We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter

Prognostics 102: Efficient Bayesian-Based Prognostics Algorithm in MATLAB

Ting Dong, Dawn An and Nam H. Kim

IntechOpen

Abstract

An efficient Bayesian-based algorithm is presented for physics-based prognostics, which combines a physical model with observed health monitoring data. Unknown model parameters are estimated using the observed data, from which the remaining useful life (RUL) of the system is predicted. This paper focuses on the Bayesian method for parameter estimation of a damage degradation model where epistemic uncertainty in model parameters is reduced with the observed data. Markov-chain Monte Carlo sampling is used to generate samples from the posterior distribution, which are then propagated through the physical model to estimate the distribution of the RUL. A MATLAB script of 76 lines is included in this paper with detailed explanations. A battery degradation model and crack growth model are used to explain the process of parameter estimation, the evolution of degradation and RUL prediction. The code presented in this paper can easily be altered for different applications. This code may help beginners to understand and use Bayesian method-based prognostics.

Keywords: Bayesian method, physics-based prognostics, remaining useful life, MATLAB code, crack growth, battery degradation

IntechOpen

1. Introduction

Structural health monitoring (SHM) [1, 2] is the process of identifying damage and evaluating the safety of a system based on online and/or off-line data. It uses an array of sensors to obtain measurement data that are directly or indirectly related to damage. The statistical analysis of these measurements can help predict the future state of the system and thus improve the safety of the system. SHM can be found in a wide variety of applications such as bridges and dams, buildings, stadiums, platforms, airframes, turbines, etc. Prognostics is an extension of SHM, which is the process of estimating the time beyond which a system can no longer function to meet desired performances [3]. The time, in terms of cycles/hours, remaining to run the system before it fails is called the remaining useful life (RUL).

There are two types of prognostics methods: data-driven and physics-based approaches. The data-driven approaches are advantageous when many training data are available for a complex system, while the physics-based approaches are good when a physical model of damage degradation is available. The physics-based approach is used for prognostics in this paper with a well-defined physics model.

Measured data is used to estimate model parameters, which are then used to predict the RUL.

Recently, many prognostics algorithms have been published in the literature [4–8]. However, many of the proposed algorithms are complex and not easily applicable. This complexity can present a serious hurdle for the beginner. In addition, using commercial programs may not be the best choice in teaching algorithms to students. As a continuation of our educational paper on prognostics algorithm [9], the objective of this paper is to explain the fundamentals of a Bayesian-based prognostics method and demonstrate how to use it using a simple MATLAB code.

The MATLAB code consists of 76 lines, which is further divided into three parts: (1) problem definition; (2) prognostics using the Bayesian method (BM); and (3) post-processing. The program is structured in such a way that the users only need to modify the problem definition part for their own application. This paper shows an example of battery degradation and crack growth models, and attempts to explain prognostics using BM with MATLAB code.

The remaining sections are organized as follows: In Section 2, the overall process of BM is explained; in Section 3, implementation of the code is explained with details using battery degradation example; and in Section 4, modification of the code for crack growth example is described, followed by conclusions in Section 5.

2. Methodology

In this section, a physics-based approach is explained using the procedure shown in **Figure 1**. The theoretical discussions in this section are mainly to help understand the MATLAB implementation in Section 3. The physics-based approach comprises of the following steps: (1) developing or identifying a physical model that describes the degradation of system health, (2) collecting data by operating the system under usage conditions and measuring degradation at a sequence of times/cycles, (3) estimating the model parameters by fitting the measured data, (4) progressing the physical model to the future times/cycles, and (5) predicting the RUL. A statistical

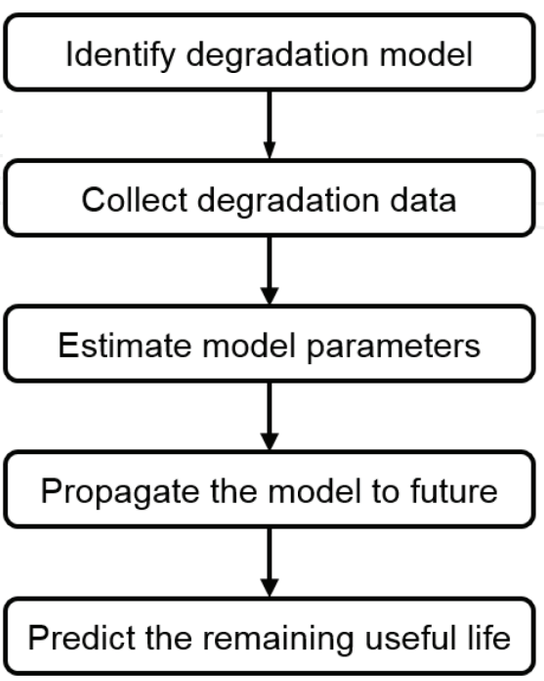


Figure 1.
Flowchart of physics-based prognostics.

inference technique called the Bayesian method (BM) is used in this paper to estimate the model parameters based on measured data. Many other methods, such as particle filter and Kalman filter, also use Bayesian inference to estimate the model parameters. In BM, all model parameters are estimated in the form of a joint probability density function (PDF), whose distribution can be represented using samples. Among various sampling methods, Markov-chain Monte Carlo (MCMC) algorithm is employed to draw samples from the distribution. These samples of model parameters are then substituted in the physical model to calculate the samples of RUL, from which the statistical distribution is evaluated.

2.1 Model definition

In this section, a degradation model of a battery is used to explain the physics-based prognostics algorithm using the Bayesian method. The degradation model of crack growth will also be explained in Section 4. It is expected that the users develop a degradation model for their own application. This section explains the basic requirements of a degradation model.

In a lithium-ion battery, it is well known that the capacity of a secondary cell degrades over cycles in use. Therefore, the capacity can be used as a degradation feature. The degradation feature is an output of the degradation model that shows a monotonic trend as a function of time. The system is considered failed when the degradation feature goes beyond a threshold. In the case of a lithium-ion battery, the failure threshold is defined when the charging capacity fades by 30% of that of a pristine battery. In this paper, the C/1 capacity (capacity at a nominally rated current of 1A) of the battery is used as a degradation feature. Since the C/1 capacity is inversely proportional to the sum of the transfer resistance and the electrolyte resistance, it represents the overall performance of a battery.

Although the degradation process of a battery is complicated, a simple empirical model is available when the usage of the battery is the repetition of fully charging-discharging cycles. In such a case, the degradation model can be written as a function of time only. Since the capacity of a battery degrades over time, the ratio of the capacity compared to that of the pristine battery is expressed by an exponential decaying model as [10]

$$\tilde{y}(t;b) = \exp(-bt) \tag{1}$$

where b is the model parameter, t is the time, and $\tilde{y}(t;b)$ is the relative degradation of the C/1 capacity compared to the pristine capacity. The notation in Eq. (1) is chosen such that the degradation model depends on both the time of usage and model parameters. In general, the degradation model is either monotonically decreased (e.g., the capacity decay of a battery) or monotonically increased (e.g., crack growth).

The main goal of the physics-based prognostics is to predict the degradation behavior using the degradation model. If the model is perfect, then it can be used to find the time t_{EOL} at the end of life from

$$\tilde{y}(t_{\mathrm{EOL}};b) = y_{\mathrm{threshold}} \tag{2}$$

where $y_{\mathrm{threshold}}$ is the failure threshold. Since the relative capacity is used as a degradation feature, the failure threshold is defined at $y_{\mathrm{threshold}} = 0.7$. Let the current time be t_{CUR}, then the RUL can be defined as

$$t_{\mathrm{RUL}} = t_{\mathrm{EOL}} - t_{\mathrm{CUR}} \tag{3}$$

In practice, however, the degradation model is not perfect in the sense that the model form, as well as the model parameters, may not be accurate. The error in the model form can be handled by introducing a model form error and identifying the error using measured data, which would be considered as the out of the scope of this paper. Interested readers can refer to Guan et al. [11].

Once the model form is accepted, the next task is to identify the accurate model parameters. In the case of the battery degradation model in Eq. (1), the parameter b needs to be identified. In most cases, the model parameters are not an intrinsic property but depend on operating conditions and environment. Therefore, these parameters can be different for different batteries and need to be identified for the specific battery of interest. In fact, the major task of physics-based prognostics is to identify the model parameters.

The model parameter b for a specific battery can be identified by measuring the capacity degradation during regular operation. The measuring process is often called health monitoring, where the degradation feature is measured over time. It is possible that the degradation feature can be monitored online. However, for the purpose of prognostics, the real-time continuous monitoring may not be necessary. Therefore, it is often suggested to collect data in a discrete set of times. Then, many different physics-based prognostics algorithms use these data to identify the model parameters so that the degradation model represents the degradation feature the best. For example, nonlinear least-squares method minimizes the error between measured data and the model prediction. Kalman filter, particle filter, and Bayesian methods are using Bayesian inference to estimate the model parameters. Different methods use different assumptions and different numerical approaches. Interested readers are referred to Kim et al. [3] for details of these methods.

