics and a limited stock of spare components. *Reliability Engineering & System Safety*, *214*, Article number: 107761.

1. Mitici, M., & de Pater, I. (2021). Online model-based Remaining-Useful-Life prognostics for aircraft cooling units using time-warping degradation clustering. *Aerospace*, *8*(6), Article number: 168.

Peer-reviewed conference papers

6. de Pater, I., & Mitici, M. (2023, September 3-7). Constructing health indicators for systems with few failure instances using unsupervised learning. *Proceedings of the 33st European Safety and Reliability Conference*, Southampton, UK, Pages: 3066–3073.

5. Mitici, M., de Pater, I., Zeng, Z., & Barros, A. (2023, September 3-7). Predictive maintenance planning using renewal reward processes and probabilistic RUL prognostics–Analyzing the influence of accuracy and sharpness of prognostics. *Proceedings of the 33st European Safety and Reliability Conference*, Southampon, UK, Pages: 1034–1041.

4. de Pater, I., & Mitici, M. (2022, July 6-8). Novel metrics to evaluate probabilistic Remaining Useful Life prognostics with applications to turbofan engines. *Proceedings of the 7th European Conference of the Prognostics and Health Management (PHM) Society*, *7*, Turin, Italy, Pages: 96–109.

3. Lee, J., de Pater, I., Boekweit, S., & Mitici, M. (2022, July 6-8). Remaining-Useful-Life prognostics for opportunistic grouping of maintenance of landing gear brakes for a fleet of aircraft. *Proceedings of the European Conference of the Prognostics and Health Management (PHM) Society*, *7*(1), Turin, Italy, Pages: 278–285.

2. de Pater, I., & Mitici, M. (2021, June 28 - July 2). Model-based Remaining-Useful-Life prognostics for aircraft cooling units. *Proceedings of the European Conference of the Prognostics and Health Management (PHM) Society*, *6*, Virtual, Pages: 1–8.

1. de Pater, I., del Mar Carillo Galera, M., & Mitici, M. (2021, September 19-23). Criticality-based predictive maintenance scheduling for aircraft components with a limited stock of spare components. *Proceedings of the 31st European Safety and Reliability Conference*, Angers, France, Pages: 55–62.

AWARDS

4. Professor Stein Haugen paper award, *European Safety and Reliability Conference*, 2023. For the paper "Constructing health indicators for systems with few failure instances using unsupervised learning".

3. Best paper award, second price, *European Conference of the Prognostics and Health Management Society*, 2022. For the paper "Remaining-Useful-Life prognostics for opportunistic grouping of maintenance of landing gear brakes for a fleet of aircraft".

2. Anna Valicek Award first place, *Annual student paper competition from the Airline Group of the International Federation of Operational Research Societies (AGIFORS)*, 2021. For the paper "Predictive maintenance for multi-component systems of repairables with Remaining-Useful-Life prognostics and a limited stock of spare components".

1. Best innovation award, *Presentation at the Annual AGIFORS Airline Operations & Maintenance Conference*, 2021.