Reliability Engineering and System Safety 184 (2019) 86–100

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

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

Noise-dependent ranking of prognostics algorithms based on discrepancy without true damage information

Yiwei Wang [a], Christian Gogu [a], Nam H. Kim [b,*], Raphael T. Haftka [b], Nicolas Binaud [a], Christian Bes [a]

[a] Université de Toulouse, Institut Clément Ader (UMR CNRS 5312) INSA/UPS/ISAE/Mines Albi, Toulouse 31400, France
[b] Mechanical & Aerospace Engineering, University of Florida, Gainesville, FL 32611, USA

A B S T R A C T

In this paper, an interesting observation on the noise-dependent performance of prognostics algorithms is presented. A method of evaluating the accuracy of prognostics algorithms without having the true degradation model is proposed. This paper compares the four most widely used model-based prognostics algorithms, i.e., Bayesian method, particle filter, Extended Kalman filter, and nonlinear least squares, to illustrate the effect of random noise in data on the performance of prediction. The mean squared error (MSE) that measures the difference between the true damage size and the predicted one is used to rank the four algorithms for each dataset. We found that the randomness in the noise leads to a very different ranking of the algorithms for different datasets, even though they are all from the same damage model. In particular, even for the algorithm that has the best performance on average, poor results can be obtained for some datasets. In absence of true damage information, we propose another metric, mean squared discrepancy (MSD), which measures the difference between the prediction and the data. A correlation study between MSE and MSD indicates that MSD can be used to estimate the ranking of the four prognostics algorithms without having the true damage information. Moreover, the best algorithm selected by MSD has a high probability of also having the smallest prediction error when used for predicting beyond the last measurement. MSD can thus be particularly useful for selecting the best algorithm for predicting into the near future for a given set of measurements.

1. Introduction

Model-based prognostics approaches can provide a better performance than data-driven approaches when a degradation model is available [1,2]. The most widely used model-based prognostics methods in the literature include Extended Kalman filter [3-4], particle filter [5,6], Bayesian method [7,8] and nonlinear least squares [9]. Some review articles [10,11] introduce and evaluate these algorithms (or part of them) in terms of a general descriptive explanation of their respective advantages and disadvantages. Some publications quantitatively compare different model-based prognostic methods through specific examples [1,9,12]. However, we have not found studies of the effect of randomness in the measurement data on the ranking of algorithms, which is the objective of the present paper since measurement noise is one of the most important uncertainty sources.

One of the major challenges in prognostics and health management (PHM) is dealing with prediction uncertainty. Long-term prediction of remaining useful life (RUL) or probability of failure in a certain time horizon increases the bound of prediction uncertainty due to various sources, such as measurement noise, future load, and usage uncertainty, model assumptions and inaccuracies, etc. [13]. An important issue encountered in making meaningful predictions is to treat these uncertainties properly as they directly affect the prognostics results, thus affecting the associated decision-making process [14]. Uncertainty leads to significant deviation of prognostics results from the actual situation. For example, in the application of fatigue crack growth, the Paris model is often used. Among the two Paris model parameters, the exponent, m, of aluminum alloy is known to be in the range of 3.6 and 4.2. This corresponds to 16% variability. But the life cycle can differ by 500% [15]. Sankararaman discussed a series of issues regarding uncertainties in PHM including the causes of uncertainty, the ways of how to interpret uncertainty, how to facilitate effective treatment of uncertainty and how to accurately quantify the uncertainty [16].

Measurement noise is one of the most significant uncertainty sources, which must be represented and managed properly. In model-based prognostics techniques, the estimation of model parameters depends on the measurement data. Using a different set of data will result in different estimations of model parameters, i.e., the uncertainty in measurement data propagates into the uncertainty in model parameters, which significantly affects the performance of the prognostics. In addition, even small errors at the initial state caused by measurement noise can accumulate and grow over time, and consequently, severely distort the predicted probability distribution over a long time period. It is necessary to account for measurement uncertainty right from the initial stages of

* Corresponding author.
E-mail address: nkim@ufl.edu (N.H. Kim).

https://doi.org/10.1016/j.ress.2017.09.021
Available online 7 October 2017
0951-8320/

system-level design. Therefore, well representing, propagating, quantifying and managing measurement uncertainty is very important. In this research, we seek to compare the most commonly used model-based prognostics techniques among themselves for their suitability in various datasets.

An observation from this study is that the performance of the algorithms depends on the particular realization of random noise. Therefore, there may not be a significant meaning to rank the generic performance of algorithms. In this sense, it would be more desirable to use the best algorithm for the given noisy data. The selection of the best algorithm is challenging when the true damage information is not available. Most degradation metrics in the literature are based on the knowledge of true degradation information [17]. Therefore, they are useful for evaluating the generic performance of an algorithm. A useful conclusion from this study is that the mean squared discrepancy is a good representative of the mean squared error. The former one can be used instead of the latter in the absence of true degradation. In addition, we observe that the best algorithm selected based on the past measurement data is highly likely to be among the best for future prediction. Therefore, the proposed method of selecting the best algorithm for the specific noisy data set can be practical for future prognostics.

Existing metrics such as the prognostic horizon, $\alpha - \lambda$ accuracy, (cumulative) relative accuracy and convergence have been proposed for offline assessment of the performance of a prognostic model from accuracy, precision or robustness aspects. Typically, computation of these metrics requires the "true" degradation information, e.g., the true End of Life (EOL), the true remaining useful life (RUL), the availability of historical run-to-fail data, etc. However, in practice, the entire degradation process from the beginning of operation until the equipment failure may not yet be observed. Therefore, this makes it challenging to assess the performance of prognostic methods and difficult for users to suitably choose the appropriate method for their problems. The main objective of this paper is to propose a method for online assessment of the performance of model-based prognostics algorithms based on past degradation measurement data. Hu et al. [18] proposed a metric that enables assessing the performance of a model-based prognostic approach from past measurement data in three cases characterized by different levels of knowledge on the degradation process. In [18], the PF is chosen as the prognostics method to test the proposed metric.

In this paper, we focus on four of the most commonly used model-based algorithms, Bayesian method, particle filter, Non-linear least squares and Extended Kalman filter, and verify their performance through a simple degradation model with multiple simulated measurement data sets. Random multiple datasets are generated using the same noise level. This makes sense since for a particular engineering application (e.g., the sensors are embedded into the aircraft structures such as fuselage panels, wings, bulkheads to monitor the fatigue cracks), once the sensor is installed, the level of measurement uncertainty is deterministic since it mainly stems from the sensor limitations, which is an intrinsic attribute of the sensor.

The conventional metric, the mean squared error (MSE) measuring the difference between predicted and the true crack size, is firstly utilized to rank the four algorithms in terms of accuracy assuming the true information on crack growth is available. We examine how much the ranking changes from one dataset to another due to randomness in the noise. We assume that difference in performance from one dataset to another is caused by specific realizations of the noise, which may be friendly to one algorithm and unfriendly to another. Then a new metric based on measurement data, called mean squared discrepancy (MSD), which measures the difference between predicted crack sizes and measured data, is proposed to be a performance indicator in the absence of true crack size. This metric is used to rank the four methods when different datasets are employed. Based on our numerical tests, it shows that even using a simple degradation model, when dealing with multiple measurement data sets among which the inner difference only stems from random noise with even the same noise level, the performance of one algorithm varies from one data set to another. No method can perform consistently well and always be the best for handling all datasets. The ranking based on the mean squared error (MSE) can be mostly preserved when the ranking based on the mean squared discrepancy (MSD) is used. The former requires the true model while the latter does not. This indicates that MSD can be considered to rank the algorithms when the true crack size is not available.

The paper is organized as follows. Section 2 briefly reviews the basic concepts and some key issues of the four algorithms. The degradation model used for testing the algorithms is also presented in this section. Two forms of the model, recursive and non-recursive, are given for adapting the characteristics of different algorithms. Section 3 states the strategy for implementing the performance comparison and the metrics for performance evaluation. A numerical study is implemented in Section 4 where the four algorithms are tested based on 100 datasets with randomly generated noise. The proposed metric is used to rank the prognostic performance of the four algorithms. Section 5 summarizes and concludes the paper.

2. Model-based prognostic and four most commonly used algorithms

In a model-based prognostics method, it is assumed that the damage state propagates over time, which is an unobserved process governed by some physical damage model, and the damage related quantities are measured at successive time-points. The physical damage model describing the degradation process is assumed to be known. However, the model parameters are generally unknown and need to be identified along with the damage state from noisy measurements. This process usually resorts to estimation algorithms. Once the model parameters are identified, they are substituted into the damage model to predict future behavior of damage and thereby to obtain prognostics information such as the evolution of damage distribution or the distribution of remaining useful life. This process can be implemented either through Monte Carlo sampling method or derived in an analytical form [19,20].

In this paper, we investigate four of the most commonly used model-based prognostics algorithms: Bayesian method (BM), particle filter (PF), Extended Kalman filter (EKF) and nonlinear least square (NLS). We illustrate their performance through a simple degradation model with multiple measurement datasets, which differ in random noise with the same noise level.

The first three methods are based on Bayesian inference, which uses measurements to update the prior knowledge on unobserved damage as well as model parameters, while the last one is a conventional regression method. The estimation methods can also be classified in terms of estimation criteria. The Bayesian estimate criterion includes maximum a posteriori and minimum mean squared error criterion while the non-Bayesian criterion involves the least squares criterion and maximum likelihood criterion. BM and PF employ the maximum a posteriori criterion while EKF is a minimum mean squared error estimator. The advantages of these four algorithms over point estimation methods such as maximum likelihood estimate lie in their capability of estimating the uncertainty at the same time while giving the expected value of the identified parameters.

The following subsections rapidly present some basic concepts and key issues of the four algorithms followed by the damage model used in this paper. The details of the four algorithms are elaborated in Appendix A, for interested readers.

2.1. Bayesian method (BM)

Among different ways of applying Bayesian inference, the Bayesian method here processes all measurement data simultaneously. The joint posterior distribution of parameters at current time step is expressed as a single equation, in which all the likelihood function prior to current

time are multiplied together along with the prior distribution of parameters. Once the expression of the posterior distribution is obtained, a sampling method can be used to draw samples from this distribution. The Markov Chain Monte Carlo (MCMC) method [21] is usually used. It starts from an arbitrary initial sample (old sample) and a new sample is drawn from a proposal distribution centered at the old sample. The two samples are compared with each other based on an acceptance criterion to decide either the new sample is accepted or the old one is reselected. This process is repeated as many times as necessary until a sufficient number of samples are obtained. The MCMC sampling results are affected by the proposal distribution and the initial sample. The more similar the proposal distribution is to the posterior distribution, the better the samples characterize the target distribution. Also, if the location of the initial sample is far from the mean of the posterior distribution, then many iterations (samples) would be required to converge to the posterior distribution.

2.2. Particle filter (PF)

Particle filter-based prognostics approaches have been applied in many engineering problems such as fatigue crack growth [5,22], Li-ion batteries [23–25], PEM fuel cells [26,27], and machine tools [6]. PF is a numerical solution of Bayesian inference with the advantage for dealing with the nonlinear non-Gaussian systems. The key idea is to represent the required joint posterior distribution of state and parameters by a set of random particles (or samples) and their weights. Note that technically, the parameters are seen as additional state variables and appended onto the true state vector to form an augmented state-parameter vector so that particles of state-parameter are processed by PF simultaneously. As a new measurement arrives, the weight of each particle is adjusted by comparing the particles and the newly arrived measurement, i.e., the particles having a higher similarity with the measurement will be assigned a higher weight. PF is composed of two steps at each iteration process: (1) prediction - the particles at the previous time step propagate through the damage model to form particles at the current time, which is seen as the prior distribution at the current step (2) update – adjust each particle's weight according to how close the particle is to the current measurement, which is quantified by the likelihood function. After many iterations of prediction-update steps, only a few particles will have a non-negligible weight, known as the degeneracy phenomenon, which implies that a large computational effort is devoted to updating particles whose contribution to the approximated posterior distribution is negligible. The degeneracy problem typically resorts to resampling methods, whose basic idea is to duplicate the high weighted particles while eliminating the low ones. Conventional resampling algorithms include multinomial, residual, stratified and systematic resampling. Guo et al. [28] compares PF prognostics results with the above four resampling approaches and concludes that systematic and stratified resampling show the best results with a slight advantage to systemic resampling which is proven theoretically superior. Some novel resampling methods involve support vector regression-based resampling [29] and monotonic resampling [30]. Note that resampling is not necessary to be implemented at each iteration since it can easily cause the problem of particle impoverishment, i.e., loss of diversity among the particles. This arises due to the fact that in the resampling stage, samples are drawn from a discrete distribution rather than a continuous one. The particles with a high weight are duplicated too often while the ones with a small weight are discarded. After several iterations, the posterior distribution is approximated by only a few high-weight particles. If this problem is not addressed properly, it may lead to "particle collapse", a severe case of sample impoverishment that all particles occupy the same point, giving a poor representation of the posterior distribution. Many efforts have been made to address the particle impoverishment [31,32], according to which, possible ways to address impoverishment include kernel smoothing method, MCMC move method, and regularized particle filter.