If the measured data are accurate, then a small number of measured data should be good enough to estimate the model parameters. In reality, however, most measured data include noise and bias, which make the estimation process difficult. Noise is a random fluctuation of signals due to uncontrollable factors in the measurement environment, while bias is a systematic departure from the average data. If the measurement is repeated, noise can be changed, while the bias may remain the same. The bias can occur because of calibration error of the sensors, but it may also occur due to the model form error. The effect of the model form error can partially be addressed by introducing the bias in the estimation process. Bias can be added in the model as an extra term and estimated in the same way as other parameters. The distribution of estimated bias is a good indicator if the model can represent degradation data well enough. If estimated bias is widely distributed, it means model form error is large. If it is narrowly distributed and the mean is close to 0, it means the model is accurate. Since noise is random, it is important to compensate for its effect in the parameter estimation process. It is obvious that a large level of noise makes the process difficult. Therefore, it is important to keep the signal-to-noise ratio as high as possible. It is also important to understand the statistical characteristics of the noise. In this paper, it is assumed that the noise follows a Gaussian distribution with a zero mean and unknown standard deviation. On the other hand, the effect of bias will not be considered. Therefore, in addition to the unknown model parameters, it is necessary to estimate the unknown standard deviation of noise in data.

Because of noise and bias, it is often expected that a large number of data be required to estimate the model parameters accurately. In prognostics, it is often assumed that N_{data} data are collected from a start time to the current time $t \in [t_0, t_{\text{CUR}}]$. Time does not have to be a physical time; it can also be the number of cycles of operation. Then, the model parameters are estimated using N_{data} data, and future degradation is predicted using the degradation model with the estimated

parameters. In particular, the goal is to accurately predict the end of life in Eq. (2) and the RUL in Eq. (3).

It is important to note that the data should show a significant change in the damage feature over time. In the case of crack growth in Section 4, for example, when the crack size is small, it grows very slowly. Therefore, the measurement data in an early stage do not show a significant change in the crack size. In such a case, the signal-to-noise ratio is too low and it is difficult to estimate the model parameters.

In this paper, instead of measuring the degradation of a real battery, the degradation data are generated based on an assumed true model. This has a couple of advantages. First, since the true model and its model parameters are known, it is possible to evaluate the accuracy of the estimation process and that of the RUL. It also allows us to investigate the effect of noise on the performance of prognostics algorithms. In this paper, the relative capacity data are generated based on Eq. (1) with the true model parameter $b_{\text{true}} = 0.012$. It is assumed that the C/1 capacity of the battery is measured once a week up to the ninth week. In order to simulate the real measurement environment, a Gaussian noise $\varepsilon \sim N(0, 0.005^2)$ is added to the true data. The following MATLAB commands can be used to generate the measured data:

```
>> time=(0:9)';
>> b=0.012;
>> trueData=exp(-b*time);
>> measuData=trueData+0.005*randn(10,1);
```

Once the measurement data are generated, the true model parameters and the information of noise are not used. **Table 1** and **Figure 2** show the true degradation data and simulated measurement data up to the current time $t_{\text{CUR}} = 9$ weeks. Based on the true model, the end of life of the battery is $t_{\text{EOL}} = 29.72$ weeks, and thus, the true RUL should be $t_{\text{RUL}} = 20.72$ weeks.

2.2 Bayesian parameter estimation

Once the measurement data are available, the next step would be to estimate the model parameters. Among many parameter estimation methods, the Bayesian inference is explained in this section. In the following explanation, Θ represents the random variable of the unknown model parameter, and Y represents the random variable of degradation feature. A variable with an upper case denotes a random variable, while a variable with a lower case denotes a realization of the random variable. Bayesian inference estimates the degree of belief in a hypothesis based on collected evidence. Bayes [12] formulated the degree of belief using the identity in conditional probability:

$$P(\Theta \cap Y) = P(\Theta|Y)P(Y) = P(Y|\Theta)P(\Theta) \tag{4}$$

where $P(\Theta|Y)$ is the conditional probability of Θ given Y. In the case of estimating the model parameter using measured data, the conditional probability of Θ when the probability of measured data Y is available can be written as

Time (weeks)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
True degradation	1.000	0.988	0.976	0.965	0.953	0.942	0.931	0.919	0.909	0.898
Measured degradation	0.995	0.983	0.975	0.974	0.942	0.938	0.930	0.920	0.911	0.895

Table 1.
Measurement data (relative capacity) for the battery degradation example.

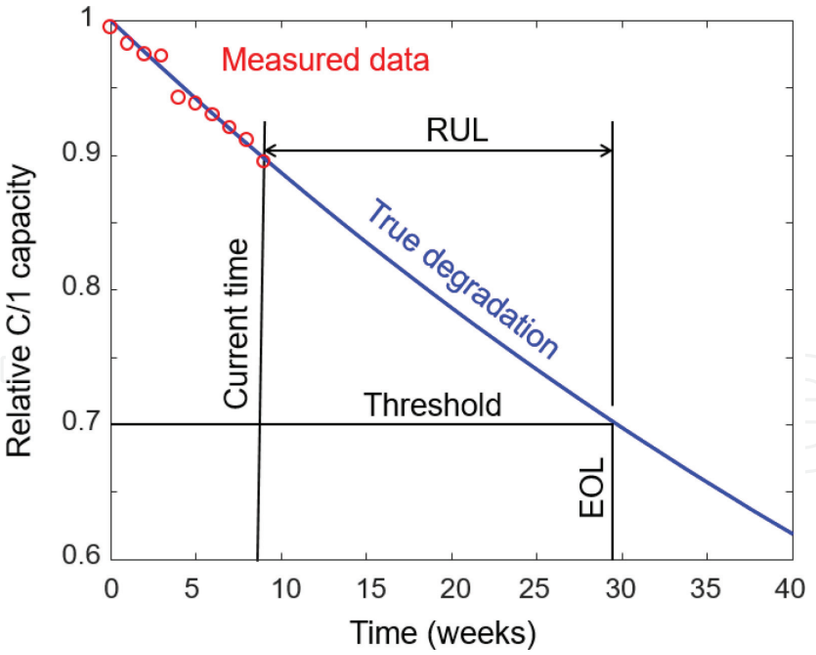


Figure 2.
True degradation curve and measured data for the relative capacity.

$$P(\Theta|Y) = \frac{P(Y|\Theta)P(\Theta)}{P(Y)} \tag{5}$$

where $P(\Theta)$ is the prior probability of parameter Θ, which represents the preexisting knowledge on the parameter. $P(\Theta|Y)$ is the posterior probability of parameter Θ after updating the prior with measurement data Y. $P(Y|\Theta)$ is the likelihood function or the probability of obtaining data Y for a given parameter Θ. The measurement data affect the posterior probability through the likelihood function. The denominator, $P(Y)$, is the marginal probability of Y and acts as a normalizing constant. The above equation can be used to improve the knowledge of $P(\Theta)$ when additional information $P(Y)$ is available.

If the Bayes' theorem in Eq. (5) is going to be used for identifying unknown model parameters, it would be better to express the theorem in the form of a probability density function (PDF) [13], which is used in the present paper. Let $f_\Theta(\theta)$ be a PDF of model parameter Θ. When there are more than one model parameters, $f_\Theta(\theta)$ can be a joint PDF of multiple parameters. If the health monitoring measures a degradation feature Y, the measurement variability can be represented using PDF, $f_Y(y)$. Then, the conditional PDFs between Θ and Y can be related to the joint PDF and the marginal PDF, $f_\Theta(\theta)$ and $f_Y(y)$, as

$$f_{\Theta Y}(\theta,y) = f_\Theta(\theta|Y=y)f_Y(y) = f_Y(y|\Theta=\theta)f_\Theta(\theta) \tag{6}$$

It is obvious that the joint PDF can be written as $f_{\Theta Y}(\theta,y) = f_\Theta(\theta)f_Y(y)$ when Θ and Y are independent, and Bayesian inference cannot be used to improve the probability distribution of $f_\Theta(\theta)$. Similar to Eqs. (5) and (6) can be used for obtaining the Bayesian inference in the form of PDF as [14].

$$f_\Theta(\theta|Y=y) = \frac{f_Y(y|\Theta=\theta)f_\Theta(\theta)}{f_Y(y)} \tag{7}$$

Since the denominator $f_Y(y)$ is a constant and since the integral of $f_\Theta(\theta|Y=y)$ is one from the property of PDF, the denominator in Eq. (7) can be considered as a

normalizing constant. Similar to Eq. (5), $f_{\Theta}(\theta|Y=y)$ is the posterior PDF of parameter Θ that is updated from the prior PDF $f_{\Theta}(\theta)$ with the likelihood function $f_Y(y|\Theta=\theta)$, which is the probability density value of measured data y given model parameter $\Theta=\theta$. The process of updating the posterior distribution $f_{\Theta}(\theta|Y=y)$ of the model parameter using the measured data y is called Bayesian inference.

