Predicting Crack Growth and Fatigue Life with Surrogate Models

Simon Pfingstl\textsuperscript{a,\ast}, Jose Ignacio Rios\textsuperscript{a}, Horst Baier\textsuperscript{a} and Markus Zimmermann\textsuperscript{a}

\textsuperscript{a}Laboratory for Product Development and Lightweight Design, Technical University of Munich, Boltzmannstr. 15, 85748 Garching, Germany

ARTICLE INFO

Keywords:
- prognostic health monitoring
- surrogate models
- crack growth prediction
- fatigue damage prognosis
- probabilistic prediction
- uncertainty quantification

ABSTRACT

Fatigue-induced damage is still one of the most uncertain failures in structural systems. Prognostic health monitoring together with surrogate models can help to predict the fatigue life of a structure. This paper demonstrates how to combine data from previously observed crack evolutions with data from the currently observed structure in order to predict crack growth and the total fatigue life. We show the application of one physics-based model, which is based on Paris’ law, and four mathematical surrogate models: recurrent neural networks, Gaussian process regression, k-nearest neighbors, and support vector regression. For a coupon test, we predict the time to failure and the crack growth with confidence intervals. Moreover, we compare the performance of all proposed models by the mean absolute error, coefficient of determination, mean of log-likelihood, and their confidence intervals. The results show that the best mathematical surrogate models are Gaussian process regression and recurrent neural networks. Furthermore, this paper shows that the mathematical surrogate models tend to have conservative confidence intervals, whereas the physics-based model exhibits overly optimistic (too small) confidence intervals.

1. Introduction

Motivation. Fatigue-induced damage is still one of the most uncertain failures in structures [1]. Since this type of damage is difficult to compute precisely, inspection intervals for assuring the health of a structure cannot be scheduled perfectly. As seen recently, a big aircraft manufacturer is facing fatigue problems, where cracks occurred earlier than expected, which led to many recalls for inspection purposes [2]. This example shows that there is a need for methods in order to reduce these fatigue damage uncertainties.

Prognostic health monitoring could help to reduce this problem. The idea is to use machine learning algorithms to learn the fatigue behavior of a structural system through the previously collected data in order to predict the time to failure. Then, with the trained model, inspections can be scheduled for every structure individually and more precisely. However, these models are often not applied yet, e.g. in the aerospace industry.

Purpose and Concept. Figure 1 illustrates the concept of this paper. Herein, the predictions are based on two different types of data: first, data from tests with different specimens covering the entire fatigue life until failure (referred to as data set 1). Second, measurement data of the structure while being monitored (referred to as data set 2). Most publications relate their predictions only on data set 2. However, often there is a data set 1 available, e.g. the collected data during the development process. Since data set 2 provides valuable information about the structural system, we present the application of one physics-based model (PBM) and four mathematical surrogate models, namely recurrent neural networks (RNN), Gaussian process regression (GPR), k-nearest neighbors (KNN), and support vector regression (SVR), which are based on both data set types. After applying the surrogate models to a crack growth data set, we compare the accuracy of them.

For our study, the data set is divided into two sets: the training set, represented by data set 1, which is used to train the models, and the test set, which is data set 2 and considered as the currently collected data for monitoring the structure. This data serves as the input for the models and is also used to quantify the performance of the trained models. Data set 2 can be supplied sequentially to the models, which leads to a continuously updated fatigue life prediction. The models are trained and tested using a set of crack growth measurement data, published by Virkler [3].
Predicting Crack Growth and Fatigue Life with Surrogate Models

Following this introduction section, the paper presents a literature review and describes the crack growth data set afterwards. In the fourth and the fifth sections, the mathematical surrogate models and the PBM are explained. Section 6 compares the performance of the results. In the last two sections, these results are discussed and summarized.

2. Literature Review

Due to the increasing research in fields such as supervised machine learning, surrogate modeling and uncertainty quantification, many researchers have recently proposed different types of models for predicting fatigue life \[4\]. For example in \[5\], the authors predict the remaining useful life of a turbofan engine with RNN and the authors of \[6\] use RNN in order to predict the fatigue crack growth. In contrast to both papers, we use RNN as a mixture density model \[7\] in order to additionally obtain confidence intervals of the predicted time to failure. In \[8\], RNN is combined with a particle filter which also leads to a probabilistic prediction. However, using a particle filter requires physical knowledge about the structural system. As the estimated parameters of the physical behavior are not updated, and thus differ from the actual ones, the model leads to poor predictions.

The GPR model has recently been used frequently for probabilistic predictions. Mohanty et al. \[9\] measure the fatigue cycles, minimum load, maximum load, and load ratio and uses GPR to predict the current crack length of a compact-tension specimen. They assume a non-stationary multi-layer perceptron covariance function, whose hyperparameters are optimized by minimizing the negative log-likelihood. Moreover, Kwon et al. \[10\] utilize GPR to predict the remaining useful life of solder joints using radio-frequency impedance. They assume a squared exponential covariance function and train it with data set type 2. Hong and Zhou \[11\] also make assumptions about the covariance function (also called kernel). In \[11\], they use three different covariance functions, namely squared exponential, Matern class and a new type based on neural networks, and are able to predict the next period of the system response of a rolling element bearing. All previously mentioned papers have in common that the used GPR models have a limited look-ahead time. Therefore, Liu et al. \[12\] aim to overcome the limited look-ahead time by introducing a linear and a quadratic mean function and predict the health state of a lithium-ion battery using GPR. For this purpose, they add new hyperparameters which have to be optimized. Therefore, they need many current data points before they can start the prediction. In their study, they need 100 data points in order to predict the upcoming 68 events. To overcome the limited look-ahead time and/or a poor prediction quality (which is a result of a poorly prescribed kernel), we additionally consider previous data, which is illustrated in Figure 1. Since we know that the kernel must be different from the ones usually used (see Figure 2), we approximate it by computing the covariance matrix of the previously collected data. This covariance matrix can be interpolated by a mathematical surrogate model for predicting points between the collected input states. Therefore, no assumptions about the kernel need to be made, which leads to an unlimited look-ahead time and to a model that is usable from the first collected data point of type 2.

Additionally, we enable a KNN model \[13\] to predict the probabilistic crack growth behavior. This type of mathematical surrogate model has also been used in the context of health monitoring, e.g. in \[14\] they use KNN in order to detect anomalies in a cooling fan with ball bearings. In \[15\], the author uses KNN to identify five different gear crack levels. However, in the present paper, we use this model to predict the entire crack growth trajectory and the fatigue...
Predicting Crack Growth and Fatigue Life with Surrogate Models

**Computed covariance matrix**

**Trained squared exponential kernel**

**Trained rational quadratic kernel**

Figure 2: Comparison of kernels trained on Virkler crack growth data. Trained kernels do not match the computed covariance matrix and thus are not able to represent the system behavior correctly.

Moreover, SVR has been used to predict the fatigue life of a structure. For example in [16], the authors predict the remaining useful life of an aero-propulsion system with SVR, and in [17] they use XFEM and SVR to predict the fatigue crack growth. In the present paper, we combine SVR with a KNN error estimator for additionally obtaining confidence intervals.

3. Sequential Data of Crack Growth

To apply models on a prognostic health monitoring example, a sequential data set, which shows the deterioration of the structure, has to be collected in advance. Methods for sensor placing can be found, for example, in [18, 19, 20] and examples for finding the most appropriate sensors for the condition indicator in [21, 22]. It is possible to combine measurement data into a single condition indicator, a so-called damage index. Herein, a crack growth example is considered, where the crack length $a$ serves as the damage index.

