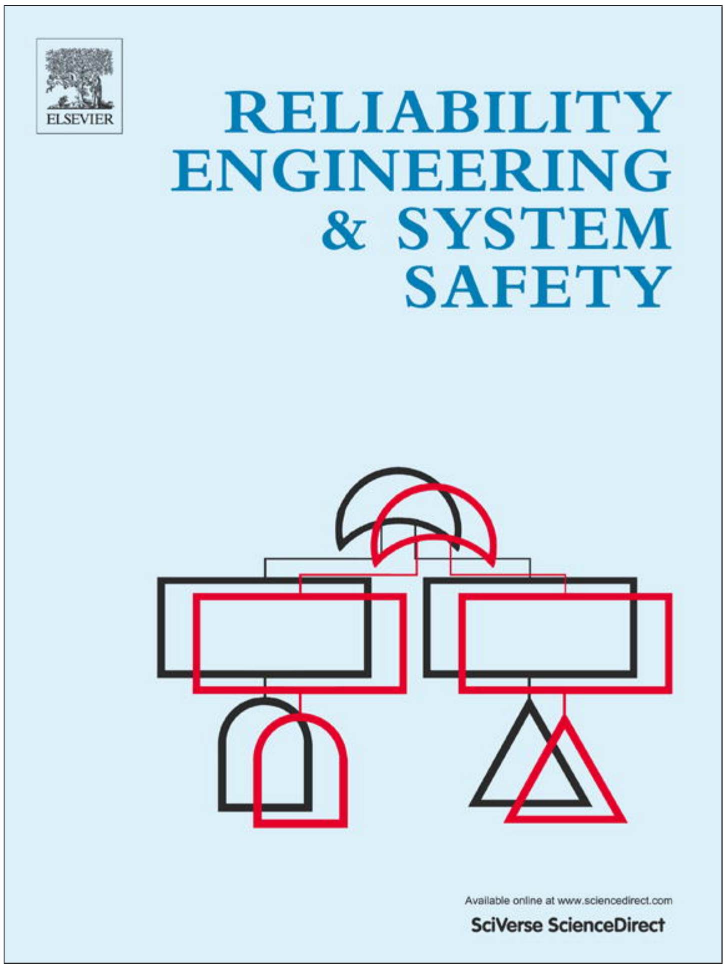

ELSEVIER
RELIABILITY ENGINEERING & SYSTEM SAFETY
Available online at www.sciencedirect.com
SciVerse ScienceDirect

Reliability Engineering and System Safety 115 (2013) 161–169

Contents lists available at SciVerse ScienceDirect

Reliability Engineering and System Safety

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

Prognostics 101: A tutorial for particle filter-based prognostics algorithm using Matlab

Dawn An [a,b,1], Joo-Ho Choi [a,2], Nam Ho Kim [b,*]

[a] Department of Aerospace & Mechanical Engineering, Korea Aerospace University, 100 Hanggongdae-gil, Hwajeon-dong, Deokyang-gu, Goyang-si, Gyeonggi-do 412-791, Republic of Korea
[b] Department of Mechanical & Aerospace Engineering, University of Florida, Gainesville, FL 32611, USA

ARTICLE INFO

Article history:
Received 19 September 2012
Received in revised form
17 January 2013
Accepted 24 February 2013
Available online 14 March 2013

Keywords:
Battery degradation
Crack growth
Matlab code
Model-based prognostics
Particle filter
Remaining useful life

ABSTRACT

This paper presents a Matlab-based tutorial for model-based prognostics, which combines a physical model with observed data to identify model parameters, from which the remaining useful life (RUL) can be predicted. Among many model-based prognostics algorithms, the particle filter is used in this tutorial for parameter estimation of damage or a degradation model. The tutorial is presented using a Matlab script with 62 lines, including detailed explanations. As examples, a battery degradation model and a crack growth model are used to explain the updating process of model parameters, damage progression, and RUL prediction. In order to illustrate the results, the RUL at an arbitrary cycle are predicted in the form of distribution along with the median and 90% prediction interval. This tutorial will be helpful for the beginners in prognostics to understand and use the prognostics method, and we hope it provides a standard of particle filter based prognostics.

1. Introduction

Although many prognostics methods have been presented in literature [1–6], it is still difficult for engineers to use them for their applications. The objective of this paper is to demonstrate how to use a prognostics method using a simple Matlab code as short as 62 lines.

In general, prognostics methods can be categorized into data-driven, model-based, and hybrid approaches [7]. The data-driven method does not use any particular physical model and largely depends on measured data. On the other hand, the model-based approach assumes that a physical model describing the behavior of damage or degradation is available and combines the model with measured data to identify model parameters. Hybrid approaches combine the above-mentioned two methods to improve the prediction performance.

Among the abovementioned prognostics methods, the model-based approach is considered since if there exist physical model, it is easy to establish standard algorithm logically compared to other approaches. In this approach, the model parameters which have an effect on model behavior are often unknown and need to be identified as a part of the prognostic process. There are several methods to estimate model parameters, such as the Kalman filter (KF) that gives an exact PDF in analytical form in the case of a linear model with a Gaussian noise [8]; Particle filter (PF), in which the posterior distribution of model parameters is expressed as a number of particles and their weights [9–11]; and Bayesian method (BM) that is to estimate the model parameters using measurement data, which are incorporated into a single posterior distribution [12–14]. In this paper, PF is employed because it can be used for a nonlinear model with non-Gaussian noise and is the most widely used in the field of prognostics.

The Matlab code is composed of 62 lines including detailed explanations, which is further divided into three parts: (1) problem definition; (2) prognostics using PF; and (3) post-processing. Users are required to modify the first part according to their application. For demonstration purposes, examples of battery degradation and crack growth are presented.

The rest sections are organized as follows: in Section 2, the overall process of model-based prognostics is explained with the Matlab code; in Section 3, the usage is explained with a battery degradation example; and in Section 4, various cases are described with a crack growth example, followed by conclusions in Section 5.

2. Model-based prognostics

The process of model-based prognostics is illustrated in Fig. 1, in which the degradation model is expressed as a function of usage conditions U, elapsed cycle or time t, and model parameters $\boldsymbol{\theta}$. The usage conditions and time are given, while the model parameters characterizing the damage behavior should be identified. Then, the remaining useful life (RUL) which represents the remaining time to failure is calculated based on the estimated model parameters.

* Corresponding author. Tel.: +1 352 575 0665; fax: +1 352 392 7303.
E-mail addresses: dawnan@ufl.edu (D. An), jhchoi@kau.ac.kr (J.H. Choi), nkim@ufl.edu (N.H. Kim).
[1] Tel.: +1 352 870 4774; fax: +1 352 392 7303.
[2] Tel.: +82 2 300 0117; fax: +82 2 3158 2191.