2.3. Extended Kalman filter (EKF)

EKF assumes the state-parameter distribution at each time step to be a multivariate normal distribution, which is characterized by its mean vector and covariance matrix, and enables to recursively compute the mean and covariance from measurements. Therefore, EKF can be regarded as the explicit solution for recursive Bayesian inference given the Gaussian assumption. To deal with the nonlinear system, EKF does the linearity approximation that the damage model is expanded as the first-order Taylor series around the prior mean of the state variable. By ignoring the higher order terms, the state prediction propagates through the nonlinear system equation whilst the state error covariance propagates through a separate first-order linearization of the nonlinear system [33]. Similar to PF, EKF is also composed of two steps: prediction and update. However, instead of processing a large collection of particles, EKF only needs to calculate the mean and covariance matrix, leading to a significant computation cost saving. Therefore, despite Gaussian assumption and linear approximations, from a practical application perspective, EKF has been proven to be a powerful tool and has been successfully applied in various engineering state-parameter identification problems [34–36]. In terms of state-parameter estimation, EKF is typically implemented through joint filtering [37] that defines the parameter vector of interest as an additional state variable and appends it onto the true state vector. The appended portion of augmented state vector does not change beyond the effects of process noise during the time-update process while the augmented error covariance matrix is propagated as a whole (i.e., the parameters inherently do not depend on time evolution and remain constant).

2.4. Nonlinear least squares (NLS)

The nonlinear least squares method is the form of least squares analysis used to fit a set of observations with a model that is nonlinear with unknown parameters. In prognostics context, when the damage model is a nonlinear combination of unknown model parameters, NLS can be used to estimate the parameters such that the damage model which fits best the given measurement data in the least squares sense, that is, the sum of the squares of the residuals, defined as the difference between the measurement data and the model prediction, is minimized. This process can be implemented using an iterative process based on optimization techniques such as Levenberg-Marquardt algorithm that combines the gradient descent and the Gauss-Newton method. NLS proceeds the measurement data in a batch way, i.e., all measurement data up to the current time are used to identify the model parameters. In prognosis, the estimated parameters depend on noisy measurement data, and the optimized parameters can be changed when a different set of measurement are used, meaning that the uncertainty in measurements is propagated into the uncertainty in parameters. Assuming Gaussian white noise, NLS is able to yield the uncertainty in the estimated parameters in the form of covariance. Once the model parameters, as well as the covariance, are estimated, the damage growth behavior can be predicted based on the samples of estimated parameters.

2.5. Crack growth model

The Paris model is adopted here as the crack growth model. The simple Paris model is used to illustrate the effect of noise. We will show that even with the same model, the randomness in the noise leads to a different ranking of the algorithms. We expect that if that happens for a simple model, there is a reasonable chance that noise will affect even more strongly on a complex model that requires estimation of a larger number of parameters. The Paris model is given in the form of crack growth rate as

$$\frac{da}{dk} = C(\Delta\sigma\sqrt{\pi a})^m \tag{1}$$

where a is the half crack size, k the loading cycles, $\Delta\sigma$ the range of stress loading, and m and C the model parameters governing the behavior of crack growth, which has epistemic uncertainty due to the lack of knowledge and needs to be identified through measurements on crack size a.

Eq. (1) can be rewritten in the recursive or non-recursive form where the crack size a is a function of the number of loading cycles k, and parameters m and C. The non-recursive form can be obtained by integrating the differential equation of Eq. (1) and solving for a. The analytical expression of crack size a after k cycles is given as

$$a_k = \left[kC\left(1-\frac{m}{2}\right)\left(\Delta\sigma\sqrt{\pi}\right)^m + a_0^{1-\frac{m}{2}}\right]^{\frac{2}{2-m}} \tag{2}$$

where a_0 is the initial crack size.

Alternatively, for some algorithms such as PF and EKF, the crack size at the current cycle depends on the previous step that needs to be computed progressively over time. In such cases, it is more desirable to use the recursive form of Paris' model. Euler method is employed here to discretize Eq. (2). The discrete Paris model is written in a recursive form at each cycle k as:

$$a_k = a_{k-\Delta k} + C\left(\Delta\sigma\sqrt{\pi a_{k-\Delta k}}\right)^m \Delta k \tag{3}$$

The two forms (recursive and non-recursive) should be equivalent theoretically but might be different in numerical implementation. Specifically, the prediction accuracy of the recursive form depends on the size of discrete time step; the smaller the step is the more precise the crack size would be. When the discrete step is small enough, the two forms should yield the identical results. Here we set the step $\Delta k = 1$, which is a minimal possible value from the physical and practical point of view, to minimize the discrete error. However, the measurement data do not need to be collected at every time step. Therefore, the time step for integrating the recursive damage model is different from measurement time step. The two parameters, m and C, are regarded as the model parameters in this example.

As discussed before, in BM, at time step k, all the measurements up to time step k are used simultaneously to construct the posterior distribution of the parameters. Similarly, in NLS, the measurements proceed in a batch way. Therefore, in these two methods, the non-recursive form of Paris model is used. While in PF and EKF, the joint posterior distribution of crack size and parameter are updated progressively over time, therefore, the recursive form is used.

The techniques of applying the four algorithms on this specific application to implement prognostics are detailed in Appendix A rather than here since (1) it is difficult to elaborate the prognostics process with only a few basic concepts and notations, and (2) it might be disruptive to our main contribution of showing the effect of random noise on ranking algorithms.

3. Strategy for comparison and metrics for performance

When multiple predictions are available from different algorithms, it is important for the users to evaluate their performance. Some works compared the performance of several model-based prognostics algorithms using one specific data set and concluded that one method outperforms others, trying to provide users the guidance of selecting the best method. However, comparing multiple algorithms with only one data set may lead to a wrong conclusion since the conclusion may not hold when different datasets are used. In fact, as illustrated in this paper, the performance of an algorithm varies with different datasets with the same level of noise. Due to the randomness in data, there is no one method that performs consistently well for all data sets.

In this paper, we assess the performances of algorithms based on multiple randomly simulated datasets. The datasets are generated with the same noise level. This makes sense, since for a particular engineering application (e.g., the sensors are embedded into the aircraft structures such as fuselage panels, wings, bulkheads to monitor the fatigue cracks), once the sensor is installed, the level of measurement uncertainty is deterministic since it mainly stems from the sensor limitations, which is an intrinsic attribute of the sensor.

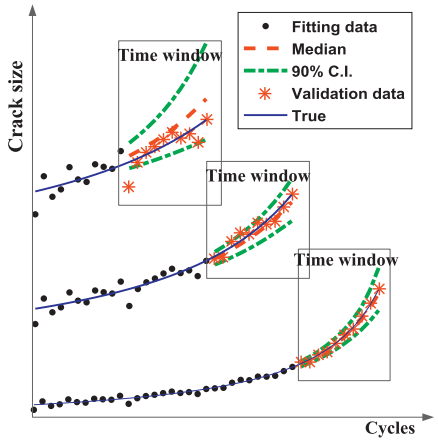

Fig. 1. Schematic illustration of moving time window strategy.

One common way of dealing with randomness in the data is to assess the average performance over a large number of realizations and find out which algorithm performs best on average. While we perform such comparison here, we are also interested in gauging the effect of randomness on the ranking for individual datasets. We will show that this effect is large so that selecting an algorithm based on average performance may lead to poor performance.

In this section, the strategy for implementing performance comparison taking into account the randomness in data is introduced firstly, followed by the metrics used for performance evaluation.

3.1. Strategy for implementing performance comparison

We use a moving time window as an experiment strategy, as shown in Fig. 1, to examine how well a given algorithm predicts future crack propagation with increasing number of measurement data. Suppose that the data are collected every ΔT cycles and currently N_m measurement data are available. These N_m data are divided into two sequential parts. The first N_f data, called fitting data, are used for estimating the distribution of model parameters. The rest N_v data, called validation data, are used for verifying the performance of the algorithms. Note that $N_m = N_f + N_v$. The time range during which the rest N_v data are collected is referred to as a time window.

In Fig. 1, the solid blue curve represents the true crack size, (denoted by a in the following text and equations), red dash line the median crack size predicted by the algorithm (denoted by $\hat{a}$) based on the first N_f data points, black solid dots the fitting data, and red asterisks the validation data (denoted by a_v). Three different time windows for three different datasets are shown. It should be noted that the true crack size is a simulated one under the assumption that the true model parameters for crack growth are known. We simulated the measured crack size by adding random noise to the true crack size. Once the measured crack sizes at different cycles are generated, neither the true parameters nor the true crack sizes are used in the estimation process. For a given current cycle, the simulated measured crack sizes up to the current cycle are used for estimating model parameters, while the data beyond the current cycle are used for the purpose of validation.

In the fitting strategy, we provide each algorithm with the first N_f data for estimating the model parameters. Then based on the estimated parameters, we predict the distribution of crack growth trajectory within the time window, i.e., from cycle $(N_f+1)\Delta T$ to $(N_f+N_v)\Delta T$. Since the model parameters are estimated in the form of a probability distribution, Monte Carlo simulation is utilized to obtain the distribution of crack trajectory, from which the median and 90% confidence intervals can be

estimated. By gradually increasing the number of fitting data, multiple time windows can be obtained to predict the performance at different stages of crack growth.

For the purpose of verifying the prediction performance, it is good to compare the true crack size with the predicted ones. However, in practice, the true crack size is unknown. A straightforward way is to compare the predicted crack size with data. In the following sections, we will show the feasibility of using the difference between validation data and predicted crack size to evaluate the prediction accuracy of the algorithms in the absence of true crack size.

3.2. Dealing with randomness in data

The strategy in Section 3.1 is developed for one algorithm using one data set. We randomly generate N_d datasets with the same noise level to assess the algorithms' performance in different realizations of noise. These N_d datasets act as a database shared by all algorithms. For each dataset, the moving time window strategy is used, i.e., the previous N_m data are used to identify the parameters while the following N_v data are used for validation. There could be multiple time windows when using one data set to test one algorithm.

3.3. Metrics for performance evaluation

3.3.1. Mean squared error (MSE)

The mean squared error (MSE) integrated over the time window is used to measure the accuracy of an algorithm. MSE is defined as

$$MSE = \frac{1}{N_v}\sum_{i=1}^{N_v}(a_{(N_f+i)\Delta T} - \hat{a}_{(N_f+i)\Delta T})^2 \tag{4}$$

where a is the true crack size, $\hat{a}$ the predicted crack size, and subscript the time step (refer to Fig. 1 for illustration). Notice that MSE can only be calculated when the true crack size is available. Recall that there are multiple time windows for testing an algorithm. MSE is calculated every time when we test one algorithm by one data set in the one-time window. Therefore, for each algorithm, there are N_d MSEs in one time window.

3.3.2. Mean squared discrepancy (MSD)

In practice, the MSE is unavailable since the true crack size cannot be available. We attempt to use another metric to assess the performance of an algorithm. A straightforward way is to compare the predictions with data. The difference between prediction and data is referred to as *discrepancy*. We consider the mean squared discrepancy (MSD) as a possible candidate for performance metric, as

$$MSD = \frac{1}{N_v}\sum_{i=1}^{N_v}(a_{v,(N_f+i)\Delta T} - \hat{a}_{(N_f+i)\Delta T})^2 \tag{5}$$

where a_v is the validation data, and $\hat{a}$ the predicted crack size by the algorithm. In the numerical case study, we will show the feasibility of using MSD as the performance indicator to rank the algorithms for an individual data set. Similar to MSE, MSD is calculated every time when test one algorithm by one data set in the one-time window, thus adding up to N_d MSDs for one method in one time window.