The measurements of the used data set were performed by Virkler in 1977 [3]. The 68 test specimens with a total width of $W = 152.4$ mm were made of aluminum 2024-T3, which is a widely used material in the aerospace industry. The center-cracked plates with an initial half crack length of $a_0 = 9$ mm were tested under a constant amplitude loading with a stress range of $\Delta \sigma_{\infty} = 48.26$ MPa until a final half crack length of $a_f = 49.8$ mm was reached. In this study, the final fracture is assumed to occur at this final crack length. During the test, the number of cycles was measured every $\Delta a = 0.2$ mm until $a = 36.2$ mm, every $\Delta a = 0.4$ mm until $a = 44.2$ mm, and every $\Delta a = 0.8$ mm until $a = 49.8$ mm. Therefore, each crack growth trajectory consists of 164 data points (number of cycles). To reduce the sources of uncertainty, all specimens were taken from the same sheet of aluminum. Moreover, all tests were conducted by one single operator on the same test machine. Due to the scatter in material properties, the number of cycles at the final crack length $N_f$ varied. Figure 3 and 4 show the crack growth data set and the test setting, respectively.

In order to represent data set 1 and 2, the data is split into two sets: a training set (measurements 1-47) and a test set (measurements 48-68) (see Figure 3). Since the specimens were tested in a random order [3], this can be seen as a random choice. Again, the training set, which is used to train the models, represents data set type 1. In contrast, the test set, which is used to predict the fatigue life, represents data set type 2. The test data serves as the sequential input of the
model and is used to evaluate the performance. For two representative test trajectories, #48 and #50, the probabilistic predictions of a slow and a moderate crack growth are visualized. Figure 3 shows these two test trajectories.

4. Mathematical Surrogate Models

All of the presented models gain information from the training data. From this, the current monitored structural system produces more and more data over time. This data can be entered sequentially to the model in order to obtain a new prediction every time step. All proposed models are based on common literature. However, for each model specific extensions are made which are explained in the following.

4.1. Recurrent Neural Network (RNN)

Developed in the 1980s, RNNs are a class of neural networks (NN) that capture time dynamics. They constitute a set of traditional NNs in order to process information across time steps. Due to their internal state memory, they can treat a sequence of inputs [23]. This type of algorithm is regarded as an effective method to address time series [4].

Recently, they have been used in many applications for state predictions, see [6, 24, 25]. Due to the vanishing gradient problem that RNNs face during backpropagation, long short-term memory (LSTM) and gated recurrent units were developed [26, 27, 28]. Again, the idea is to provide more and more input data to the model as time passes. Therefore, a standard LSTM is used in this paper. For example in [6], LSTM is used to predict the crack growth. However, in the present paper the authors combine LSTM with a mixture density network (MDN) in order to obtain probabilistic predictions. This MDN represents the conditional probability density function (PDF) of the target variables [7]. Note, that we use non-italic letters for random variables. In this case, the RNN has two output variables: the mean \( \mu_{N_f} \) and the standard deviation \( \sigma_{N_f} \) of the random variable \( N_f \) (time to failure). A schematic representation of the model is depicted in Figure 5.

\[
\begin{align*}
\hat{\mu}_{N_f,t+2} & \quad \hat{\sigma}_{N_f,t+2} \\
[a_f^i, N_f^i] & \quad [a_{f+1}, N_{f+1}^i] & \quad [a_{f+2}, N_{f+2}^i]
\end{align*}
\]

Figure 5: Schematic representation of the RNN.

As in [29, 30], a normal distribution for the time to failure is assumed:

\[
p(N_f | a_f) = \frac{1}{\sqrt{2\pi\sigma_{N_f}}^2} e^{-\frac{(N_f - \mu_{N_f})^2}{2\sigma_{N_f}^2}} \tag{1}
\]

Like in the traditional NNs, the hidden unit, or rather the neuron in a layer, represents a basis function, which is connected with other neurons of the next layer. The training set serves to train, or rather optimize, the parameters of these basis functions of the NN. The objective function is a log-likelihood function as proposed in [31]. For a batch of one true measurement point and \( n \) prediction distributions, each prediction provides a PDF value and all together a likelihood of the batch

\[
p(N_f | a_f, \hat{\mu}_{N_f}^i, \hat{\sigma}_{N_f}^i) = \prod_{i=1}^{n^j} p(N_f | a_f, \hat{\mu}_{N_f, i}^j, \hat{\sigma}_{N_f, i}^j) \tag{2}
\]

where \( N_f^{i, j} \) is a random variable based on the \( i \)th realization and \( n \) indicates the total number of predictions for the
jth trajectory. In our case it is \( n' = n = 164 \). Due to very small likelihoods and the fact that logarithms transform multiplications to sums, this can be rewritten into

\[
\log(p(N_{j,f} \mid a_f, \mu_{N_f,j}, \sigma_{N_f,j})) = \sum_{i=1}^{n'} \log(N_{t,i} \mid a_f, \mu_{N_f,j}, \sigma_{N_f,j}))
\]

and the log-likelihood over all trajectories to

\[
\log(p(N_{j,f} \mid a_f, \mu_{N_f,j}, \sigma_{N_f,j})) = \sum_{j=1}^{t} \sum_{i=1}^{n'} \log(p(N_{t,i} \mid a_f, \mu_{N_f,j}, \sigma_{N_f,j}))
\]

where \( t \) is the total number of trajectories. Additionally, in order to avoid dependencies on the batch size, or rather the number of predictions and trajectories, the function is averaged

\[
\bar{L} = \log(p(N_{j,f} \mid a_f, \mu_{N_f,j}, \sigma_{N_f,j})) = \frac{1}{t} \sum_{j=1}^{t} \frac{1}{n'} \sum_{i=1}^{n'} \log(p(N_{t,i} \mid a_f, \mu_{N_f,j}, \sigma_{N_f,j}))
\]

which in this paper will be referred to as the mean of the log-likelihood \( \bar{L} \). Herein, this objective function is to be maximized with respect to the parameters of the basis functions. Since the Adam optimizer was proven to outperform other alternatives in [32], it was used in this study. According to [32], the hyperparameters, such as learning rate, number of epochs, exponential decay rate and small value for numerical stability, were set to the recommended default values. After the training process, the model can be used to predict the mean and the standard deviation of the time to failure. This prediction can be done sequentially (see Algorithm 1). The figures in Tables 2 and 3 in the results section visualize the fatigue life prediction of the RNN for trajectories #48 and #50 with a 95% confidence interval.

