A Tutorial for Model-based Prognostics Algorithms based on Matlab Code

Dawn An[1], Joo-Ho Choi[2], and Nam Ho Kim[3]

[1,2]*Korea Aerospace University, Goyang-City, Gyeonggi-do, 412-791, Korea*
skal34@nate.com
jhchoi@kau.ac.kr

[3] *University of Florida, Gainesville, FL, 32611, USA*
nkim@ufl.edu

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 in model-based prognostics. 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.

1. INTRODUCTION

Although many prognostics methods have been presented in literature (Daigle & Goebel, 2011; DeCastro et al., 2009; Luo et al., 2008), 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.

Among different prognostics methods, the model-based approach is considered, which assumes that a physical model describing the behavior of damage or degradation is available. In this approach, the model parameters are often unknown and need to be identified as a part of the prognostic process. The method combines the model with measured data to identify the model parameters and predict its behavior under future loading conditions. There are several methods to estimate model parameters, such as the Kalman filter (KF) (Kalman, 1960), Particle filter (PF) (Orchard & Vachtsevanos, 2007; Zio & Peloni, 2011; Li et al., 2003), and Bayesian method (BM) (An et al., 2011; An et al., 2012; Payne, 2005). 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.

This paper is 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 Figure 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.

The model parameters are estimated using an algorithm such as PF by integrating the damage model with the

Dawn An et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

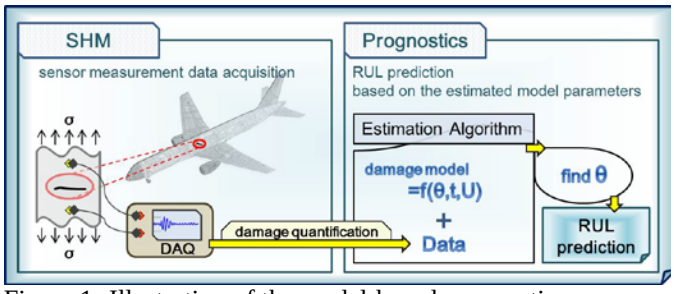


Figure 1. Illustration of the model-based prognostics process

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 (Goebel et al., 2008):

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

where a, b are model parameters, t is the time or cycles, and λ is internal battery performance, such as electrolyte resistance R_E or transfer resistance R_{CT}. The internal battery performance is normally observed instead of capacity. Also, there is a relationship between $R_E + R_{CT}$ and C/1 capacity (capacity at nominally rated current of 1A); $R_E + R_{CT}$ is typically inversely proportional to the C/1 capacity. Therefore, in this paper, the 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)*.

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

Table 1. Measurement data for battery degradation problem

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 (Bayes, 1763):

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

where $\boldsymbol{\Theta}$ is a vector of unknown parameters, $\mathbf{z}$ is a vector of observed data, $L(\mathbf{z}|\boldsymbol{\Theta})$ is the likelihood or the PDF value of $\mathbf{z}$ conditional on the given $\boldsymbol{\Theta}$, $p(\boldsymbol{\Theta})$ is the prior PDF of $\boldsymbol{\Theta}$, and $p(\boldsymbol{\Theta}|\mathbf{z})$ is the posterior PDF of $\boldsymbol{\Theta}$ 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 (Orchard & Vachtsevanos, 2007; Zio & Peloni, 2011). The general process of PF is based on the state transition function f and the measurement function h (Zio & Peloni, 2011; Li et al., 2003):

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

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

where k is the time step index, x_k is the damage state, $\boldsymbol{\theta}_k$ is a vector of model parameters, z_k is measurement data. ν_k and ω_k are, respectively, process and measurement noise. In this paper, 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 ν_k is ignored because it can be handled through the uncertainty in model

* For simplicity, it is assumed that a = 1 is given.