3.3.3. Evaluation index taking into account the confidence interval of the prediction

For the decision makers, in addition to using MSE to evaluate the accuracy of the algorithms, it is also important to know how well one can trust the prediction. A good prognostics not only provides accurate and precise prediction but also specifies the level of confidence associated with such predictions. In this sense, the confidence interval (C.I.) is considered as an important factor to assess the prognostics quality from a conservative decision-making point of view. To this end, we develop a synthetic index comprising of two component indexes to evaluate the performance of the algorithms taking into account the C.I. of prediction.

The first component E_1 measures the relative width of the 90% C.I. with respect to the true crack size for each prediction point and then averages over the validation domain, as given in Eq. (6), where $\hat{a}^u$ and $\hat{a}^l$ are the upper and lower bounds of the 90% C.I.. The true crack size in the denominator acts as a normalizing constant to facilitate the comparison among the algorithms since the true crack size is independent of the prediction results given by different algorithms. A smaller E_1 indicates a narrower 90% C.I., thus a smaller prediction variation.

$$E_1 = \frac{1}{N_v}\sum_{i=1}^{N_v}\frac{\hat{a}^u_{(N_f+i)\Delta T} - \hat{a}^l_{(N_f+i)\Delta T})}{a_{(N_f+i)\Delta T}} \tag{6}$$

The second component E_2 tells whether the C.I. covers the true crack size. In one time window, instead of using a fixed C.I. for all N_v prediction points, we calculate how wide the C.I. needs to be at the minimum to cover the true crack size at each prediction point and accordingly assign that point a different mark. E_2 is then taken as the average of these N_v marks. Specifically, at time step $k+i$, $i=1,2,\ldots, N_v$, the minimal α% C.I. such that this α% C.I. can cover the true crack size is calculated. A smaller α indicates a more reliable prediction, thus a higher mark should be assigned. A series of discrete values increased from 90 to 99 with one increment are taken for α and the mark is computed from a mapping defined as $m = -0.01\alpha + 1$. For example, to the prediction points from step $k+1$ to $k+10$, if the smallest C.I. needed to cover the true crack size are successively 95%, 94%, 92%, 91% and 90% for the remaining six, then the marks 0.5, 0.6, 0.8, 0.9 and 1 for the remaining six are assigned to these ten prediction points and $E_2 = 0.88$. A larger E_2 means a more reliable prediction, thus a better performance of the algorithm.

The above two indexes evaluate the prognostics quality and dispersion in different ways. It would be beneficial to combine them together to give a more comprehensive index. A linear combination of the two indices is calculated as a synthetic index, denoted as EI. Based on their importance, a weight of −0.7, 0.3 are assigned to E_1 and E_2, respectively, and accordingly, $EI = -0.7E_1 + 0.3E_2$. Note that the weight for E_1 is negative since for E_1, the smaller the better. In this sense, the higher EI is, the better the prediction would be. Note that we use the coefficient of −0.7 and 0.3 for illustrative purpose and the users may assign different values based on their experience or preference. Based on the strategy in Section 3.1, EI is calculated every time when we test one algorithm by one data set in the one-time window. Therefore, for each algorithm, there are N_d EIs in one time window.

The calculation of E_1 and E_2 depends on the true crack size, which is not available in practice. Therefore, instead of using the true crack size, we recalculate the indices based on the validation data, denoted as $\hat{E}_1$ and $\hat{E}_2$ for the components and $\hat{EI}$ for synthetic index. In this sense, $\hat{E}_1$ calculates the relative width of the 90% C.I. with respect to the data for each prediction point. $\hat{E}_2$ measures how wide a C.I. is at least needed to cover the data. In the following study, we will show the feasibility of using $\hat{EI}$ as the indicator to rank the algorithms in the absence of the true crack size.

4. Numerical case study

In this section, we investigate the four algorithms by assessing their prognostics performance to $N_d = 100$ randomly generated measurement data sets. All datasets are generated using the same crack growth model, but different because of the randomly generated noise. The strategy for implementing performance comparison and the metrics for performance evaluation presented in Section 3 are employed. For each dataset, we test the prognostics behavior of each algorithm and rank them in terms of metrics.

The crack size is assumed initially to grow from $a_0 = 8$ mm. The true values of Paris' model parameters are assumed as $m_{true} = 3.8$ and $C_{true} = 1.5e{-}10$. In Paris' law, the exponent, m, is the slope of the fatigue crack curve in log-log scale (i.e., $\log(da/dN)$ vs $\log(\Delta K)$), while the parameter C corresponds to y-intercept at $\Delta K = 1$ of the fatigue curve. The values of m and C are normally calculated by fitting fatigue test data

under controlled laboratory environment. Here the fatigue test data refer to the data point of crack growth rate da/dN versus the stress intensity factor ΔK. Articles studying the relation between m and C for different materials are provided in [38–40]. In this paper, $m_{true} = 3.8$ and $C_{true} = 1.5e{-}10$ are taken based on references [8,12], which corresponds to the mean crack growth of Aluminum panel. The measurement noise is Gaussian white noise with mean zeros and standard deviation $\sigma = 0.8$ mm, which is equivalent to 10% coefficient of variation (COV) with respect to the initial crack size of 8 mm (over the 30 measurements during 3000 cycles considered in this paper, the crack grows up to 23 mm. The true crack size at different cycles is given in Appendix B for reference). The 100 random measurement datasets are generated by (1) using a_0, m_{true}, C_{true} to compute the true crack size according to the Paris model for given time steps, and (2) adding the measurement noise $V \sim N(0, \sigma^2)$ to the true crack size. The measurements are collected every $\Delta T = 100$ cycles until 4000 cycles, adding up to 40 measurement data. The above steps are repeated 100 times to obtain 100 measurement data sets, each containing 40 data. These datasets act as the database used by all algorithms.

Based on the strategy for performance comparison in Section 3.1, here $N_f = 10$, 20, and 30 (hence 3 time windows) are used to test the prediction performance in different crack propagation stages: from the slightly nonlinear case in earlier stage to the relatively strong nonlinear case in a later stage. N_v is fixed to 10. For example, when $N_f = 10$, the measurements collected from 100th cycle to 1000th cycle (the first 10 measurements) are used to predict the crack propagation trajectory from 1100th cycle to 2000th cycle; when $N_f = 20$, the measurements collected from 100th cycle to 2000th cycle (the first 20 measurements) are used to predict the trajectory from 2100th to 3000th cycle; and when $N_f = 30$, measurements collected from 100th cycle to 3000th cycle (the first 30 measurements) are used to predict the trajectory from 3100th to 4000th cycle.

Here the minimal number of data points we used for inferring the model parameters is 10. For the Bayesian inference-based algorithms, i.e., EKF, PF, and BM, they process the data either sequentially (EKF and PF) or in a batch way (BM). Theoretically, they don't have limitations on the number of data needed for inferring the parameters. But the number of data indeed affects the uncertainties in the estimated parameters. For the regression-based algorithm, NLS, at least three data are needed since the number of parameters to be estimated is two (m and C). Therefore, we can consider that at least three data are needed in order to make all algorithm work. However, too few data will result in a large uncertainty in the estimated parameters, consequently, leading to a large uncertainty in prediction. In such case, the prediction might not be very informative as the predicted results could fall in a very wide range, which is less meaningful in practice. Therefore, based on our experience and literature [1,12], we use at least 10 data to perform the parameter estimation.

4.1. Ranking of prognostics algorithms is sensitive to noise in data

We first show that even for our simple degradation model, when dealing with multiple measurement datasets from different realizations of random noise, the performance of an algorithm varies from one data set to another, and none of the methods performs best for all datasets. The accuracy metric MSE is used to assess the performance of the algorithms. Later, we will show how to determine the ranking of algorithms in absence of MSE, which is meaningful to help decision makers to choose the proper method when facing a particular data set.

The MSE is displayed in two ways. Table 1 compares the average MSE over 100 datasets for the three time windows (i.e.,$N_f = 10$, 20, and 30, respectively), while Fig. 2 illustrates the empirical cumulative distribution function (CDF) of the MSE using 100 datasets for each method. For BM, PF and EKF, the empirical CDF curves in the three time windows are plotted in the same figure, while in NLS, the MSEs in different time windows are not of the same magnitude. The curves are separately plotted otherwise the figure becomes hardly readable.

Table 1
Average MSE (in m^2) over 100 datasets with different numbers of measurement data.

Methods	$N_m = 20$ ($N_f = 10$)	$N_m = 30$ ($N_f = 20$)	$N_m = 40$ ($N_f = 30$)
BM	1.20e−6	6.54e−7	3.02e−6
PF	1.22e−6	7.85e−7	4.52e−6
NLS	0.27	0.034	1.04e−4
EKF	2.91e−6	2.54e−6	5.48e−5

Table 2
Statistics of MSE (Means Square Error) rank.

$N_m = 20$ ($N_f = 10$)		$N_m = 30$ ($N_f = 20$)		$N_m = 40$ ($N_f = 30$)	
Cases	Times	Cases	Times	Cases	Times
EBPN	6	ENBP	1	EPBN	1
NEPB	1	EPBN	2	EBPN	3
NEBP	1	EBPN	4	NPBE	2
NPBE	5	EBNP	1	NBPE	5
NBPE	4	NEPB	1	PNEB	2
PNBE	2	NPBE	3	PNBE	3
PBEN	39	NBPE	5	PEBN	2
PBNE	10	NBEP	5	PBEN	9
BPNE	10	PNBE	1	PBNE	26
BPEN	20	PEBN	5	BNPE	8
BEPN	2	PBEN	21	BPNE	25
		PBNE	7	BPEN	14
		BNPE	4		
		BNEP	1		
		BPNE	9		
		BPEN	27		
		BEPN	3		

Table 1 shows that in terms of average performance, BM and PF outperform the other two while NLS yields the largest MSE, especially in the earlier stage when very few data are available. For BM, PF and EKF, the MSE is not monotonically reduced as more measurements are used but tends to be large at the steep section of the crack growth curve. This indicates the prediction error increases when the crack grows fast. For NLS, the MSE in the $N_f = 10$ and $N_f = 20$ are large. The reason is that MSE of NLS method has some abnormally large values. These outliers of large MSE contribute to the average value, which makes the average MSE much greater than other three algorithms. Fig. 2 tells that BM and PF show very similar behavior. This is expected since they are both based on Bayesian inference. NLS performs poorly when very few measurement data are used, implying that NLS is not able to infer the model parameters very well in the case of a small amount of data without prior information. Bad estimates to parameters lead to very uncertain crack size prediction. When more data are given, the MSE of NLS method reduces significantly. The reason for poor performance of NLS in this case is likely that no prior information is used under strongly correlated parameters. However, the effect of noise is large enough so that for some realizations of noise, NLS can outperform other methods.

Next, we rank the four algorithms in terms of their MSE for each dataset. To present the results we use letters B, P, N, and E to index the methods BM, PF, NLS, and EKF, respectively. There are 24 possible permutations of rank, and the number of times each permutation appears is presented in Table 2 for $N_f = 10$, 20, and 30. For example, for $N_f = 10$, 11 permutations out of the 24 occur. Among these, the ranking PBEN, which indicates PF > BM > EKF > NLS, occurs the most frequently. It is interesting to note that even NLS, the worst performer on average, outperforms the others for 11 out of the 100 datasets.