**Algorithm 1:** Training and prediction of fatigue life with RNN

**Result:** \( \hat{\mu}_{N_f,j} \) and \( \hat{\sigma}_{N_f,j} \).

Train RNN model with maximization of the log-likelihood \( \rightarrow \) mean and standard deviation prediction; 

\( i = 1; \)

while \( i \leq n \) (for every new current crack length) do

- Get current crack lengths \( a^{(2)}_{1,i} \) and cycles \( N^{(2)}_{1,i} \) of data set 2;
- RNN model with \( [a^{(2)}_{1,i}, N^{(2)}_{1,i}] \) as inputs \( \rightarrow [\hat{\mu}_{N_f,j}, \hat{\sigma}_{N_f,j}] \);

\( i = i + 1; \)

end

**4.2. Gaussian Process Regression (GPR)**

A Gaussian process is a generalization of the Gaussian probability distribution. A stochastic process governs the properties of functions, whereas a probability distribution describes random variables, which are scalars or vectors (for multivariate distributions). Loosely thinking of a function as a very long vector with entries of the function values \( f(x) \) for a particular input \( x \), where \( f(x) \) follows a Gaussian distribution, leads to the main idea of Gaussian processes [33]. According to [33], a Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution. A joint Gaussian distribution is fully specified by a mean vector \( \mu \) and a covariance matrix \( \Sigma \) with

\[
f \sim N(\mu, \Sigma)
\]

Here, the number of cycles based on the deterioration of the structure represented by the damage index (which is in this case, the crack length \( a \)) is seen as a Gaussian process \( f(x) = N(a) \)

\[
N(a) \sim \mathcal{GP}(m(a), K(a, a'))
\]
with the mean function \(m(\mathbf{a})\)

\[m(\mathbf{a}) = \mathbb{E}[\mathbf{N}(\mathbf{a})]\]  
(8)

and the covariance function \(K(\mathbf{a}, \mathbf{a}')\)

\[K(\mathbf{a}, \mathbf{a}') = \mathbb{E}[(\mathbf{N}(\mathbf{a}) - m(\mathbf{a}))(\mathbf{N}(\mathbf{a}') - m(\mathbf{a}'))]\]  
(9)

Since the Virkler data set is measured with a fixed crack increment \(\Delta \mathbf{a}\), the input variable is represented by the crack length and the output is the distribution over \(\mathbf{N}\). In reality it would be the other way around, since the damage index is usually measured over a fixed time step. In realistic modeling situations, there is no access to the function value itself, but only to a noisy version of it. Therefore, an additive independent identically distributed Gaussian noise with a variance \(\sigma_{\mathbf{N}}^2\) is assumed. Thererfore,

\[p(\mathbf{N}(\mathbf{a}_+, \mathbf{a}_-)) = \mathcal{N}_+(\mathbf{N}^+ | \mathbf{N}_+ \sim \mathcal{N}
\left(\begin{bmatrix} \mathbf{m}_+ \\ \mathbf{m}^+ \end{bmatrix}, \begin{bmatrix} \mathbf{K}(\mathbf{a}_+, \mathbf{a}_+) & \mathbf{K}(\mathbf{a}_+, \mathbf{a}_+) \\
\mathbf{K}(\mathbf{a}_+, \mathbf{a}_+) & \mathbf{K}(\mathbf{a}_+, \mathbf{a}_+) + \sigma_{\mathbf{N}}^2 \mathbf{I} \end{bmatrix}\right)\right)\]  
(10)

where (+) is used for known states, and (*) for states to predict. Note that

\[K^T_{+*} = \mathbf{K}_{+*} = \mathbf{K}(\mathbf{a}_+, \mathbf{a}_*)\]  
(11)

In order to make predictions dependent on the currently measured data, the known data acts as a conditional on the distributions

\[p(\mathbf{N}_+ | \mathbf{N}_+) = \frac{p(\mathbf{N}_+ | \mathbf{N}_+)}{p(\mathbf{N}_+)}\]  
(12)

\[= \mathcal{N}(\mathbf{m}_+ + \mathbf{K}_{+*} \mathbf{K}^{-1}_{+*}(\mathbf{N}_+ - \mathbf{m}_+), \mathbf{K}_{+*} - \mathbf{K}_{+*} \mathbf{K}^{-1}_{+*} \mathbf{K}^T_{+*})\]  
(13)

where the kernel is updated with the currently collected realizations \(\mathbf{N}_+\). Herein, the mean function \(m(\mathbf{a})\) is approximated by taking the mean of the training set, which represents the already gained knowledge of the structural system (data set 1). Moreover, the computed covariance matrix of the training set serves as the discrete approximation of the covariance function \(K(\mathbf{a}, \mathbf{a}')\). Therefore, the distributions over \(\mathbf{N}\) can only be computed at the discrete points of \(\mathbf{a}\). If other states of \(\mathbf{a}\) shall be computed, \(m(\mathbf{a})\) and \(K(\mathbf{a}, \mathbf{a}')\) can be interpolated between the supporting points. The only parameter to be tuned is the measurement noise represented by the variance \(\sigma_{\mathbf{N}}^2\). Usually this is a scalar. However, it is stated in [3] that the crack length is measured with the same accuracy during the test. Since the variance \(\sigma_{\mathbf{N}}^2\) is related to the fatigue life and not to the crack lengths, the measurement noise is dependent on the cycles. Therefore, the authors computed a mean crack growth rate \(\Delta \mathbf{a} / d \mathbf{N}\) with Equations 18, 19, and 20. Additionally, a multiplication factor \(\theta\) is introduced, which has to be tuned.

\[\sigma_{\mathbf{N}}^2 = \theta^2 \frac{d \mathbf{a}}{d \mathbf{N}}\]  
(14)

In order to optimize \(\theta\), a leave-one-out cross validation is conducted. The leave-one-out cross validation is a special type of cross validation, where \(k_\times\) is equal to the amount of data. In this case, the training data is represented by 47 trajectories. Therefore, the performance is evaluated 47 times, each with a different trajectory as the validation set. In this optimization problem, the mean of the log-likelihood \(\mathcal{L}\) serves as the objective function. \(\theta = 909790 \sqrt{\text{cycle}^3 / \text{m}}\) is found to give the highest \(\mathcal{L}\). If the measurement noise is a scalar, the optimization can be done directly with respect to \(\sigma_{\mathbf{N}}\).

Now, with this model, the mean and the variance of the remaining trajectory can be predicted. This can be done sequentially by computing the new conditional distribution due to the recently provided measurement point with Equation 13. The procedure is shown in Algorithm 2. The figures in Tables 2 and 3 in the appendix show the probabilistic fatigue life prediction of the GPR model, whereas the figures in Table 4 and 5 in the results section display the GPR crack growth predictions for trajectories #48 and #50 at two different time states.
Algorithm 2: Training and prediction of crack growth with GPR

Result: $\hat{\mu}_{N_{i+1:end}}$ and $\hat{\sigma}_{N_{i+1:end}}$

Train and set measurement noise $\sigma_N$;
Compute mean of number of cycles at each crack length $a^{(1)}_i$ of data set 1;
Use interpolation scheme for getting a mean function $m(a)$;
Compute covariance matrix of number of cycles between different trajectories of data set 1;
Use interpolation scheme for getting a covariance function $K(a_i, a_j)$ (all combinations between array components of $a_i$ and $a_j$ are considered e.g. with a mesh grid);
$i = 1$;

while $i \leq n$ (for every new current crack length) do

Get current crack lengths $a^{(2)}_{1:i}$ and number of cycles $N^{(2)}_{1:i}$ of data set 2;
Compute mean of number of cycles at given crack lengths $m_s = m(a_{1:i})$;
Compute covariance of number of cycles at given crack lengths $K_{++} = K(a_{1:i}, a_{1:i}) + \sigma_{N,1:i}I$;
Compute mean of number of cycles at crack lengths to predict $m^* = m(a_{i+1:end})$;
Compute covariance of number of cycles at crack lengths to predict $K_{++} = K(a_{i+1:end}, a_{i+1:end})$;
Compute covariance between given and predicted crack lengths $K_{+*} = K(a_{i+1:end}, a_{1:i})$;
Compute updated mean prediction $\hat{\mu}_{N_{i+1:end}} = m_s + K_{+*}K_{++}^{-1}(N^{(2)}_{1:i} - m^*)$;
Compute standard deviation of prediction $\hat{\sigma}_{N_{i+1:end}} = \text{diag} \left( K_{++} - K_{+*}K_{++}^{-1}K_{+*}^T \right)$;

$i = i + 1$;
end

Figure 6: 14 nearest neighbors of KNN model for trajectory #50 at $a = 10\, \text{mm}$ (left) and at $a = 20.4\, \text{mm}$ (right).

