

# A generic framework for multisensor degradation modeling based on supervised classification and failure surface

Changyue Song, Kaibo Liu & Xi Zhang

To cite this article: Changyue Song, Kaibo Liu & Xi Zhang (2019) A generic framework for multisensor degradation modeling based on supervised classification and failure surface, IISE Transactions, 51:11, 1288-1302, DOI: [10.1080/24725854.2018.1555384](https://doi.org/10.1080/24725854.2018.1555384)

To link to this article: <https://doi.org/10.1080/24725854.2018.1555384>



Published online: 06 May 2019.



Submit your article to this journal [↗](#)



Article views: 515



View related articles [↗](#)




View Crossmark data [↗](#)



Citing articles: 3 View citing articles [↗](#)



# A generic framework for multisensor degradation modeling based on supervised classification and failure surface

Changyue Song<sup>a</sup> , Kaibo Liu<sup>a</sup>, and Xi Zhang<sup>b</sup> 

<sup>a</sup>Department of Industrial and Systems Engineering, University of Wisconsin-Madison, Madison, WI, USA; <sup>b</sup>Department of Industrial Engineering and Management, Peking University, Beijing, P.R. China

## ABSTRACT

In condition monitoring, multiple sensors are widely used to simultaneously collect measurements from the same unit to estimate the degradation status and predict the remaining useful life. In this article, we propose a generic framework for multisensor degradation modeling, which can be viewed as an extension of the degradation models from one-dimensional space to multi-dimensional space. Specifically, we model each sensor signal based on random-effect models and characterize failure events by a multi-dimensional failure surface, which is an extension of the conventional definition of the failure threshold for a single sensor signal. To overcome the challenges in estimating the failure surface, we transform the degradation modeling problem into a supervised classification problem, where a variety of classifiers can be incorporated to estimate the degradation status of the unit based on the underlying signal paths, i.e., the collected sensor signals after removing the noise. As a result, the proposed method gains great flexibility. It can also be used for sensor selection, can handle asynchronous sensor signals, and is easy to implement in practice. Simulation studies and a case study on the degradation of aircraft engines are conducted to evaluate the performance of the proposed framework in parameter estimation and prognosis.

## ARTICLE HISTORY

Received 15 February 2018

Accepted 18 November 2018

## KEYWORDS

Asynchronous sensor signals; data fusion; sensor selection

## 1. Introduction

With the rapid development of sensor technology, condition monitoring has been widely adopted in attempts to limit or even prevent unexpected failures and reduce the maintenance cost of critical units such as machines, automotive batteries, and aircraft engines. In condition monitoring, the degradation modeling and analysis of the signals collected by sensors play a critical role in estimating the degradation status and predicting the Remaining Useful Life (RUL) of units (Nelson, 1990; Meeker and Escobar, 1998). Currently, most of the literature on degradation modeling focuses on analyzing a single sensor signal (Si *et al.* 2011; Ye and Xie, 2015). However, as discussed in Brotherton *et al.* (2002) and Jardine *et al.* (2006), a single sensor signal is often insufficient to fully characterize the degradation status of the unit, as one sensor only collects measurements with respect to one characteristic of the degradation process. In order to gather information from different characteristics and predict the RUL more accurately, it has become common practice to deploy multiple sensors to monitor one unit simultaneously. This creates a pressing need for multisensor degradation modeling.

The key problem in multisensor degradation modeling lies in how to effectively fuse the useful information from multiple sensor signals to obtain a more accurate estimation

of the degradation status. In the literature, data fusion methods have been widely employed for multisensor degradation modeling. Depending on the level at which the fusion operation is performed, data fusion methods can be generally categorized into decision-level fusion and data-level fusion (Hall and Llinas, 1997; Jardine *et al.*, 2006). Decision-level fusion methods combine different prognostic results. For example, Hu *et al.* (2012) calculated the weighted average of the predicted RULs from multiple algorithms as the final prediction, where the weights were determined by cross validation. A similar approach was adopted by Baraldi *et al.* (2012) where the weights were dynamically determined using a Kalman filter. As a common limitation, decision-level fusion methods are heuristic and only produce a point estimation of the RUL without insights on the underlying degradation process.

In contrast, data-level fusion methods directly combine measurements or extracted features from multiple sensor signals. For instance, Tian (2012) and Loutas *et al.* (2013) proposed to rely on machine learning algorithms, such as neural networks, to directly predict the RUL based on the most recent sensor measurements. However, these approaches fail to utilize the unique characteristics of degradation modeling. State-space models have also been used for multisensor degradation modeling (Xu *et al.*, 2008; Saha *et al.*, 2009). However, state-space models are limited by the

assumption of the Markov property, i.e., the future degradation status depends only on the current degradation status but not the past, which may be invalid in practice (Bae and Kvam, 2004; Chen and Tsui, 2013). Fang, Gebraeel and Paynabar (2017) proposed to extract features from sensor signals using Functional Principal Component Analysis (FPCA) and predict the RUL of units by a (log)-location-scale regression model. However, the extracted features are difficult to interpret in practice, and all signals are required to share the same time domain. A recent development is the creation of multisensor degradation models that are based on a Health Index (HI) (Liu *et al.*, 2013; Liu and Huang, 2016; Liu *et al.*, 2017; Song *et al.*, 2018; Song and Liu, 2018). The main idea of HI-based methods is to construct a composite HI via a combination of multiple sensor signals to better characterize the underlying degradation process. Then the constructed HI is regarded as a single sensor signal and analyzed based on an appropriate degradation model. Although the HI-based methods facilitate visualization and decision making, they are limited by the flexibility to explore complex relationships between the underlying degradation process and the sensor signals. Specifically, HI-based methods assume an analytical function with known form (e.g., a linear function as considered by most of these studies) to combine sensor measurements into the HI, but in practice, the function form may be complex and unknown. Moreover, a common limitation of the data-level fusion methods is that they only consider synchronous sensor signals, i.e., all sensors collect measurements at the same time points, which may not be true in practice since different sensors may have different sampling frequencies and some sensor measurements can be missing during data collection and transmission.

To the best of our knowledge, the existing literature still lacks a generic multisensor degradation model that is: (i) tailored for degradation process; (ii) sufficiently flexible to be able to explore different relationships between the underlying degradation status and the sensor signals; (iii) suitable for use with asynchronous sensor signals; and (iv) able to automatically screen out non-informative sensor signals. In this article, we aim to fill this gap in the literature and propose a generic framework for multisensor degradation modeling. The proposed method can be viewed as an extension of degradation models from one-dimensional space to multi-dimensional space. Existing degradation models based on a single sensor signal commonly assume that a failure occurs when the underlying signal path, i.e., the collected sensor signal after removing the noise, crosses a failure threshold (Lu and Meeker, 1993; Gebraeel *et al.*, 2005). Accordingly, with multiple sensors, we consider that a failure occurs when the trajectory of the multiple signal paths crosses a *failure surface* in the multi-dimensional space. We illustrate how the failure threshold can be generalized to a failure surface in Figure 1. Figure 1(a) shows the single sensor case. In particular, in the left part of Figure 1(a), the dots denote the collected sensor measurements at three different time points, the dashed line denotes the underlying signal path, and the failure threshold is represented by a horizontal line. The unit fails when the underlying signal

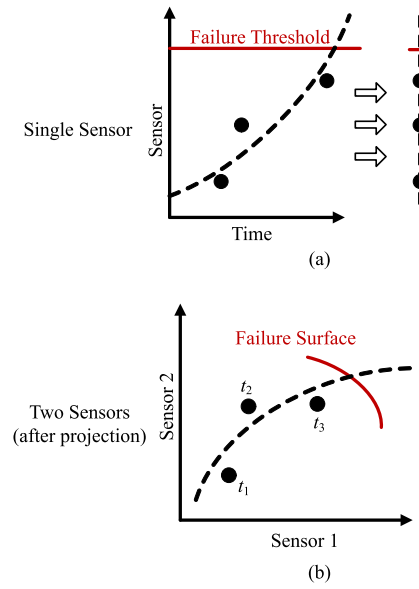


Figure 1. Illustration of failure threshold and failure surface.

path crosses the failure threshold. The right part of Figure 1(a) is obtained by projecting the signal to the vertical axis, i.e., we use the sensor (measurement) as the only coordinate and discard the time coordinate. Since the failure threshold does not rely on the time, it collapses into a point. In this way, we characterize the failure event in a one-dimensional space. As the unit is in operation, the underlying signal path starts from the bottom and evolves upwards until hitting the failure threshold. Similarly, in the case of two sensors, we can project the signals into a curve and represent the failure event in a two-dimensional space with the two sensors (measurement) as the two coordinates, which is shown in Figure 1(b). The three dots denote the collected sensor measurements at three different time points with  $t_1 < t_2 < t_3$ , and the trajectory of the sensor paths evolves from bottom-left to top-right as the dashed curve shows. In this case, the unit fails when the trajectory crosses the failure surface, which is a curve in the two-dimensional space. Similarly, with more sensors, a multi-dimensional surface can be used to define the occurrence of failure events.