The above discussion illustrates that even with a simple crack growth model, the rank of the methods varies from dataset to dataset, even if the difference among data sets comes only from random noise with the same noise level. In addition to the rank, random noise can also lead to a large difference in algorithm performance, that is to say, the best algorithm on

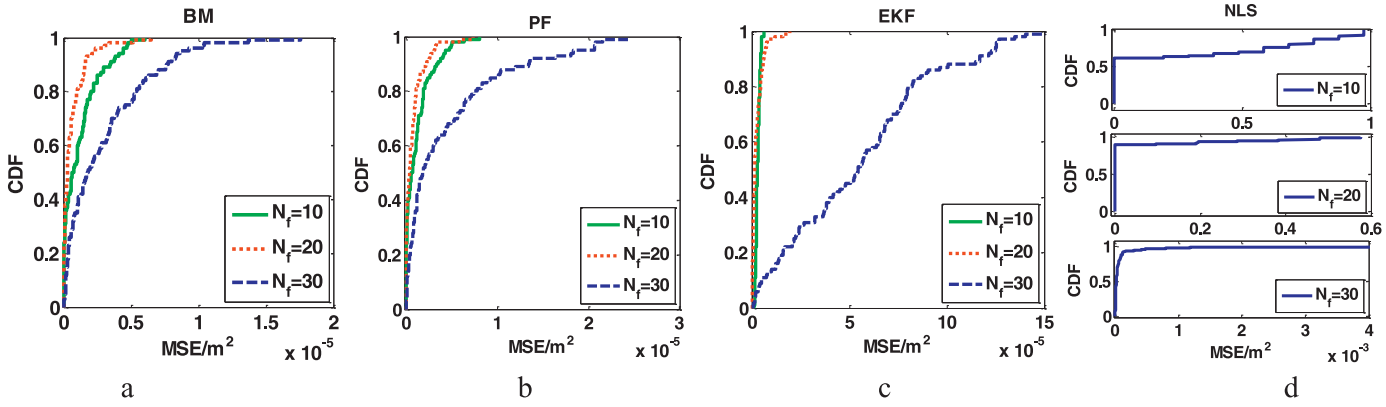

Fig. 2. CDF of MSE of different methods. (a) BM (b) PF (c) EKF (d) NLS.

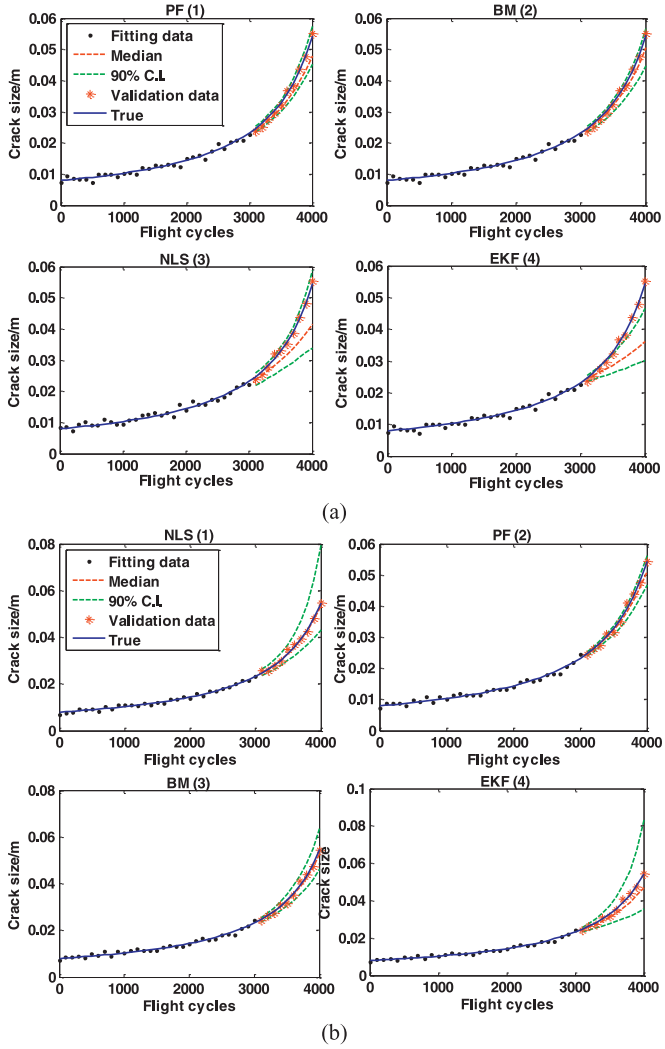

Fig. 3. An example to show the rank of the four algorithms tested by two datasets. (a) tested on dataset #4 (b) tested on data set #14.

average can have a very poor accuracy for some datasets. Specifically, from Table 2, we see that PF is the best algorithm on average; i.e., PF is the best 127 times out of 300 (51, 34, and 42 times in $N_f = 10$, 20, and 30 cases, respectively). However, Table 2 also shows that for nine cases out of 300 datasets PF is the worst algorithm.

Fig. 3 gives an example showing the prediction performance for two different datasets (dataset #4 and #14) where the four algorithms rank differently. In this example, 30 data are used for fitting and 10 data are used for validation. The numbers in parentheses are their MSE-rank. It is worth noting that while the measurements for these two data sets look similar the difference in terms of the prediction for the four algorithms can be quite significant, thus highlighting the importance to correctly determine the best performing algorithm for a given data set.

In summary, the performance of prognostics algorithms strongly depends on a specific dataset. Therefore, it may not have much sense to say one algorithm is better than the other. It would be better to choose the best algorithm based on a given specific dataset. We also see that the number of measurements can make a big difference in the ranking. For example, PF is the best 51% of the time for $N_f = 10$, 34% of the time for $N_f = 20$, and 42% for $N_f = 30$. When a large number of data are available, we would expect all algorithms to do well. The problem is with relatively small number of data, in which case even basic quantities like the level of noise in the data is not well estimated. The different algorithms filter noise in a different manner so that the departure of the sample from the ideal distribution misleads them differently. Our suggestion of how to accommodate this is to use all of the algorithms rather than select one in advance. The proposed MSD measure can then allow choosing the best performing algorithm for the given measurements.

4.2. Ranking the algorithms in absence of true damage information

4.2.1. MSD can rank the algorithms in terms of accuracy

In practice, the MSE is unavailable since the true crack sizes are unknown. We attempt to use another metric to assess the performance of algorithms. A straightforward way of evaluating the performance is to compare the predictions with validation data, which is the MSD presented in Section 3.3. To verify our hypothesis, we consider the correlation between MSD (Eq. (5)) and MSE (Eq. (4)). The correlation is presented in Fig. 4, which includes a scatter plot of MSE and MSD as well as the correlation coefficient (abbreviated as "cc" in Fig. 4). We can see that in general, MSD is highly correlated with MSE for all algorithms in all time-window scenarios, meaning that MSD could be considered as a performance indicator to assess the algorithm performance.

When MSE is unknown, we naturally ask whether MSD can be used to rank the algorithms in terms of accuracy. This can be achieved by studying how consistent the rank based on MSE is with that based on MSD. Specifically, for each dataset, we rank the four algorithms based on their MSE. This is the *actual rank* in terms of accuracy. Then we re-rank the algorithms based on their MSD, which is referred to as *predicted rank*, and compare these two ranks to see to what extent they match.

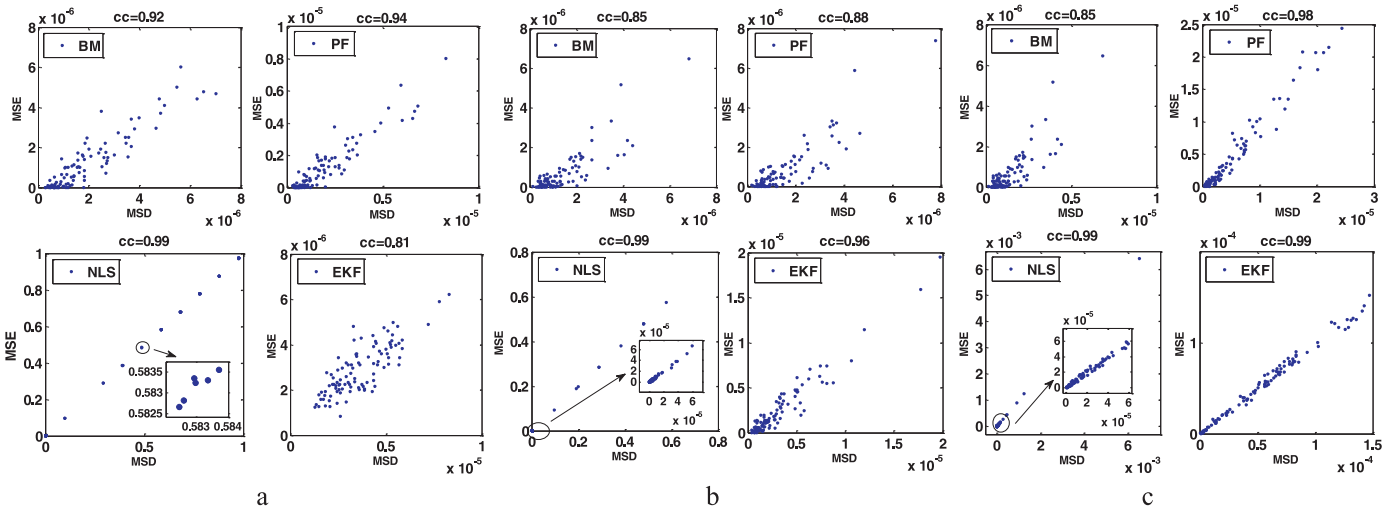

Fig. 4. Correlation between MSE and MSD for each method.
(a) $N_m = 20$ ($N_f = 10$) (b) $N_m = 30$ ($N_f = 20$) (c) $N_m = 40$ ($N_f = 30$).

Table 3
Statistics of match extent of MSE rank and MSD rank.

$N_m = 20$ ($N_f = 10$)		$N_m = 30$ ($N_f = 20$)		$N_m = 40$ ($N_f = 30$)	
D_R	times	D_R	times	D_R	times
0	81	0	65	0	86
3	1	3	4	3	1
5	2	5	6	7	11
7	11	7	13	9	1
15	1	10	1	15	1
21	4	15	2		
		21	6		
		25	3		

The standard approach for comparing ranks would be to use the Spearman correlation coefficient r_s. For four ranks, r_s is given as

$$r_s = 1 - 0.1\sum_{i=1}^{4} d_i^2 \tag{7}$$

where d_i is the discrepancy in rank at the i-th position (we make the MSE ranking always 1234 rather than assign a number to an algorithm). For example, if MSE rank is NBPE and MSD rank is BNPE, then the two ranks are 1234 and 2134 respectively, and accordingly, $d_1 = -1$, $d_2 = 1$, $d_3 = d_4 = 0$. However, this correlation coefficient does not distinguish between switching #1 and #2, versus switching #3 and #4. We opt instead for a weighted measure D_R of the discrepancy in ranking, which assigns a weight of four to a discrepancy in the first place and one to a discrepancy in the last place. The weight is introduced to take into account the importance of mistaking different positions. We assume that the importance of mismatching the ith position diminishes with increasing i. A higher weight is assigned to a smaller ith position for accounting this importance. Therefore, the smaller D_R is, the better the two ranks match each other. The calculation of D_R is given in Eq. (8). For example, if the first two places are permuted (2134) then $D_R = 7$, if the last two places are permuted (1243) $D_R = 3$, and if the order is reversed (4321) $D_R = 50$.

$$D_R = \sum_{i=1}^{4} (5 - i)d_i^2 \tag{8}$$

The statistics of the D_R for the 100 datasets are given in Table 3 where $D_R = 3$ corresponds to switching 3rd and 4th positions, $D_R = 5$ to switching 2nd and 3rd, and $D_R = 7$ to switching 1st and 2nd. It is seen from the table that out of all 300 datasets, 232 have a perfect agreement

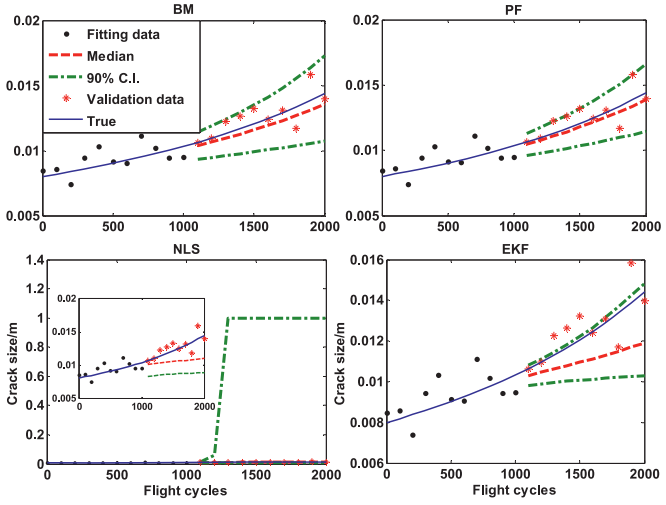

Fig. 5. Performance comparison, tested on data set #4.

in rank and 49 others have a single adjacent permutation so that only 19 have a substantial difference in rank.