4.3. K-Nearest Neighbors (KNN)

The k-nearest neighbors model is a widely used method for classification and regression problems [34]. For regression, the output is the average of the $k$ nearest neighbor values. Moreover, it is possible to compute the standard deviation of them. In our case, we use this model to predict the crack growth trajectory. We compute the Euclidean distance between data set 1 and the current line (data set 2), resulting in a ranking of trajectories. The ranking indicates which previous crack growth curve is the closest to the current one and which is the furthest. After computing the ranking, the $k$ closest trajectories are taken in order to compute the mean and the standard deviation. This serves as the prediction for the current crack growth. The procedure is shown in Algorithm 3. The authors also applied a Gaussian weighting scheme to the $k$ nearest neighbors, which led to a worse result than without using it. Figure 6 illustrates the idea of the applied KNN model for two different input sizes, with the 14 nearest neighbors of the current data shown. The grey lines indicate the previous data from which the closest lines are chosen. In order to find the best value for $k$, a leave-one-out cross validation with a mean of the log-likelihood $\bar{L}$ as the performance metric is conducted. The highest $\bar{L}$ is reached for $k = 14$. Now, as described before, the probabilistic prediction can be made by computing the mean and the standard deviation of the 14 nearest neighbors. The figures in Tables 2 and 3 in the appendix show the probabilistic fatigue life prediction of the KNN model, whereas the figures in Tables 4 and 5 in the results section display the KNN crack growth predictions for trajectories #48 and #50 at two different time states.
Algorithm 3: Training and prediction of crack growth with KNN

\textbf{Result:} \( \hat{\mu}_{N_f} \) and \( \hat{\sigma}_{N_f} \). By setting \( a_f \) to multiple crack lengths, the entire crack path can be predicted.

Load all cycles \( N^{(1)}_{i,j} \) of data set 1;
Set \( k \) for nearest neighbors;
\( i = 1; \)
\textbf{while} \( i \leq n \) (for every new current crack length) \textbf{do}
Get current cycle \( N^{(2)}_i \) of data set 2;
\( j = 1; \)
Compute the Euclidean distance compared to data set 2;
\textbf{while} \( j \leq t \) (for every trajectory of data set 2) \textbf{do}
\begin{align*}
d_j &= \sqrt{\sum_i (N^{(1)}_{1:i,j} - N^{(2)}_{1:i})^2}; \\
j &= j + 1;
\end{align*}
\textbf{end}
Sort trajectories of data set 1 \( N^{(1)}_{f,j} \) with respect to Euclidean distance \( d_j \);
\( \hat{\mu}_{N_f} = \text{mean}(N_{f,j=1:k}); \)
\( \hat{\sigma}_{N_f} = \text{standard deviation}(N_{f,j=1:k}); \)
\( i = i + 1; \)
\textbf{end}

4.4. Support Vector Regression (SVR)

The SVR model [35, 36, 37, 38] is essentially a regression algorithm, and like most regression algorithms, it requires a fixed number of inputs. In a real scenario, new data points of the current measurement, in this case the crack length and the number of cycles, become available over time. Inspired by previous work on remaining useful life estimation [16], a sliding window with a fixed size (4 data points on average, which is \( \approx 10\% \) of the fatigue life) is used to collect the most recent data points. Each time a new measurement is available, the window slides one step to the right, maintaining its fixed size. The mean of those data points as well as the trend coefficient (the slope of a linear regression fitted for those points) will be used as the two inputs for the SVR model in order to predict the time to failure. This is shown in Figure 7. For further information, see [16].

Due to the fact that SVR models do not provide a probabilistic output, an error estimator is applied. This estimates the error made by the SVR model, which will serve as the standard deviation for an assumed normal distribution of the time to failure. The model chosen to predict the error is KNN, as it has been proven in other cases to work better than alternatives for predicting this error [39]. For this process, the SVR model is trained first in order to make the mean prediction. Following this, the KNN model is trained to output the standard deviation.

For the non-linear \( \varepsilon \)-SVR model, which is implemented in the Scikit-Learn library, the default radial basis function (RBF) kernel is used:

\[ k(x_n, x_m) = e^{-\gamma ||x_n - x_m||^2} \quad (15) \]
There are three hyperparameters to tune: \( C_{\text{svr}} \), \( \epsilon \) and \( \gamma \). The parameters \( C_{\text{svr}} \) and \( \epsilon \) are related to the optimization problem associated with the SVR, whereas \( \gamma \) is related to the kernel function. The cost function to minimize is

\[
L = \frac{1}{2} \| w \|^2 + C_{\text{svr}} \sum_{i=1}^{n} (\xi_i + \xi_i^*)
\]

subject to

\[
\begin{align*}
|y_i - \langle w, x_i \rangle - b| & \leq \epsilon + \xi_i \\
\langle w, x_i \rangle + b - y_i & \leq \epsilon + \xi_i^* \\
\xi_i, \xi_i^* & \geq 0
\end{align*}
\]

where \( x_i \) are the independent variables, \( y_i \) the dependent ones, and \( w \) the weights assigned to each variable. The prediction is \( \langle w, x_i \rangle + b \) (bias). A margin of tolerance is defined by \( \epsilon \), where the predictions that fall within this interval do not penalize the cost function. The predictions that fall outside of it by \( \xi_i \) or \( \xi_i^* \) are penalized linearly with the penalization constant \( C_{\text{svr}} \).

In order to select these three parameters in a reasonable manner, a grid search with different values for the hyperparameters is conducted. This grid contains all possible combinations of parameter values shown in Table 1. The performance of the SVR model with all combinations of the grid is obtained through a k-fold cross validation with \( k_x = 5 \). In a k-fold cross validation, the training set is divided into \( k_x \) parts, where \((k_x - 1)\) parts serve as the training set and the remaining part serves as the validation set for the performance evaluation. The training procedure is repeated \( k_x \) times with different validation sets and the performance is averaged over all \( k_x \) performance evaluations. In this, the coefficient of determination \( R^2 \) serves as the performance metric. However, in general the parameters are picked close to the default values.

According to [40], the \( C_{\text{svr}} \) parameter has only a negligible effect on the generalization. Moreover, \( \gamma \) has only a minor effect on the performance. Therefore, the two parameters are selected in order to over-fit less and thus provide a better generalization potential. The bigger \( C_{\text{svr}} \), the more penalization is assigned to the values that fall outside of the tolerance margin. That is why smaller values are preferred and \( C_{\text{svr}} = 1 \) is selected. As for \( \gamma \), the higher it is, the smaller range of influence each point has. Therefore, smaller values are also preferred and \( \gamma \) is set equal to 1. Following this, the highest \( R^2 \) value is found for \( \epsilon = 10^{-2} \). These parameters could be more finely tuned, but it would end up in the best version for the specific available data set and not necessarily the best version for the general problem.