There are two major tasks for degradation modeling of a single sensor signal: (i) modeling the underlying path of the single sensor signal; and (ii) estimating the failure threshold. Accordingly, for multisensor degradation modeling, the tasks are: (i) characterizing the underlying paths of multiple sensor signals; and (ii) estimating the failure surface. For the first task, existing techniques can be modified to model the underlying path of each sensor signal by adding an extension that considers the correlation among multiple sensor signals. However, for the second task, whereas estimating the failure threshold in the situation of a single sensor signal is relatively easy, estimating the failure surface for multiple sensor signals has not been investigated, due to its significant challenges. The difficulty lies in that the form of the failure surface is generally unknown and may be complex and even random. In the literature, HI-based methods address this problem by assuming the failure surface to be a hyperplane, and thus equivalently, failure is defined as the time when a linear combination of the underlying signal

paths is greater than a failure threshold. However, this assumption is too restrictive and may not be valid in practice.

To resolve this problem, our innovative idea is to transform the multisensor degradation modeling problem into a supervised classification problem, where a classifier is incorporated to define the failure surface and estimate the probability of failure based on the underlying signal paths. Since a variety of parametric and nonparametric classifiers can be employed, the failure surface constructed by our framework is very flexible. Consequently, the framework has great potential for implementation in various applications. In addition, we can further show that HI-based method is only a special case of the proposed framework.

Another advantage is that the proposed framework is intrinsically capable of solving the sensor selection problem in multisensor degradation modeling. In practice, it is common that some sensor signals are not related to the degradation process and only act as noise. These sensors should be screened out to avoid overfitting and enhance the prognostic performance. However, the literature on sensor selection for degradation modeling and prognosis is still sparse. Most existing studies heavily rely on domain knowledge and manual screening for sensor selection (Liu et al., 2013; Song et al., 2018). Fang, Paynabar and Gebraeel (2017) extended Fang, Gebraeel and Paynabar (2017) and employed a penalized (log)-location-scale regression with group non-negative garrote penalty for automatic sensor selection, but the procedure was not well integrated with prognosis and still shared the same limitations with Fang, Gebraeel and Paynabar (2017), as previously mentioned. Recently, Kim et al. (2019) developed a novel HI-based method that could screen out non-informative sensors. Specifically, they penalized the weights which were used to linearly combine the multiple sensor signals to construct the HI. In our proposed method, since the degradation modeling problem is transformed into a classification problem, the sensor selection problem can be regarded as the feature selection problem in classification. As a result, various feature selection methods can be applied.

Last but not least, the proposed framework is easy to implement and able to consider asynchronous sensor signals, since the classifier relies on the underlying signal paths but not the collected sensor measurements. Existing software packages of popular classifiers can be directly utilized in this framework. This greatly enhances the applicability of our method in practice. The rest of this article is organized as follows. In Section 2, we describe the generic framework in detail. The proposed method is tested and verified in Section 3 and Section 4 with simulation studies and a case study on the degradation of aircraft turbofan engines. Section 5 presents conclusions and discusses future work.

## 2. Methodology

### 2.1. Model formulation

Suppose  $s$  sensors are used to monitor each unit simultaneously. Let  $L_{ij}(t)$  be the sensor measurement for unit  $i$ , sensor  $j$ , at time  $t$ , where  $j = 1, \dots, s$ . We decompose  $L_{ij}(t)$  into the underlying signal path  $D_{ij}(t)$  and the noise  $\varepsilon_{ij}(t)$ :

$$L_{ij}(t) = D_{ij}(t) + \varepsilon_{ij}(t).$$

Following existing degradation models on a single sensor signal (Lu and Meeker, 1993), we model  $D_{ij}(t)$  as

$$D_{ij}(t) = \eta_j(t, \Gamma_{ij}), \quad (1)$$

where  $\eta_j(\cdot)$  is a sensor-specific function, and  $\Gamma_{ij} \in \mathcal{R}^{d_j \times 1}$  is an unknown random-effect parameter for unit  $i$ , sensor  $j$  with dimension  $d_j$ . Denote  $p(\Gamma_i)$  as the prior distribution of  $\Gamma_i = [\Gamma_{i,1}; \dots; \Gamma_{i,s}] \in \mathcal{R}^{d \times 1}$  for unit  $i$ , where  $d = \sum_{j=1}^s d_j$ . The form of  $\eta_j(\cdot)$  should be specified according to the real application. If domain knowledge and historical data are available, a corresponding parametric form can be adopted. Otherwise, if no such information is available, we may consider a generic form  $\eta_j(t, \Gamma_{ij}) = \psi_j(t)^T \Gamma_{ij}$ , where  $\psi_j(t) \in \mathcal{R}^{d_j \times 1}$  is composed of a series of basis functions with respect to time  $t$  for sensor  $j$ . In the literature, there are numerous studies on how to model a single sensor signal as a specific form of Equation (1) (Bae and Kvam, 2004; Gebraeel, 2006; Yu, 2006; Bae et al., 2007; Zhou, Serban and Gebraeel, 2014; Zhou, Serban, Gebraeel and Muller, 2014). In this article to highlight our main ideas, we assume the form of  $\eta_j(t, \Gamma_{ij})$  is already acquired.

The main idea of the proposed framework is to infer the degradation status of a unit based on the underlying signal paths instead of the original sensor measurements. Specifically, let the binary variable  $b_i(t)$  be the status of unit  $i$  at time  $t$  where  $b_i(t) = 1$  means unit  $i$  has already failed at time  $t$  and  $b_i(t) = 0$  otherwise. We assume  $b_i(t)$  can be inferred based on the underlying signal paths  $\mathbf{D}_i(t) = [D_{i,1}(t), \dots, D_{i,s}(t)]^T \in \mathcal{R}^{s \times 1}$  at time  $t$  via a classifier  $z(\cdot)$ :

$$p(b_i(t) = 1 | \mathbf{D}_i(t)) = z(\mathbf{D}_i(t)).$$

For example,  $z(\mathbf{D}_i(t)) = I(D_{ij}(t) \geq l)$  compares the value of  $D_{ij}(t)$  with a threshold  $l$  and unit  $i$  fails when  $D_{ij}(t) \geq l$ , where  $I(\cdot)$  is the indicator function. This represents the situation when a single sensor signal sufficiently characterizes the degradation process and the degradation models based on a single sensor signal can be applied. As another example,  $z(\mathbf{D}_i(t)) = I(\mathbf{D}_i(t)^T \mathbf{w} \geq l)$  compares a linear combination of  $\mathbf{D}_i(t)$  with the threshold  $l$ , where  $\mathbf{w} = [w_1, \dots, w_s]^T \in \mathbb{R}^{s \times 1}$  is the weight vector to combine the potential signal paths. This represents the main idea of HI-based methods which construct the HI  $h_i(t)$  for unit  $i$  at time  $t$  by  $h_i(t) = \sum_{j=1}^s L_{ij}(t)w_j = \mathbf{D}_i(t)^T \mathbf{w} + \sum_{j=1}^s \varepsilon_{ij}(t)w_j = \mathbf{D}_i(t)^T \mathbf{w} + \varepsilon'_i(t)$  to characterize the underlying degradation process, where  $\varepsilon'_i(t) = \sum_{j=1}^s \varepsilon_{ij}(t)w_j$ . Therefore, the HI-based method is only a special case of the proposed generic framework.

### 2.2 Classifier estimation

Our task is to estimate the classifier  $z(\cdot)$  based on  $m$  historical units with the collected sensor measurements. Suppose for unit  $i$ , sensor  $j$ , measurements  $L_{ij}(t)$  are collected at time  $t = t_{i,j,1}, \dots, t_{i,j,n_{ij}}$ . We use a column vector  $\mathbf{L}_{ij} = [L_{ij}(t_{i,j,1}), \dots, L_{ij}(t_{i,j,n_{ij}})]^T \in \mathcal{R}^{n_{ij} \times 1}$  to denote all collected measurements for unit  $i$ , sensor  $j$ , and denote  $\mathbf{L}_i = [\mathbf{L}_{i,1}; \dots; \mathbf{L}_{i,s}] \in \mathcal{R}^{n_i \times 1}$  as all available sensor measurements for



unit  $i$ , where  $n_i = \sum_{j=1}^s n_{i,j}$ . It is worth noting that, as mentioned before, the collected sensor measurements can be asynchronous, i.e., we do not require that  $n_{i,j_1} = n_{i,j_2}$  or  $t_{i,j_1,n} = t_{i,j_2,n}$  for any  $j_1, j_2$  and  $n$ . Such flexibility stems from the fact that our proposed classifier  $z(\cdot)$  relies on the underlying signal paths  $\mathbf{D}_i(t)$  but not the collected sensor measurements  $\mathbf{L}_i(t)$ .