The Bayesian inference can be extended to the case when many data are available. In general, it is possible that the posterior PDF can be obtained by applying all data simultaneously or by iteratively applying each data at a time. Although two approaches are theoretically equivalent, they end up numerically different methods. For example, the particle filter method uses a single measurement to update the posterior distribution, and the previous posterior distribution is used as a prior distribution for the following measurement. On the other hand, Bayesian method uses all measurement data together to build a single posterior distribution, which is used in this paper. Let us consider that $\mathbf{y}=\left\{y_1,y_2,\ldots,y_{N_{\text{data}}}\right\}$ is the vector or N_{data} measurements. In such a case, the Bayes' theorem can be written as

$$f_{\Theta}(\theta|Y=\mathbf{y})=\frac{1}{K}\prod_{i=1}^{N_{\text{data}}}\left[f_Y(y_i|\Theta=\theta)\right]f_{\Theta}(\theta) \tag{8}$$

where K is the product of all marginal PDFs. However, it can be considered as a normalizing constant to make the integration of the posterior PDF to be one. It is noted that the total likelihood function is the product of the likelihood functions of individual data, which is then multiplied by the prior PDF followed by normalization to yield the posterior PDF.

In contrast to the traditional least-squares method, the Bayes' theorem can estimate not only the best values of parameters but also the uncertainty structure of the identified parameters. Since these uncertainty structures are derived from that of the prior distribution and likelihood function, the uncertainty of the posterior distribution is directly related to that of the likelihood and the prior distribution.

In the Bayesian method, it is assumed that the users know the prior distribution of model parameters and the distribution type of measurement noise. In this paper, it is assumed that the prior distribution is given as a uniform distribution with a lower- and upper-bound. It is also assumed that the measurement noise is a Gaussian distribution; that is $\varepsilon\sim N(0,s^2)$, where s is the standard deviation of noise. However, users can change these assumptions easily. For example, the case when noise in data follows a lognormal distribution is considered in the crack growth problem in Section 4. In most cases, since the standard deviation of noise is unknown, it should be a part of unknown model parameters. In the case of the battery model, therefore, the vector of unknown model parameters is defined as $\Theta=\{b,s\}$. By assuming that the two model parameters are statistically independent, the prior joint PDF of the two parameters can be defined as

$$f_{\Theta}(\theta)=f(b)\times f(s),\quad f(b)\sim U(0,0.05),\ f(s)\sim U\left(10^{-5},0.1\right) \tag{9}$$

Once the prior distribution is determined, it is necessary to build the likelihood function using the measurement data and to yield the posterior distribution shown in Eq. (8). The meaning of the likelihood function is the PDF value of obtaining the measured data y_k for given model parameters $\theta=\{b,s\}$. Since the measured data are fixed, the likelihood function is a function of model parameters, which makes the likelihood function different from the PDF. If the model prediction is close to the measured data, then the likelihood is large, while the likelihood is small when the two values are significantly different. In order to build the likelihood, it is

necessary to measure degradations at different times. Since the measured degradation data (y_k, t_k), $k = 1, 2, ..., N_{\text{data}}$ are given at discrete times, the degradation model is also evaluated at the same discrete times as $\tilde{y}_k(b) = \tilde{y}(t_k; b), k = 1, 2, ..., N_{\text{data}}$. Since the times between the measurement and the model are synchronized, $\tilde{y}_k(b)$ is only a function of model parameter b. The measured data y_k include the random noise that is governed by s, while the model prediction $\tilde{y}_k(b)$ depends on b. Then, the likelihood function of the k-th measured data can be defined as

$$f_Y(y_k|\theta) = \frac{1}{s\sqrt{2\pi}} \exp\left[-\frac{1}{2s^2}\left(y_k - \tilde{y}_k(b)\right)^2\right], \quad k = 1, 2, ..., N_{\text{data}} \tag{10}$$

As shown in Eq. (8), the likelihoods of multiple data can be multiplied to obtain the posterior distribution. With N_{data} data, $\mathbf{y} = \left\{y_1, y_2, ..., y_{N_{\text{data}}}\right\}$, the posterior joint PDF can be calculated by multiplying all likelihood functions and the prior PDF as

$$f_\Theta(\theta|\mathbf{y}) = \frac{1}{Ks^{N_{\text{data}}}} \exp\left[-\frac{1}{2s^2}\sum_{k=1}^{N_{\text{data}}}\left(y_k - \tilde{y}_k(b)\right)^2\right] f_\Theta(\theta) \tag{11}$$

where K is again a normalizing constant.

2.3 Markov chain Monte Carlo sampling

Bayesian parameter estimation in Eq. (11) shows the functional expression of the posterior joint PDF of unknown model parameters. When the prior and posterior are conjugate, the posterior distribution can be expressed in the form of a standard probability distribution. In general cases, however, the posterior distribution can be expressed as a product of complex functions, such as the posterior PDF shown in Eq. (11).

The posterior PDF is then used to calculate the degradation trend and predict the RUL. For complex nonlinear models, it is difficult to propagate uncertainty in the parameters to the degradation model. Instead, samples of model parameters are generated from the posterior distribution, and the degradation model with the threshold in Eq. (2) is used to propagate these samples to calculate the samples of the end of life, and thus, the samples of the RUL in Eq. (3). Therefore, it is important to generate samples that follow the posterior distribution of parameters.

In general, the inverse cumulative distribution function (CDF) method is the easiest way of generating samples from a non-standard probability distribution, but it requires the functional expression of CDF, not PDF. For practical engineering applications, it is likely that the posterior PDF may be different from a standard probability distribution, or the posterior PDF is complicated due to the complex correlation structures between parameters. In such a case, sampling-based methods can be used to generate samples of parameters. There are many sampling methods, such as the grid approximation [15], rejection sampling [16], importance sampling [17], and the Markov Chain Monte Carlo (MCMC) method [18]. In this paper, the MCMC method using the Metropolis-Hastings (MH) algorithm is employed. MCMC is a simulation technique used to estimate quantities of interest by sampling consecutive random variables wherein the future state depends only on the current state [19].

The MCMC sampling method uses a Markov chain model in a random walk, where the distribution of the next sample depends only on the current sample (see **Figure 3**). As the algorithm generates more and more samples, the samples more closely approximate the posterior PDF. Specifically, Starting with an arbitrary initial sample (current sample), a new candidate sample is drawn from a proposal

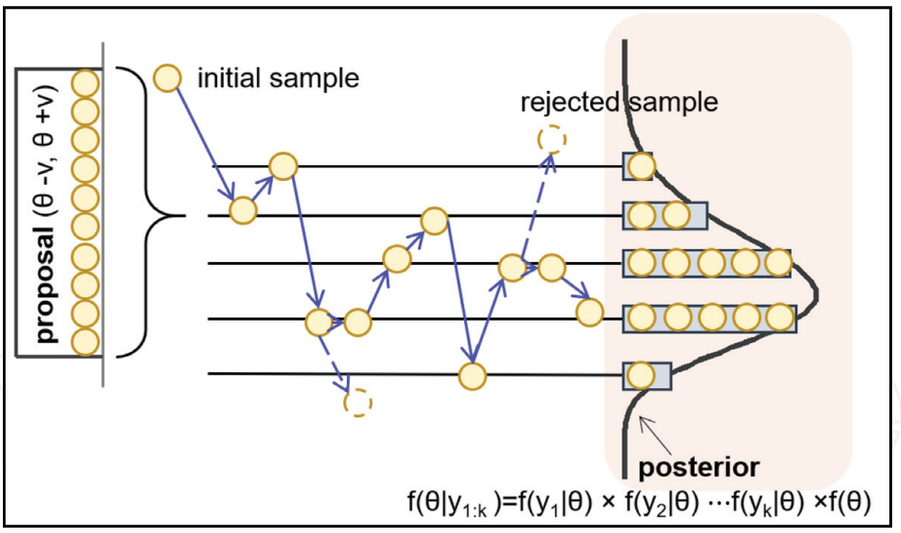


Figure 3.
Markov chain Monte Carlo sampling using random walk.

distribution centered at the current sample. In this paper, a uniformly distributed proposal distribution is used. Therefore, it is expected that the users provide the initial sample of parameters and the width of the proposal distribution. At i-th iteration, it is expected that the current sample $\theta^{(i-1)}$ is available, and the new candidate sample θ^* is drawn from the following proposal distribution that is uniformly distributed centered at $\theta^{(i-1)}$:

$$g\left(\theta^*, |\theta^{(i-1)}\right) \sim U\left[\theta^{(i-1)} - \mathbf{w}, \theta^{(i-1)} + \mathbf{w}\right] \tag{12}$$

where $\mathbf{w}$ is the user-provided width of the proposal distribution. It is noted that the proposal distribution is symmetric; that is $g\left(\theta^*, |\theta^{(i-1)}\right) = g\left(\theta^{(i-1)}, |\theta^*\right)$.