0951-8320/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.ress.2013.02.019

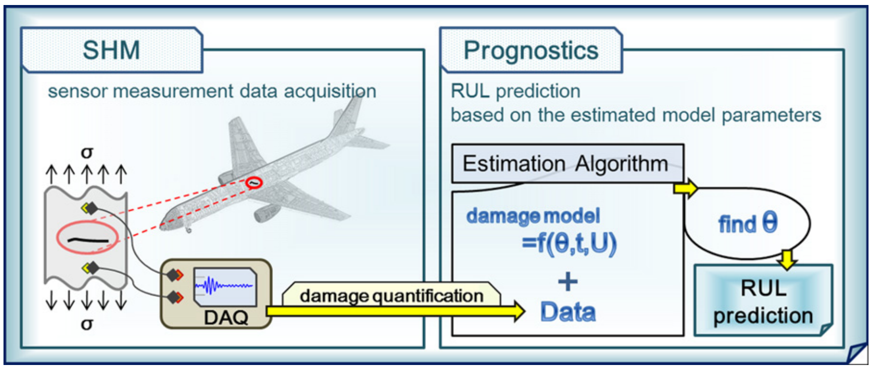


Fig. 1. Illustration of the model-based prognostics process.

The model parameters are estimated using an algorithm such as PF by integrating the damage model with the damage data that represent the system's health state at the time the data are obtained. Since damage cannot be directly measured in most cases, a damage quantification process is required from sensor measurement data, which is called structural health monitoring (SHM). This tutorial assumes that data are available in terms of the level of damage at various times.

2.1. *Model definition: battery degradation*

In the following explanation, 'line' or 'lines' in a parenthesis indicates the line number of the code in Appendix. In the degradation of a battery (line 2), it is well known that the capacity of a secondary cell such as a Lithium-ion battery degrades over cycles in use, and the failure threshold is defined when the capacity fades by 30% of the rated value (line 7). A simple form of the empirical degradation model is expressed by an exponential growth model as follows [15]:

$$\lambda = a\exp(-bt) \tag{1}$$

where a and b are model parameters, t is time or cycles, and λ is the internal battery performance which is stated as either the electrolyte resistance R_E or the transfer resistance R_{CT}. There is a relation that C/1 capacity, i.e., capacity at nominally rated current of 1 A is inversely proportional to the sum of the two resistances R_E+R_{CT} [15]. Therefore, even though the internal battery performance is normally observed instead of capacity, C/1 capacity is considered as λ, and observed data are assumed to be given as a form of $C/1$ capacity for the purpose of demonstrating the prognostics algorithm.

The $C/1$ capacity data (lines 5–6) measured at every 5 weeks (lines 3–4) are given in Table 1. The data are generated by: (a) assuming that the true model parameters $a_{\text{true}}=1$ and $b_{\text{true}}=0.012$; (b) calculating the true $C/1$ capacity according to Eq. (1) for the given time steps; and (c) adding Gaussian noise $\varepsilon \sim N(0,0.05^2)$ to the true $C/1$ capacity data. The true values of parameters are only used to generate observed data. Then, the goal of prognosis is to estimate b using the data (lines 16–39).[3]

2.2. *Estimation algorithm: particle filter (PF)*

PF uses a statistical method called Bayesian inference, in which observations are used to estimate and update unknown parameters as a form of the probability density function (PDF). Bayesian inference is based on the following Bayes′ theorem [16]:

$$p(\mathbf{\Theta}|\mathbf{z}) \propto L(\mathbf{z}|\mathbf{\Theta})p(\mathbf{\Theta}) \tag{2}$$

where $\mathbf{\Theta}$ is a vector of unknown parameters, $\mathbf{z}$ is a vector of observed data, $L(\mathbf{z}|\mathbf{\Theta})$ is the likelihood or the PDF value of $\mathbf{z}$ conditional on the given $\mathbf{\Theta}$, $p(\mathbf{\Theta})$ is the prior PDF of $\mathbf{\Theta}$, and $p(\mathbf{\Theta}|\mathbf{z})$ is the posterior PDF of $\mathbf{\Theta}$ conditional on $\mathbf{z}$.

In PF, the Bayesian update is processed in a sequential way with particles (or samples) having probability information of unknown parameters; When a new measurement is available, the posterior at the previous step is used as the prior information at the current step, and the parameters are updated by multiplying it with the likelihood. Therefore, PF is also known as the sequential Monte Carlo method [9,10]. The general process of PF is based on the state transition function f and the measurement function h [10,11]:

$$x_k = f(x_{k-1},\mathbf{\theta}_k,v_k) \tag{3}$$

$$z_k = h(x_k,\omega_k) \tag{4}$$

where k is the time step index, x_k is the damage state, $\mathbf{\theta}_k$ is a vector of model parameters, z_k is measurement data. v_k and ω_k are, respectively, process and measurement noise. In the prognostics area, the state transition function f is referred to as a damage model.

According to the damage model in Eq. (3), the battery degradation model in Eq. (1) can be rewritten in the following form (line 29):

$$x_k = \exp(-b_k\Delta t)x_{k-1} \tag{5}$$

with $t_k = t_{k-1}+\Delta t$. In this case, process noise v_k is ignored because it can be handled through the uncertainty in model parameters. For the measurement function, it is assumed that z_k is the same as $C/1$ capacity including measurement noise ω_k. Gaussian noise, $\omega_k \sim N(0,\sigma)$, is used with unknown standard deviation σ. Therefore, the unknown parameters become $\mathbf{\Theta}=[x,\mathbf{\theta}(=[b]),\sigma]^T$, including the damage state x_k which is obtained based on the model parameter b_k (see Eq. (5)) (line 8).

The process of PF is based on the Bayes' theorem illustrated in Fig. 2 with one parameter estimation. At the first time step, i.e., $k=1$, n samples of the parameters are drawn from the initial (prior) distribution (lines 9, 16–21). Then, the following three steps are employed. In the first prediction step (lines 25–30), the posterior distributions of the model parameters at the previous ($k-1$th) step are used for the prior at the current (kth) step in the form of samples (lines 26–27). Also, the damage state at the current time is transmitted from the samples of the damage model at the previous step based on the model parameters (lines 28–29). The samples in this step correspond to $p(\mathbf{\Theta}_k)$ in Fig. 2. Next is the updating step (lines 31–33), which is related to the likelihood of measurement data $L(z_k|\mathbf{\Theta}_k)$ in Fig. 2. Assuming ω_k is normally distributed, the likelihood of the measurement can be expressed as (line 33):

$$L(z_k|x_k^i,b_k^i,\sigma_k^i) = \frac{1}{\sqrt{2\pi}\sigma_k^i}\exp\left[-\frac{1}{2}\left(\frac{z_k-x_k^i(b_k^i)}{\sigma_k^i}\right)^2\right], \quad i=1,\ldots,n \tag{6}$$

[3] For simplicity, it is assumed that $a=1$ is given.