Generally, to train a classifier, we need to collect a set of training samples with known responses  $y_i$  and covariates  $\mathbf{X}_i$  ( $i = 1, \dots, M$ ), and solve the following equation:

$$\hat{z} = \underset{z}{\operatorname{argmin}} \frac{1}{M} \sum_{i=1}^M \mathcal{Q}[y_i, z(\mathbf{X}_i)] + \mathcal{J}(z),$$

where  $\mathcal{Q}(\cdot, \cdot)$  is the loss function, and  $\mathcal{J}(\cdot)$  is the penalty function. In our formulation, the responses  $b_i(t)$  and covariates  $\mathbf{D}_i(t)$  are time series, and thus if  $\mathbf{D}_i(t)$  are known, the classifier can be estimated by

$$\hat{z} = \underset{z}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^m \int \mathcal{Q}[b_i(t), z(\mathbf{D}_i(t))] dt + \mathcal{J}(z). \quad (2)$$

However, the integral is usually difficult to compute in practice, and existing software packages on popular classifiers cannot be directly implemented in Equation (2). Therefore, we take samples at time  $t = \tau_{i,1}, \dots, \tau_{i,2N}$  and approximate Equation (2) by

$$\hat{z} = \underset{z}{\operatorname{argmin}} \frac{1}{2mN} \sum_{i=1}^m \sum_{n=1}^{2N} \mathcal{Q}[b_i(\tau_{i,n}), z(\mathbf{D}_i(\tau_{i,n}))] + \mathcal{J}(z). \quad (3)$$

It is worth noting that the sampling time points  $\tau_{i,n}$  are not the same as the time points  $t_{i,j,n}$  when sensor measurements are collected. The main difference is that  $\tau_{i,n}$  is associated with the underlying signal paths  $\mathbf{D}_i(t)$  and can be appropriately set by practitioners to approximate the integral in Equation (2), whereas  $t_{i,j,n}$  is associated with measurements  $L_{i,j}(t)$  of sensor  $j$ , and thus it is determined by the data acquisition time. Since we regard the signal path  $D_{i,j}(t)$  as a function of time  $t$  according to Equation (1), the sampling time points  $\tau_{i,n}$  can be any time before or after the unit failure. In Equation (3), we take the same number of samples  $(\mathbf{D}_i(\tau_{i,n}), b_i(\tau_{i,n}))$  from each unit to ensure each unit is equally important in the classifier. Also, the number of samples should be balanced in each class as well; otherwise, it may lead to an imbalanced classification problem with poor prediction performance (Japkowicz, 2000; Weiss and Provost, 2001). Specifically, if historical unit  $i$  is known to fail at time  $T_i$ , then the responses are

$$b_i(t) = \begin{cases} 0, & \forall t < T_i \\ 1, & \forall t \geq T_i \end{cases},$$

given that the degradation is irreversible, i.e., a failed unit will stay in that status unless maintenance is performed. In order to balance the number of samples from the two classes with  $b_i(t) = 0$  and  $b_i(t) = 1$  in the training set, we choose  $\tau_{i,n} = T_i + (n - N - 1)\delta$  with  $\delta > 0$  denoting the difference between two adjacent sampling points. In this way, we take  $2N$  samples from each unit. Since  $\tau_{i,N} = T_i - \delta < T_i$ ,  $\tau_{i,N+1} = T_i$ , and  $\tau_{i,n}$  increases with respect to  $n$ , we have that for the first  $N$  samples  $b_i(\tau_{i,n}) = 0$ ,  $n = 1, \dots, N$ , and for

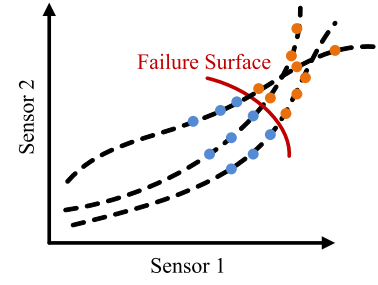


Figure 2. Illustration of the classification problem.

the last  $N$  samples  $b_i(\tau_{i,n}) = 1$ ,  $n = N + 1, \dots, 2N$ . With a larger  $N$  and a smaller  $\delta$ , our approximation becomes more accurate. Figure 2 illustrates our idea for the case of two sensors, where each dashed curve represents the trajectory of the signal paths  $\mathbf{D}_i(t) = [D_{i,1}(t), D_{i,2}(t)]$  for one unit, and the dots represent the sampled points  $\mathbf{D}_i(t^*)$  on the trajectory of the signal paths at different time points  $t^*$ . For now, we regard the signal paths  $\mathbf{D}_i(t)$  as known. Although sensor measurements  $\mathbf{L}_i(t)$  are usually censored, the signal paths  $\mathbf{D}_i(t)$  are functions of time  $t$  according to Equation (1) and thus the trajectory can go beyond the failure surface. In Figure 2, for illustration, we take  $N = 3$  samples before and after the failure of each unit on the trajectory of the signal paths, and then a classifier can be used to define the failure surface to separate the two groups of samples. In practice, the values of  $N$  and  $\delta$  should be determined based on the real application.

Another challenge in the above formulation is that the underlying signal paths  $\mathbf{D}_i(\tau_{i,n}) = [D_{i,1}(\tau_{i,n}), \dots, D_{i,s}(\tau_{i,n})]^T = [\eta_1(\tau_{i,n}, \Gamma_{i,1}), \dots, \eta_s(\tau_{i,n}, \Gamma_{i,s})]^T$  are usually unknown, which depend on the latent random-effect parameter  $\Gamma_i$ , and thus the estimation method of Equation (3) still cannot be applied in our case. To solve this problem, we further modify Equation (3) and consider the expected loss regarding the posterior distribution of  $\Gamma_i$

$$\hat{z} = \underset{z}{\operatorname{argmin}} \frac{1}{2mN} \sum_{i=1}^m \sum_{n=1}^{2N} \int \mathcal{Q}[b_i(\tau_{i,n}), z(\mathbf{D}_i(\tau_{i,n}))] p(\Gamma_i | \mathbf{L}_i) d\Gamma_i + \mathcal{J}(z). \quad (4)$$

Here  $p(\Gamma_i | \mathbf{L}_i)$  is the posterior distribution of  $\Gamma_i$  given the collected sensor measurements  $\mathbf{L}_i$ , which can be calculated according to  $p(\Gamma_i | \mathbf{L}_i) \propto p(\mathbf{L}_i | \Gamma_i) p(\Gamma_i)$ . With conjugate distribution families, we can obtain analytical expression for the posterior distribution. For example, if the noises are independent and normally distributed  $\varepsilon_{i,j}(t) \sim \mathcal{N}(0, \sigma_j^2)$ , and the underlying signal path has the form  $\eta_j(t, \Gamma_{i,j}) = \psi_j(t)^T \Gamma_{i,j}$ , then  $\mathbf{L}_{i,j} | \Gamma_{i,j} = \Psi_{i,j} \Gamma_{i,j} + \varepsilon_{i,j} \sim \mathcal{N}_{n_{ij}}(\Psi_{i,j} \Gamma_{i,j}, \sigma_j^2 \mathbf{I})$ , where  $\Psi_{i,j} = [\psi_j(t_{i,j,1}), \dots, \psi_j(t_{i,j,n_{ij}})]^T \in \mathcal{R}^{n_{ij} \times d_j}$  is the design matrix,  $\varepsilon_{i,j} = [\varepsilon_{i,j}(t_{i,j,1}), \dots, \varepsilon_{i,j}(t_{i,j,n_{ij}})] \in \mathcal{R}^{n_{ij} \times 1}$  contains the noises, and  $\mathbf{I}$  is the identity matrix. Therefore, the conditional distribution  $\mathbf{L}_i | \Gamma_i \sim \mathcal{N}_{n_i}(\Psi_i \Gamma_i, \Omega_i)$  where  $\Psi_i = \operatorname{diag}(\Psi_{i,1}, \dots, \Psi_{i,s}) \in \mathcal{R}^{n_i \times d}$  and  $\Omega_i = \operatorname{diag}(\sigma_1^2 \mathbf{I}, \dots, \sigma_s^2 \mathbf{I}) \in \mathcal{R}^{n_i \times n_i}$  are two block-diagonal matrices. In this case, if the prior distribution is also normal  $\Gamma_i \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ , we can obtain the posterior distribution  $\Gamma_i | \mathbf{L}_i \sim \mathcal{N}_d(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ , where

$\mu_i = \Sigma_i(\Psi_i^T \Omega_i^{-1} L_i + \Sigma^{-1} \mu)$ , and  $\Sigma_i = (\Psi_i^T \Omega_i^{-1} \Psi_i + \Sigma^{-1})^{-1}$ . If there is no analytical solution for the posterior distribution, numerical methods such as Monte Carlo Markov Chain can be used to obtain samples from the posterior distribution  $p(\Gamma_i | L_i)$ .

Now the main challenge lies in calculating the integral in Equation (4). Directly calculating the integral is computationally expensive, and may be even impossible. To reduce the computation and ensure the easy implementation such that practitioners can directly utilize the existing software packages of popular classifiers, we decided to consider some approximation techniques. In the literature, there are a number of approximation techniques for calculating the integral (Pinheiro and Bates, 1995; Bae and Kvam, 2004). In this article we consider two techniques: (i) the Monte Carlo method; and (ii) Laplacian approximation as a demonstration, since these two techniques require fewer restrictions on the specific form of Equation (4) and can be applied to many classifiers. In practice, other approximation techniques can also be employed, depending on the application context.