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 $\Theta = [x, \theta(=[b]), \sigma]^T$, 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 Figure 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 (k th) 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 $p(\Theta_k)$ in Figure 2. Next is the updating step (lines 31-33), which is related to the likelihood of measurement data $L(z_k \mid \Theta_k)$ in the figure. Assuming ω_k is normally distributed, the likelihood of the measurement can be expressed as (line 33):

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

In Eq. (6), the PDF value of z_k at the given i th samples of the unknown parameters $\Theta = x, b, \sigma$ corresponds to the weight of the i th samples; the weight is proportional to the magnitude of the PDF value, which is expressed as the length of the vertical bar in Figure 2. Finally, the samples with high or low weight are duplicated or eliminated, respectively, at the resampling step (lines 34-39). Among

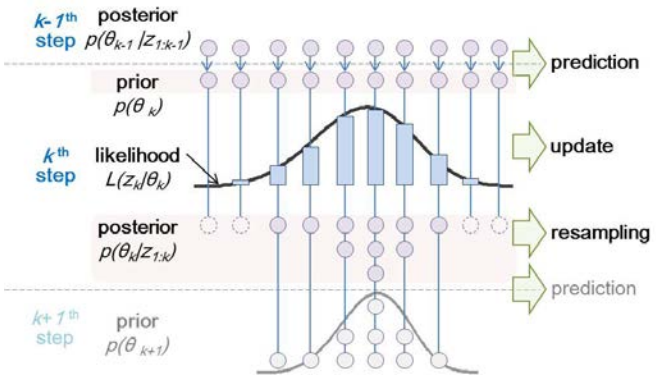


Figure 2. Illustration of the PF process

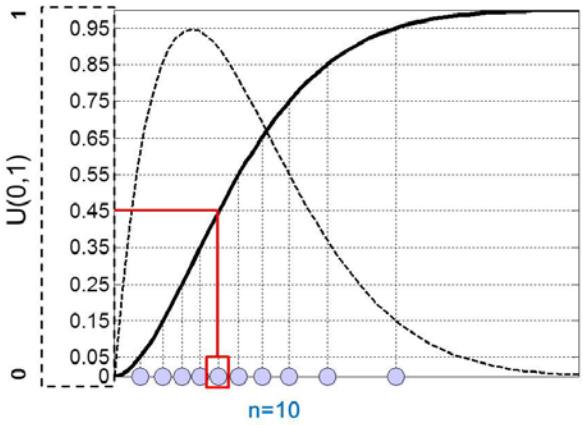


Figure 3. Illustration of resampling method

several methods, the inverse CDF method (Zio & Peloni, 2011) is used, which is illustrated in Figure 3. Firstly, a CDF is constructed from the likelihood function in Eq. (6) (line 35), which is illustrated as solid curve in the figure. Next, a random value is generated from $U(0,1)$ (line 37), which becomes a CDF value (e.g., 0.45 in the figure). Finally, a sample of the parameter having the CDF value is found (line 38), which is marked by a rectangle in the figure. 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 $p(\Theta_k \mid \mathbf{z}_{1:k})$ in Figure 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 Figure 4 (lines 24-30, 40-47). In the figure, the two dashed curves and the PDF shape, respectively,

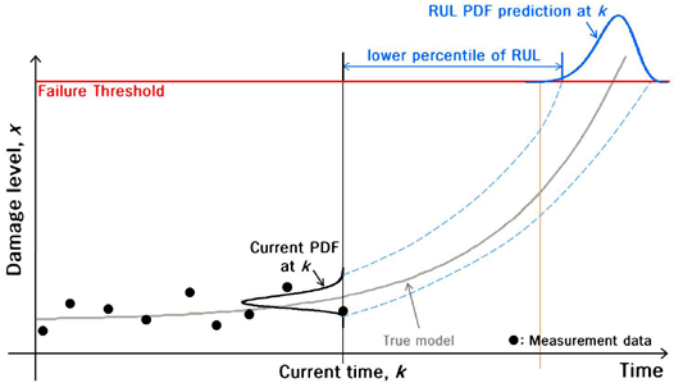


Figure 4. Illustration of RUL prediction

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 Figure 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