We further probe the correlation between MSD and MSE with different magnitude of the noise level. We test other two noise levels, i.e., $\sigma = 0.3$ mm and $\sigma = 2.85$ mm, which are equivalent to 3.75% and 35% coefficient of variation (COV) with respect to the initial crack size (8 mm). In fact, given initial crack size 8 mm, 35% is almost the largest COV we could try, because for larger COVs negative values of crack length appear. The correlation between MSE and MSD as well as the statistics of match extent of MSE rank and MSD rank in those two noise levels are reported in Appendix C. It shows that the correlation between MSE and MSD in both two noise levels are high, except the case of EKF with 35% COV noise level in the case of $N_f = 10$, which is relatively low (0.64). In addition, in both two noise levels, the match extent between MSE rank and MSD rank are satisfactory. The results indicate that the MSD rank is tolerant to a relatively large measurement noise.

An example is given here to show the process discussed above. The comparison of the performances of the four algorithms for data set #4 is elaborated. Fig. 5 illustrates the prognostics behavior when $N_f = 10$. It is seen that BM and PF are comparable. NLS performs worst while EKF is between the best two methods and the worst one. Table 4 compares the methods quantitatively by presenting their MSE and MSD. We can see

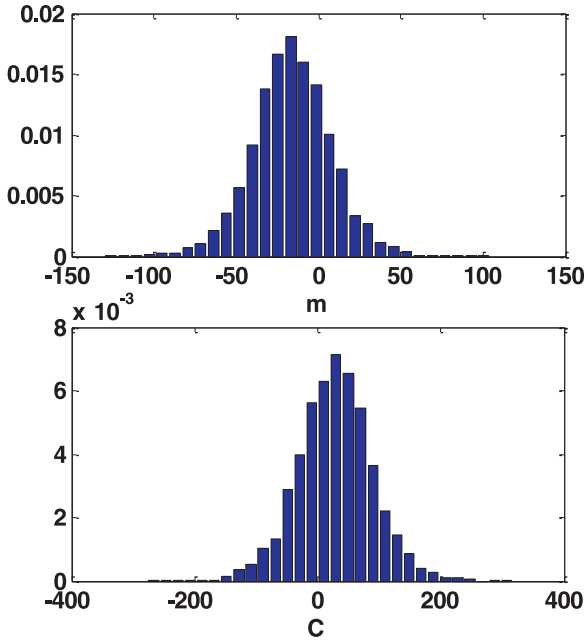

Fig. 6. Histogram of estimated parameters by NLS, $N_f = 10$, tested on dataset #4.

Table 4
Quantitative comparison of four algorithms in the case of $N_f = 10$, tested by dataset #4.

Method	MSE/m^2	MSD/m^2
BM	1.47e−7	1.32e−6
PF	1.42e−7	1.31e−6
NLS	4.05e−6	5.90e−6
EK	2.19e−6	3.79e−6

that the actual MSE rank of the four methods is PBEN and the predicted MSD rank matches the MSE rank perfectly.

Since the prediction using NLS in Fig. 5 is quite different from that of others, it would be beneficial to study the NLS further. Since the upper bound of 90% C.I. rises sharply (Note that we set an upper limit of one meter for the predicted crack size to prevent positive infinite numbers in the numerical simulation), it is difficult to see the median of the prediction. The subplot embedded in the original figure is shown after deleting the upper bound, from which we see that the predicted median can hardly trace the true crack size. The inferior performance of NLS is due to a bad estimation of model parameters when only a few data are available. Fig. 6 shows the histograms of the model parameters estimated by NLS when $N_f = 10$. The parameters are poorly identified with

Table 5
Correlation study between EI and $\hat{EI}$.

$N_m = 20$ ($N_f = 10$)				$N_m = 30$ ($N_f = 20$)				$N_m = 40$ ($N_f = 30$)			
BM	PF	NLS	EKF	BM	PF	NLS	EKF	BM	PF	NLS	EKF
0.77	0.76	0.99	0.84	0.81	0.85	0.99	0.98	0.84	0.92	0.99	0.96

Table 6
Statistics of match extent of EI rank and $\hat{EI}$ rank.

$N_m = 20$ ($N_f = 10$)		$N_m = 30$ ($N_f = 20$)		$N_m = 40$ ($N_f = 30$)	
D_R	times	D_R	times	D_R	times
0	70	0	76	0	77
3	20	3	21	3	19
5	4	5	1	5	1
9	4	7	2	7	2
15	2			15	1

a wide range (remind that $m_{true} = 3.8$ and $C_{true} = \ln(1.5e{-}10)$), leading very uncertain prediction for crack size growth trajectories. The estimation difficulty can be mitigated as more data are available, as shown in Fig. 7.

4.2.2. $\hat{EI}$ can rank the algorithms taking into account the confidence interval of prediction

We introduce the synthetic index in Section 3.3.3 to account for the confidence interval of prediction. It is noted that the calculation of EI depends on true crack size while that of $\hat{EI}$ depends on validation data. We study the correlation between EI and $\hat{EI}$ for each algorithm. The results are reported in Table 5. We see that EI highly correlates with $\hat{EI}$ in different scenarios, indicating that $\hat{EI}$ can be used to evaluate the performance taking into account the confidence interval when the true crack sizes are absence.

Naturally, we want to know whether $\hat{EI}$ can be used for ranking the algorithms since EI is not available in practice. This is achieved by studying how well the EI-rank matches $\hat{EI}$-rank. Specifically, for each dataset, we rank the four algorithms based on their EI and on $\hat{EI}$ and then compare these two ranks to see to what extent they match through calculating the weighted measure D_R. The results are reported in Table 6. We see from the table that out of all 300 datasets, 223 have a perfect agreement in ranking and 70 others have a single adjacent permutation (corresponding to the cases $D_R = 3$, 5, and 7). Only 7 have a substantial difference in ranking.

In addition, we find that if we seek to predict a conservative estimate of the crack growth, the most important aspect affecting the quality of this prediction is the algorithms rather than the dataset used. We obtain always the similar ranking with either PF or BM the best independently of the dataset (The EI-rank is either PF > BM > NLS > EKF

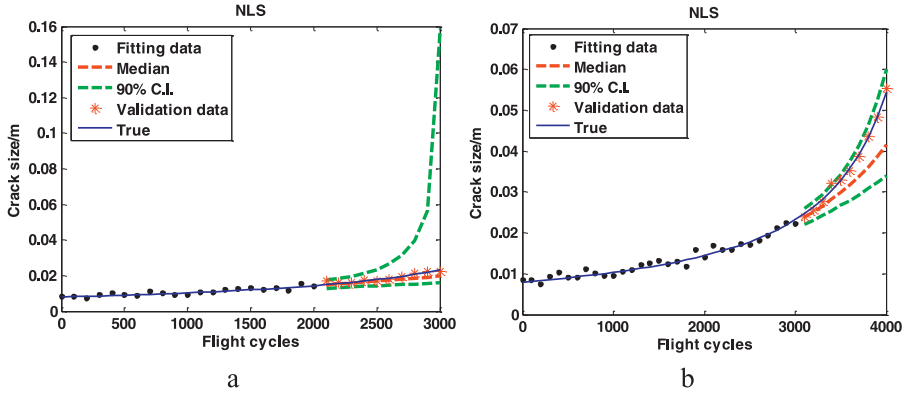

Fig. 7. Prediction behavior of NLS tested on dataset #4. (a) $N_f = 20$ (b) $N_f = 30$.

or BM > PF > NLS > EKF). This makes sense in reality. There are cases where an accurate estimate is more useful than a conservative estimate and there are cases the other way around. Take the predictive maintenance of an aircraft fleet in an airline as an example [41], if the users are more concerned about how many aircraft in a fleet will need maintenance, the data decide which algorithm to use. In this case, MSD can be used to choose the best algorithm in terms of accuracy for a given data set. If the users need to make a decision on one particular aircraft which is close to critical size, then conservativeness is more important, in which case it is best to pick the algorithm involving the smallest prediction uncertainty, which we found is mostly independent of the dataset.

4.3. MSD can improve the success rate of selecting the best algorithm for predicting the future

From the decision makers' point of view, instead of the ranking of the algorithms, it may be more desirable to know the best method to predict the future damage growth for a given data set. The numerical example in Table 2 shows that PF is the best 127 times out of 300 (51, 34, and 42 times in 10, 20, and 30 measurements case respectively). However, in spite of its overall good performance, the rate of being best can be as low as around 42% (127/300). That is to say, if the users always select the PF as the single algorithm for all datasets, more than 50% chance they cannot select the best algorithm.

For a given set of measurements, the users may be concerned with (1) how to select the best algorithm based on the available data, and (2) how well the selected algorithm performs when used for predicting into the future (i.e. beyond the last measurement). The first concern has been addressed in Section 4.2. To investigate the second one, we predict a certain number of cycles further from the last measurement, and calculate the relative error with respect to the true crack size in the future cycle to see if the "best algorithm given by MSE" (refer to "MSE-best" hereafter) also leads to the lowest error when predicting into the future where no measurements are available. Specifically, when $N_m = 20$, the first 10 data (collected from 100th cycle to 1000th cycle) are used for fitting ($N_f = 10$) and the rest 10 data (collected from 1100th to 2000th) are used for validation ($N_v = 10$). MSE and MSD are calculated in the validation domain. Then we predict 1000 cycles further from the last measurement (corresponding to 2000th cycle) up to 3000th cycle and calculate the relative error to the true crack size of each algorithm at the 3000th cycle to see whether the MSE-best remains the best (i.e. has the lowest error with respect to the true crack size). Moreover, given MSE-best remains the best, we also calculate how often MSD-best (refer to the best algorithm given by MSD) is consistent with the MSE-best (since in practice MSE is unknown). We also investigate how many MSD-best is inconsistent with the MSE-best for current. For $N_m = 30$, we predict 1000 cycles further from the last measurement up to 4000th cycle while for $N_m = 40$, we predict 500 cycles further up to 4500th cycle. In this last case, we limit the prediction to 500 additional cycles due to the exponential growth of the crack, which leads to crack sizes exceeding the critical values after 1000 more cycles (refer to Appendix B).

To facilitate the description, we define the following events. A = {MSE-best remains the best in the future}, B = {MSD-best is consistent with MSE-best}. B|A = {B occurs given A occurs}. Accordingly, $\bar{B}$ is the complementary event of B. Then for $N_m = 10$, 20, and 30, we investigate the times of occurrence of event A and event B|A for all the 100 datasets. The results are reported in Table 7. We see that when $N_m = 20$, the event that MSE-best remains the best at future prediction occurs in 73 out of 100 datasets, and 70 out of these 73 datasets MSD succeeds in selecting the same best algorithm as MSE does. For $N_m = 30$, the percentage is a bit lower but still better than the selecting PF as the single algorithm (which is 42%). For $N_m = 40$, the rate is more satisfactory. We take the most frequent MSE ranking when $N_m = 40$, PBNE, (refer to Table 2) as an example and show how to count the frequency of A and B|A in Appendix D.

Table 7
Times of occurrence of A and B|A, A = {MSE-best remains the best in the near future}, B = {MSD-best is consistent with MSE-best}.

	Times of occurrence of A	Times of occurrence of B|A	Times of occurrence of $\bar{B}$
$N_m = 20$ ($N_f = 10$)	87	78	16
$N_m = 30$ ($N_f = 20$)	66	56	25
$N_m = 40$ ($N_f = 30$)	73	70	12

Table 8
Times of occurrence of A* and B|A*, A* = {MSE-best remains the best in the near future or is the second best given that the difference of relative errors between the second best the best is less than 5%}.

	Times of occurrence of A*	Times of occurrence of B|A*	Times of occurrence of $\bar{B}$
$N_m = 20$ ($N_f = 10$)	95	80	16
$N_m = 30$ ($N_f = 20$)	84	66	25
$N_m = 40$ ($N_f = 30$)	87	82	12

In addition, we found that for some datasets, the MSE-best switch to the second best in the future but the difference between MSE-best and the future-best in terms of relative error is small. For example, in Appendix D, for data set #58, the MSE-best, the particle filter, switches to the second best at the 4500th cycle. However, the difference between PF and the best one at 4500th cycle in terms of relative error is very small (less than 1%). In such cases, we consider that MSE-best performs well in the future. To take into account such cases, we define the event A*. A* = {MSE-best remains the best at future prediction or is the second best given that the difference of relative errors between the second best and the best is less than 5%}. Under this definition, the number of times of occurrence of event A* and event B|A* among all 100 datasets are reported in Table 8. We see that the percentage that MSD selects a good algorithm for future prediction is improved. The results thus indicate that the selected algorithm based on MSD has a high probability of remaining good performance for predicting the near future.