### 2.3. Monte Carlo method

The first approximation technique we consider is the Monte Carlo method. Since

$$\int \mathcal{Q}[b_i(\tau_{i,n}), z(\mathbf{D}_i(\tau_{i,n}))] p(\Gamma_i | L_i) d\Gamma_i \\ \approx \frac{1}{K} \sum_{k=1}^K \mathcal{Q}[b_i(\tau_{i,n}), z(\mathbf{D}_i^{(k)}(\tau_{i,n}))],$$

according to the law of large numbers, where

$$\mathbf{D}_i^{(k)}(\tau_{i,n}) = [D_{i,1}^{(k)}(\tau_{i,n}), \dots, D_{i,s}^{(k)}(\tau_{i,n})]^T \\ = [\eta_1(\tau_{i,n}; \Gamma_{i,1}^{(k)}), \dots, \eta_s(\tau_{i,n}; \Gamma_{i,s}^{(k)})]^T,$$

and  $\Gamma_i^{(k)}$  is a draw from the posterior distribution  $p(\Gamma_i | L_i)$ , we can estimate the classifier by

$$\hat{z} = \underset{z}{\operatorname{argmin}} \frac{1}{2mNK} \sum_{i=1}^m \sum_{k=1}^K \sum_{n=1}^{2N} \mathcal{Q}[b_i(\tau_{i,n}), z(\mathbf{D}_i^{(k)}(\tau_{i,n}))] + \mathcal{J}(z). \quad (5)$$

In other words, for each unit  $i$ , we draw  $K$  samples from the posterior distribution  $p(\Gamma_i | L_i)$ . For each sample  $\Gamma_i^{(k)}$ , we calculate the underlying signal paths  $\mathbf{D}_i^{(k)}(t)$  and then take samples at time  $t = \tau_{i,1}, \dots, \tau_{i,2N}$ . Finally, the sampled  $\mathbf{D}_i^{(k)}(\tau_{i,n})$  and the corresponding  $b_i(\tau_{i,n})$  are used for classifier estimation, where existing software packages can be directly utilized.

There are several advantages of the Monte Carlo method. First, it is flexible with no additional restrictions imposed on the structure of the model. Second, it does not require the posterior distribution  $p(\Gamma_i | L_i)$  to have an analytical form, and thus,  $p(\Gamma_i | L_i)$  can be calculated using numerical methods. Third, although Equation (5) is derived in the setting of a parametric classifier, it can also be used to train a non-parametric classifier such as  $k$ -nearest neighborhood. And last, the approximation error that is created by use of the Monte Carlo method does not depend on the dimension of

$\Gamma_i$ , and thus it can be used in the case where the dimension of  $\Gamma_i$  is high. However, one potential limitation is that the Monte Carlo method may require a large set of training samples. Specifically, there are  $2mNK$  samples in the training set, which may be challenging to the computational resource. In the next section, we discuss another approximation technique, Laplacian approximation.

### 2.4. Laplacian approximation

To facilitate the Laplacian approximation technique, we rewrite Equation (4) as

$$\hat{z} = \underset{z}{\operatorname{argmin}} \frac{1}{2mN} \sum_{i=1}^m \int \exp[f(\Gamma_i)] d\Gamma_i + \mathcal{J}(z), \quad (6)$$

where  $f(\Gamma_i) = \log \left\{ \sum_{n=1}^{2N} \mathcal{Q}[y_i(\tau_{i,n}), z(\mathbf{D}_i(\tau_{i,n}))] \right\} + \log p(\Gamma_i | L_i)$ . The basic idea of Laplacian approximation is to approximate  $f(\Gamma_i)$  by the second-order Taylor expansion at the maximum point. Specifically, given the classifier  $z(\cdot)$ , we find the maximum point

$$\Gamma_i^* = \underset{\Gamma_i}{\operatorname{argmax}} f(\Gamma_i) \\ = \underset{\Gamma_i}{\operatorname{argmax}} \log \left\{ \sum_{n=1}^{2N} \mathcal{Q}[y_i(\tau_{i,n}), z(\mathbf{D}_i(\tau_{i,n}))] \right\} + \log p(\Gamma_i | L_i). \quad (7)$$

Then according to the Taylor expansion,  $f(\Gamma_i)$  can be approximated as

$$f(\Gamma_i) \approx f(\Gamma_i^*) + \frac{1}{2} (\Gamma_i - \Gamma_i^*)^T \nabla^2 f(\Gamma_i^*) (\Gamma_i - \Gamma_i^*),$$

where  $\nabla^2 f(\Gamma_i^*)$  is the Hessian matrix at  $\Gamma_i = \Gamma_i^*$ , and the first-order derivative of  $f(\Gamma_i)$  vanishes since  $\Gamma_i^*$  is a stationary point. Consequently, the integral can be approximated by

$$\int \exp[f(\Gamma_i)] d\Gamma_i \\ \approx \int \exp \left[ f(\Gamma_i^*) + \frac{1}{2} (\Gamma_i - \Gamma_i^*)^T \nabla^2 f(\Gamma_i^*) (\Gamma_i - \Gamma_i^*) \right] d\Gamma_i \\ = \exp[f(\Gamma_i^*)] \cdot \sqrt{\frac{(2\pi)^d}{|\nabla^2 f(\Gamma_i^*)|}}.$$

Here the operator  $|\cdot|$  calculates the determinant. In this way, Equation (6) is approximated as

$$\hat{z} = \underset{z}{\operatorname{argmin}} \frac{1}{2mN} \sum_{i=1}^m \sum_{n=1}^{2N} \omega_i \mathcal{Q}[y_i(\tau_{i,n}), z(\mathbf{D}_i^*(\tau_{i,n}))] + \mathcal{J}(z), \quad (8)$$

where

$$\omega_i = p(\Gamma_i^* | L_i) \sqrt{(2\pi)^d / |\nabla^2 f(\Gamma_i^*)|},$$

$$\mathbf{D}_i^*(t) = [D_{i,1}^*(t), \dots, D_{i,s}^*(t)]^T = [\eta_1(t; \Gamma_{i,1}^*), \dots, \eta_s(t; \Gamma_{i,s}^*)]^T,$$

and  $\Gamma_i^*$  can be regarded as a function of  $z(\cdot)$ . Therefore,  $\hat{z}(\cdot)$  can be obtained by solving Equation (8).

Based on Laplacian approximation, the size of the training set is  $2mN$ , which is less than the Monte Carlo method.

However, the training process in Laplacian approximation involves nested optimization, which requires additional computation. Also, restrictions apply for Laplacian approximation. First, the classifiers that can be incorporated are limited. For example, nonparametric classifiers without a proper loss function may not be used with Laplacian approximation. Second, to facilitate solving Equation (7), it is desired that the posterior distribution  $p(\Gamma_i|L_i)$  has analytical expression or close-form approximations. And last,  $f(\Gamma_i)$  must be twice-differentiable to calculate the Hessian matrix.

## 2.5. RUL prediction

For an in-field unit  $r$  that has not yet failed, the estimated classifier  $\hat{z}$  can be used to predict the RUL. Specifically, suppose the collected sensor signals for unit  $r$  are  $L_r \in \mathcal{R}^{n_r \times 1}$ , the probability of failure before time  $t$  can be predicted as

$$p(T_r < t | L_r) = p(b_r(t) = 1 | L_r) = \int p(b_r(t) = 1 | D_r(t)) p(\Gamma_r | L_r) d\Gamma_r, \quad (9)$$

where  $T_r$  is the failure time of unit  $r$ , and the underlying signal paths  $D_r(t)$  depend on the latent random variable  $\Gamma_r$ . Since the probability  $p(b_r(t) = 1 | D_r(t))$  can be estimated by  $\hat{z}(D_r(t))$ , Equation (9) can be approximated by

$$p(T_r < t | L_r) = E_{\Gamma_r | L_r}[\hat{z}(D_r(t))] = \frac{1}{K} \sum_{k=1}^K \hat{z}(D_r^{(k)}(t)).$$

Here

$$D_r^{(k)}(t) = [D_{r,1}^{(k)}(t), \dots, D_{r,s}^{(k)}(t)]^T = \left[ \eta_1(t; \Gamma_{r,1}^{(k)}), \dots, \eta_s(t; \Gamma_{r,s}^{(k)}) \right]^T,$$

and  $\Gamma_r^{(k)}$  is a draw from the posterior distribution  $p(\Gamma_r | L_r)$ , which can be computed by  $p(\Gamma_r | L_r) \propto p(L_r | \Gamma_r) p(\Gamma_r)$  as discussed in Section 2.2. Let  $t_r^c$  be the current time and assume that unit  $r$  has not failed by time  $t_r^c$ . Then, we can update the distribution of the failure time  $T_r$  by

$$p(T_r < t | T_r > t_r^c, L_r) = \frac{p(T_r < t | L_r) - p(T_r < t_r^c | L_r)}{1 - p(T_r < t_r^c | L_r)}.$$

Since the distribution may be skewed, we use the median as the point estimator of the failure time, i.e., we find  $\hat{T}_r$  such that:

$$p(T_r < \hat{T}_r | T_r > t_r^c, L_r) = 0.5.$$

In this way, the RUL can be predicted as  $\hat{T}_r - t_r^c$ . Similarly, for a number  $\nu$  with  $0 < \nu < 1$ , the corresponding quantile  $\hat{T}_r^{(\nu)}$  of the failure time can be calculated according to

$$p(T_r < \hat{T}_r^{(\nu)} | T_r > t_r^c, L_r) = \nu.$$

Then a prediction interval for the RUL with level  $\alpha$  can be established as  $(\hat{T}_r^{(\alpha/2)} - t_r^c, \hat{T}_r^{(1-\alpha/2)} - t_r^c)$ .