Table 1
Measurement data for battery degradation problem.

Time step, k	Initial, 0	1	2	3	4	5	6	7	8	9
Weeks	0	5	10	15	20	25	30	35	40	45
$C/1$ (Ahr)	1.0000	0.9351	0.8512	0.9028	0.7754	0.7114	0.6830	0.6147	0.5628	0.7090

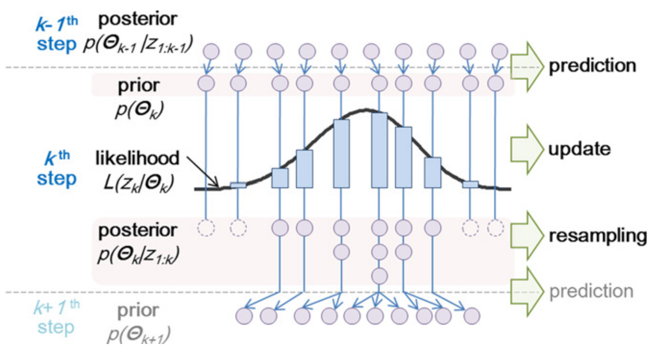


Fig. 2. Illustration of the PF process.

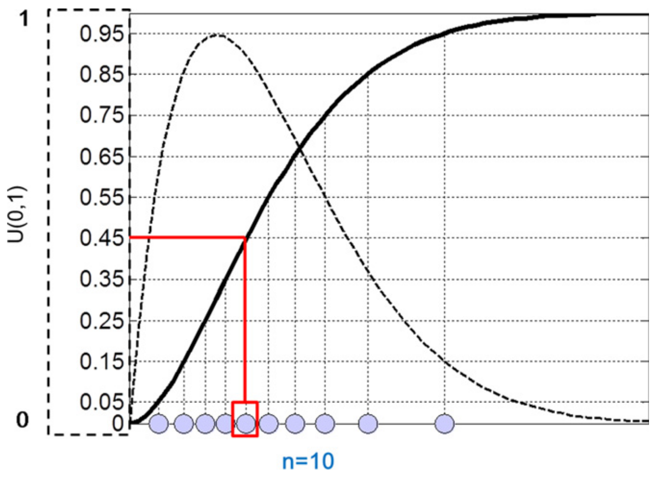


Fig. 3. Illustration of resampling method.

In Eq. (6), the PDF value of z_k at the given ith samples of the unknown parameters $\mathbf{\Theta}=x,b,\sigma$ corresponds to the weight of the ith samples; the weight is proportional to the magnitude of the PDF value, which is expressed as the length of the vertical bar in Fig. 2. Finally, the samples with high or low weight are duplicated or eliminated, respectively, at the resampling step (lines 34–39). Among several methods, the inverse CDF method [10] is used, which is illustrated in Fig. 3. Firstly, a CDF is constructed from the likelihood function in Eq. (6) (line 35), which is illustrated as solid curve in Fig. 3. Next, a random value is generated from $U(0,1)$ (line 37), which becomes a CDF value (e.g., 0.45 in Fig. 3). Finally, a sample of the parameter having the CDF value is found (line 38), which is marked by a rectangle in Fig. 3. By repeating this process n times, n samples are obtained (line 36). Note that since samples exist in a discrete form, the sample having the closest value to the CDF value is selected. Consequently, the resampled results become the posterior distribution $p(\mathbf{\Theta}_k|\mathbf{z}_{1:k})$ in Fig. 2, which corresponds to the posterior distribution at the current step (line 38), and is also used as the prior distribution at the next ($k+1$th) step (lines 25–30).

2.3. Prognosis: predicting the damage state and RUL

Once the estimated parameter is obtained (lines 25–39, line 43), the future damage state and RUL can be predicted by progressing the damage state until it reaches the threshold as shown in Fig. 4 (lines

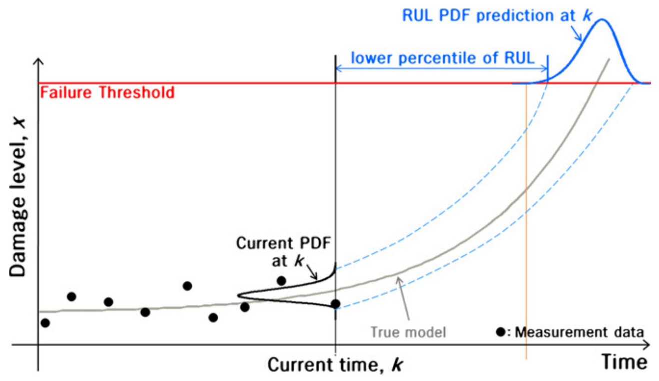


Fig. 4. Illustration of RUL prediction.

24–30, 40–47). In Fig. 4, the two dashed curves and the PDF shape, respectively, represent the prediction interval of the damage state and the distribution of time when the damage state reaches the threshold. The distribution of RUL can be obtained by subtracting this PDF from the threshold. In the prognosis step, only the damage state is transmitted (lines 25–30) without updating model parameters (lines 40–41). At this time, the measurement error with the updated standard deviation is added to the damage state (lines 44–46).

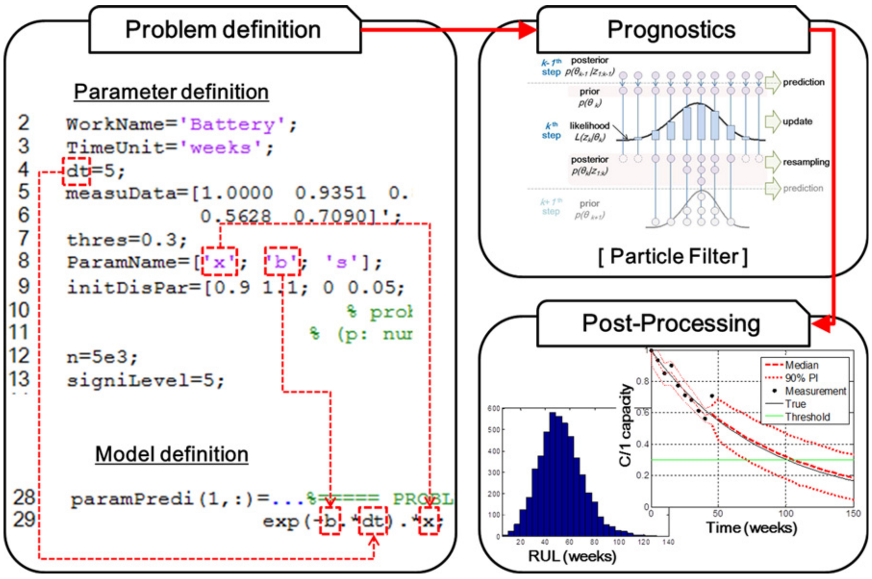
3. Matlab implementation

In this section, the usage of the 62-line Matlab code is explained. The code is divided into three parts: (1) problem definition for user-specific applications, (2) prognostics using PF, and (3) post-processing for displaying results. The block diagram of the code is illustrated in Fig. 5. Only the problem definition part needs to be modified for different applications, which are further divided into two sections: parameter definition and model definition. In the parameter definition, all known parameters as well as the initial estimate of unknown parameters are defined, such as the name of parameters to be estimated, the probability parameters of initial distributions of the unknown parameters and measured data, etc. (lines 1–14). Next, the damage equation or state transition function is defined in model definition (line 29). Once these two are completed, users can obtain the RUL distribution at the current time and its percentiles, median and 90% prediction interval. Detailed explanations are given in the following subsections with an example of battery degradation, in which italicized bold letters represent the Matlab code in the Appendix.