In the next step, the error estimator model is trained. For this, the absolute error between the realization and the SVR prediction is computed. This error shall be predicted by the error estimator model and is used as the predicted standard deviation. In our case, we use a KNN model, where a neighbor is a set of mean and slope. In order to choose the number of neighbors, the coefficient of determination \( R^2 \) of different \( k \) values, obtained through a k-fold cross validation with \( k_x = 10 \), is compared again. The number of folds used is larger due to the fact that there is only one parameter being explored. The evaluated number of neighbors ranges from 50 neighbors to 400, with steps of 50. The part of the training set has 3,754 data points, which means that 50 neighbors are the closest 1.33% and 400 are the closest 10.66% of this training set. A lower \( k \) value means using less points to make predictions, and might lead to over-fitting. Therefore, higher values are preferred. All results have a small \( R^2 \) value close to 0.2, as this model tries to predict the error of the SVR. High \( R^2 \) values would mean that the SVR model is consistently making similar errors and therefore has a bias issue. Due to a performance drop at \( k = 250 \), \( k = 200 \) is selected as the number of neighbors.

After the two training procedures have been completed, the models can predict the mean and the standard deviation.
Predicting Crack Growth and Fatigue Life with Surrogate Models

of the time to failure sequentially (see Algorithm 4). The figures in Tables 2 and 3 in the results section show the probabilistic fatigue life prediction of the SVR model for trajectories #48 and #50 with a 95% confidence interval.

Algorithm 4: Training and prediction of fatigue life with SVR

Result: \( \hat{\mu}_{N_f} \) and \( \hat{\sigma}_{N_f} \)

Train SVR model \( \rightarrow \) mean prediction;

Compute absolute errors between SVR model and true value;

Train KNN model to predict absolute errors \( \rightarrow \) standard deviation prediction;

\( i = 1 + 4 \);

while \( i \leq n \) (for every new current crack length from the fifth data point) do

Get current crack lengths \( a_{i-4:i}^{(2)} \) and cycles \( N_{i-4:i}^{(2)} \) of data set 2;

Compute mean and slope of \( a_{i-4:i}^{(2)}, N_{i-4:i}^{(2)} \);

SVR model with mean and slope as inputs \( \rightarrow \hat{\mu}_{N_f} \);

KNN model with mean and slope as inputs \( \rightarrow \hat{\sigma}_{N_f} \);

\( i = i + 1 \);

end

5. Physics-based Model

Additionally to the mathematical surrogate models we present a PBM in order to predict the crack growth. The Paris law \[41\] is a widely applied model used to describe the crack growth behavior in a metal plate

\[
\frac{da}{dN} = C(\Delta K)^{m^*}
\]

where \( C \) and \( m^* \) are material properties, which could be correlated according to \[42\]. \( da/dN \) is the crack growth rate in \( m/\text{cycle} \) and \( \Delta K \) is the range of the stress intensity factor in \( MPa\sqrt{m} \). \( \Delta K \) can be computed as

\[
\Delta K = \beta \Delta \sigma_{\infty} \sqrt{\pi a}
\]

where \( \beta \) is a geometry factor, which is in general dependent on the crack length. According to \[43\], for the present test case, \( \beta \) can be expressed as

\[
\beta = \frac{1}{\sqrt{\cos(\frac{\pi a}{W})}}
\]

As in \[29, 30\], this study assumes \( m^* \) to be fixed. According to \[30\], the exponent is set to \( m^* = 2.9 \). This is also quite close to what is proposed in \[44\]. The material parameter \( C \) can be approximated from the Virkler data as

\[
C_i = \frac{a_{i+1} - a_i}{N_{i+1} - N_i} (\Delta K)^{-m^*}
\]

After doing this for the entire training data, we estimate the parameter \( C \) for each measurement sequence by taking the mean of all corresponding \( C_i \). Then, the mean and the standard deviation over all measurement sequences can be estimated. This leads to \( \hat{\mu}_C = 8.7096 \times 10^{-11} \) and \( \hat{\sigma}_C = 6.5680 \times 10^{-12} \) (\( da/dN \) expressed in \( m/\text{cycle} \) and \( \Delta K \) in \( MPa\sqrt{m} \)), respectively. The procedure is summarized in Algorithm 5.

In \[29, 30\], the authors assume a normal distribution of \( C \), even though the material parameter is only defined on the positive domain. Sampling \( C \) from a normal distribution could lead to realizations that are non-physical. Since in \[30, 45\], it is stated that the PDF \( p(N|a) \) of \( N \) over a fixed crack length \( a \) follows predominantly a lognormal distribution, it can be derived that \( C \) follows a lognormal distribution, too. By making the assumption that \( C \) does not depend on \( a \),
Algorithm 5: Estimate mean and standard deviation of $C$ with data set 1

**Result:** $\hat{\mu}_C$ and $\hat{\sigma}_C$

- Initialization of $C_i$ and $C_j$;
- Set parameters $W$, $\Delta \sigma_\infty$ and $m^*$;
- $i = 1$ and $j = 1$;

**while** $j \leq t$ (for every trajectory of data set 1) **do**

- Load crack lengths $a$ and cycles $N$ of trajectory $j$;
- **while** $i \leq n-1$ (trajectory length minus 1) **do**
  - $\beta = \frac{1}{\sqrt{\cos^2 \left( \frac{\pi a}{W} \right)}}$;
  - $\Delta K = \beta \Delta \sigma_\infty \sqrt{\pi a}$;
  - $C_i = \frac{a_{i+1} - a_i}{N_{i+1} - N_i} (\Delta K)^m$;
  - $i = i + 1$;

- $C_j = \text{mean}(C_{i=1:n-1})$;
- $j = j + 1$;

**end**

$\hat{\mu}_C = \text{mean}(C_{j=1:t})$;

$\hat{\sigma}_C = \text{standard deviation}(C_{j=1:t})$;

with the initial condition $N_0 = 0$, the differential equation of (18) can be solved as

$$N_f = \frac{1}{C \Delta \sigma_\infty^m} \int_{a_0}^{a_f} \sqrt{\frac{\pi a}{\cos \left( \frac{\pi a}{W} \right)}}^{-m^*} \, da$$

(22)

with

$$\alpha = \frac{\int_{a_0}^{a_f} \sqrt{\frac{\pi a}{\cos \left( \frac{\pi a}{W} \right)}}^{-m^*} \, da}{\Delta \sigma_\infty^m}$$

(23)

Then, the lognormal distribution of $C$ reads:

$$Z \sim \mathcal{N}(0, 1)$$

$$\log(N_f) \sim \mathcal{N}(\hat{\mu}_{N_f}, \hat{\sigma}_{N_f})$$

$$\log(N_f) = \hat{\mu}_{N_f} + \hat{\sigma}_{N_f} Z$$

$$\log\left( \frac{1}{N_f} \right) = -\hat{\mu}_{N_f} + \hat{\sigma}_{N_f} Z$$

$$\log\left( \frac{\alpha}{N_f} \right) = \log(\alpha) + \log\left( \frac{1}{N_f} \right)$$

$$C = \frac{\alpha}{N_f} = e^{\log(\alpha) + \log\left( \frac{1}{N_f} \right)}$$

$$C = e^{\log(\alpha) - \hat{\mu}_{N_f} + \hat{\sigma}_{N_f} Z}$$

$$\log(C) \sim \mathcal{N}(\log(\alpha) - \hat{\mu}_{N_f}, \hat{\sigma}_{N_f})$$

However, this is only for sampling reasons assuring positive numbers for $C$. By comparing a lognormal and normal distribution of $p(N_f|a_f)$, the authors found out that in this example the differences are negligible. Therefore, for simplicity a normal distribution at $a = a_f$ for $p(N_f|a_f)$ is assumed. This is also the case in all proposed mathematical surrogate models. As explained before, the mean and the standard deviation of $C$ result from the training set. Now, several values for $C$ can be sampled. From there, the current crack length of the test data as well as the Monte-Carlo
sampled \( C \) are put into Equation 22 in order to compute the crack growth path and the time to failure. This can be done several times and for multiple crack lengths. In this example, the authors computed the entire crack path 100 times. Every computed crack path has a different randomly chosen \( C \). From the computed trajectories, the mean and the standard deviation can be evaluated and then compared to the actual measurement results. As the measurement setup provides a new data point, the computation can be repeated with the new initial crack length. The model procedure and the results for two different time states of trajectory #50 are shown in Algorithm 6 and in Figure 8.