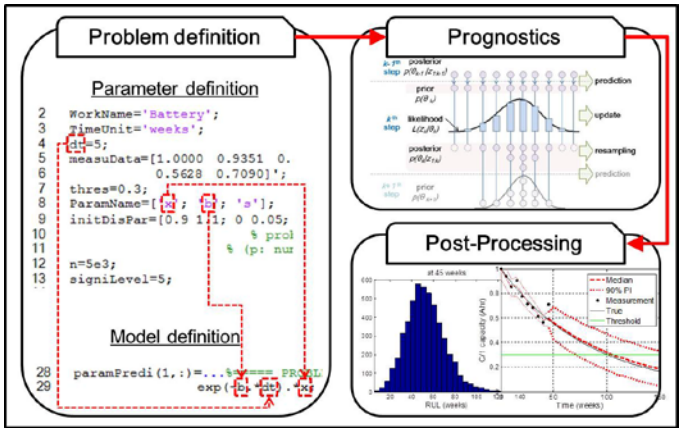


Figure 5. Block diagram of the code

is a $k1\times1$ vector. According to the definition of failure threshold in Section 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 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:

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

Equation (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, 1,000~5,000 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. (Pitt et al., 2012).

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)

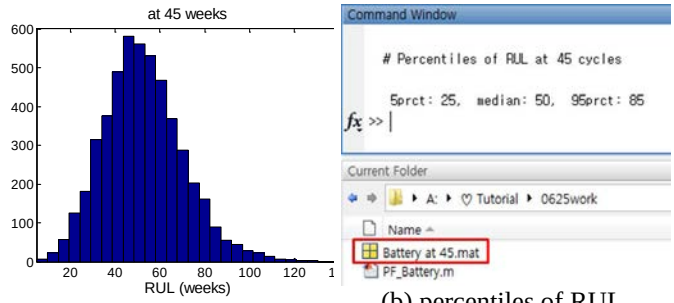
The prognostics process is composed of three steps: (1) the initial distributions of the parameters (lines 16-21), (2) estimation process (lines 25-39, 43), and (2) prognosis (lines 24-30, 40-47). In terms of the code usage, there are two issues that can be considered according to users' intention: the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

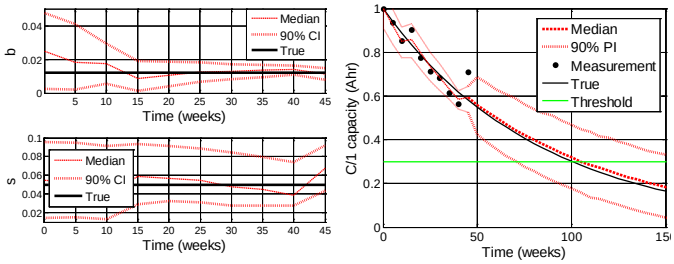
Once problem definition is completed and the code is implemented, distribution and its percentiles of RUL at the current time can be displayed. Figure 6 shows RUL results at 45 weeks after the updating process is progressed up to $k=9$ (see Table 1; $k=9$ corresponds to ***k1=10*** in the script since the initial, $k=0$ is stored in ***k1=1***). Figure 6(b) shows 5, 50 (median), and 95 percentiles, which are caused by ***signiLevel=5*** (line 13). Also, the solid box in the figure represents that the 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



(a) RUL distribution (b) percentiles of RUL distribution

Figure 6. Visual results obtained from the code: battery degradation example



(a) trace of parameter update (b) C/1 capacity prediction

Figure 7. Visual results obtained from the additional code: battery degradation example

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

Table 2. Measurement data for crack growth problem

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 Eq. (1) or Eq. (5). The additional visual results are shown in Figure 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.

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 (Paris & Erdogan, 1963) as

$$\frac{\mathrm{d}a}{\mathrm{d}N}=C\left(\Delta K\right)^{m},\ \Delta K=\Delta\sigma\sqrt{\pi a} \tag{8}$$

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:

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

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 $\Delta\sigma$ =78MPa, which is listed in Table 2. First the true crack size data are generated using Eq. (9) with $m_{\text{true}}=3.8$ and $C_{\text{true}}=1.5\times10^{-10}$. The measured crack size data are then generated by multiplying noise, which is lognormally distributed with standard deviation of $0.001/a_k$ (m). For the

RUL calculation, the critical crack size is determined as 0.0463m. More specific information for crack growth problem is in the literature by An et al. (2012).