Once the candidate sample is generated, it is either accepted as a new sample or rejected based on an acceptance criterion. When accepted, the candidate sample is added to a new sample and used in the next iteration. When rejected, the candidate sample is discarded, and the current sample is reused in the next iteration. In the original MH algorithm, it is suggested to use a function that is proportional to the posterior distribution for the acceptance/rejection test. In this paper, however, the posterior distribution is directly used as its evaluation is not computationally expensive. Since the proposal distribution is symmetric, the following acceptance ratio can be defined:

$$Q\left(\theta^{(i-1)}, \theta^*\right) = \frac{f_\Theta\left(\theta^*|\mathbf{y}\right)}{f_\Theta\left(\theta^{(i-1)}|\mathbf{y}\right)} \tag{13}$$

The acceptance ratio compares the posterior probability of the new candidate sample against that of the current sample. If the candidate sample has a higher probability than that of the current sample; i.e., $Q\left(\theta^{(i-1)}, \theta^*\right) > 1$, then it is always accepted as a new sample. When $0 < Q\left(\theta^{(i-1)}, \theta^*\right) < 1$; that is, the probability of the candidate sample is lower than that of the current sample, the acceptance is determined based on the ratio. A high acceptance ratio has a more probability to be accepted, while a low ratio is occasionally accepted. This can be achieved by generating a sample from a uniform distribution, $u \sim U[0, 1]$, and the candidate sample is accepted if $u < Q\left(\theta^{(i-1)}, \theta^*\right)$; otherwise, it is rejected. Intuitively, this is why this algorithm works, and returns samples that follow the desired distribution $f_\Theta\left(\theta|\mathbf{y}\right)$. In **Figure 3**, two

Algorithm: Metropolis-Hastings algorithm

Initial sample $\theta^{(0)}$ and posterior $f_{\Theta}(\theta^{(0)}|\mathbf{y})$

For i = 1 to Ns

  Sample from proposal dist. $\theta^* \sim g(\theta^*|\theta^{(i-1)})$ and posterior $f_{\Theta}(\theta^*|\mathbf{y})$

  Acceptance ratio: $Q(\theta^{(i-1)}, \theta^*) = \dfrac{f_{\Theta}(\theta^* \mid \mathbf{y})}{f_{\Theta}(\theta^{(i-1)} \mid \mathbf{y})}$

  Acceptance probability u ~ U(0,1)

  If $u < Q(\theta^{(i-1)}, \theta^*)$

    Accept the candidate sample: $\theta^{(i)} = \theta^*$

  otherwise

    Reject the candidate sample: $\theta^{(i)} = \theta^{(i-1)}$

  end

end

Figure 4.
Metropolis-Hastings algorithm for generating samples from a posterior distribution.

dashed circles mean that these candidate samples are not selected according to the criterion. In such a case, the current sample is selected again. This process is repeated as many times as necessary until a sufficient number of samples are obtained. **Figure 4** summarizes the MCMC sampling procedure using the MH algorithm.

The performance of MCMC sampling depends on the initial sample and the selection of the proposal distribution. A too-narrow proposal distribution can yield destabilization by not fully covering the posterior distribution, while a too-wide distribution can yield many duplications in sampling result by not accepting new samples. In addition, if the initial sample is located far away from the posterior distribution, many iterations (samples) will be required to converge to the posterior distribution. To prevent the effect of inaccurate initial samples, an initial portion of the samples can be discarded in estimating the posterior distribution, which is called the burn-in. In this paper, the first 20 percent of the samples are discarded as a burn-in.

2.4 Prognostics

Once the samples of parameters are obtained based on the posterior distribution, the future damage state and the RUL can be predicted by substituting the samples of parameters in the degradation model and estimating the RUL using Eqs. (2) and (3). In general, since the degradation model is a nonlinear implicit function of time, solving for $t_{\mathrm{EOL}}^{(i)}$ with a given sample $b^{(i)}$ in Eq. (2) may require an iterative process. Instead, in this paper, the degradation model is evaluated at a set of discrete future times, and then, the end of life is calculated using a simple interpolation. More specifically, let the set of discrete times is defined as

$$\text{time} = [t_0 \;\; t_1 \;\; \cdots \;\; t_{\mathrm{CUR}} \;\; \cdots \;\; t_{\mathrm{end}}] \tag{14}$$

Then, the measurement data are available between t_0 and t_{CUR}. Bayesian parameter estimation in the previous section uses the measurement data between t_0 and

t_{CUR} to estimate the posterior PDF of model parameters. Using the estimated model parameters, the degradations in the future times between t_{CUR} and t_{end} are calculated in the prediction stage. If two consecutive degradations cross the threshold; that is,

$$\left(\tilde{y}_k(b) - y_{\text{threshold}}\right) \cdot \left(\tilde{y}_{k+1}(b) - y_{\text{threshold}}\right) \leq 0 \tag{15}$$

then t_{EOL} exists between t_k and t_{k+1}, which can be found by a simple linear interpolation. When the set of futures times do not include the end of life, it can extrapolate based on the trend of data. It is also possible that the degradation model never reaches the threshold; that is, the system has an infinite of life. In such a case, the sample is deleted from the calculation. Once the samples of the end of life are obtained, the samples of RUL can be calculated using Eq. (3).

Once the samples of RUL are available, the confidence interval and/or the prediction interval is used to evaluate the accuracy or precision of the RUL. The confidence interval represents how good the RUL is. Therefore, the confidence interval of 95% means that the true RUL will be within this interval with the probability of 95%. That is, the confidence interval tells us about the likely location of the true RUL. In the case of RUL samples, the 95% confidence interval can be calculated by taking the lower 2.5 percentile and the upper 2.5 percentile from the samples. On the other hand, the prediction interval shows the possible location of the next sample. Knowing that the next sample will have additional randomness from the predicted RUL, the prediction interval is calculated by adding additional randomness to the data. In practice, the RUL estimation is important for scheduling maintenance. Therefore, only the lower confidence/prediction bound is of interest in the practical application. **Figure 5** shows a representative result of prognostics, which shows the statistical distribution and the confidence interval of the RUL.

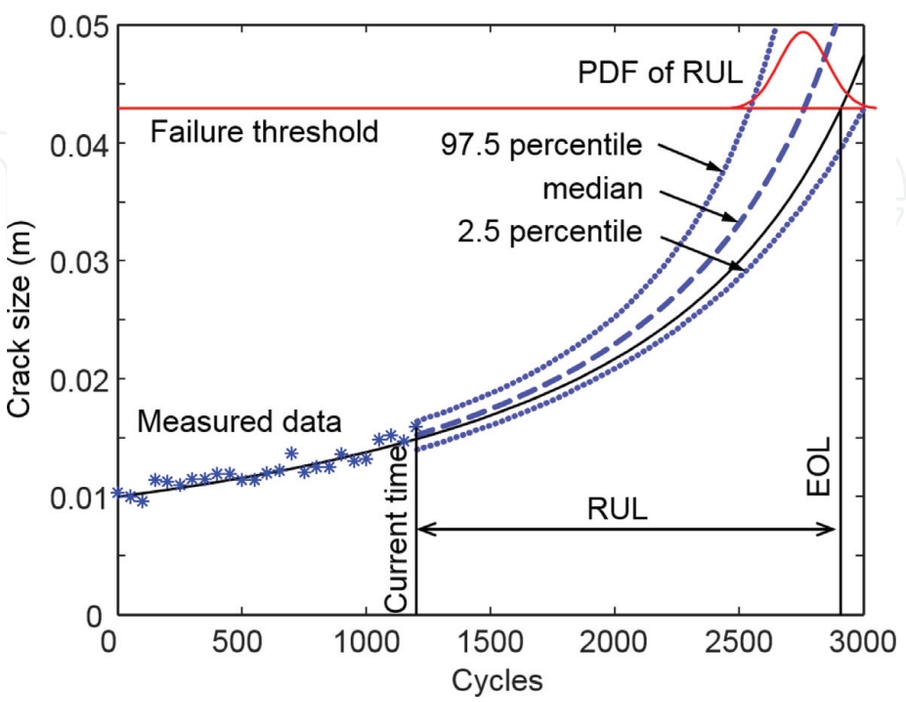


Figure 5.
Statistical distribution and the confidence interval of the remaining useful life.

3. MATLAB implementation

In this section, MATLAB implementation of prognostics using the Bayesian method is discussed. In the following explanation, 'line' or 'lines' in a parenthesis indicated the number of the line of the code in Appendix. The code is divided into three parts: (1) Problem Definition (lines 2–15, 65–67) (2) Bayesian parameter estimation and MCMC sampling (lines 16–29, 60–76) (3) Post-processing for displaying results (lines 40–57). Only the Problem Definition part needs to be changed for different applications. Detailed explanations are given in the subsequent sections with an example of battery degradation.