### Algorithm 6: Predict crack growth with PBM

Result: \( \hat{\mu}_{N_f} \) and \( \hat{\sigma}_{N_f} \). By setting \( a_f \) to multiple crack lengths, the entire crack path can be predicted.

1. Set parameters \( W, \Delta \sigma, m^*, \hat{\mu}_C, \) and \( \hat{\sigma}_C \);
2. \( i = 1 \);
3. while \( i \leq n \) (for every new current crack length) do
   - Get current crack length \( a_i \) and cycle \( N_i \) of data set 2;
   - \( \alpha = \frac{\int_{a_i}^{a_f} \frac{a}{\cos(\alpha)} \, da}{\Delta \sigma^m} \);
   - \( p = 1 \);
   - while \( p \leq 100 \) (do 100 times) do
     - Sample \( C \) with \( C \sim \mathcal{N}(\hat{\mu}_C, \hat{\sigma}_C) \);
     - \( N_{f,p} = \frac{a}{C} \);
     - \( p = p + 1 \);
   - end
   - \( \hat{\mu}_{N_f} = \text{mean}(N_{f,p=1:100}) \);
   - \( \hat{\sigma}_{N_f} = \text{standard deviation}(N_{f,p=1:100}) \);
   - \( i = i + 1 \);
end

### 6. Results

For convenience, the resulting prediction plots of trajectories #48 and #50 are listed in Tables 2, 3, 4, and 5. Tables 2 and 3 show the prediction of the fatigue life with confidence intervals. There, the x-axis indicates the number of cycles of the currently collected data. As the test continues, the input data becomes bigger and the prediction is redone. Moreover, Tables 4 and 5 show the probabilistic predictions of the entire crack growth path at two different times.
Table 2
Probabilistic prediction of cycles to failure $N_f$ and crack growth of trajectory #48.

| Method | PBM | RNN | SVR | GPR | KNN |
|--------|-----|-----|-----|-----|-----|
|        | ![Graph](image1) | ![Graph](image2) | ![Graph](image3) | ![Graph](image4) | ![Graph](image5) |
|        | ![Graph](image6) | ![Graph](image7) | ![Graph](image8) | ![Graph](image9) | ![Graph](image10) |
Table 3
Probabilistic prediction of cycles to failure $N_f$ and crack growth of trajectory #50.

| Model | PBM | RNN | SVR | GPR | KNN |
|-------|-----|-----|-----|-----|-----|
| $N_f$ | \(3 \times 10^5\) | \(3 \times 10^5\) | \(3 \times 10^5\) | \(3 \times 10^5\) | \(3 \times 10^5\) |
| Cycle to failure $N_f$ | \(5 \times 10^5\) | \(5 \times 10^5\) | \(5 \times 10^5\) | \(5 \times 10^5\) | \(5 \times 10^5\) |
| Confidence | \(95\%\) | \(95\%\) | \(95\%\) | \(95\%\) | \(95\%\) |

Simon Pfingstl et al.; Preprint submitted to Mechanical Systems and Signal Processing
Table 4
Probabilistic prediction of the entire path for trajectory #48.
Table 5
Probabilistic prediction of the entire path for trajectory #50.
Furthermore, different performance measures are used in order to compare the accuracy of the models. In [46], several performance measures were published in the context of prognostic health monitoring. However, the authors do not list any probabilistic performance measures. Besides a deterministic measure, we suggest two additional probabilistic measures in order to account for the probabilistic behavior of our predictions: the mean of the log-likelihood $\bar{\mathcal{L}}$ of Equation 5 and a method to compare the provided confidence intervals. In total, four different performance metrics are evaluated for each model.

First, the mean prediction is compared to the actual measurement values of the test set. For this purpose, two performance metrics are evaluated: the coefficient of determination $R^2$, which is a relative measure

$$R^2 = 1 - \frac{\sum_{j=1}^{t}(N^f_j - \bar{\mu}_N)^2}{\sum_{j=1}^{t}(N^f_j - \bar{N}_f)^2}$$  \hfill (24)$$

where

$$\bar{\mu}_N = \frac{1}{n} \sum_{i=1}^{n} \hat{\mu}_{N,f,i}$$  \hfill (25)$$

and

$$\bar{N}_f = \frac{1}{t} \sum_{j=1}^{t} N^f_j$$  \hfill (26)$$

and, as listed in [46], the mean absolute error (MAE)

$$MAE = \frac{1}{t} \sum_{j=1}^{t} \frac{1}{n} \sum_{i=1}^{n} |N^f_{j,i} - \hat{\mu}_{N,j,i}|$$  \hfill (27)$$

For both, the $R^2$ value and the MAE, the difference only to the time to failure $N^f$ is evaluated, since two models, i.e. RNN and SVR, do not provide any prediction for the entire path. Furthermore, due to the fact that all models provide probabilistic predictions, the mean of the log-likelihood $\bar{\mathcal{L}}$ of Equation 5 is evaluated. In Table 6, the values for $R^2$, MAE, and $\bar{\mathcal{L}}$ are compared.

For the confidence intervals, we review how often the actual test measurement values of the time to failure lie within the predicted confidence interval. This can be written

$$\delta_i^j = \begin{cases} 1, & \text{when } \big(\hat{\mu}_{N,f,i} - \gamma \hat{\sigma}_{N,f,i}\big) \leq N^f_j \leq \big(\hat{\mu}_{N,f,i} + \gamma \hat{\sigma}_{N,f,i}\big) \\ 0, & \text{when } \big(\hat{\mu}_{N,f,i} - \gamma \hat{\sigma}_{N,f,i}\big) > N^f_j \lor N^f_j > \big(\hat{\mu}_{N,f,i} + \gamma \hat{\sigma}_{N,f,i}\big) \end{cases}$$  \hfill (28)$$

1this is interpretable even without knowing the range of the values

Table 6
Prediction performance of models.

| Model | $R^2$  | MAE  | $\bar{\mathcal{L}}$ |
|-------|--------|------|---------------------|
| PBM   | 0.863  | 5682 | 3.43                |
| RNN   | 0.962  | 4698 | 2.73                |
| SVR   | 0.884  | 5483 | 2.40                |
| GPR   | 0.963  | 5019 | 2.59                |
| KNN   | 0.596  | 8802 | 1.95                |

Simon Pfingstl et al.: Preprint submitted to Mechanical Systems and Signal Processing
Predicting Crack Growth and Fatigue Life with Surrogate Models

Figure 9: Confidence measures of models. Above dashed line means conservative, below means overly optimistic.

\[ \hat{P} = \frac{1}{l} \sum_{j=1}^{l} \frac{1}{n'} \sum_{i=1}^{n'} \delta_i \]  

(29)

where \( \gamma \) is the confidence level and adjusts the width of the confidence interval. For different \( \gamma \) values, the models are compared in Figure 9, where the black dashed line indicates a perfect model.