## 2.6. Sensor selection

As previously mentioned, the proposed framework is intrinsically capable of solving the sensor selection problem in multisensor degradation modeling. This is because as the degradation modeling problem is transformed into a classification problem, the sensor selection problem can be regarded as the feature selection or variable selection in the classification. Therefore, a variety of feature selection methods can be used for sensor selection, such as the wrapping methods and shrinkage methods. An introduction to feature selection approaches can be found in Guyon and Elisseeff (2003). Specifically, wrapping methods typically rely on cross validation or another technique to select the subset of variables that minimizes the prediction error on the validation set. Thus, in theory, wrapping methods can be incorporated into any multisensor degradation model for sensor selection. However, wrapping methods are usually time-consuming and heuristic, and thus have no guarantee that the selected subset of features is optimal. Shrinkage methods, on the other hand, rely on the penalty function  $\mathcal{J}(z)$  to screen out unrelated features. Shrinkage methods have the advantage of less computation and often offer a desirable theoretical property that ensures that the correct features are selected with some specific penalty function, such as the LASSO and the adaptive LASSO penalty (Tibshirani, 1996; Knight and Fu, 2000; Zou, 2006). Compared with other multisensor degradation models, the proposed framework has the advantage that it easily incorporates shrinkage methods for sensor selection, and the existing related software packages can be directly utilized as discussed in Section 2.3.

## 3. Simulation studies

In this simulation section, we conduct a series of studies to evaluate the proposed framework using a simulated dataset. Specifically, we consider the prognostic performance and failure surface estimation of our method with different classifiers in Section 3.2 and test the sensor selection performance with adaptive LASSO in Section 3.3. Section 3.4 describes the sensitivity to sparse data, and Section 3.5 discusses the selection of tuning parameters in the proposed framework. The simulation studies are conducted on a server with 2.20 GHz 16-core CPU and 192 GB memory using MATLAB 2017a.

### 3.1. Dataset generation

In this simulation study, we consider four sensors and generate the underlying signal paths for each unit by

$$\begin{aligned} D_{i,1}(t) &= \Gamma_{i,1}^{(0)} + \Gamma_{i,1}^{(1)}(0.2t + \sin(0.2t)) = [1, 0.2t + \sin(0.2t)] \Gamma_{i,1}, \\ D_{i,2}(t) &= \Gamma_{i,2}^{(0)} + \Gamma_{i,2}^{(1)} \left( 0.6t^{0.8} - \frac{100}{t+5} \right) = \left[ 1, 0.6t^{0.8} - \frac{100}{t+5} \right] \Gamma_{i,2}, \\ D_{i,3}(t) &= \Gamma_{i,3}^{(0)} + \Gamma_{i,3}^{(1)} \frac{t^{1.4}}{40} = \left[ 1, \frac{t^{1.4}}{40} \right] \Gamma_{i,3}, \\ D_{i,4}(t) &= \Gamma_{i,4}^{(0)} + \Gamma_{i,4}^{(1)} \left( \frac{t^{1.6}}{100} + 3 \cos(0.1t) \right) = \left[ 1, \frac{t^{1.6}}{100} + 3 \cos(0.1t) \right] \Gamma_{i,4}, \end{aligned}$$

with the prior distribution

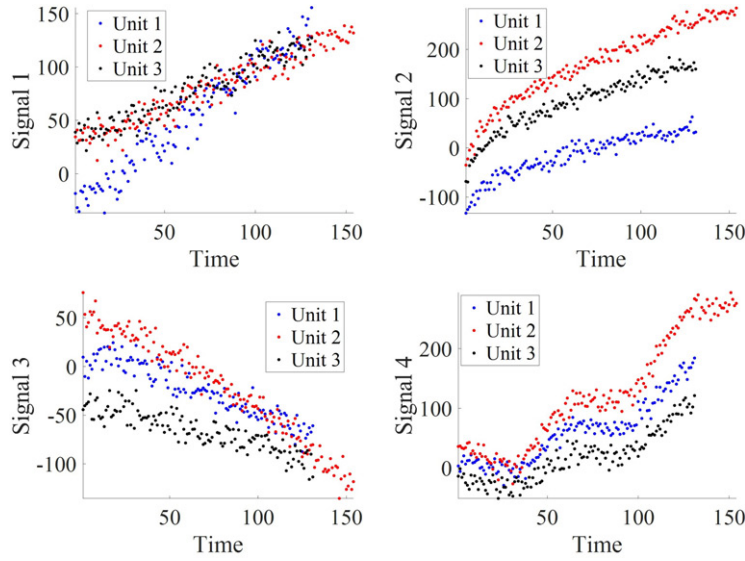


Figure 3. The four signals of selected three units in the generated dataset.

$$\mathbf{\Gamma}_i = \begin{bmatrix} \mathbf{\Gamma}_{i,1} \\ \mathbf{\Gamma}_{i,2} \\ \mathbf{\Gamma}_{i,3} \\ \mathbf{\Gamma}_{i,4} \end{bmatrix} \sim \mathcal{N}_8 \left( \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} -1 \\ 5 \end{bmatrix}, \mathbf{I}_4 \otimes \begin{bmatrix} 900 & 1 \\ 1 & 2 \end{bmatrix} \right).$$

Here  $\otimes$  denotes the Kronecker product and  $\mathbf{I}_4$  is the  $4 \times 4$  identity matrix. In this simulation study, to highlight our main idea and better evaluate the proposed framework, the true models of the underlying signal paths described above are used for classifier estimation and RUL prediction. In addition, we consider  $\mathbf{\Gamma}_{i,j}$ ,  $j = 1, 2, 3, 4$  are mutually independent for simplicity. The four sensor signals are randomly generated by  $L_{i,j}(t) = D_{i,j}(t) + \varepsilon_{i,j}(t)$ , where  $\varepsilon_{i,j}(t) \sim \mathcal{N}(0, 10^2)$ . As mentioned in Section 2.2, in this case, the posterior distribution  $\mathbf{\Gamma}_i$  is also normal and has an analytical expression. A random failure surface is used to define unit failure, which is a hyperplane for each unit. Specifically, failure time of unit  $i$  is defined as

$$T_i = \underset{t}{\operatorname{argmin}} D_{i,1}(t) + 0.05D_{i,2}(t) - 0.2D_{i,3}(t) \geq l_i,$$

where  $l_i$  is a random number subject to a logistic distribution with location parameter  $\mu_l = 15 \times 12 = 180$  and scale parameter  $\sigma_l = 12$ . Obviously, only the first three sensors are related to the degradation process and the last sensor should be screened out using a sensor selection algorithm. After the failure time  $T_i$  is calculated, the sensor measurements of each unit are generated at time  $t = 1, \dots, \lfloor T_i \rfloor$ , where  $\lfloor T_i \rfloor$  is the largest integer smaller or equal to  $T_i$ . As an example, the four generated signals for three different units are shown in Figure 3.

The true cumulative distribution function of the failure time  $T_i$  is

$$\begin{aligned} p(T_i \leq t) &= p(l_i \leq D_{i,1}(t) + 0.05D_{i,2}(t) - 0.2D_{i,3}(t)) \\ &= \frac{1}{\exp(-[1, \mathbf{D}_i(t)/\sigma_l] \boldsymbol{\beta}_0) + 1}, \end{aligned}$$

where  $\boldsymbol{\beta}_0 = (-15, 1, 0.05, -0.2, 0)^T$  and  $\sigma_l = 12$ . Then, the true distribution of  $b_i(t)$  is determined by Logistic Regression (LR) based on  $\mathbf{D}_i(t)$  with the true coefficient being  $\boldsymbol{\beta}_0$ . This classifier is treated as the oracle classifier and

the corresponding prognostic performance is treated as the benchmark to evaluate the proposed framework.