5. Conclusions

In this paper, the four most commonly used algorithms, Bayesian method, particle filter, nonlinear least squares, and Extended Kalman filter, are applied on a simple crack growth model with simulated random measurement noise. We investigate their performance statistically by testing their performance using 100 randomly generated measurement datasets with the same noise level. The mean squared error (MSE) is used as a metric to rank the four algorithms in terms of accuracy for each data set. It is found that the performance of prognostics algorithms strongly depends on the realization of random noise, and none of the algorithms can be the best on all realizations. It was found that on average the two algorithms based on Bayesian inference substantially outperformed the other two. However, the statistics of MSE rank showed that the performance of the algorithms varies substantially from one dataset to another even if the data are generated with the same noise level. As a result, for some data sets the worst on-average algorithm can substantially outperform the best one. Since the exact solution is not available in practice, the discrepancy between predictions and measurements (MSD) has to stand for the actual error (MSE). We found a very good correlation between MSE and MSD and that ranking the algorithms based on discrepancy with measurements is a good stand in for their true rank in terms of accuracy. In addition, the best algorithm selected by MSD has a high probability of maintaining the good performance in the near future. This is particularly useful for users to choose a good algorithm for their specific case.

Appendix A. Apply the four algorithms on crack growth model

The objective is to employ the four model-based algorithms to obtain the posterior distribution of parameters given noisy measurements collected up to current time step k, denoted as $y_{1:k}$. As detailed in Section 4, the measurement noise is assumed as white Gaussian noise, in which the measurements at any pair of times are identically distributed and statistically independent. The theories of the four algorithms will not be elaborated here. Readers refer to [42] for BM, [43] for PF, [44] for EKF and [45] for NLS. Only their implementations on Paris model are presented here. Bayesian method (BM) and Nonlinear least squares (NLS) employ the non-recursive Paris model (see Section 2.5 for details) to obtain the posterior distribution as a single expression. Particle filter (PF) and Extended Kalman filter use the recursive Paris model and get the posterior distribution recursively. In BM and PF, the posterior distribution of parameter at time step j, ($j = 1,2,\ldots,k$) is presented in the forms of samples. These samples are propagated through Paris model to predict the crack growth trajectories from time k. In EKF and NLS, due to the normality assumption, the posterior distribution of parameter at time step j is multivariate normal distribution and t-distribution characterized by mean and covariance matrix. Samples are drawn from these distributions and used to predict the crack growth trajectories from time k.

First of all, the likelihood function used in BM and PF are given. The lognormal distribution is employed such that

$$p(y_j|a_j^i(m_j^i, C_j^i)) = \frac{1}{y_j\sqrt{2\pi}\zeta_j^i}\exp\left(-\frac{1}{2}\left(\frac{\ln y_j - \eta_j^i}{\zeta_j^i}\right)^2\right), i = 1, 2, \ldots, n_s \quad (9)$$

in which

$$\zeta_j^i = \sqrt{\ln\left(1 + \left(\frac{\sigma}{a_j^i(m_j^i, C_j^i)}\right)^2\right)}$$

$$\eta_j^i = \ln\left(a_j^i(m_j^i, C_j^i)\right) - \frac{1}{2}\left(\zeta_j^i\right)^2$$

In Eq. (9), n_s is the number of samples, here $n_s = 5000$ for all four methods, $a_j^i(m_j^i, C_j^i)$ is the i-th sample at time step j. The parentheses indicate that the crack size is a function of parameters m and C. Eq. (9) can be understood as following. When at a specific time step j that a measurement y_j is given, the likelihood is the function of crack size $a_j{}^i$; that is, the function of $m_j{}^i$ and $C_j{}^i$. The computation of likelihood at time j depends on the samples of the parameters $m_j{}^i$ and $C_j{}^i$, and the data y_j. In state-parameter estimation context, the parameters are uncertain. In this case, the explanation of likelihood can be expressed as the probability to obtain the measured y_j for a given parameter value $m_j{}^i$ and $C_j{}^i$. The idea of parameters identification is to select the $m_j{}^i$ and $C_j{}^i$ that make $a_j{}^i$ closest to the measurement y_j.

A.1. Bayesian method (BM)

Eq. (9) gives the way of calculating the likelihood value for a given measurement and a given sample. When multiple data are available, the posterior distribution of parameters at the current step k is obtained by a single equation, in which all the likelihood functions of measured data up to the current time step are multiplied together. Specifically, given data $y_{1:k} = \{y_1, y_2, \ldots, y_k\}$, the joint posterior distribution of parameters m and C at current time step k is calculated by

$$p_{m_k,C_k|Y_{1:k}} = \frac{1}{\sqrt{2\pi}\sigma_m}\exp\left(-\frac{[m-\mu_m]^2}{2{\sigma_m}^2}\right)\times\frac{1}{\sqrt{2\pi}\sigma_C}\exp\left(-\frac{[C-\mu_C]^2}{2{\sigma_C}^2}\right) \times\prod_{j=1}^{k}\frac{1}{y_j\sqrt{2\pi}\zeta_j^i}\exp\left(-\frac{1}{2}\left(\frac{\ln y_j - \eta_j}{\zeta_j}\right)^2\right) \quad (10)$$

which is the product of prior PDF function of m and C (the first two terms) times the likelihood functions of all measurements up to k (the third term). The prior distribution of the parameters m and C in BM are assumed normally distributed, with mean and standard deviation equal to $\mu_m = 4$, $\mu_C = \ln(1.0\text{e}{-10})$, and $\sigma_m = 0.2, \sigma_C = 1.1$. Once the expression of joint posterior distribution is obtained, the MCMC sampling method can be used to generate as many samples as needed. Tutorial on MCMC sampling can be found in [21]. Given measurements $\{y_{1:k}\}$, the process of sampling parameters from the joint posterior distribution, that is Eq. (10), is described as follows.

Step 1 Choose an initial value for the damage model parameters, denoted as $\boldsymbol{\theta}^0 = [m^0, C^0]$. In this application, the initial values are $m^0 = 4$, $C^0 = \ln(1.0\text{e}{-10})$. Then the crack size from time step 1 to k given $\boldsymbol{\theta}^0$ can be calculated through Paris' model, i.e.,$[a_{1:k}(m^0, C^0)] = [a_1(m^0, C^0), a_2(m^0, C^0), \ldots, a_k(m^0, C^0)]$. The joint posterior PDF value given $\boldsymbol{\theta}^0$ at time k is accordingly obtained through Eq. (10), denoted as $p(\boldsymbol{\theta}^0|Y_{1:k})$.

Step 2 For $i = 1, 2, \ldots, n_s$,

(a) Generate a new sample $\boldsymbol{\theta}^* = [m^*, C^*]$ from a proposal distribution whose probability density function is denoted as T. and accordingly calculate the crack size up to step k given $\boldsymbol{\theta}^*$, i.e., $[a_{1:k}(m^*, C^*)] = [a_1(m^*, C^*), a_2(m^*, C^*), \ldots, a_k(m^*, C^*)]$. Then calculate the joint posterior PDF of m and C at time step k given the new parameters $\boldsymbol{\theta}^*$ through Eq. (10), denoted as $p(\boldsymbol{\theta}^*|Y_{1:k})$.

(b) Accept the new sample $\boldsymbol{\theta}^*$ as the next sample, i.e., $\boldsymbol{\theta}^i = \boldsymbol{\theta}^*$ if

$$u \le \min\left\{1, \frac{p(\boldsymbol{\theta}^*|Y_{1:k})T(\boldsymbol{\theta}^{i-1}|\boldsymbol{\theta}^*)}{p(\boldsymbol{\theta}^{i-1}|Y_{1:k})T(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{i-1})}\right\}$$

otherwise, reselected the old sample $\boldsymbol{\theta}^{i-1}$ as the next sample, i.e., $\boldsymbol{\theta}^i = \boldsymbol{\theta}^{i-1}$.

(c) Set $i = i + 1$, go back to step (a) until $i = n_s$.

In this application, the proposal distribution is chosen as the uniform distribution which is symmetric. Therefore $T(\boldsymbol{\theta}^{i-1}|\boldsymbol{\theta}^*) = T(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{i-1})$ and the acceptance criterion reduces to $u \le \min\{1, \frac{p(\boldsymbol{\theta}^*|Y_{1:k})}{p(\boldsymbol{\theta}^{i-1}|Y_{1:k})}\}$.

A.2. Particle filter (PF)

There are many versions of PF [42]. The technique we used here includes three steps at each iteration, prediction, update, and resampling. The implementation of PF on Paris model is described as follows:

Step 1 Initialization

The initial distribution of the parameters m and C in PF are assumed normally distributed, i.e., $m_0 \sim N(\mu_m, \sigma_m), C_0 \sim N(\mu_C, \sigma_C)$, with $\mu_m = 4, \mu_C = \ln(1.0\text{e}{-10}), \sigma_m = 0.2$, and $\sigma_C = 1.1$. Draw n_s particles of m, C from their initial distribution, denoted as $[m_0^i, C_0^i], i = 1, 2, \ldots, n_s$.

Step 2 For $j = 1, 2, \ldots, k$,

Prediction. In this step, the crack size at previous time j-1 is propagated to the current time j based on Paris' model. For this, samples of the prior distribution need to be generated. First, n_s samples of $[m_j^i, C_j^i]$ are propagated from previous step j-1. It should be noticed that model parameters inherently do not depend on time evolution. Therefore, n_s samples of $[m_j^i, C_j^i]$ become the same as the previous particles $[m_{j-1}^i, C_{j-1}^i]$. Next, n_s samples of the previous crack size a_{j-1}^i and model parameters $[m_{j-1}^i, C_{j-1}^i]$ are used in the Paris model to propagate n_s samples of a_j^i. In summary, the crack size and parameters propagate through Eq. (11).

$$\begin{bmatrix} a_j^i \\ m_j^i \\ C_j^i \end{bmatrix} = \begin{bmatrix} a_{j-1}^i + C_{j-1}^i\left(\Delta\sigma\sqrt{\pi a_{j-1}^i}\right)^{m_{j-1}^i} \\ m_{j-1}^i \\ C_{j-1}^i \end{bmatrix} \quad (11)$$

Update. Calculate the weight of each particle at time step j. The weight is defined by normalizing the likelihood, as given in Eq. (12)

$$we_j^i = \frac{p(y_j|a_j^i(m_j^i, C_j^i))}{\sum_{i=1}^{n_s} p(y_j|a_j^i(m_j^i, C_j^i))} \quad (12)$$

Resampling. The inverse CDF method is used as the resampling algorithm. It includes the following steps: (a). construct CDF of y_j from the likelihood (b). Find the samples of $a^i_j(m^i_j, C^i_j)$ which make the CDF of y_j be the same value as (or the closest to) the randomly chosen value from a uniform distribution U(0,1). By repeating this process N times, N samples of $a^i_j(m^i_j, C^i_j)$ are obtained, which represents an approximation of the posterior distribution. After resampling process, every sample is assigned the same weight, $we^i_j = 1/N$.

A.3. Extended Kalman filter

EKF deals with state-parameter estimation by appending the parameters vector onto the true state vector to form an augmented vector and estimate the state and parameter simultaneously. In EKF, due to the normality assumption, the posterior distribution of state-parameter is a joint normal distribution that is characterized by mean and covariance. Therefore, instead of using a collection of samples to describe the posterior distribution, EFK gives only the mean and covariance matrix of the state-parameter vector. The recursive Paris model in Eq. (3) is used in EKF. The transition function is written as Eq. (13). Note that in typical EKF, the system transition process contains process noise, whose variance is denoted as Q and will be used in EKF process. However, in the purpose of prognostics in this paper, the process noise is ignored.

$$\begin{bmatrix} a_j \\ m_j \\ C_j \end{bmatrix} = \begin{bmatrix} a_{j-1} + C_{j-1}\left(\Delta\sigma\sqrt{\pi a_{j-1}}\right)^{m_{j-1}} \\ m_{j-1} \\ C_{j-1} \end{bmatrix} \tag{13}$$

In the following EKF process, we use the symbol "^" to represent an estimate and subscript "j" to denote the time step. Symbols "-" and "+" in the upper right corner are used to indicate prior estimate and posterior estimate respectively. For example, $[\hat{a}^-_j, \hat{m}^-_j, \hat{C}^-_j]$ represents a priori estimate of the crack size and parameters at time step j while $[\hat{a}^+_j, \hat{m}^+_j, \hat{C}^+_j]$ denotes the a posteriori estimate. Similar, $P^-{}_j$ is the a priori estimate for state error covariance matrix at time step j while $P^+{}_j$ is the posterior one. EKF consists two steps: perdition and update.