7. Discussion

Tables 4 and 5 show the probabilistic prediction of the entire crack growth trajectory. The tables visualize how the models update their prediction by getting new current data. Due to new data, the PBM and the GPR model narrow the confidence interval. In contrast, the KNN model does not do this, since the 14 nearest neighbors have not changed. Moreover, Table 4 shows an offset between the prediction of the KNN model and the last current data point. This is due to the fact that the mean of the nearest neighbors does not necessarily intersect the last current data point. The poor prediction of the KNN model is also visualized in Tables 2 and 3. There, the predictions are illustrated along the number of cycles of the last input. For all models except for KNN, as the test continues, the confidence interval becomes smaller and therefore the prediction becomes more precise. For the PBM, RNN, SVR, and GPR, the course of the mean is similar. The confidence intervals of the SVR model are the broadest, indicating a too-conservative prediction. The PBM exhibits, especially towards the end, the most narrow confidence intervals, which lead to a high mean of the log-likelihood \( \hat{L} \) but also to predictions that are too optimistic.

Mean prediction. By looking at the \( R^2 \) value, it can be seen that all models have a value greater than 0.85 except for KNN. This might be attributed to the limited amount of data. Based on the \( R^2 \) value, the other mathematical surrogate models perform even better than the PBM. The MAE indicates the same behavior, as the errors of the three mathematical surrogate models are smaller than the PBM. The best mean prediction is possible with RNN and GPR based on the MAE and \( R^2 \), respectively.

Estimated confidence interval of models. Based on the mean of the log-likelihood \( \hat{L} \), the worst model is KNN and the best model is RNN, which is close to GPR. By comparing them to the PBM, it can be seen that the PBM performs significantly better than the mathematical surrogate models. Due to the smaller predicted standard deviation especially towards the end, the PBM gains a higher \( \hat{L} \) value. By looking at the confidence intervals, the KNN model is the closest to the black line, indicating that this model maintains the wanted confidence level best. The most distant from the black line and therefore the worst confidence interval predictor is the SVR model, due to the poor error estimation model. Furthermore, the RNN and GPR models are the second closest and therefore predict the confidence intervals quite well. The PBM is the second worst model based on the confidence interval metric. Due to the limited amount of data, all models reach a value of 1 for a wanted confidence level of 99.9%.
Computational cost. The most costly model for predicting the fatigue life is the PBM, where 100 computations are made for every new input data. In this case, a differential equation is used, which has to be solved numerically. However, in a more realistic engineering scenario, where the stress intensity factor cannot be computed by an analytical equation, a finite element analysis has to be conducted, which would increase the computational cost significantly. In contrast, all mathematical surrogate models compute the predictions in a real-time capable manner. However, the GPR and KNN models could have problems for big data applications, as the covariance matrix has to be inverted and all nearest neighbors have to be computed, respectively.

Criticality of model usage. All mathematical surrogate models in Figure 9 lie above the black line, indicating a conservative prediction. This means that these models overestimate their confidence intervals. Therefore, making decisions based on the mathematical surrogate models, e.g. when to conduct an inspection, would lead to a safer structural system. In contrast to this, the PBM lies below the black line and thus underestimates the confidence intervals, which could result in a catastrophic decision.

Generalization of the results. As shown in the paper, the KNN model exhibits weak results. If the nearest lines are found by the KNN model, either the predictions cannot be improved anymore or the prediction will change suddenly. This is due to the limited amount of data. Hence, the KNN model should not be used in this context since usually there are only a few trajectories available. The SVR model performs poorly too. The bad performance results from the strong dependency on the kernel function. Finding an appropriate kernel function is often not possible which leads to a worse performance than other mathematical surrogate models.

In contrast, the RNN model shows one of the best results since it is a highly flexible model which is only constrained by the number of neurons and hidden layers. Furthermore, there is no prior assumption like for SVR on the kernel. The GPR model leads also to good results. This is because it is highly flexible and not limited to certain functions. All information of data set 1 is considered by the GPR model since it is contained in the computed covariance and mean function. Moreover, GPR is able to handle noisy data.

Surprisingly, the PBM does not perform better than the mathematical surrogate models. This can be explained by the fact that the proposed PBM considers all variability as a result of its parameters (here only parameter $C$). In the presence of other uncertainty sources, e.g., the load or measurement noise, the accuracy of the model will decrease. Often it is hard to know all uncertainty sources which is why a PBM might lead to poor results and even to overly optimistic confidence intervals like in this paper.

However, the results are based on only one data set. Therefore, further data sets have to be considered for a common validation of this generalization.

8. Summary

Four mathematical surrogate models and one PBM were applied for predicting the fatigue life with confidence intervals based on sequential data. The paper shows that the best models for this purpose are RNN and GPR. While GPR has the advantage of predicting the entire crack growth trajectory, RNN tends to be faster, in particular for large data sets. Moreover, the PBM, in which the implemented physical equations are widely used in the aerospace industry, exhibits non-conservative behavior. The proposed models enable prognostic health monitoring and can be used to better schedule inspections.

Acknowledgment

This research was conducted as part of the Strubatex research project in the national aeronautical research program LuFo V and was funded by the Federal Ministry for Economic Affairs and Energy based on a decision by the German Bundestag. The authors declare no conflict of interest.
References

[1] M. Gobbato, J. B. Kosmatka, J. P. Conte, A recursive bayesian approach for fatigue damage prognosis: An experimental validation at the reliability component level, Mechanical Systems and Signal Processing 45 (2014) 448–467.

[2] Federal Aviation Administration, Airworthiness directives: the boeing company airplanes: Amendment 39-19755, 84 fr 52754, 03-10-2019.

[3] D. A. Vinkler, B. Hillberry, P. Goel, The statistical nature of fatigue crack propagation, Journal of Engineering Materials and Technology 101 (1979) 148–153.

[4] J. Sikorska, M. Hodkiewicz, L. Ma, Prognostic modelling options for remaining useful life estimation by industry, Mechanical Systems and Signal Processing 25 (2011) 1803–1836.

[5] N. Gugulothu, V. TV, P. Malhotra, L. Vig, P. Agarwal, G. Shroff, Predicting remaining useful life using time series embeddings based on recurrent neural networks, arXiv preprint arXiv:1709.01073 (2017).

[6] D. T. Do, J. Lee, H. Nguyen-Xuan, Fast evaluation of crack growth path using time series forecasting, Engineering Fracture Mechanics 218 (2019) 106567.

[7] C. M. Bishop, Mixture density networks (1994).

[8] M. MA, Z. MAO, Rotating machinery prognostics via the fusion of particle filter and deep learning, Structural Health Monitoring 2019 (2019).

[9] S. Mohanty, A. Chattopadhyay, P. Peralta, S. Das, Bayesian statistic based multivariate gaussian process approach for offline/online fatigue crack growth prediction, Experimental Mechanics 51 (2011) 833–843.

[10] D. Kwon, M. H. Azarian, M. Pecht, Remaining-life prediction of solder joints using rf impedance analysis and gaussian process regression, IEEE Transactions on Components, Packaging and Manufacturing Technology 5 (2015) 1602–1609.

[11] S. Hong, Z. Zhou, Remaining useful life prognosis of bearing based on gauss process regression, in: 2012 5th International Conference on BioMedical Engineering and Informatics, IEEE, 2012, pp. 1575–1579.

[12] D. Liu, J. Pang, J. Zhou, Y. Peng, M. Pecht, Prognostics for state of health estimation of lithium-ion batteries based on combination gaussian process functional regression, Microelectronics Reliability 53 (2013) 832–839.