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

- initial distribution:

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

- likelihood function:

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

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

4.2. Modifying the Code for the Crack Growth Problem

4.2.1. Problem Definition

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

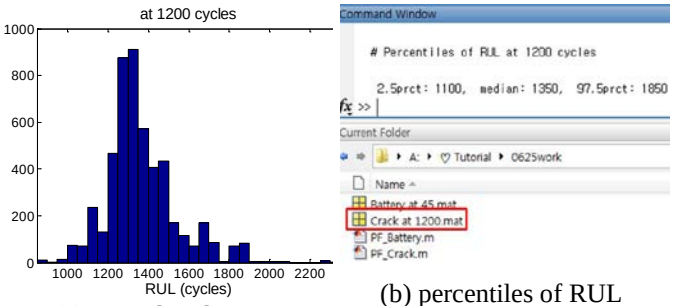
- line 2: ***WorkName='Crack';***
- line 3: ***TimeUnit='cycles';***
- line 4: ***dt=50;*** (or ***dN=50***, but should be matched with line 29)
- lines 5-6: ***measuData=[0.0119 0.0103 0.0118 0.0095 0.0085 0.0122 0.0110 0.0120 0.0113 0.0122 0.0110 0.0124 0.0117 0.0138 0.0127 0.0115 0.0135 0.0124 0.0141 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 is 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 (0.001m) 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;***
- 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:

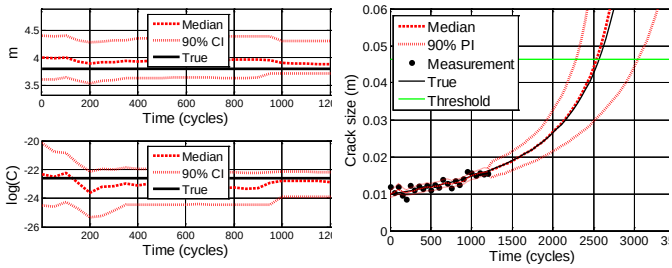
- line 18: ***param(j,:)=normrnd(initDisPar(j,1),initDisPar(j,2),1,n);***
- line 33: ***sigl=sqrt(log(1+(paramPredi(end,:)./paramPredi(1,:)).^2)); mul=log(paramPredi(1,:))-0.5*sigl.^2; likel=lognpdf(measuData(k),mul,sigl);***
- line 45: ***sigl=sqrt(log(1+(param(end,:)./param(1,:)).^2));mul=log(param(1,:))-0.5*sigl.^2;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



(a) RUL distribution (b) percentiles of RUL distribution

Figure 8. Visual results obtained from the code: crack growth example



(a) trace of parameter update (b) crack growth prediction

Figure 9. Visual results obtained from the additional code: crack growth example

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 Figure 8. Also, users can obtain Figure 9 using the stored results of the parameters; ***aResul***, ***mResul***, ***CResul*** which can be known by typing ***ParamResul*** in the command window.

5. CONCLUSION

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 a crack growth example. This will be helpful for the beginners in prognostics to use the prognostics method for their applications.

ACKNOWLEDGEMENT

This work was supported by the International Cooperation of the Korea Insitute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Korea government Ministry of Knowledge Economy (No. 20118520020010).

REFERENCES

An, D., Choi, J. H., Schmitz, T. L., & Kim, N. H., (2011). In-Situ Monitoring and Prediction of Progressive Joint Wear using Bayesian Statistics, *Wear*, vol. 270(11-12), pp. 828-838.

An, D., Choi, J. H., & Kim, N. H., (2012). Identification of Correlated Damage Parameters under Noise and Bias Using Bayesian Inference. *Structural Health Monitoring*, vol. 11(3), pp. 292-302.

Bayes, T., (1763). An Essay towards solving a problem in the doctrine of chances, *Philosophical Transactions of the Royal Society of London*, vol. 53, pp. 370-418.