It is expected that the MATLAB script is saved as a file with the name of '**BM.m**', which has two input arguments, **para0** and **weigh** (line 1). The first argument, **para0**, is the initial sample of model parameters, and **weigh** is the width **w** of the proposal distribution in Eq. (12). The size of each array should be the same as the number of model parameters. The following is an example of calling the code in the command window of MATLAB:

```
samplResul=BM([0.011 0.02]', [0.001 0.003]');
```

In the above MATLAB command, **para0 = [0.011 0.02]'** is the initial sample of b and s, and **weigh = [0.001 0.003]'** is the width of proposal distribution of b and s. Since the convergence and accuracy depend on these two variables, it is suggested to try with different values. Since the users know the prior distribution of the model parameters, it is a good practice to start with the mean of the prior distribution as an initial sample. If the code ran successfully, it will return the samples of model parameters in the **samplResul** array and will generate two plots. The first plot is the trace of MCMC samples, and the second plot is the histogram of the RUL.

3.1 Problem definition (lines 2–15, 65–67)

The problem definition means defining the degradation equation using model parameters and time/cycle. All known parameters, as well as the initial estimate of unknown parameters, parameter names, and model data, need to be defined. The problem definition consists of two parts: parameter definition and model definition. For parameter definition (lines 2–15), '**Battery**' is used as a **WorkName** (line 3). The capacity is measured every week, so **TimeUnit** is '**weeks**' (line 4). In line 5, **time** is an array of discrete times in the units of **TimeUnit**. Measurements and predictions will be done at these times. The relative C/1 capacity data is stored as **measuData** (lines 6–7), which has 10 weeks of measurement; that is, $N_{\text{data}} = 10$ (k1 in line 18). These data correspond to the first 10 times in **time** array. Since time starts from 0, the 10th time corresponds to 9 weeks, which is the current time; that is, $t_{\text{CUR}} = 9$ week. The degradation will be predicted for future times that are from 10 to 50 weeks. The measurement data are generated using the MATLAB script in Section 2.1. It is noted that due to random noise, the users may experience different realizations of measurement data.

In line 8, the degradation threshold $y_{\text{threshold}}$ is defined using variable '**thres**' with the value of **0.7**. **ParamName** in line 9 is the name of model parameters: model parameter '**b**' and standard deviation of noise '**s**'. Since the parameter name will actually be used in the model, it is important to use the actual name of variables here. The number of parameters is stored in the variable '**p**' in line 17. It is noted that the last parameter should be the standard deviation of noise '**s**'. **prioDisPar** stores the information of the prior distribution of model parameters, as given in

Eq. (9). When a uniform prior distribution is used, each row contains the lower- and upper-bounds of the distribution.

During MCMC sampling, the number of samples **Ns** is set to **5000** (line 12). Since the sampling process takes many iterations to converge, the initial 20% of the samples are discarded in calculating the posterior distribution by command **burnIn = 0.2** (line 13). In order to keep 5000 samples, **Ns/(1-burnIn)** = 6250 samples (line 21) are generated first, and then, **nBurn** = 1250 samples (line 30) are discarded.

Since the RUL is represented by **Ns** samples, the confidence interval or the prediction interval is often used to support the decision-making process. **signiLevel** (line 14) is the significance level of this interval in percentage. When **signiLevel** = 5, the code will return the lower 5 percentile, median, and the upper 5 percentile. Following is the sample output from **BM.m**:

```
# Percentiles of RUL at 9 cycles
5prct: 18.7182, median: 20.381, 95prct: 22.1576
```

For the model definition, the degradation model $\tilde{y}(t;b)$ in Eq. (1) is defined in lines 65–67. In this equation, t is the time of measurement, and b is the model parameter as defined in line 9. The model equation needs to be defined in such a way that component-by-component operations are possible. This is because the time and model parameters can be an array of samples.

3.2 Bayesian parameter estimation with MCMC (lines 16–31)

In the Bayesian parameter estimation process, the posterior distribution is expressed in terms of the product of the prior distribution and the likelihoods of all measured data. In the MATLAB code **BM.m**, the degradation model and the posterior distribution are calculated in the function **BMappl** (lines 60–76). First, the parameter samples in **param** are assigned to the variables using the **eval** function (line 62–64):

```
for j=1:size(param,1)
eval([ParamName(j,:) '=param(j,:);']);
end
```

In the case of the battery example, this command is equivalent to.

```
b = param(1,:);
s = param(2,:);
```

This is why the **ParamName** in line 9 must have the same name with the actual variable. Then, these parameters are used to calculate the degradation model with given time t (line 66).

If measurement data (**measuData**) is empty, then **BMappl** only calculates the degradation model with given parameter samples at a given time t. This corresponds to propagating the degradation model using parameter samples to the future time for prognostics. If measurement data are provided (lines 71–74), then **BMappl** calculates the values of the posterior distribution at the parameter samples. In this case, the time should be given as an array of N_{data} components from the start time to the current time. The posterior distribution in Eq. (11) (line 74) is the multiplication of the prior distribution (line 71) in Eq. (9) with the likelihood of all measured data (lines 72–73) in Eq. (10). The calculated posterior distribution from **BMappl** is used for calculating the acceptance ratio (line 23).

MCMC sampling using the MH algorithm starts with the initial sample that is provided by the users (line 19) and the value of the posterior PDF (line 20). In the

loop of MCMC sampling (lines 21–29), a candidate sample is randomly generated from a uniform distribution, centered at the current sample and the width of $\pm$**w** (line 22). The value of posterior PDF for the candidate sample is also calculated (line 23). If the acceptance ratio in Eq. (13) is greater than a randomly generated number u, then the candidate sample is accepted as a new sample (lines 25–26). Otherwise, the current sample and its posterior PDF are kept as a new sample and PDF. Once the MCMC sampling loop is over, the first 20% of the samples are discarded as a burn-in process (line 31). At the end of Bayesian parameter estimation, **samplResul** array contains **Ns** samples of model parameters.

3.3 Remaining useful life prediction (lines 32–39)

Once the samples of model parameters are obtained based on the posterior distribution, they can be used to find the RUL, which is the time when the degradation prediction reaches the threshold. First, for all samples in **samplResul**, the degradation is predicted in the future times between $t_{\text{CUR}} =$ k1 and t_{end}.

```
for k=1:length(time(k1:end))
  [degrPreCon(k,:),~]=BMappl(samplResul,ParamName,time(k1-1+k),[],[]);
end
```

Once the degradations in the future times are calculated, MATLAB function **interp1** is used to find the time when $\tilde{y}(t_{\text{EOL}}; b) = y_{\text{threshold}}$. Once t_{EOL} is found, the RUL can be calculated using Eq. (3).

```
for i=1:Ns
 RUL(i)=interp1(degrPreCon(:,i),time(k1:end),thres,'pchip') - time(k1);
end
```

The option 'pchip' in **interp1** uses a shape-preserving piecewise cubic interpolation, which preserves C^1-continuity.

3.4 Postprocessing (lines 40–58)

In the postprocessing stage, the results given in samples are interpreted in terms of statistical quantities or in the form of graphs. First, in the **RUL** array, those components that have an infinite life should be removed (line 41). Then, the confidence intervals of [5%, median, 95%] are calculated from the **RUL** array and stored in **rulPerce** (line 42–43).

The MATLAB code plots two figures. The first figure plots the trace of MCMC samples (lines 45–50) as shown in **Figure 6(a)**. This trace shows the quality of MCMC samples. If samples are distributed randomly and symmetrically around the mean with a constant bound, then it means that the samples are stabilized and well represent the posterior distribution. If the trace of samples shows an irregular behavior as shown in **Figure 6(b)**, the samples may not represent the posterior PDF properly. This can happen when the initial sample was far away from the mean and when the width of the proposal distribution is too narrow or too wide.

The second plot is the histogram of RUL (lines 51–53) as shown in **Figure 7**. At the end of the code, the confidence intervals of [5%, median, 95%] are printed on the command window (lines 54–56). All variables are saved in the computer file so that they can be loaded to the memory for further analysis (line 57). The name of the saved database is "WorkName at t_{CUR}.mat". For example, in the battery case, the saved database name is "Battery at 9.mat".