### 3.2. Prognostic performance and failure surface estimation

First, we evaluate the prognostic performance of our proposed framework. Specifically, we randomly generate  $m$  historical units and estimate the classifier  $z(\cdot)$ . The estimated  $\hat{z}(\cdot)$  is applied to another 100 testing units to predict the RUL based on truncated sensor signals, and we calculate the prediction error as

$$e_i = \frac{|T_i - \hat{T}_i|}{T_i}, \quad (10)$$

where  $T_i$  is the true failure time of unit  $i$  and  $\hat{T}_i$  is the predicted failure time. The average prediction error  $\bar{e}$  across all testing units is then calculated. This process is repeated for 50 times to obtain the mean and standard deviation of  $\bar{e}$ . In this study, LR is considered as  $z(\cdot)$  because the oracle classifier is an LR model. In addition, we also employ a Support Vector Machine (SVM) to consider the case when a different classifier is used. For LR, the deviance (defined as  $-2\log L_p$ , where  $L_p$  is the likelihood function) is regarded as the loss function and no penalty function is considered, that is

$$Q(b, z(\mathbf{D})) = -2\{\mathbf{D}\boldsymbol{\rho} + \rho_0 - \log[\exp(\mathbf{D}\boldsymbol{\rho} + \rho_0) + 1]\}$$

and  $\mathcal{J}(z) = 0$ , where  $\boldsymbol{\beta} = [\rho_0; \boldsymbol{\rho}]$  are the coefficients to be estimated. For the SVM, we use the hinge loss function  $Q(b, z(\mathbf{D})) = [1 - b(\mathbf{D}\boldsymbol{\rho} + \rho_0)]_+$  and penalty function  $\mathcal{J}(z) = 0.5 \cdot \boldsymbol{\rho}^T \boldsymbol{\rho}$ , where  $[a]_+ = \max(0, a)$ ,  $b \in \{-1, 1\}$  with  $-1$  representing non-failure and  $1$  representing failure, and  $\boldsymbol{\beta} = [\rho_0; \boldsymbol{\rho}]$  are the coefficients to be estimated.

When training the classifiers, both approximation methods, the Monte Carlo method and Laplacian approximation, are applied. Tuning parameters are set as  $N = 200$ ,  $\delta = 0.2$ ,  $K = 50$ , i.e., we take samples at 200 different values of  $\tau_{i,n}$  before the failure time  $T_i$  and 200 different values after  $T_i$



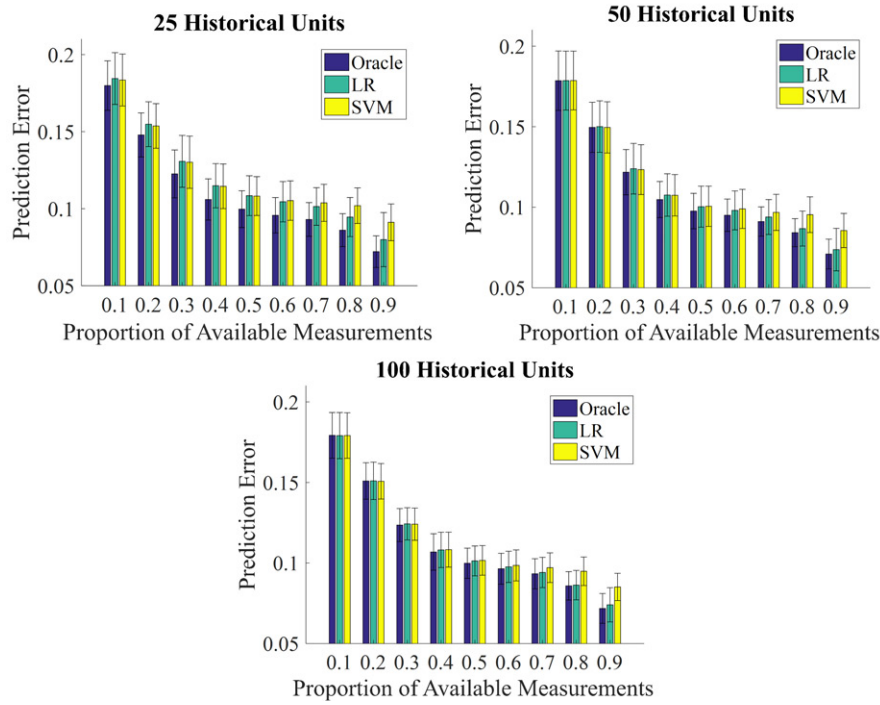


Figure 4. Prognostic performance of the proposed method with LR and the SVM compared with the oracle classifier.

with an equal space of 0.2, and we draw 50 random samples  $\Gamma_i^{(k)}$  from the posterior distribution  $p(\Gamma_i|L_i)$  when implementing the Monte Carlo method. More discussions on how to choose the tuning parameters will be presented in Section 3.4. The computational time is recorded and compared between the Monte Carlo method and Laplacian approximation. As an example, when 100 historical units are available, for the Monte Carlo method, the training set contains  $2 \times 10^6$  samples and the training procedure takes around 10 seconds for LR and 78 seconds for the SVM on average. For Laplacian approximation, the training set contains  $4 \times 10^4$  samples and the training procedure takes around 20 minutes for LR and 23 minutes for the SVM on average. Clearly, although the Monte Carlo method requires a larger memory, it is much faster than the Laplacian approximation.

The prognostic results of our method using Monte Carlo method based on 25, 50, and 100 historical units are shown in Figure 4, where the bars denote the mean of  $\bar{e}$  for each classifier with one standard deviation, and the  $x$ -axis denotes different levels of available sensor measurements. For example, “0.4” means that the sensor signals of each testing unit  $i$  are truncated at  $0.4T_i$ . The result of Laplacian approximation is almost the same as the Monte Carlo method and thus is omitted. As shown in the figure, as more historical units become available, the prognostic performance of the proposed method based on LR becomes closer to the oracle classifier. The prediction error of the SVM is slightly greater than the oracle classifier, but is still satisfactory. This indicates that the proposed framework can accurately predict the RUL even with a different classifier.

In addition, to verify the accuracy of our proposed method in estimating the failure surface, we compare the estimated failure surface with the true one. In this

simulation study, since  $\beta_0 = (-15, 1, 0.05, -0.2, 0)^T$ , the underlying true failure surface can be expressed as

$$-15 + 1 \cdot x_1 + 0.05 \cdot x_2 - 0.2 \cdot x_3 + 0 \cdot x_4 = \xi,$$

where the random variable  $\xi \sim \text{logistic}(0, 1)$ , and  $x_j$  represents the scaled signal path of sensor  $j$ , i.e.,  $D_{ij}(t)/\sigma_l$ . With 100 historical units available and LR as the classifier, the estimated failure surface is

$$-15 + 1.001 \cdot x_1 + 0.049 \cdot x_2 - 0.207 \cdot x_3 - 0.004 \cdot x_4 = 1.110 \cdot \xi.$$

As we can see, despite a slight scale difference in the estimated coefficient, the estimated failure surface is very close to the true failure surface. Similarly for the SVM, the estimated failure surface is

$$-15 + 1.001 \cdot x_1 + 0.050 \cdot x_2 - 0.205 \cdot x_3 - 0.003 \cdot x_4 = 0.$$

For the SVM, the estimated failure surface is fixed, and is very close to the expectation of the true failure surface. As a result, the estimated failure surfaces of both LR and SVM are very close to the true failure surface.

### 3.3. Sensor selection

To screen out non-informative sensors, we regard LR as  $z(\cdot)$  and adopt the adaptive LASSO penalty due to its oracle properties (Zou, 2006). The idea of adaptive LASSO is to penalize each entry of the LR coefficient  $\beta$  except the intercept with different weights, and the penalty function is given as  $\mathcal{J}(z) = \lambda \sum_{j=1}^s \alpha_j |\beta_j|$ , where the weight  $\alpha_j$  is estimated by  $\hat{\alpha}_j = 1/|\hat{\beta}_j^{\text{MLE}}|^\gamma$ ,  $\hat{\beta}^{\text{MLE}} = (\hat{\beta}_1^{\text{MLE}}, \dots, \hat{\beta}_s^{\text{MLE}})^T$  is the estimated coefficient by maximum likelihood estimation without penalty, and  $\gamma$  is a positive number. In this simulation, we

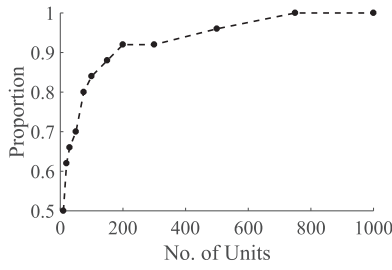


Figure 5. Proportion of repetitions that the correct sensors are selected.

choose  $\gamma = 1$  as a demonstration. In particular, we randomly generate  $m$  historical units. Then starting with a large value of  $\lambda$ , which excludes all the sensors, we gradually decrease  $\lambda$  until three of the four sensors are selected. We repeat this process for 50 times and calculate the proportion of repetitions that Sensor 4 is the only excluded sensor, which indicates that the sensors are correctly selected. We evaluate the sensor selection performance with different  $m$  and the result is shown in Figure 5. This figure clearly indicates that as more historical units become available, the algorithm is more likely to select the correct sensors. The sensor selection performance of the proposed method has been further verified based on another simulated dataset which contains 10 non-informative sensors out of 25 sensors. The conclusion is similar and thus the details are omitted here.