Daigle, M., & Goebel, K., (2011). Multiple Damage Progression Paths in Model-based Prognostics. *Aerospace Conference, 2011 IEEE*.

DeCastro, J. A., Tang, L., Loparo, K. A., Goebel, K., and Vachtsevanos, G., (2009). Exact Nonlinear Filtering and Prediction in Process Model-based Prognostics. *Annual Conference of the Prognostics and Health Management Society*.

Goebel, K., Saha, B., Saxena, A., Celaya, J. R., & Christophersen, J., (2008). Prognostics in Battery Health Management. *IEEE Instrumentation and Measurements Magazine*, vol. 11(4), pp. 33-40.

Kalman, R. E., (1960). A New Approach to Linear Filtering and Prediction Problems. *Transaction of the ASME—Journal of Basic Engineering*, vol. 82(1), pp. 35-45.

Luo, J., Pattipati, K.R., Qiao, L., & Chigusa, S., (2008). Model-based Prognostic Techniques Applied to a Suspension System. *IEEE Transactions on System, Man and Cybernetics*, vol. 38(5), pp. 1156-1168.

Orchard, M. E., & Vachtsevanos, G. J., (2007). A Particle Filtering Approach for On-Line Failure Prognosis in a Planetary Carrier Plate. *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 7(4), pp. 221-227.

Paris, P. C. & Erdogan, F., (1963). A Critical Analysis of Crack Propagation Laws, *ASME Journal of Basic Engineering*, vol. 85, pp. 528-534.

Payne, S. J., (2005). A Bayesian Approach for the Estimation of Model Parameters from Noisy Data Sets. *IEEE Signal Processing Letters*, vol. 12(8), pp. 553-556.

Zio, E., & Peloni, G., (2011). Particle Filtering Prognostic Estimation of the Remaining Useful Life of Nonlinear Components. *Reliability Engineering and System Safety*, vol. 96(3), pp. 403-409.

Li, P., Goodall, R. & Kadirkamanathan, V., (2003). Parameter Estimation of Railway Vehicle Dynamic Model using Rao-Blackwellised Particle Filter, *European Control Conference*.

Pitt, M. K., Silva, R. S., Giordani, P., & Kohn, R., (2012). On Some Properties of Markov Chain Monte Carlo Simulation Methods based on the Particle Filter. *Journal of Econometrics*, In Press (available online, July 2012).

BIOGRAPHIES

Dawn An received the B.S. degree and M.S. degree of mechanical engineering from Korea Aerospace University in 2008 and 2010, respectively. She is now a joint Ph.D. student at Korea Aerospace University and the University of Florida. Her current research is focused on the Bayesian inference, correlated parameter identification and the methodology for prognostics and health management and structural health monitoring.

Joo-Ho Choi received the B.S. degree of mechanical engineering from Hanyang University in 1981, the M.S. degree and Ph.D. degree of mechanical engineering from Korea Advanced Institute of Science and Technology (KAIST) in 1983 and 1987, respectively. During the year 1988, he worked as a Postdoctoral Fellow at the University of Iowa. He joined the School of Aerospace and Mechanical Engineering at Korea Aerospace University, Korea, in 1997 and is now Professor. His current research is focused on the reliability analysis, design for life-time reliability, and prognostics and health management.

Nam Ho Kim received the B.S. degree of mechanical engineering from Seoul National University in 1989, the M.S. degree and Ph.D. degree of mechanical engineering from Korea Advanced Institute of Science and Technology (KAIST) in 1991 and the University of Iowa in 1999, respectively. He worked as a Postdoctoral Associate at the University of Iowa from 1999 to 2001. He joined the Dept. of Mechanical & Aerospace Engineering at the University

of Florida, in 2002 and is now Associate Professor. His current research is focused on design under uncertainty, design optimization of automotive NVH problem, shape DSA of transient dynamics (implicit/explicit) and structural health monitoring.

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
        for i=1:n; cdf(i)=sum(likel(1:i)); end; cdf=cdf./max(cdf);
        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 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);            %% Work Save
```