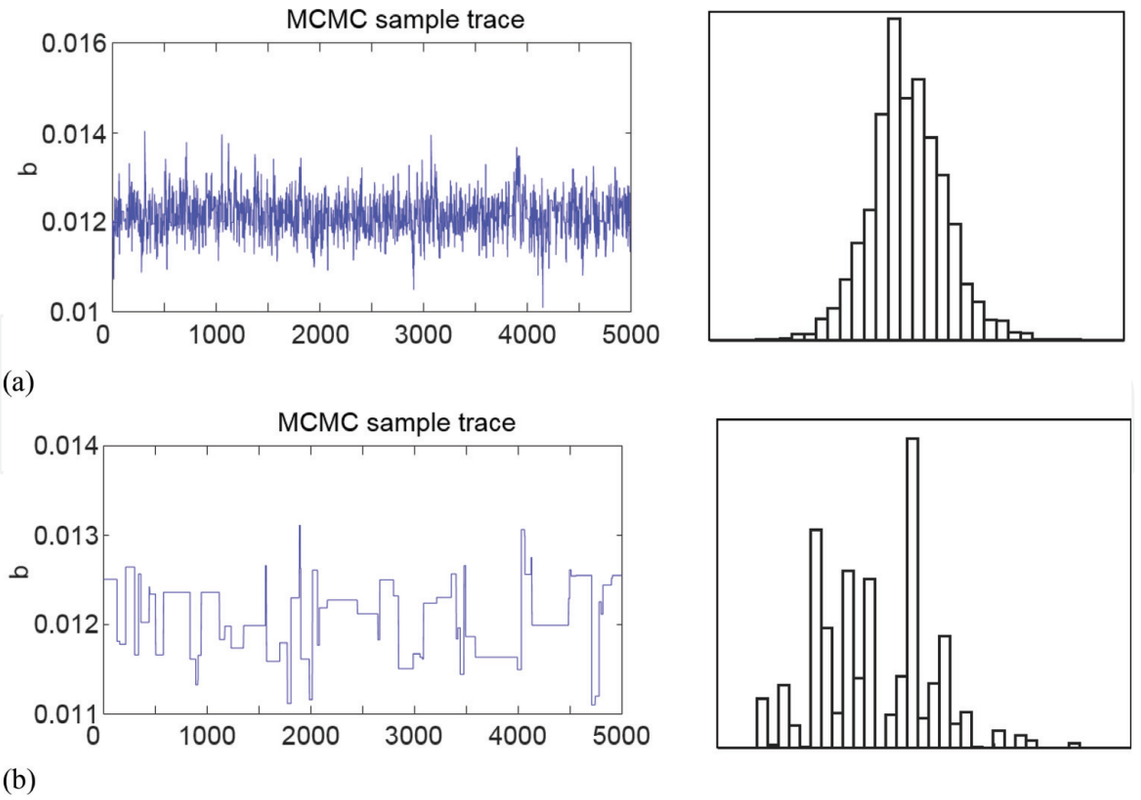


Figure 6.
Trace of samples from MCMC sampling (a) proper samples and (b) improper samples.

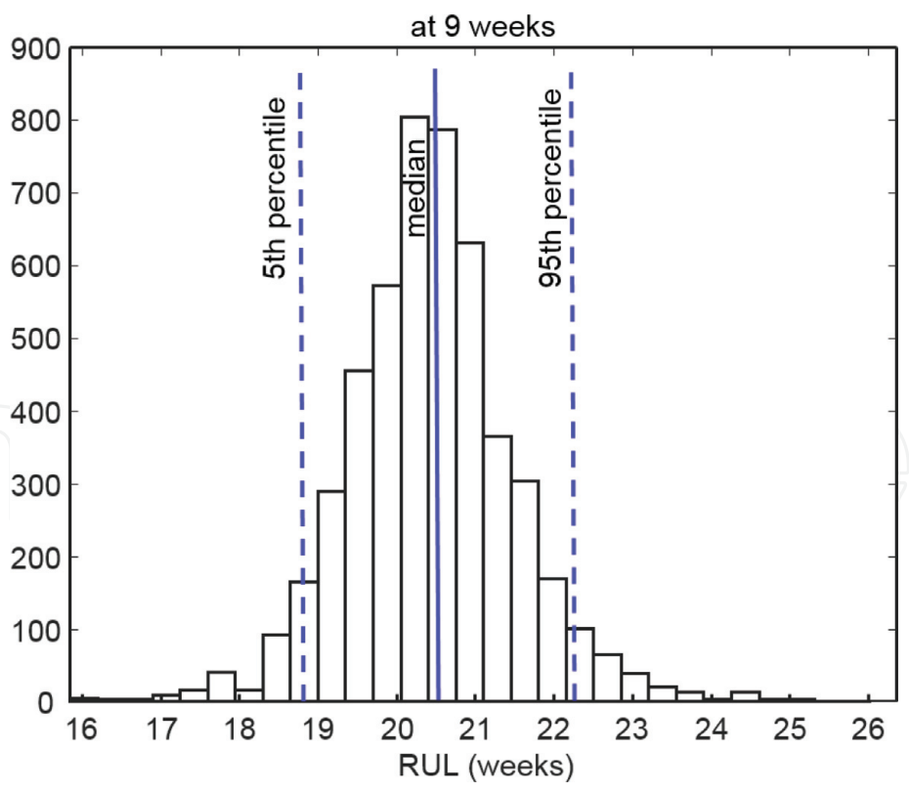


Figure 7.
Histogram of the remaining useful life.

Although the MATLAB code plots two figures, it is possible that the users can plot different figures using the saved database. After calling the **BM.m** function, the saved database has to be loaded to the memory using the following commands:

```
clear; clc; load('Battery at 9.mat')
```

In addition to the trace of samples shown in **Figure 6**, it is possible to plot the histogram of model parameters using the same samples. The following commands plot the histograms of all model parameters.

```
figure; % histogram of parameters
for j=1:p;
  subplot(1,p,j);
  hist(samplResul(j,:),30);
end
```

When the true degradation model is known, it is possible to compare the predicted degradation with the true one. The following MATLAB script plots the median and confidence intervals of the predicted degradation along with the true degradation and the threshold.

```
figure;
degraTrue=exp(-0.012.*time);
degraPI=prctile(degraPredi',perceValue)';
plot(time,degraTrue,'k'); hold on;
plot(time(1:k1),measuData,'*b');
plot(time(k1:end),degraPI(:,1),'-r');
plot(time(k1:end),degraPI(:,2:3),':r');
plot([0 time(end)],[thres thres],'g');
xlabel('weeks'); ylabel('Relative C/1 capacity');
```

The degradation curves up to 50 weeks are shown in **Figure 8** for the battery example. The true degradation with $b = 0.012$ is shown with the black curve. The red curves show 5, 50 (median) and 95 percentiles of the predicted degradation, which are caused by **signiLevel** = 5 (line 14). The plot also shows the threshold (green line) and measurement data (blue asterisk marks). Based on the true model, the end of life of the battery is $t_{\mathrm{EOL}} = 29.72$ weeks, and thus, the true RUL should be $t_{\mathrm{RUL}} = 20.72$ weeks. The prediction shows that the median of RUL is

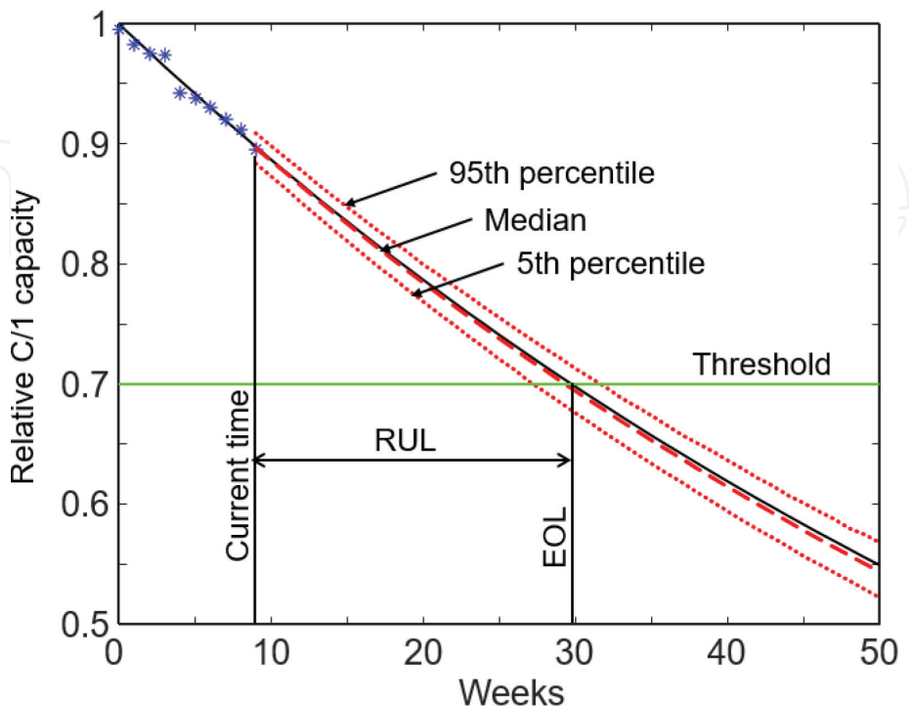


Figure 8.
Comparison of the predicted degradation with the true degradation.

20.38 weeks, which is close to the true RUL. In addition, the 90% confidence interval is about 3.4 weeks; that is, the uncertainty in the prediction is about 17%.

4. Application to crack growth prognostics

The code can be modified easily by the users for various applications. In this section, an example of crack growth is used to explain how the MATLAB code **BM.m** can be modified.