### 3.4. Sensitivity to sparse data

In practice, the collected sensor measurements may be sparse, due to sensor malfunction, limited transmission bandwidth, data loss, etc. In this simulation study, we consider the case when only a small number of sensor measurements are available. Specifically, we randomly generate 100 units and use LR as the classifier. The LR coefficient  $\hat{\beta}^1$  estimated based on all available sensor measurements is regarded as the baseline. Then for each signal of a unit, we randomly draw a fraction  $c$  of measurements and estimate the LR coefficient  $\hat{\beta}^c$  based only on these measurements. Since the measurements are drawn separately for each signal, the available sensor measurements are asynchronous in this study, to better mimic the practical situation. To show the sensitivity of our method to the data sparsity, we calculate the relative absolute difference  $\Delta_j^c$  between each entry of  $\hat{\beta}^1$  and  $\hat{\beta}^c$  as

$$\Delta_j^c = \frac{|\hat{\beta}_j^1 - \hat{\beta}_j^c|}{|\hat{\beta}_j^1|},$$

for  $j = 0, 1, 2, 3$ . We do not consider  $j = 4$  because  $\hat{\beta}_4^1$  is very close to zero. If the proposed method is not sensitive to sparse data, the relative absolute difference  $\Delta_j^c$  will be small. This procedure is repeated for 20 times to obtain the average and standard deviation of  $\Delta_j^c$ . All other parameters are the same as in Section 3.2. The averaged  $\Delta_j^c$  as well as the standard deviations of the average using Monte Carlo method and Laplacian approximation are shown in Figure 6 and Figure 7, respectively. Overall, the results are satisfactory because the average relative difference is only around

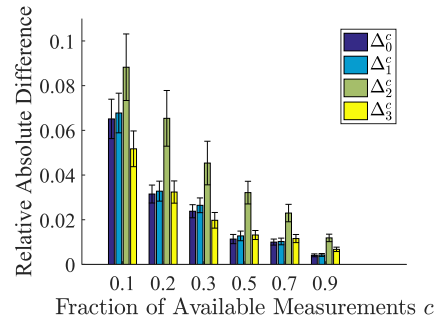


Figure 6. Sensitivity of the estimated coefficient to sparse data using Monte Carlo method.

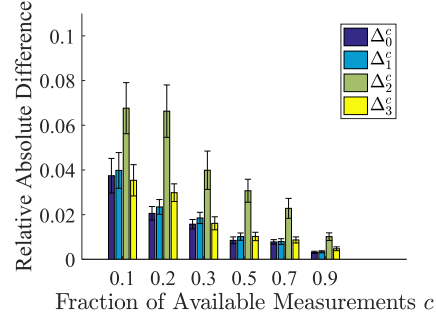


Figure 7. Sensitivity of the estimated coefficient to sparse data using Laplacian approximation.

10%, even if there are only 10% measurements available. A more detailed comparison between the two figures reveals that the Laplacian approximation seems to be more robust to sparse data than the Monte Carlo method. One possible reason for this behavior is that when the available data is sparse, the posterior distribution  $p(\Gamma_i|L_i)$  has large variance. Consequently, if  $K$  is not large enough, the random samples  $\Gamma_i^{(k)}$  cannot represent the posterior distribution  $p(\Gamma_i|L_i)$  well, which adds additional variation that increases  $\Delta_j^c$ .

### 3.5. Discussion of tuning parameters

There are three tuning parameters including  $N$ ,  $\delta$ , and  $K$  in the proposed framework. Generally,  $N$  and  $K$  should be large and  $\delta$  should be small to obtain an accurate estimation. However, too large values of  $N$  and  $K$  may require a long period of computation and may not be necessary in practice, and also, as will be discussed later,  $\delta$  cannot be too small when  $N$  is not large enough. In this subsection, we run a series of simulation studies to assess the effects of these tuning parameters, which can provide guidance on the parameter settings. Due to the page limit, we discuss our conclusions here and the details can be found in the Appendix.

We recognize that a proper value of  $N \cdot \delta$  is more important for an accurate approximation than  $N$  or  $\delta$  itself. The physical meaning of  $N \cdot \delta$  can be interpreted as the window length that we take samples before and after the failure time  $T_i$ , and it approximates  $T_i - \tau_{i,1}$  or  $\tau_{i,2N} - T_i$ . One possible reason is that usually the failure surface is random in some “zone” of the multi-dimensional space. If  $N \cdot \delta$  is too small, most of the samples  $(D_i(t), b_i(t))$  in the training set are still within a subspace of this “zone” and thus are not

enough to fully characterize the failure surface. On the contrary, with a relatively large  $N \cdot \delta$ , the training set  $(\mathbf{D}_i(t), b_i(t))$  will contain samples throughout the “zone,” leading to a more accurate estimation.

We also find that the value of  $N \cdot \delta$  has different effects on different classifiers. Some classifiers such as the SVM are less sensitive to small  $N \cdot \delta$ . One possible reason for this behavior is that the SVM is only related to a portion of training samples that are misclassified, which are known as the supporting vectors (Hastie *et al.*, 2009). With  $\delta$  fixed, a larger  $N$  leads to more samples  $(\mathbf{D}_i(t), b_i(t))$  that are far away from the failure surface; however, the supporting vectors may remain the same and thus a larger  $N$  does not change the SVM model to any great extent.

Another observation is that with a larger  $m$ , a smaller value of  $K$  will be enough for accurate approximation of the Monte Carlo method. This is reasonable because each unit represents a sample  $\Gamma_i$  from the population. However,  $\Gamma_i$  is unknown and can only be inferred from the posterior distribution  $p(\Gamma_i | L_i)$ . If  $m$  is small, the samples of  $\Gamma_i$  cannot characterize the full sample space and we have to exploit the subsample space of  $p(\Gamma_i | L_i)$  to obtain more information. On the other hand, if  $m$  is large, the full sample space is well characterized by samples of  $\Gamma_i$ . In this case, using a large  $K$  to fully exploit the subsample space of  $p(\Gamma_i | L_i)$  does not provide much information. Actually this is a favorable property for implementing the Monte Carlo method in practice, which allows us to use a small  $K$  when there are many historical units to reduce the computation.

Based on our experience, we summarize the procedure for specifying the tuning parameters as follows. At first,  $N \cdot \delta$  can be chosen as a value around the sample standard deviation of  $\{T_i\}_{i=1}^m$ , where  $T_i$  is the failure time of the historical unit  $i$ . The value of  $N \cdot \delta$  can be decreased if a SVM is used as the classifier. Then, the value of  $\delta$  can be specified based on the desired precision of the prognostic result. For example, if we only need to predict in which day that the unit will fail and do not mind if the unit fails in the morning or in the afternoon, we can choose  $\delta$  to be 1 day. Accordingly, the value of  $N$  can be specified. Next, we consider the value of  $K$  if Monte Carlo approximation is employed. From our study, we observe that if  $m \cdot K$  is around several thousands, the prognostic performance is usually satisfactory, and  $K$  can be determined accordingly. In addition, if the sensor measurements are sparse or the measurement error is large, the value of  $K$  should be increased. Finally, based on the fact that with Monte Carlo approximation, the number of samples in the training set for the classification problem is  $2mNK$ , we can further adjust the values of the tuning parameters according to the available computational resource.

#### 4. Application

In simulation studies of Section 3, we assume that the true degradation model for each sensor signal as well as the prior distribution  $p(\Gamma_i)$  is known. In this section, we consider the case when this information is unavailable, which is common

**Table 1.** Detailed description of the 21 sensors (Saxena *et al.*, 2008)

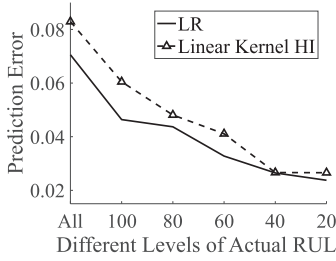
Symbol	Description	Units
T2	Total temperature at fan inlet	°R
T24	Total temperature at LPC outlet	°R
T30	Total temperature at HPC outlet	°R
T50	Total temperature at LPT outlet	°R
P2	Pressure at fan inlet	psia
P15	Total pressure in bypass-duct	psia
P30	Total pressure at HPC outlet	psia
Nf	Physical fan speed	rpm
Nc	Physical core speed	rpm
epr	Engine pressure ratio (P50/P2)	–
Ps30	Static pressure at HPC outlet	psia
phi	Ratio of fuel flow to Ps30	pps/psi
NRf	Corrected fan speed	rpm
NRc	Corrected core speed	rpm
BPR	Bypass Ratio	–
farB	Burner fuel-air ratio	–
htBleed	Bleed Enthalpy	–
Nf_dmd	Demanded fan speed	rpm
PCNFR_dmd	Demanded corrected fan speed	rpm
W31	HPT coolant bleed	lbm/s
W32	LPT coolant bleed	lbm/s

in practice. In this case study, we focus on the degradation modeling of aircraft turbofan engines based on multiple sensor signals.