3.1. Problem definition (lines 1–14, 29)

3.1.1. Parameter definition (lines 1–14)

For the battery degradation example, ***'Battery'*** is used for ***WorkName***, which is the name of the result file. The capacity is measured at every 5 weeks, so ***'weeks'*** and the number ***5*** are, respectively, typed in ***TimeUnit*** and ***dt***. The $C/1$ capacity data in Table 1 are stored in ***measuData***, which is a $k1\times 1$ vector. According to the definition of failure threshold in Sections 2.1, ***0.3*** (30% of $C/1$ capacity) is used for ***thres***. ***ParamName*** is the name of the unknown parameters to be estimated; damage state ***'x'***, model parameter ***'b'*** and the standard deviation of



```
2   WorkName='Battery';
3   TimeUnit='weeks';
4   dt=5;
5   measuData=[1.0000  0.9351  0.
6                  0.5628  0.7090]';
7   thres=0.3;
8   ParamName=['x'; 'b'; 's'];
9   initDisPar=[0.9 1.1; 0 0.05;
10                            % prol
11                         % (p: nur
12  n=5e3;
13  signiLevel=5;

28  paramPredi(1,:)=...%===== PROBL
29                exp(-b.*dt).*x;
```

Fig. 5. Block diagram of the code.

measurement error **'s'** are included. When determining the parameters' name, there are four cautions: (1) the user can define anything for the parameter's name, but the length of parameters' name should be the same as each other; (2) when assigning a one letter name, be careful not to use ***i***, ***j***, ***k***, ***n***, ***p***, ***u*** because they are already used in the code; (3) the parameter's name representing the damage state and model parameters should be used as the model equation on line 29; and (4) the parameter's name of the damage state and standard deviation, respectively, should be placed on the first and the last row. ***initDisPar*** is a $p \times q$ matrix of probability parameters of initial distributions, where p and q are the number of unknown parameters and probability parameters, respectively. Since there are no available prior information, it is assumed that the initial distributions of the three $(=p)$ unknown parameters are uniform whose probability parameters are two $(=q)$, lower and upper bounds:

$$x_o \sim U(0.9,1.1), \quad b_0 \sim U(0,0.05), \quad s_0 \sim U(0.01,0.1) \qquad (7)$$

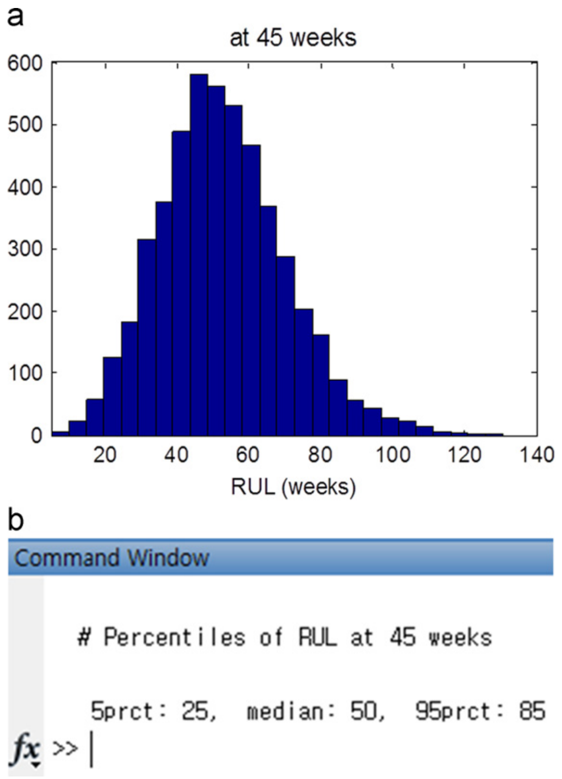
Eq. (7) can be typed as [***0.9 1.1***; ***0 0.05***; ***0.01 0.1***]; in ***initDisPar***. The rest of the required parameters are the number of particles (or samples) ***n*** and significance level ***signiLevel*** for calculating the confidence interval (C.I.) and prediction interval (P.I.). Usually, 1000–5000 particles and a 5, 2.5 or 0.5 significance level for 90%, 95% or 99% intervals are used. In this example, ***5000*** and ***5*** are set for ***n*** and ***signiLevel***, respectively. To consider the effect according to the number of samples, users can refer to Ref. [17].

3.1.2. Model definition (line 29)

The damage model equation in Eq. (5) is used in line 29. In the equation, the time interval Δt is expressed as ***dt*** in the script, which was defined in line 4. Also, the model parameter b_k and the damage state at the previous step x_{k-1}, respectively, are expressed as ***b*** and ***x***, which were defined in line 8. The algebraic expressions should use component-wise operations (i.e., using '.') since damage state is a vector with n samples.

3.2. Prognostics using PF (lines 15–47)

The prognostics process is composed of three steps: (1) the initial distributions of the parameters (lines 16–21), (2) estimation process (lines 25–39, 43), and (2) prognosis (lines 24–30, 40–47). In terms of the code usage, there are two issues that can be considered according to users' intention: the initial distribution (line 18) and the likelihood



```
Command Window

# Percentiles of RUL at 45 weeks

5prct: 25,  median: 50,  95prct: 85
fx >>
```

Fig. 6. Visual results obtained from the code: battery degradation example: (a) RUL distribution and (b) percentiles of RUL distribution.

function (line 33). In the code, the default options for the initial distribution and the likelihood function, respectively, are uniform and normal distribution. The other probability distributions can also be employed, and this will be introduced in Section 4.