4.1 Model definition: crack growth

In fatigue crack growth, the failure criterion is given in terms of the crack size. Therefore, it would be appropriate to use the size of crack as a degradation feature. In this case, the degradation feature monotonically increases, while it was monotonically decreased for the battery example. Assuming that a through-the-thickness center crack exists in an infinite plate under mode-I loading condition, the rate of fatigue crack growth can be expressed using the Paris-Erdogan model as

$$\frac{\mathrm{d}a}{\mathrm{d}N} = C(\Delta K)^m \tag{16}$$

where a is the half crack size, N is the number of cycles, m and C are model parameters, $\Delta K = \Delta\sigma\sqrt{\pi a}$ is the range of stress intensity factor, and $\Delta\sigma$ is the stress range. It is assumed that time is the number of fatigue loading cycles. For the consistent notation, the crack size and cycles are replaced with $\tilde{y} = a$ and $t = N$ in the following explanation. Since the degradation model requires the crack size as a function of time and model parameters, Eq. (16) can be integrated to obtain the following degradation model:

$$\tilde{y}(t; m, C) = \left[tC\left(1 - \frac{m}{2}\right)\left(\Delta\sigma\sqrt{\pi}\right)^m + a_0^{1-\frac{m}{2}}\right]^{\frac{2}{2-m}} \tag{17}$$

The system is under fatigue loading with the range of stress being $\Delta\sigma = 75\text{MPa}$ at each cycle. It is assumed that the health monitoring is performed every 50 cycles to measure the crack size y_k until the current time $t_{\text{CUR}} = 1,200$ cycles, and the initial size of the crack is $a_0 = 0.01\text{m}$. Similar to the battery degradation example, the measurement data are simulated by adding random noise to the true crack size. First, the true crack size data are generated at every 50 cycles using Eq. (17) with $m_{\text{true}} = 3.8$ and $C_{\text{true}} = 1.5 \times 10^{-10}$. The measured crack size data are then generated by adding Gaussian noise $\varepsilon \sim N(0, s^2), s = 0.0005\text{m}$ to the true crack sizes. The measured crack size data are used to identify three model parameters, $\theta = \{m, \ln(C), \ s\}$. In the Paris-Erdogan model, the y-intercept C is very small but changes its magnitude by several orders. Therefore, it would be better to identify logarithm of C. For RUL calculation, the critical crack size is determined as 0.043 m.

For the Bayesian method, it is necessary to define the prior distribution and the likelihood function. In the battery example, it was assumed that noise in data follows a normal distribution. However, when the distribution type of measurement noise is unknown, it is possible that the likelihood function might be different from the true noise distribution. The same is true for the prior/initial distribution. Therefore, it would be a good exercise to study the effect of different distribution types by changing the MATLAB codes. In this example, the lognormal distribution is employed for the likelihood function as:

$$f(y_k|\theta) = \frac{1}{y_k \zeta_k \sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{\ln y_k - \eta_k}{\zeta_k}\right)^2\right], \quad k = 1, \ldots, N_{\text{data}} \tag{18}$$

where $\zeta_k = \sqrt{\ln\left[1 + \left(s/\tilde{y}_k\right)^2\right]}$ and $\eta_k = \ln\left(\tilde{y}_k\right) - 0.5(\zeta_k)^2$ are the standard deviation and mean of a lognormal distribution, respectively. In the above equation, $\tilde{y}_k(\theta)$ is the model prediction from Eq. (17) at time t_k with given model parameters m and C.

Also, the prior distribution of each parameter is assumed as a normal distribution as $f(m) \sim N(4, 0.2^2)$, $f(\ln C) \sim N(-22.33, 0.5^2)$, and $f(s) \sim N\left(5 \times 10^{-4}, \left(1 \times 10^{-4}\right)^2\right)$. Therefore, the joint prior distribution can be obtained from the independence assumption as

$$f_{\Theta}(\theta) = f(m) \times f(\ln C) \times f(s) \tag{19}$$

The posterior distribution can be obtained by multiplying the prior distribution in Eq. (19) with the likelihood function in Eq. (18).

4.2 Modifying the code

For the crack growth example, the code in Appendix needs to be changed as follows. First, the problem definition part in lines 2–15 is replaced with the following code:

```
%===== PROBLEM DEFINITION 1 (Required Parameters)==================
WorkName='Crack';                    %work results are saved by WorkName
TimeUnit='cycles';                                  % time unit name
time=(0:50:3600)'; % time including both at measurement and at prediction
measuData=[1.03 1.00 0.96 1.14 1.13 1.10 1.15 1.15 1.19 1.19 1.14 ...
           1.14 1.20 1.22 1.37 1.21 1.25 1.25 1.36 1.30 1.32 1.48 ...
           1.52 1.47 1.59]'*0.01;
thres=0.043;                                  % threshold - critical value
ParamName=['m'; 'C'; 's'];          % model parameters' name to be estimated
prioDisPar=[4 0.2; -22.33 .5; 5E-4 1E-4];   % parameter prior distributions
Ns=10000;                          % num. of samples for MCMC simulation
burnIn=0.2;                                       % burn-in fraction
signiLevel=2.5;                        % significance level for C.I. and P.I.
%==================================================================
```

A total of $N_{\text{data}} = 25$ measurement data are provided up to $t_{\text{CUR}} = 1,200$ cycles, and the degradation is predicted until $t_{\text{end}} = 3,600$ cycles. Since the prior distributions are assumed a normal distribution, the first column of **prioDisPar** is the mean, and the second column is the standard deviation.

Next, the model definition part in lines 65–67 is replaced with the following code:

```
%===== PROBLEM DEFINITION 2 (model equation)=======================
a0=0.01; dsig=75; coef=1-m/2;
degraModel=(t.*exp(C).*coef.*(dsig*sqrt(pi)).^m + a0.^coef).^(1./coef);
loca=imag(degraModel)~=0; degraModel(loca)=real(degraModel(loca));
%==================================================================
```

DOI: http://dx.doi.org/10.5772/intechopen.82781

Initially, a fatigue crack grows slowly and then grows rapidly just before becoming unstable. The crack growth model in Eq. (17) is only valid when the crack growth is stable. Normally the threshold $y_{\text{threshold}}$ is set before the crack becomes unstable. When the crack becomes unstable, Eq. (17) yield a complex number. Therefore, the last line of the model definition code identifies if a prediction yields a complex number and converts it to a real number by ignoring the complex part.

In addition to the problem definition part, the posterior distribution part also needs to be modified because instead of a uniform distribution, a normal distribution is used for the prior distribution. Also, lognormal distributions are used instead of normal distributions for the likelihood function. The posterior distribution part in lines 71–74 needs to be modified as follows:

```
prior=prod(normpdf(param,prioDisPar(:,1),prioDisPar(:,2)));
likel=1;
for k=1:length(measuData)
  zeta=sqrt(log(1+(s./degraModel(k)).^2)); eta=log(degraModel(k))-0.5*zeta.^2;
  likel=lognpdf(measuData(k),eta,zeta).*likel;
end
poste=likel.*prior;
```

4.3 Results

Similar to the battery example, the MATLAB code can be used to plot the trace of MCMC sampling and the histogram of model parameters and RUL. Using a similar code provided in Section 3.4, the degradation trend could also be plotted. For example, **Figure 5** shows the predicted degradation with the true degradation. Even if the median (1553 cycles) has a relatively large error with the true RUL (1709 cycles), the 95% confidence interval covers the true RUL.

An important information that was not discussed before is the correlation between model parameters. It is well known that the Paris-Erdogan model parameters, m and C, are strongly correlated [20]. Therefore, it would be beneficial to plot the MCMC samples in the parameter space. The following MATLAB script plots the MCMC samples of the parameters.

```
clear; clc; load('Crack at 1200.mat');
[m, C]=meshgrid((3.4:0.009:4.3)',(-24:0.025:-21.5));
for i=1:101; for j=1:101
   para=[m(i,j); C(i,j); 0.0005];
   [~,post(i,j)]=BMappl(para,ParamName,time(1:k1),measuData,prioDisPar);
end; end
plot(samplResul(1,:),samplResul(2,:),'.','color',[0.5 0.5 0.5]); hold on
contour(m,C,real(post));
plot(3.8,log(1.5E-10),'pk','markersize',14);
```

Figure 9 shows the MCMC samples in the parameter space along with the exact value of the parameters (star marker). The figure also shows the contour of the joint posterior PDF based on the grid method. It can be observed that the MCMC samples represent the joint posterior PDF well, and the joint PDF covers the true parameter values. However, due to a strong correlation between the two Paris-Erdogan model parameters, the joint PDF shows a narrow but long tail. Any combination of model parameters along the correlation line may yield a similar damage growth trend.

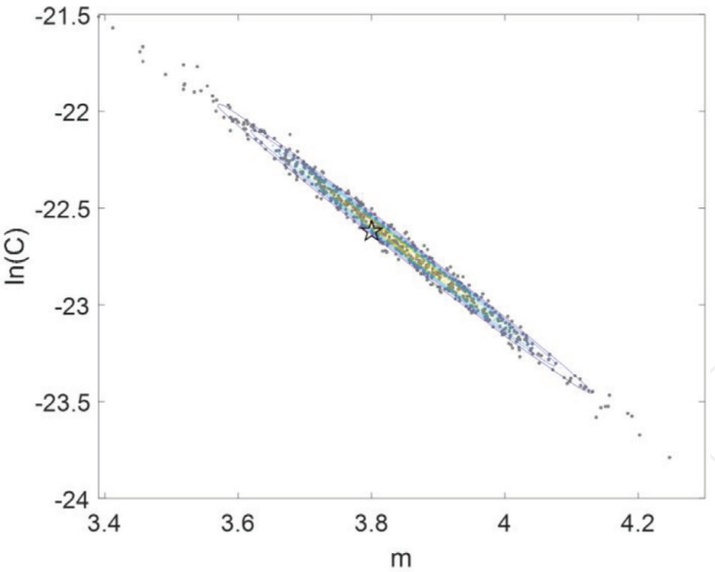


Figure 9.
Correlation between the two Paris-Erdogan model parameters.