Before running EKF process, the following initial values are set. (1) The initial guess for crack size and parameters, denoted as $\hat{a}^+_0 \hat{m}^+_0, \hat{C}^+_0$, are drawn randomly from a uniform distribution centered at the true value, a_0, m_{true} and C_{true}, with the range of 50% around these values. Reminder that $a_0 = 0.8$ mm, $m_{\text{true}} = 3.8$, and $C_{\text{true}} = \ln(1.5\text{e}{-10})$. (2) The initial covariance matrix P_0 (in our case, a 3-by-3 matrix) is chosen depending on how much confidence one has to the initial estimates. (3) The variance of process noise Q is set to a small value of 1e−10 in order to make the EKF run while R is the measurement noise variance.

Prediction

The posterior estimates of state-parameter at time step j−1 are propagated to time j through the following transition function

$$\begin{bmatrix} \hat{a}^-_j \\ \hat{m}^-_j \\ \hat{C}^-_j \end{bmatrix} = \begin{bmatrix} \hat{a}^+_{j-1} + C\left(\Delta\sigma\sqrt{\pi\hat{a}^+_{j-1}}\right)^m \\ \hat{m}^+_{j-1} \\ \hat{C}^+_{j-1} \end{bmatrix} \tag{14}$$

The covariance P propagates as

$$P^-_j = \Phi_{j-1} P^+_{j-1} \Phi^T_{j-1} + Q \tag{15}$$

where Φ_{j-1} is the Jacobian matrix of the transition function with respect to the variables in the augmented vector, a, m, C, at the point $[\hat{a}^-_{j-1}, \hat{m}^-_{j-1}, \hat{C}^-_{j-1}]^T$. Specifically, Φ_{j-1} is calculated as

$$\Phi_{j-1} = \begin{bmatrix} 1 + C\frac{m}{2}\pi^{\frac{m}{2}}(\Delta\sigma)^m (a)^{\frac{m}{2}-1} & C(\Delta\sigma\sqrt{\pi a})^m \ln(\Delta\sigma\sqrt{\pi a}) & (\Delta\sigma\sqrt{\pi a})^m \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\Big|_{a=\hat{a}^-_{j-1}, m=\hat{m}^-_{j-1}, C=\hat{C}^-_{j-1}}$$

Update

The Kalman gain K_j is computed from Eq. (16), where H_j is the Jacobin matrix of the measurement function with respect to a, m and C at the point $[\hat{a}^-_j, \hat{m}^-_j, \hat{C}^-_j]^T$. In our case, the crack size is directly measured and the measurement data is simulated by adding noise to crack size. The measurement function can be written as $y_j = a_j + v_j$, where v_j is the noise. In such a case, H_j is a constant that $H = [1\ 0\ 0]$ and accordingly, K_j is a 3-by-1 matrix.

$$K_j = P^-_j H^T{}_j [H_j P^-_j H^T_j + R]^{-1} \tag{16}$$

The estimated measurement can be computed as

$$\hat{y}_j = \hat{a}^-_j \tag{17}$$

The posterior estimate of state is obtained from Eq. (18)

$$\begin{bmatrix} \hat{a}^+_j \\ \hat{m}^+_j \\ \hat{C}^+_j \end{bmatrix} = \begin{bmatrix} \hat{a}^-_j \\ \hat{m}^-_j \\ \hat{C}^-_j \end{bmatrix} + K_j\left(y_j - \hat{y}_j\right) \tag{18}$$

The error covariance matrix is updated as follow

$$P^+_j = [I - K_j H_j] P^-_j \tag{19}$$

The prediction and update steps are repeated at each measurement $y_{j,\ j=1,2,\ldots k}$, until the current time k. Then the posterior estimate at time k, $[\hat{a}^+_k, \hat{m}^+_k, \hat{C}^+_k]$, and the covariance matrix P^+_k are obtained. Samples can be drawn from a multivariate normal distribution with mean $\hat{a}^+_k, \hat{m}^+_k, \hat{C}^+_k$ and covariance matrix P^+_k. These samples are used for predicting the crack size behavior from the current time step k.

A.4. Nonlinear least squares

NLS finds the model parameters by minimizing the weighted sum of squared errors, $\text{SS}_{\text{E,W}}$.

$$SS_{E,W} = \sum_{j=1}^{k} \frac{(a_j(m,C) - y_j)^2}{w_j{}^2} = [\mathbf{a} - \mathbf{y}]^T \mathbf{W} [\mathbf{a} - \mathbf{y}] \tag{20}$$

where $w_j{}^2$ is a weight at the measurement point y_j, $\mathbf{W}$ a diagonal matrix with the inverse of weights $1/\ w_j{}^2$ as the diagonal elements, $a_j(m, C)$ the output from Paris' model, and $(a_j(m, C)\text{-}y_j)$ the residual. It is assumed that the magnitudes of the errors at all measured points are the same. Then $w_j{}^2$ becomes constant and identical to the variance of noise in the measurement data, which can be determined by the estimated variance of noise in data, as given in Eq. (21), in which k is the number of data and 2 is the number of parameters. Therefore, $\mathbf{W}$ is a constant and has no effect on the minimizing process to find parameters.

$$w_j^2 = \frac{[\mathbf{a} - \mathbf{y}]^T [\mathbf{a} - \mathbf{y}]}{k - 2} \tag{21}$$

In nonlinear least squares, the crack size a is not a linear combination of parameters m and C. In this case, the parameters are determined using an iterative process based on optimization techniques. The optimization process is not detailed, but the Matlab function 'lsqnonlin' is used instead. Readers refer the definition of 'lsqnonlin' to Matlab documentation of optimization toolbox. The input arguments of 'lsqnonlin' in our application are as follows. The objective function is a vector-valued function, that is, a k-component vector containing the residuals from time step 1 to k such that $\mathbf{F} = [a_1(m, C) - y_1, a_2(m, C) - y_2, \ldots, a_k(m, C) - y_k]$, in which m and C are design variables updating over the iterative process whose initial values should be designated by users. Here, 4 and

ln(1.0e−10) are assigned as the initial values for m and C, respectively. Levenberg-Marquardt method is chosen as the optimization algorithm for 'lsqnonlin'. 'lsqnonlin' gives the following outputs, the estimated parameters, denoted as $\hat{m}$ and $\hat{C}$, the residuals that can be used to calculated w_j^2 in Eq. (21), and a Jacobian matrix that will be used in the following calculation of the uncertainty in estimated parameters.

In general, the iterative process yields a single set of parameters' value that minimizes the $SS_{E,W}$ error. However, in prognostics, the optimum parameters depend on measurement data that include measurement variability. If a different data are used, the optimized parameters can be changed. That is, the estimated parameters have uncertainty. In NLS, the uncertainty of estimated parameters is characterized by its covariance matrix given by the following equation:

$$\sum_{\hat{m},\hat{C}} = [\mathbf{J}^T\mathbf{W}\mathbf{J}]^{-1} \tag{22}$$

where $\mathbf{J}$ is a Jacobian matrix of the first-order partial derivatives of the Paris model with respect to the model parameters m and C calculated at all measurements. In our case, $\mathbf{J}$ is a k-by-2 matrix such that

$$\mathbf{J} = \begin{bmatrix} \partial a_1(m,C)/\partial m & \partial a_1(m,C)/\partial C \\ \vdots & \vdots \\ \partial a_k(m,C)/\partial m & \partial a_k(m,C)/\partial C \end{bmatrix} \tag{23}$$

Eq. (22) can be rewritten as

$$\sum_{\hat{m},\hat{C}} = w_j{}^2[\mathbf{J}^T\mathbf{J}]^{-1} \tag{24}$$

Note that the residuals needed to compute w_j^2 and Jacobian matrix $\mathbf{J}$ can be obtained directly from 'lsqnonlin' function. The covariance matrix $\sum_{\hat{m},\hat{C}}$ can then be calculated. Once the model parameters and its variance are obtained, the distribution of parameters can be obtained. The crack growth trajectory is predicted based on the samples of estimated parameters.

Appendix B. True crack size at different flight cycles used in this paper

Table 9 tlists the true crack size (mm) at different flight cycles use in this paper. The crack grows exponentially at the end stage. To prevent some complex or abnormal value (e.g., Inf or NaN in MATLAB) in the numerical simulation, we set a limit of upper bound of 1000 mm for the crack size. Therefore, the value at 4900th and 5000th cycle does not necessarily mean that the crack size is 1000 mm.

Table 9
True crack size at different cycles, unit in mm.

Cycles	Crack size	Cycles	Crack size	Cycles	Crack size	Cycles	Crack size
Initial	8.0	1400	11.7	2800	20.7	4200	73.0
100	8.2	1500	12.0	2900	21.9	4300	87.5
200	8.4	1600	12.5	3000	23.2	4400	108.7
300	8.6	1700	12.9	3100	24.7	4500	142.3
400	8.8	1800	13.4	3200	26.3	4600	203.1
500	9.0	1900	13.9	3300	28.2	4700	343.9
600	9.3	2000	14.4	3400	30.4	4800	969.9
700	9.5	2100	15.0	3500	32.8	4900	1000
800	9.8	2200	15.6	3600	35.7	5000	1000
900	10.0	2300	16.3	3700	39.2		
1000	10.3	2400	17.0	3800	43.3		
1100	10.6	2500	17.8	3900	48.3		
1200	11.0	2600	18.7	4000	54.5		
1300	11.3	2700	19.7	4100	62.5		

Appendix C. Correlation between MSE and MSD in different noise levels

Table 10
Correlation between MSE and MSD in 3.75% COV noise level.

$N_m = 20$ ($N_f = 10$)				$N_m = 30$ ($N_f = 20$)				$N_m = 40$ ($N_f = 30$)			
BM	PF	NLS	EKF	BM	PF	NLS	EKF	BM	PF	NLS	EKF
0.94	0.90	0.99	0.95	0.86	0.91	0.99	0.99	0.97	0.99	0.99	0.99

Table 11
Statistics of match extent of MSE rank and MSD rank in 3.75% COV noise level.

$N_m = 20$ ($N_f = 10$)		$N_m = 30$ ($N_f = 20$)		$N_m = 40$ ($N_f = 30$)	
D_R	Times	D_R	Times	D_R	Times
0	88	0	87	0	87
3	1	3	1	3	1
5	3	5	6	5	6
7	5	7	5	7	6
15	2	21	1		
21	1				

Table 12
Correlation between MSE and MSD in 35% COV noise level.

$N_m = 20$ ($N_f = 10$)				$N_m = 30$ ($N_f = 20$)				$N_m = 40$ ($N_f = 30$)			
BM	PF	NLS	EKF	BM	PF	NLS	EKF	BM	PF	NLS	EKF
0.83	0.72	0.99	0.64	0.92	0.92	0.99	0.80	0.96	0.94	0.99	0.95

Table 13
Statistics of match extent of MSE rank and MSD rank in 35% COV noise level.

$N_m = 20$ ($N_f = 10$)		$N_m = 30$ ($N_f = 20$)		$N_m = 40$ ($N_f = 30$)	
D_R	Times	D_R	Times	D_R	Times
0	88	0	86	0	79
3	2	3	1	3	2
5	1	5	3	5	5
7	5	7	8	7	10
15	1	15	1	10	1
24	1	21	1	15	2
25	1			24	1
51	1				

Appendix D. An example to show the calculation of events A and B|A

The most frequent MSE ranking in the case of $N_m = 40$ ($N_f = 30$), PBNE, (refer to Table 2) is presented as an example to show that how to count the occurrence times of event A, B|A, and B|A*. There are 26 out of 100 datasets give the MSE ranking of PBNE, as listed in Table 14. To each dataset, each algorithm predicts 500 cycles further from the last measurements (corresponds to 4000th cycle) up to 4500th cycle. The relative errors of the predicted crack size with respect to the true crack size at 4500th cycle are reported in Table 14 (Note that the true crack size at 4500th cycle is 142.3 mm, refer to Appendix B). For reference, the predicted crack size at 4500th cycle is given in Table 15.