3.3. Post-processing (lines 48–62)

Once problem definition is completed and the code is implemented, distribution and its percentiles of RUL at the current time can be displayed. Fig. 6 shows RUL results at 45 weeks after the updating process is progressed up to $k=9$ (see Table 1; $k=9$ corresponds to $\boldsymbol{k1}=\mathbf{10}$ in the script since the initial, $k=0$ is stored in ***k1***=**1**). Fig. 6(b) shows 5, 50 (median), and 95 percentiles, which are caused by ***signiLevel***=***5*** (line 13). The result of median can be compared with the true RUL, which is obtained by subtracting the current time (45 weeks) from the threshold time. Also, the threshold time is when the $C/1$ capacity reaches 0.3 and results in 100.33 weeks, which is calculated using Eq. (1) and true model parameters given in Section 2.1. Therefore, the median of RUL prediction, 50 weeks is fairly accurate compared with the true RUL, 55 weeks. In addition, RUL prediction can be more accurate by reducing the time interval after the current time. All results are saved as a name of '***WorkName*** (line 2) at current time.mat'.

The other results, such as the trace of parameters and prediction of the damage state, can be displayed by using sampling results during the updating process. Users can display the sampling results of any variable at each step by entering ***ParamResul*** into the Matlab command window; ***xResul***, ***bResul*** and ***sResul*** are forms of adding ***ParamName*** (line 8) to ***Resul***. Therefore, users can draw the percentiles of the damage state prediction by coding ***plot***(***repmat*** (***time***,***1***,***3***), ***prctile***(***xResul'***,***perceValue***)'), and for the other cases, ***xResul*** is replaced with ***bResul*** or ***sResul***. If the true values of the model parameters are known, the results can be compared with the true values. In this problem, the true values of b=0.012 and s=0.05, and the true damage state are calculated using Eqs. (1) or (5). The additional visual results are shown in Fig. 7.

4. Practical use

The code can be easily adapted by users for more practical use. As an example, the usage algorithm with a crack growth example is considered in the following subsections.

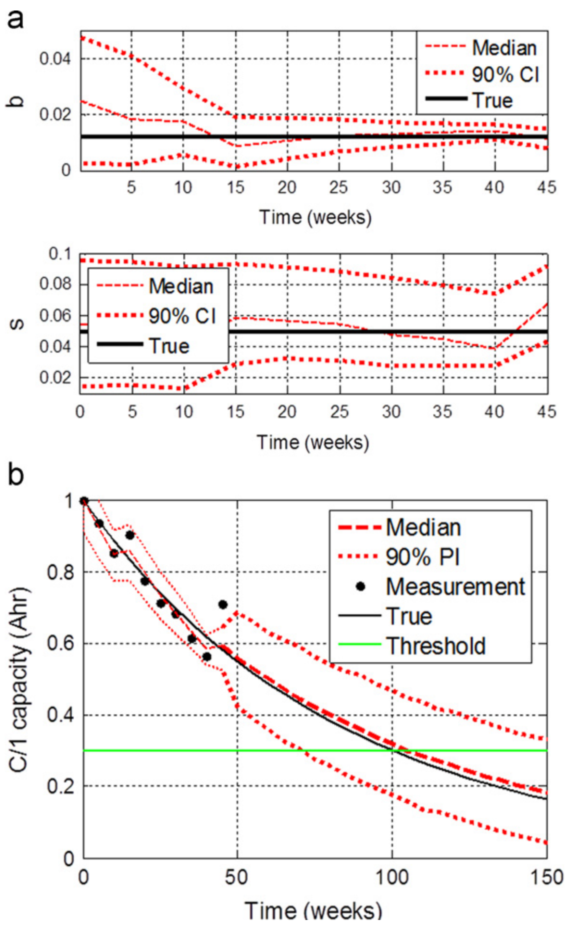


Fig. 7. Visual results obtained from the additional code: battery degradation example: (a) trace of parameter update and (b) $C/1$ capacity prediction.

4.1. *Model definition: crack growth*

It is assumed that a through-the-thickness center crack exists in an infinite plate under the mode I loading condition. The rate of damage growth can be expressed using the Paris model [18] as

$$\frac{da}{dN} = C(\Delta K)^m, \quad \Delta K = \Delta\sigma\sqrt{\pi a} \tag{8}$$

where a is the crack size, N is the number of cycles, m and C are damage model parameters, ΔK is the range of the stress intensity factor, and $\Delta\sigma$ is the stress range. The model can be rewritten in the form of the state transition function:

$$a_k = C_k(\Delta\sigma\sqrt{\pi a_{k-1}})^{m_k} dN + a_{k-1} \tag{9}$$

The model parameters m_k and C_k as well as the damage state a_k are estimated using the measured crack size z_k at every 50 cycles under loading condition $\Delta\sigma = 78$ MPa, which is listed in Table 2. First the true crack size data are generated using Eq. (9) with $m_{\text{true}} = 3.8$ and $C_{\text{true}} = 1.5 \times 10^{-10}$. The measured crack size data are then generated by multiplying noise, which is lognormally distributed with standard deviation of $0.001/a_k$(m). Actually, it has been shown that the distribution of crack size follows a lognormal distribution [19]. For the RUL calculation, the critical crack size is determined as 0.0463 m. More specific information for crack growth problem is in Ref. [13].

It is assumed that the standard deviation of measurement, σ, is known as 0.001 m. Also, the initial distribution of the parameters and the likelihood function are, respectively, normal and lognormal distributions, which are as follows:

– initial distribution:

$$a_0 \sim N(0.01,(5\times10^{-4})^2), \quad m_0 \sim N(4,0.2^2), \\ \log C_0 \sim N(-22.33,1.12^2) \tag{10}$$

– likelihood function:

$$L(z_k|a_k^i,m_k^i,C_k^i) = \frac{1}{z_k\sqrt{2\pi}\zeta_k^i}\exp\left[-\frac{1}{2}\left(\frac{\ln z_k - \lambda_k^i}{\zeta_k^i}\right)^2\right], \quad i=1,\ldots,n \tag{11}$$

where $\zeta_k^i = \sqrt{\ln[1+(\sigma/a_k^i(m_k^i,C_k^i))^2]}$ and $\lambda_k^i = \ln[a_k^i(m_k^i,C_k^i)] - 1/2(\zeta_k^i)^2$.