[13] S. A. Dudani, The distance-weighted k-nearest-neighbor rule, IEEE Transactions on Systems, Man, and Cybernetics (1976) 325–327.

[14] J. Tian, M. H. Azarian, M. Pecht, Anomaly detection using self-organizing maps-based k-nearest neighbor algorithm, in: Proceedings of the European Conference of the Prognostics and Health Management Society, Citeseer, 2014.

[15] D. Wang, K-nearest neighbors based methods for identification of different gear crack levels under different motor speeds and loads: Revisited, Mechanical Systems and Signal Processing 70 (2016) 201–208.

[16] R. Khelif, B. Chebel-Morello, S. Malinowski, E. Laajili, F. Fnaiech, N. Zerhouni, Direct remaining useful life estimation based on support vector regression, IEEE Transactions on Industrial Electronics 64 (2016) 2276–2285.

[17] W. Song, Z. Jiang, H. Jiang, Predict the fatigue life of crack based on extended finite element method and svr, in: AIP Conference Proceedings, volume 1967, AIP Publishing LLC, 2018, p. 030024.

[18] H. Sohn, C. R. Farrar, Damage diagnosis using time series analysis of vibration signals, Smart Materials and Structures 10 (2001) 446.

[19] S. Pfingstl, M. Zimmermann, Strain-based structural health monitoring: Computing regions for critical crack detection, Structural Health Monitoring 2019 (2019).

[20] S. Pfingstl, M. Steiner, O. Tusch, M. Zimmermann, Crack detection zones: Computation and validation, Sensors 20 (2020) 2568.

[21] L. J. Antolick, J. S. Branning, D. R. Wade, P. J. Dempsey, Evaluation of gear condition indicator performance on rotocraft fleet (2010).

[22] A. Ginart, I. Barlas, J. Goldin, J. L. Dorrity, Automated feature selection for embeddable prognostic and health monitoring (phm) architectures, in: 2006 IEEE Autotestcon, IEEE, 2006, pp. 195–201.

[23] V. Nguyen, M. Kefalas, K. Yang, A. Apostolidis, M. Olhofer, S. Limmer, T. Bäck, A review: Prognostics and health management in automotive and aerospace, International Journal of Prognostics and Health Management 10 (2019).

[24] S. Zhong, Z. Li, L. Lin, Y. Zhang, Aero-engine exhaust gas temperature prognostic model based on gated recurrent unit network, in: 2018 IEEE International Conference on Prognostics and Health Management (ICPHM), IEEE, 2018, pp. 1–5.

[25] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, R. X. Gao, Deep learning and its applications to machine health monitoring, Mechanical Systems and Signal Processing 115 (2019) 213–237.

[26] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (1997) 1735–1780.

[27] F. A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: Continual prediction with lstm (1999).

[28] F. A. Gers, N. N. Schraudolph, J. Schmidhuber, F. Cummins, Learning precise timing with lstm recurrent networks, arXiv preprint arXiv:1709.01073 (2017).

[29] S. Hong, Z. Zhou, Remaining useful life prognosis of bearing based on gauss process regression, in: AIP Conference Proceedings, volume 1967, AIP Publishing LLC, 2018, p. 030024.

[30] H. Sohn, C. R. Farrar, Damage diagnosis using time series analysis of vibration signals, Smart Materials and Structures 10 (2001) 446.

[31] S. Pfingstl, M. Zimmermann, Strain-based structural health monitoring: Computing regions for critical crack detection, Structural Health Monitoring 2019 (2019).

[32] S. Pfingstl, M. Steiner, O. Tusch, M. Zimmermann, Crack detection zones: Computation and validation, Sensors 20 (2020) 2568.

[33] L. J. Antolick, J. S. Branning, D. R. Wade, P. J. Dempsey, Evaluation of gear condition indicator performance on rotocraft fleet (2010).

[34] A. Ginart, I. Barlas, J. Goldin, J. L. Dorrity, Automated feature selection for embeddable prognostic and health monitoring (phm) architectures, in: 2006 IEEE Autotestcon, IEEE, 2006, pp. 195–201.

[35] V. Nguyen, M. Kefalas, K. Yang, A. Apostolidis, M. Olhofer, S. Limmer, T. Bäck, A review: Prognostics and health management in automotive and aerospace, International Journal of Prognostics and Health Management 10 (2019).

[36] S. Zhong, Z. Li, L. Lin, Y. Zhang, Aero-engine exhaust gas temperature prognostic model based on gated recurrent unit network, in: 2018 IEEE International Conference on Prognostics and Health Management (ICPHM), IEEE, 2018, pp. 1–5.

[37] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, R. X. Gao, Deep learning and its applications to machine health monitoring, Mechanical Systems and Signal Processing 115 (2019) 213–237.

[38] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (1997) 1735–1780.

[39] F. A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: Continual prediction with lstm (1999).

[40] F. A. Gers, N. N. Schraudolph, J. Schmidhuber, Learning precise timing with lstm recurrent networks, Journal of Machine Learning Research 3 (2002) 115–143.

[41] G. Cavallini, R. Lazzeri, A probabilistic approach to fatigue design of aerospace components by using the risk assessment evaluation, INTECH Open Access Publisher, 2012.

[42] W. Wang, W. Hu, N. Armstrong, Fatigue crack prognosis using bayesian probabilistic modelling, Mechanical Engineering Journal 4 (2017) 16–00702.

[43] E. B. Baum, F. Wilczek, Supervised learning of probability distributions by neural networks, in: Neural Information Processing Systems, 1988, pp. 52–61.

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

[45] C. E. Rasmussen, C. K. I. Williams, Gaussian processes for machine learning, Adaptive Computation and Machine Learning, 3. print ed., MIT Press, Cambridge, Mass., 2008.

[46] N. S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, The American Statistician 46 (1992) 175–185.

[47] V. Vapnik, Pattern recognition using generalized portrait method, 1963.

[48] V. Vapnik, A. Chervonenkis, A note on one class of perceptrons, Automation and Remote Control 25 (1964).

[49] V. Vapnik, A. Chervonenkis, Theory of pattern recognition, 1974.

[50] A. J. Smola, B. Schölkopf, A tutorial on support vector regression, Statistics and Computing 14 (2004) 199–222.
[39] Quicit, A simple technique to estimate prediction intervals for any regression model, 2018. URL: https://medium.com/žquicit/a-simple-technique-to-estimate-prediction-intervals-for-any-regression-model-2dd73f630bcb.

[40] D. Basak, S. Pal, D. C. Patranabis, Support vector regression, Neural Information Processing-Letters and Reviews 11 (2007) 203–224.

[41] P. Paris, F. Erdogan, A critical analysis of crack propagation laws (1963).

[42] A. Carpinteri, M. Paggi. Are the parisâ¬ž law parameters dependent on each other?. Frattura ed Integrità Strutturale 1 (2007) 10–16.

[43] A. International, Test method for measurement of fatigue crack growth rates, ASTM STP 738 (1981) 340–356.

[44] B. Spencer, J. Tang, M. Artley, Stochastic approach to modeling fatigue crack growth, AIAA Journal 27 (1989) 1628–1635.

[45] M. Liao, J. P. Komorowski, Corrosion risk assessment of aircraft structures, Journal of ASTM International 1 (2004) 1–16.

[46] E. Ramasso, A. Saxena, Performance benchmarking and analysis of prognostic methods for cmapss datasets, International Journal of Prognostics and Health Management 5 (2014) 1–15.