Table 14 tells that event A occurs 20 times out of 26. The six datasets, in which MSE-best fails to preserve the best in the future, are marked with an asterisk in the first column. The event B|A occurs 17 out of 20. The three datasets, in which MSD-best is not consistent with MSE-best given that the MSE-best preserves the best in the future, are marked with an asterisk in the last column. For data set #40, #58, #71 and #100, although the MSE-best (the particle filter) switch to the second best in the future, the difference with the future-best in terms of the relative error is less than 5% (2.7% for dataset #40, 3.0% for #100, and less than 1% for #58 and #71). Therefore, event A* occurs 24 times out of 26 and event B|A* occurs 20 times out of 24. In summary, based on the available data, one select the particle filter and there is 77% chance (20/26) that the particle filter will preserve good performance in the future.

Table 14
Relative error with respect to the true crack size at 4500th cycle by each algorithm, P-particle filter, B-Bayesian method, N-nonlinear least square, E-Extended Kalman filter.

Dataset #	Relative error at 4500th cycle (%)				MSD Ranking
	P	B	N	E	
4	16.95	25.17	55.38	62.02	PBNE
5	23.26	30.82	49.29	68.44	PBNE
7	16.77	27.49	52.79	66.27	PBNE
9	14.01	19.17	44.11	64.52	PBNE
12	22.41	23.58	41.37	71.98	PBNE
*20	7.85	1.88	44.09	66.61	BPNE
26	19.94	22.80	31.34	63.48	PBNE
33	0.12	21.14	25.38	63.98	*BPNE
35	0.40	15.26	33.51	63.29	PBNE
37	7.18	20.16	35.40	58.97	PBNE
*40	19.54	16.78	19.78	60.50	BPNE
44	16.05	23.62	37.36	59.13	PBNE
47	24.99	29.33	66.24	72.32	PBNE
48	5.04	17.75	51.41	71.06	PBNE
49	9.31	25.75	31.70	63.47	PBNE
*57	12.82	3.91	53.68	56.41	PBNE
*58	22.63	22.61	24.76	57.00	PBNE
61	3.29	17.46	44.80	63.75	*BPNE
62	4.14	16.76	42.76	73.99	PBNE
63	3.21	27.27	41.82	62.40	PBNE
67	9.81	20.53	31.98	56.64	PBNE
*71	10.95	10.06	38.21	62.07	PBNE
82	19.40	28.89	43.99	71.65	PBNE
84	12.46	23.95	54.80	71.97	PBNE
95	1.30	8.48	32.57	56.75	*BPNE
*100	12.73	9.71	15.11	63.14	PBNE

Table 15
The predicted crack size at 4500th cycle by each algorithm by each method, unit in mm.

Dataset #	Predicted crack size at 4500th cycle (mm)			
	P	B	N	E
4	118.18	106.48	63.50	54.04
5	109.20	98.45	72.15	44.92
7	118.44	103.18	67.18	48.00
9	122.36	115.02	79.53	50.49
12	110.42	108.75	83.43	39.87
20	131.12	139.63	79.56	47.51
26	113.92	109.86	97.71	51.96
33	142.13	112.21	106.18	51.26
35	141.74	120.58	94.61	52.23
37	132.08	113.61	91.92	58.38
40	114.49	118.43	114.16	56.21
44	119.46	108.69	89.14	58.15
47	106.73	100.56	48.04	39.38
48	135.13	117.04	69.14	41.18
49	155.56	105.66	97.19	51.98
57	124.06	147.87	218.69	62.03
58	110.10	110.13	107.07	61.19
61	137.61	117.45	78.55	51.58
62	148.20	118.45	81.46	37.02
63	137.74	103.50	82.79	53.51
67	128.34	113.09	96.79	61.70
71	126.72	127.99	196.68	53.98
82	114.69	101.19	79.70	40.35
84	124.56	108.23	64.31	39.88
95	140.45	130.24	95.95	61.54
100	124.19	128.48	120.80	52.46

References

[1] An D, Kim N-H, Choi J-H. Practical options for selecting data-driven or physics-based prognostics algorithms with reviews. Reliab Eng Syst Saf 2015;133:223–36.
[2] Baraldi P, Cadini F, Mangili F, Zio E. Model-based and data-driven prognostics under different available information. Probabilist Eng Mech 2013;32:66–79.
[3] Lu S, Tu Y, Lu H. Predictive condition based maintenance for continuously deteriorating systems. Quality Reliab Eng Int 2007;23:71–81.
[4] Bressel M, Hilairet M, Hissel D, Bouamama B. Extended Kalman filter for prognostic of proton exchange membrane fuel cell. Appl Energy 2016;164:220–7.
[5] Zio E, Peloni G. Particle filtering prognostic estimation of the remaining useful life of nonlinear components. Reliab Eng Syst Saf 2011;96:403–9.
[6] Zhang B, Sconyers C, Patrick R, Vachtsevanos G. A multi-fault modeling approach for fault diagnosis and failure prognosis of engineering systems. Annual conference of the prognostics and health management society San Diego, CA; 2009.
[7] Gobbato M, Kosmatka J, Conte J. A recursive Bayesian approach for fatigue damage prognosis: an experimental validation at the reliability component level. Mech Syst Signal Process 2014;45:448–67.
[8] Karandikar J, Kim N, Schmitz T. Prediction of remaining useful life for fatigue-damaged structures using Bayesian inference. Eng Fract Mech 2012;96:588–605.
[9] Walker E, Rayman S, White R. Comparison of a particle filter and other state estimation methods for prognostics of lithium-ion batteries. J Power Sources 2015;287:1–12.
[10] Kan M, Tan A, Mathew J. A review on prognostic techniques for non-stationary and non-linear rotating systems. Mech Syst Signal Process 2015;62:1–20.
[11] Lee J, Wu F, Zhao W, Ghaffari M, Liao L, Siegel D. Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications. Mech Syst Signal Processing 2014;42:314–34.
[12] An D, Choi JH, Kim NH. A comparison study of methods for parameter estimation in the physics-based prognostics. Proceeedings of the 53rd AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference; 2012.
[13] Pecht M, Jaai R. A prognostics and health management roadmap for information and electronics-rich systems. Microelectron Reliab 2010;50(3):317–23.
[14] Wang P, Youn BD, Hu C. A generic probabilistic framework for structural health prognostics and uncertainty management. Mech Syst Signal Process 2012;28:622–37.
[15] Kim NH, An D, Choi JH. Prognostics and health management of engineering systems. Switzerland: Springer International Publishing; 2017.
[16] Sankararaman S. Significance, interpretation, and quantification of uncertainty in prognostics and remaining useful life prediction. Mech Syst Signal Process 2015;52:228–47.
[17] Saxena A, Jose C, Bhaskar S, Sankalita S, Kai G. Metrics for offline evaluation of prognostic performance. Int J Prognost Health Manag 2010;1:4–23.
[18] Hu Y, Baraldi P, Maio F, Zio E. Online performance assessment method for a model-based prognostic approach. IEEE Trans Reliab 2016;65:718–35.

[19] Si X. An adaptive prognostic approach via nonlinear degradation modeling: application to battery data. IEEE Trans Ind Electron 2015;62:5082–96.
[20] Si X, Wang W, Hu C, Zhou D. Estimating remaining useful life with three-source variability in degradation modeling. IEEE Trans Reliab 2014;63:167–90.
[21] Andrieu C, Nando D, Doucet A, Jordan M. An introduction to MCMC for machine learning. Mach Learn 2003;50:5–43.
[22] Zio E, Peloni G. Particle filtering prognostic estimation of the remaining useful life of nonlinear components. Reliab Eng Syst Saf 2011;96:403–9.
[23] Hu Y, Baraldi P, Maio F, Zio E. A particle filtering and kernel smoothing-based approach for new design component prognostics. Reliab Eng Syst Saf 2015;134:19–31.
[24] An D, Choi J-H, Kim N-H. Prognostics 101: a tutorial for particle filter-based prognostics algorithm using Matlab. Reliab Eng Syst Saf 2013;115:161–9.
[25] Dalal M, Ma J, He D. Lithium-ion battery life prognostic health management system using particle filtering framework. Proc Inst Mech Eng Part O 2011;225:81–90.
[26] Jouin M, Gouriveau R, Hissel D, Pera M-C, Zerhouni N. Prognostics of PEM fuel cell in a particle filtering framework. Int J Hydrogen Energy 2014;39:481–94.
[27] Kimotho JK, Meyer T, Sextro W. PEM fuel cell prognostics using particle filter with model parameter adaptation. In: Prognostics and health management (PHM), 2014 IEEE conference on; 2014. p. 1–6.
[28] Guo L, Peng Y, Liu D, Luo Y. Comparison of resampling algorithms for particle filter based remaining useful life estimation. In: Prognostics and health management (PHM), 2014 IEEE conference on. IEEE; 2014. p. 1–8.
[29] Dong H, Jin X, Lou X, Wang C. Lithium-ion battery state of health monitoring and remaining useful life prediction based on support vector regression-particle filter. J Power Sources 2014;271:114–23.
[30] Yan HC, Pang CK, Zhou JH. Precognitive maintenance and probabilistic assessment of tool wear using particle filters. In: IECON 2013 - 39th annual conference of the IEEE industrial electronics society; 2013. p. 7382–7.
[31] Arulampalam M, Maskell S, Gordon N, Clapp T. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Trans Signal Process 2002;50:174–88.
[32] Nordh J. Metropolis-hastings improved particle smoother and marginalized models. In: Signal processing conference (EUSIPCO), 2015 23rd European; 2015. p. 973–7.
[33] Julier SJ, Uhlmann JK. A general method for approximating nonlinear transformations of probability distributions. Technical report, Robotics Research Group, Department of Engineering Science, University of Oxford; 1996.
[34] Garcia-velo J, Walker B. Aerodynamics parameter estimation for high-performance aircraft using extended Kalman filtering. J Guidance Control Dynamics 1997;20:1257–60.
[35] Chowdhary G, Jategaonkar R. Aerodynamics parameter estimation from flight data applying extended and unscented Kalman filter. Aerosp Sci Technol 2010;14:106–17.
[36] Bisht S, Singh M. An adaptive unscented Kalman filter for tracking sudden stiffness changes. Mech Syst Signal Process 2014;49:181–95.
[37] VanDyke MC, Schwartz JL, Hall CD. Unscented Kalman filtering for spacecraft attitude state and parameter estimation. Blacksburg, Virginia: Department of Aerospace & Ocean Engineering, Virginia Polytechnic Institute & State University; 2004.
[38] Cortie MB, Garrett GG. On the correlation between the C and m in the Paris equation for fatigue crack propagation. Eng Fract Mech 1988;30(1):49–58.
[39] Benson JP, Edmonds DV. The relationship between the parameters C and m of Paris' law for fatigue crack growth in a low-alloy steel. Scripta Metall 1978;12(7):645–7.
[40] Bilir ÖG. The relationship between the parameters C and n of Paris' law for fatigue crack growth in a SAE 1010 steel. Eng Fract Mech 1990;36:361–4.
[41] Yiwei WANG, Christian GOGU, Binaud N, Christian BES, Haftka RT. A cost driven predictive maintenance policy for structural airframe maintenance. Chin J Aeronaut 2017.
[42] An D, Choi J-H, Kim NH. Identification of correlated damage parameters under noise and bias using Bayesian inference. Struct Health Monit 2011;11:293–303.
[43] Doucet A, Johansen AM. A tutorial on particle filtering and smoothing: fifteen years later. Handbook of nonlinear filtering 2009;12:656–704.
[44] Grewal MS, Andrews AP. Kalman filtering: theory and practice with MATLAB. 4th ed. Wiley-IEEE Press; 2015.
[45] The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems. Available via Duke University. http://people.duke.edu/~hpgavin/ce281/lm.pdf. Accessed 28 May 2016.