4.2. *Modifying the code for the crack growth example*

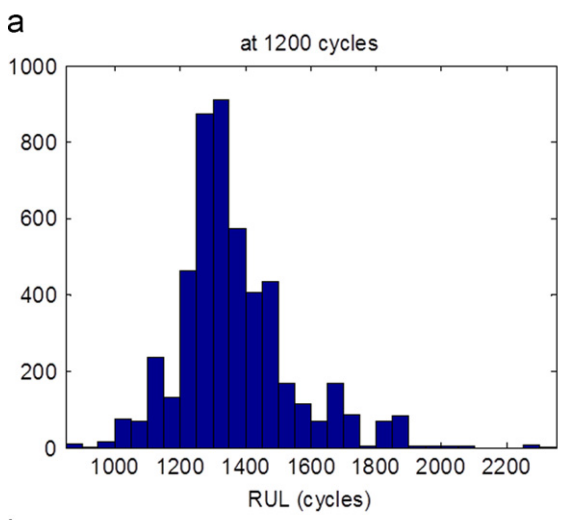
4.2.1. *Problem definition*

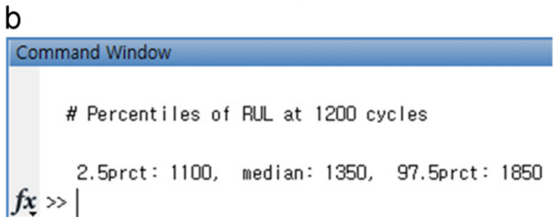
Based on the above given information, the code in the Appendix is changed as follows:

– line 2: ***WorkName***=***'Crack'***;
– line 3: ***TimeUnit***=***'cycles'***;
– line 4: ***dt***=***50***; (or ***dN***=***50***, but should be matched with line 29)
– lines 5–6: ***measuData***=[***0.0119 0.0103 0.0118 0.0095 0.0085 0.0122 0.0110 0.0120 0.0113 0.0122 0.0110 0.0124 0.0117 0.0138 0.0127 0.0115 0.0135 0.0124 0.0141 0.0160 0.0157 0.0149 0.0156 0.0153 0.0155***]';
– line 7: ***thres***=***0.0463***;
– line 8: ***ParamName***=[***'a'***;***'m'***; ***'C'***; ***'s'***];
– line 9: ***initDisPar***=[***0.01 5e-4; 4 0.2;*** −***22.33 1.12; 0.001 0***]; This corresponds to Eq. (10) except the last two values, ***0.001*** and ***0*** for σ(=***'s'***). Even if σ is a deterministic value, it should be included in lines 8–9. Therefore, users should make σ become a deterministic value (0.001 m) by using ***0.001*** and ***0***, which stand for mean and standard deviation, respectively. In other words, the probability parameters should be set to make the n samples become the same as a deterministic value.
– line 13: ***signiLevel***=***2.5***; 95% intervals are calculated.
– insert it next line 13: ***delSigma***=***78***;

Table 2
Measurement data for crack growth problem.

Time step, *k*	Initial, 0	1	2	3	4	5	6	7	8
Time (cycles)	0	50	100	150	200	250	300	350	400
Crack size (m)	0.0119	0.0103	0.0118	0.0095	0.0085	0.0122	0.0110	0.0120	0.0113
Time step, *k*		9	10	11	12	13	14	15	16
Time (cycles)		450	500	550	600	650	700	750	800
Crack size (m)		0.0122	0.0110	0.0124	0.0117	0.0138	0.0127	0.0115	0.0135
Time step, *k*		17	18	19	20	21	22	23	24
Time (cycles)		850	900	950	1000	1050	1100	1150	1200
Crack size (m)		0.0124	0.0141	0.0160	0.0157	0.0149	0.0156	0.0153	0.0155





```
# Percentiles of RUL at 1200 cycles

 2.5prct: 1100,  median: 1350,  97.5prct: 1850
fx >> |
```

Fig. 8. Visual results obtained from the code: crack growth example: (a) RUL distribution and (b) percentiles of RUL distribution.

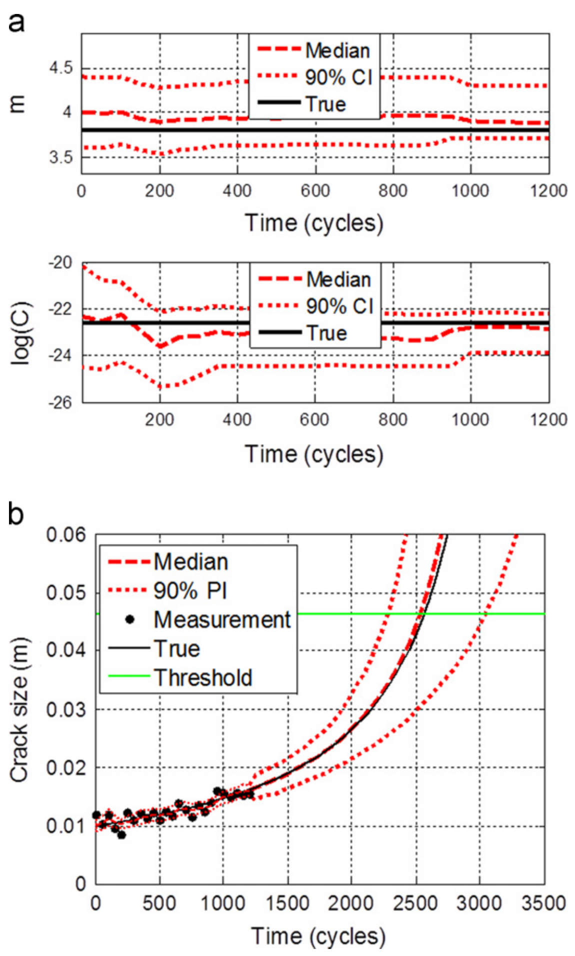


Fig. 9. Visual results obtained from the additional code: crack growth example: (a) trace of parameter update and (b) crack growth prediction.

- line 29: ***exp*(*C*). *(*delSigma*.*sqrt(pi*a)).^m.*dt+a**; This corresponds to Eq. (9), but note that log(*C*) is used instead of *C* due to a numerical problem (*C* is a very small value).

4.2.2. Prognostics using PF

The initial distribution of the parameters and the likelihood function are different from those of battery degradation. Therefore, the lines 18, 33 and 45 should be modified as follows:

- line 18: ***param*(*j*,:)=*normrnd*(*initDisPar*(*j*,1),*initDisPar*(*j*,2), 1,*n*);**
- line 33: ***sigl*=*sqrt*(*log*(1+(*paramPredi*(*end*,:)./ *paramPredi*(1, :)).^2)); *mul*=*log*(*paramPredi* (1,:))-0.5*sigl.^2; *likel* =*lognpdf*(*measuData*(*k*),*mul*,*sigl*);**
- line 45: ***sigl*=*sqrt*(*log*(1+(*param*(*end*,:)./*param*(1,:)).^2)); *mul*=*log*(*param*(1,:))-0.5*sigl.^2; *eval*([*ParamResul*(1,:) '(*k*, :)= *lognrnd* (*mul*,*sigl*,1,*n*);'*]);**

If the prior information and the distribution type of measurement error are not given, the initial distribution and the likelihood function should be assumed. It would be a good exercise to study different distribution types based on the above revision.

4.2.3. Post-processing

The results obtained from the code are shown in Fig. 8. Also, users can obtain Fig. 9 using the stored results of the parameters; ***aResul***, ***mResul***, ***CResul*** which can be known by typing ***ParamResul*** in the command window.