5. Conclusions

This paper presents a Bayesian-based prognostics algorithm with a MATLAB code. This code is constructed with simply 76 lines in the case of a battery degradation example. Users can easily modify this code as per their own application. As an example of code modification, the case of crack growth model is also presented. The paper also provided several MATLAB scripts to help plot the degradation curve and correlation between multiple parameters.

A. Appendix

```
function samplResul=BM(para0,weigh)
%===== PROBLEM DEFINITION 1 (Required Parameters)================
WorkName='Battery';                    % work results are saved by WorkName
TimeUnit='weeks';                                  % time unit name
time=(0:50)';   % time including both at measurement and at prediction
measuData=[0.9951 0.9826 0.9750 0.9736 0.9424 0.9381 ...
           0.9300 0.9203 0.9114 0.8952]';   % measured data at time (0:9)
thres=0.7;                                  % threshold - critical value
ParamName=['b'; 's'];              % model parameters' name to be estimated
prioDisPar=[0 0.05;                        % parameter prior distributions
            1e-5 0.1];
Ns=5000;                          % num. of samples for MCMC simulation
burnIn=0.2;                                        % burn-in fraction
signiLevel=5;                       % significance level for C.I. and P.I.
%================================================================
%%% Bayesian parameter estimation with MCMC
p=size(ParamName,1);                                % num. of parameters
k1=length(measuData);                                % num. of data
sampl(:,1)=para0;                      % Initial samples of parameters
[~,jPdf0]=BMappl(para0,ParamName,time(1:k1),measuData,prioDisPar);
for i=2:Ns/(1-burnIn)                                %% MCMC Process
```

DOI: http://dx.doi.org/10.5772/intechopen.82781

```
para1(:,1)=para0+weigh.*(2*rand(p,1)-1);  % sample from proposal dist.
[~,jPdf1]=BMappl(para1,ParamName,time(1:k1),measuData,prioDisPar);
if rand<(jPdf1/jPdf0) && jPdf1>0                % acceptance criterion
para0=para1;
jPdf0=jPdf1;
 end
 sampl(:,i)=para0;                                % new MCMC sample
end
nBurn=Ns/(1-burnIn)-Ns;                % No. of effective MCMC samples
samplResul=sampl(:,nBurn+1:end);            % Final Sampling results
%%% RUL prediction
for k=1:length(time(k1:end))                % degradation prediction
 [degrPreCon(k,:),~]=BMappl(samplResul,ParamName,time(k1-1+k),[],[]);
 degraPredi(k,:)=degrPreCon(k,:)+normrnd(0,samplResul(end,:));
end
for i=1:Ns                                          % RUL prediction
 RUL(i)=interp1(degrPreCon(:,i),time(k1:end),thres,'pchip') - time(k1);
end
%%% POST-PROCESSING
Index=isnan(RUL); RUL(Index)=[];
perceValue=[50 signiLevel 100-signiLevel];          % median & confi-
dence intervals
rulPerce=prctile(RUL,perceValue);                   %percentiles of RUL
figure(1);
for j=1:p % plotting MCMC sample trace
 subplot(p,1,j); % for all model parameters
 plot(samplResul(j,:));
 ylabel(ParamName(j,:));
 title('MCMC sample trace');
end
figure(2); set(gca,'fontsize',14); hist(RUL,30);            % RUL histogram
xlim([min(RUL) max(RUL)]); xlabel(['RUL' ' (' TimeUnit ')']);
titleName=['at ' num2str(time(k1)) ' ' TimeUnit]; title(titleName)
fprintf( '\n # Percentiles of RUL at %g cycles \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);  % save work
end
%
function [degraModel, poste]=BMappl(param,ParamName,t,measuData,
prioDisPar)
% Evaluate the degradation model or posterior PDF
for j=1:size(param,1)
eval([ParamName(j,:) '=param(j,:);']);
end
%===== PROBLEM DEFINITION 2 (model equation)===================
degraModel=exp(-b.*t);
%=============================================================
if isempty(measuData)
poste=0;
else
prior=prod(unifpdf(param,prioDisPar(:,1),prioDisPar(:,2)));      % prior
likel=(1./s).^length(measuData) ...                          % likelihood
```

```
73            .*exp(-0.5./s.^2.*norm(measuData-degraModel)^2);
74 poste=likel.*prior; % posterior
75 end
76 end
```

IntechOpen

IntechOpen

Author details

Ting Dong, Dawn An and Nam H. Kim*
University of Florida, Gainesville, FL, USA

*Address all correspondence to: nkim@ufl.edu

IntechOpen

© 2019 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. CC BY

References

[1] Giurgiutiu V. Structural Health Monitoring with Piezoelectric Wafer Active Sensors. 2nd Edition. Waltham, MA, USA: Academic Press; 2014

[2] Sohn H, Farrar CR, Hemez FM, Czarnecki JJ, Shunk DD, Stinemates DW, et al. "A Review of Structural Health Monitoring Literature: 1996–2001," Report Number LA-13976-MS. Los Alamos, NM: Los Alamos National Laboratory; 2004

[3] Kim NH, An D, Choi J-H. Prognostics and Health Management of Engineering Systems: An introduction. Switzerland: Springer International Publishing; 2017. DOI: 10.1007/978-3-319-44742-1

[4] Si XS, Wang W, Hu CH, Zhou DH. Remaining useful life estimation—A review on the statistical data driven approaches. European Journal of Operational Research. 2011;**213**:1-14

[5] 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. Mechanical Systems and Signal Processing. 2014;**42**(1–2):314-334

[6] Saha B, Goebel K, Christophersen J. Comparison of prognostic algorithms for estimating remaining useful life of batteries. Transactions of the Institute of Measurement and Control. 2009;**31**(3–4):293-308

[7] Xing Y, Williard N, Tsui K-L, Pecht M. A comparative review of prognostics-based reliability methods for Lithium batteries. In: Prognostics and System Health Management Conference, Shenzhen, China; 24-25 May 2011

[8] Zhang J, Lee J. A review on prognostics and health monitoring of Li-ion battery. Journal of Power Sources. 2011;**196**:6007-6014

[9] An D, Choi J-H, Kim NH. Prognostics 101: A tutorial for particle filter-based prognostics algorithm using Matlab. Reliability Engineering and System Safety. 2013;**115**:161-169. DOI: 10.1016/j.ress.2013.02.019

[10] Goebel KB, Saha A, Saxena JR, et al. Prognostics in battery health management. IEEE Instrumentation and Measurement Magazine. 2008;**11**(4):33-40

[11] Guan X, Jha R, Liu Y. Model selection, updating and averaging for probabilistic fatigue damage prognosis. Structural Safety. 2011;**33**(3):242-249

[12] Bayes T, Price R. An essay towards solving a problem in the doctrine of chances. By the late rev. Mr. Bayes, communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S. Philosophical Transactions of the Royal Society of London. 1763;**53**:370-418. DOI: 10.1098/rstl.1763.0053

[13] An D, Choi J-H, Kim NH, Pattabhiraman S. Fatigue life prediction based on Bayesian approach to incorporate field data into probability model. Structural Engineering and Mechanics. 2011;**37**(4):427-442

[14] Athanasios P, editor. Probability, Random Variables, and Stochastic Processes. New York: McGraw-Hill; 1984

[15] Gelman A, Carlin JB, Stern HS, et al., editors. Bayesian Data Analysis. New York: Chapman & Hall; 2004

[16] Casella G, Robert CP, Wells MT. Generalized Accept-Reject Sampling Schemes. Lecture Notes-Monograph Series. Vol. 45. Beachwood: Institute of Mathematical Statistics; 2004. pp. 342-347

[17] Glynn PW, Iglehart DL. Importance sampling for stochastic simulations.

Management Science. 1989;**35**(11): 1367-1392

[18] Andrieu C, Freitas DN, Doucet A, et al. An introduction to MCMC for machine learning. Machine Learning. 2003;**50**(1):5-43

[19] An D, Kim NH, Choi J-H. Practical options for selecting data-driven or physics-based prognostics algorithms with reviews. Reliability Engineering & System Safety. 2015;**133**:223-236. DOI: 10.1016/j.ress.2014.09.014

[20] An D, Choi J-H, Kim NH. Identification of correlated damage parameters under noise and bias using Bayesian inference. Structural Health Monitoring. 2012;**11**(3):293-303. DOI: 10.1177/1475921711424520

IntechOpen