Table 3
Measurement data and load conditions for VAL crack growth problem.

Time step, *k*	Initial, 0	1	2	3	4	5	6
Time (cycles)	0	5000	10,000	15,000	18,000	20,000	26,000
Crack size (m)	0.0056	0.0099	0.0180	0.0180	0.0180	0.0180	0.0180
Load conditions	(0–10,000 cycles) *p* cycle: #=5, $\sigma_{\max}$=69 MPa *q* cycle: #=45, $\sigma_{\max}$=130 MPa			(10,001–18,000 cycles) *p* cycle: #=45, $\sigma_{\max}$=69 MPa *q* cycle: #=5, $\sigma_{\max}$=100 MPa		(18,001–38,000 cycles) *p* cycle: #=40, $\sigma_{\max}$=69 MPa *q* cycle: #=10, $\sigma_{\max}$=100 MPa	

Time step, *k*	7	8	9	10	11	12	13	14	15
Time (cycles)	32,000	38,000	40,000	42,000	44,000	46,000	48,000	50,000	50,500
Crack size (m)	0.0180	0.0180	0.0187	0.0204	0.0216	0.0239	0.0263	0.0320	0.0351
Load conditions		(38,001–50,500 cycles) *p* cycle: #=40, $\sigma_{\max}$=69 MPa *q* cycle: #=10, $\sigma_{\max}$=130 MPa							

4.3. Crack growth under variable amplitude loading (VAL)

In this model, the time interval is important to obtain proper results because it determines the size of sub-intervals of numerical integration, and therefore, has an effect on the model accuracy. In the previous two examples, the time interval is given as the interval of data measurement and is employed for damage propagation. For the case of variable amplitude loading, the interval for data measurement and that of damage propagation are different. For example, damage propagation is progressed every cycle based on the physical model. When the damage propagation cycle reaches data measurement cycle, the model parameters are updated based on the particle filter algorithm.

As an example, Huang's model [20] is employed, which is one of several crack growth models under variable amplitude loading:

$$\frac{da}{dN}=C\left[(\Delta K_{eq})^m-(\Delta K_{th})^m\right] \tag{12}$$

$$\Delta K_{eq}=M_R M_P \Delta K,\ \Delta K=F\Delta\sigma\sqrt{\pi a} \tag{13}$$

$$M_R=\begin{cases}(1-R)^{-\beta_1} & (-5\le R<0)\\ (1-R)^{-\beta} & (0\le R<0.5)\\ (1.05-1.4R+0.6R^2)^{-\beta} & (0.5\le R<1)\end{cases},\ R=\frac{\sigma_{\max}}{\sigma_{\min}} \tag{14}$$

$$M_P=\begin{cases}\left(\frac{r_y}{a_{OL}+r_{OL}-a-r_\Delta}\right)^n & (a+r_y<a_{OL}+r_{OL}-r_\Delta)\\ 1 & (a+r_y\ge a_{OL}+r_{OL}-r_\Delta)\end{cases},$$
$$r_y=\alpha\left(\frac{K_{\max}}{\sigma_y}\right)^2 \text{ and } \quad r_{OL}=\alpha\left(\frac{K_{\max}^{OL}}{\sigma_y}\right)^2 \tag{15}$$

where ΔK_{eq} and ΔK_{th}, respectively, are the range of equivalent stress intensity factor and threshold stress intensity factor; M_R and M_P, respectively, are correction factors for the loading ratio and the loading sequence interaction; F is the geometry factor; β and β_1 are shaping exponents; $\sigma_{\max}$ and $\sigma_{\min}$, respectively, are the maximum and minimum stress at every cycle; r_y is the plastic zone size ahead of the crack tip; r_Δ is the increment in the plastic zone size ahead of the crack tip caused by under loading; n is the shaping exponent in the present model; α is the plastic zone size factor; $K_{\max}$ is maximum stress intensity factor; σ_y is tensile yield stress; and OL placed on sub or superscript is the case at which the overload occurs.

The model parameters $(m,C,\Delta K_{th},\beta,n,\sigma_y)$ and standard deviation of noise as well as the damage state a are estimated simultaneously using the measured crack sizes that are obtained from 150 × 170 × 4 (mm) size-specimen test under $\sigma_{\min}=3.5$ MPa and the other given loading conditions, which are listed in Table 3. In Table 3, p cycle and q cycle represent, respectively, the conditions for normal load and overload. Fig. 10 shows the result of crack growth prediction using up to 11 data points (solid dots). Users can also obtain similar results by modifying the code according to the physical model in Eqs. (12)–(15). More information can be found in the website mentioned in Section 5.

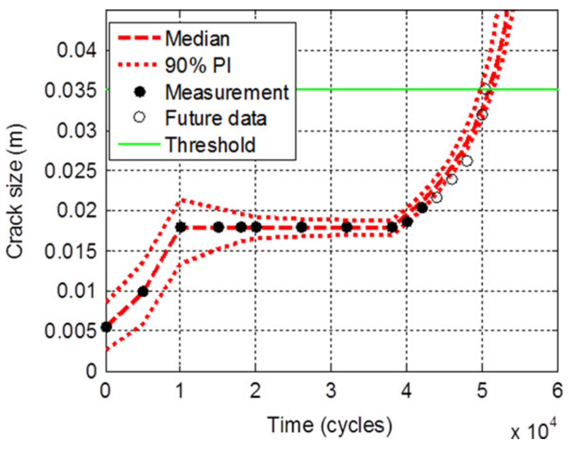


Fig. 10. Crack growth prediction with variable amplitude loading.

5. Conclusions

This paper presents a tutorial for model-based prognostics with a Matlab code. The code is simply constructed with 62 lines using an example of battery degradation, and users can easily modify the code for their specific applications. Also, more practical cases are considered with crack growth examples. This will be helpful for the beginners in prognostics to use the prognostics method for their applications. Toward this aspect, the author has established a website https://sites.google.com/site/dawnan1114/, with the hope to facilitate continuous communication with the people interested in this research.

Acknowledgment

This work was supported by a grant from the International Collaborative R&D Program (0420-2011-0161) of the Korea Institute of Energy Technology Evaluation and Planning (KETEP), funded by the Korean government's Ministry of Knowledge Economy.

Appendix. Matlab code

```
%===== PROBLEM DEFINITION: PARAMETER DEFINITION =============================
WorkName='Battery';                         % work results are saved by WorkName
TimeUnit='weeks';                                             % time unit name
dt=5;                                              % time interval (five weeks)
measuData=[1.0000  0.9351  0.8512  0.9028  0.7754  0.7114  0.6830  0.6147 ...
           0.5628  0.7090]';  % measured data at every time intervals (k1 x 1)
thres=0.3;                                         % threshold - critical value
ParamName=['x'; 'b'; 's'];              % model parameters' name to be estimated
initDisPar=[0.9 1.1; 0 0.05; 0.01 0.1];
                       % probability parameters of initial distribution, p x q
                    % (p: num. of unknown param, q: num. of probability param)
n=5e3;                                                  % number of particles
signiLevel=5;                            % significance level for C.I. and P.I.
%===========================================================================
% % % PROGNOSTICS using PARTICLE FILTER
p=size(ParamName,1);
for j=1:p;                                             %% Initial Distribution
    param(j,:)=unifrnd(initDisPar(j,1),initDisPar(j,2),1,n);
    ParamResul(j,:)=[ParamName(j,:) 'Resul'];
    eval([ParamResul(j,:) '=param(j,:);']);
end;
k1=length(measuData); k=1;                        %% Update Process or Prognosis
if measuData(end)-measuData(1)<0; cofec=-1; else cofec =1; end
while min(eval([ParamResul(1,:) '(k,:)'])*cofec)<thres*cofec;  k=k+1;
        % step1. prediction (prior)
        paramPredi=param;
        for j=1:p; eval([ParamName(j,:) '=paramPredi(j,:);']); end
        paramPredi(1,:)=...%===== PROBLEM DEFINITION: MODEL DEFINITION ======
                       exp(-b.*dt).*x;
        %==================================================================
    if k<=k1                                                % (Update Process)
        % step2. update (likelihood)
        likel=normpdf(measuData(k),paramPredi(1,:),paramPredi(end,:));
        % step3. resampling
        cdf=cumsum(likel)./max(sum(likel));
        for i=1:n;
            u=rand;
            loca=find(cdf >= u); param(:,i)=paramPredi(:,loca(1));
        end;
    else                                                         % (Prognosis)
        param=paramPredi;
    end
        for j=1:p; eval([ParamResul(j,:) '(k,:)=param(j,:);']); end;
        if k>k1;
           eval([ParamResul(1,:) '(k,:)=normrnd(param(1,:),param(end,:));']);
        end
end
% % % POST-PROCESSING
time=[0:dt:dt*(k-1)]';                                      %% RUL Calculation
perceValue=[50 signiLevel 100-signiLevel];
for i=1:n;
    loca=find(eval([ParamResul(1,:) '(:,i)'])*cofec>=thres*cofec);
    RUL(i)=time(loca(1))-time(k1);
end;
RULPerce=prctile(RUL',perceValue);
figure; set(gca,'fontsize',14); hist(RUL,30);           %% RUL Results Display
xlim([min(RUL) max(RUL)]); xlabel(['RUL' ' (' TimeUnit ')']);
titleName=['at ' num2str(time(k1)) ' ' TimeUnit]; title(titleName)
fprintf( '\n  # Percentiles of RUL at %g weeks \n', time(k1))
fprintf('\n   %gprct: %g,  median: %g,  %gprct: %g \n' , perceValue(2), ...
       RULPerce(2), RULPerce(1), perceValue(3), RULPerce(3))
Name=[WorkName ' at ' num2str(time(k1)) '.mat']; save(Name);      %% Work Save
```

References

[1] Daigle M, Goebel K. Multiple damage progression paths in model-based prognostics. In: Proceedings of IEEE aerospace conference, Big Sky, Montana, March 05–12, 2011.

[2] DeCastro JA, Tang L, Loparo KA, Goebel K, Vachtsevanos G. Exact nonlinear filtering and prediction in process model-based prognostics. In: Proceedings of the annual conference of the prognostics and health management society, San Diego, CA, September 27–October 1, 2009.

[3] Luo J, Pattipati KR, Qiao L, Chigusa S. Model-based prognostic techniques applied to a suspension system. IEEE Transactions on Systems Man and Cybernetics-Part A 2008;38(5):1156–68.

[4] Chao H, Youn BD, Wang P, Yoon JT. Ensemble of data-driven prognostic algorithms for robust prediction of remaining useful life. Reliability Engineering and System Safety 2012;103:120–35 Original Research Article.

[5] Chookah M, Nuhi M, Modarres M. A probabilistic physics-of-failure model for prognostic health management of structures subject to pitting and corrosion–fatigue. Reliability Engineering and System Safety 2011;96:1601–10.

[6] Zio E, Maio FD. A data-driven fuzzy approach for predicting the remaining useful life in dynamic failure scenarios of a nuclear system. Reliability Engineering and System Safety 2010;95:49–57.

[7] Vachtsevanos G, Lewis FL, Roemer M, Hess A, Wu B. Intelligent fault diagnosis and prognosis for engineering systems. New Jersey: John Wiley & Sons; 2006.

[8] Kalman RE. A new approach to linear filtering and prediction problems. Journal of Basic Engineering-Transactions of the ASME. 1960;82(1):35–45.

[9] Orchard ME, Vachtsevanos GJ. A particle-filtering approach for on-line fault diagnosis and failure prognosis. Transactions of the Institute of Measurement and Control 2009;31(3–4):221–46.

[10] Zio E, Peloni G. Particle filtering prognostic estimation of the remaining useful life of nonlinear components. Reliability Engineering and System Safety 2011;96(3):403–9.

[11] Li P, Goodall R, Kadirkamanathan V. Parameter estimation of railway vehicle dynamic model using Rao-Blackwellised particle filter. In: Proceedings of the European control conference, Cambridge, UK, September 1-4, 2003.

[12] An D, Choi JH, Schmitz TL, Kim NH. In-situ monitoring and prediction of progressive joint wear using Bayesian statistics. Wear 2011;270(11–12):828–38.

[13] An D, Choi JH, Kim NH. Identification of correlated damage parameters under noise and bias using Bayesian inference. Structural Health Monitoring 2012;11(3):292–302.

[14] Payne SJA. Bayesian approach for the estimation of model parameters from noisy data sets. IEEE Signal Processing Letters 2005;12(8):553–6.

[15] Goebel K, Saha B, Saxena A, Celaya JR, Christophersen J. Prognostics in battery health management. IEEE Instrumentation and Measurement Magazine 2008;11(4):33–40.

[16] Bayes T. An essay towards solving a problem in the doctrine of chances. Philosophical Transactions of the Royal Society of London 1763;53:370–418.

[17] Pitt MK, Silva RS, Giordani P, Kohn R. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. Journal of Econometrics 2012;171(2):134–51.

[18] Paris PC, Erdogan F. A critical analysis of crack propagation laws. Journal of Basic Engineering-Transactions of the ASME. 1963;85:528–34.

[19] Wang X, Rabiei M, Hurtado J, Hoffman P, Modarres M. A probabilistic-based airframe integrity management model. Reliability Engineering and System Safety 2009;94:932–41.

[20] Huang X, Torgeir M, Cui W. An engineering model of fatigue crack growth under variable amplitude loading. International Journal of Fatigue 2008;30(1):2–10.