#### 4.1. Data description

The data was generated by C-MAPSS, a widely used simulation platform to study the degradation of large commercial aircraft engines (Saxena *et al.*, 2008; Sarkar *et al.*, 2011). The dataset contains 100 historical units and 100 in-field units under the same failure mode and the same operation condition. There are 21 sensors monitoring each unit on a variety of metrics such as temperature and pressure. The detailed description of the sensors is provided in Table 1. Sensor measurements are continuously collected until the failure of the unit occurs. For historical units, the average number of available measurements from each sensor for each unit is 206. For each in-field unit, the sensor measurements are available up to some time point before the failure and the average number of measurements from each sensor for each unit is 131. The actual RULs of in-field units are recorded in a separate file. Our task is to construct a degradation model to predict the RULs of in-field units and compare with the true RULs to assess the prognostic performance of the model. The dataset is available online (Saxena and Goebel, 2008).

In the literature, there are a number of studies on this dataset, and thus we can compare our result with existing ones. Specifically, the recent studies of Fang, Gebraeel and Paynabar (2017) and Song *et al.* (2018) are selected as benchmark methods here as their results are among the best in the literature. Song *et al.* (2018) proposed a kernel HI-based method to integrate the HI-based method with kernel methods in order to extend the fusion function from linear to nonlinear functions, i.e., the HI could be constructed by a nonlinear combination of sensor signals based on a certain kernel function. Specifically, they constructed the HI in a way such that the “quality” of the constructed HI was maximized, which was measured by the signal-to-noise ratio for degradation signals (SNR<sup>d</sup>). The kernel HI-based method is



**Figure 8.** Comparison of the average RUL prediction errors on the in-field units using our proposed method with LR and the kernel HI-based method with linear kernel function based on 11 sensors.

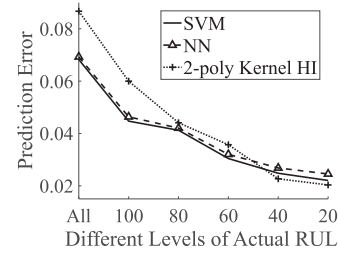
of great interest, as the HI method can be regarded as a special case of our proposed method. The second benchmark study (Fang, Gebraeel and Paynabar, 2017) utilizes FPCA to extract features from sensor signals and constructs a (log)-location-scale regression model for RUL prediction.

#### 4.2. Result and comparison

Song *et al.* (2018) manually selected 11 out of the 21 sensors for degradation modeling (i.e., T24, T50, P30, Nf, Ps30, phi, Nrf, BPR, htBleed, W31, and W32), whereas Fang, Gebraeel and Paynabar (2017) considered all 21 sensors. Since we are interested in comparing with the kernel HI method based on the same set of sensors, we first analyze the 11 sensors selected by Song *et al.* (2018) and compare our result with the kernel HI method. Then, we conduct an automatic sensor selection, analyze the selected sensors, and compare our result with both benchmark methods (Fang, Gebraeel and Paynabar, 2017; Song *et al.*, 2018). In this way, we can verify if our sensor selection method can improve the prognostic result.

We adopt the same preprocessing procedure as in Song *et al.* (2018). Specifically, we standardize the sensor measurements after taking a logarithm transformation. The quadratic degradation model  $\eta_j(t, \Gamma_{ij}) = [1, t, t^2]\Gamma_{ij}$  is chosen for each transformed sensor signal, as it provides a good fit as shown in existing studies (Liu *et al.*, 2017; Song *et al.*, 2018). Each sensor is separately modeled for simplicity with the random-effect parameters  $\Gamma_{ij}$  assumed to be normal, i.e.,  $\Gamma_{ij} \sim N(\mu_j, \Sigma_j)$ , where the prior parameters  $\mu_j$  and  $\Sigma_j$  can be estimated from the historical units based on the two-phase algorithm proposed by Lu and Meeker (1993). We also assume the noise term  $\varepsilon_{ij}(t) \sim N(0, \sigma_j^2)$ .

In this case study, we choose the tuning parameters  $N = 80$ ,  $\delta = 0.5$ , and  $K = 25$ . Specifically, we choose  $N\delta = 40$ , based on the sample standard deviation of the failure time of the historical units. Since all the true RULs of the in-field units are integers,  $\delta = 0.5$  will be precise enough for our model, and thus  $N = 80$ . With 100 historical units, a simple choice of  $K = 25$  would be able to produce a satisfactory performance. We consider three popular classifiers including LR, SVM, and Neural Network (NN). It can be shown that the LR is equivalent to the HI-based method where the HI is constructed by a linear combination of sensor signals and the failure threshold follows the logistic distribution. Thus, it is interesting to compare the LR result with the kernel



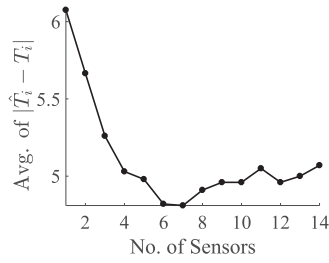
**Figure 9.** Comparison of the average RUL prediction errors on the in-field units using our proposed method with SVM and NN, and kernel HI-based method with second-order polynomial kernel function based on 11 sensors.

HI-based method where a linear kernel function is used, i.e., the HI is constructed by a linear combination of sensor signals. Despite the similarity, fundamental differences exist including: (i) the LR is estimated by solving Equation (4) while the HI is constructed by maximizing the SNR<sup>d</sup> metric; and (ii) the use of LR requires the modeling of each sensor signal whereas the HI-based method only models the constructed HI. These differences lead to different prognostic results. Song *et al.* (2018) also used a second-order polynomial kernel function to explore a nonlinear combination of sensor signals and improved the prognostic result. Accordingly, we choose nonlinear classifiers including a SVM with the second-order polynomial kernel and a NN with a single hidden layer for comparison. For the NN, the sigmoid function is used as the activation function for the hidden layer, and the number of neurons is determined to be 10 based on the 10-fold cross validation.

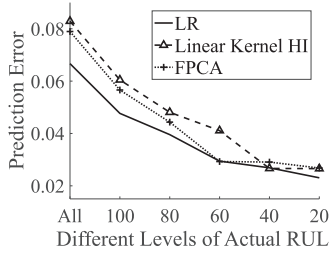
In Figure 8, we compare the prognostic results of the proposed method with the LR classifier and the kernel HI-based method with linear kernel function based on the 11 sensors. The x-axis represents different levels of actual RUL. For example, “80” means that only in-field units with an actual RUL less or equal to 80 are considered and “All” means that all in-field units are considered. The y-axis denotes the RUL prediction error defined in Equation (10). The curves represent the average prediction errors on different levels of actual RUL using different methods. It is clear that the LR outperforms the kernel HI-based method with a linear kernel function. A clear decreasing trend can be observed in this figure, indicating that the prediction error decreases with less actual RUL. This is reasonable, since with less actual RUL, more sensor measurements are collected and we only need to predict over a shorter period. Similarly, in Figure 9, we compare the prognostic results of our proposed framework with SVM and NN, and the kernel HI-based method with a second-order polynomial kernel based on the 11 sensors. This figure shows that SVM and NN perform better than the kernel HI-based method when the actual RULs of in-field units are medium or large. When the actual RUL is small, the prediction errors of SVM and NN are slightly larger.

The results shown in Figure 8 and Figure 9 are all based on the 11 sensors as identified in Song *et al.* (2018). Next, we aim to conduct sensor selection using the proposed framework and try to improve our prognostic results. Specifically, we choose LR with adaptive LASSO penalty for sensor selection. Since 7 out of the 21 sensors are constant

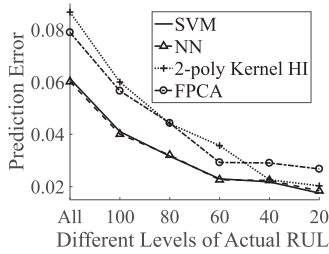




**Figure 10.** Average absolute difference between predicted RULs and actual RULs in 10-fold cross validation with different number of selected sensors.



**Figure 11.** Comparison of the average RUL prediction errors on the in-field units using our proposed method with LR based on the selected seven sensors and the benchmark methods.



**Figure 12.** Comparison of the average RUL prediction errors on the in-field units using our proposed method with SVM and NN based on the selected seven sensors and the benchmark methods.

and do not contain any information, we exclude the seven constant sensors in advance. The adaptive LASSO can produce a solution path containing different subsets of sensors corresponding to different tuning parameters  $\lambda$ , and these subsets contain different number of sensors from 1 to 14. For each subset of sensors in the solution path, we use 10-fold cross validation to evaluate the average absolute difference between the predict RULs and the actual RULs based on the historical units, i.e., the average of  $|\hat{T}_i - T_i|$ . The best subset of sensors is selected as the one with minimum absolute difference, which is shown in Figure 10.

As a result, seven sensors are selected including T24, T50, Nc, Ps30, phi, BPR, and W32. Based on the selected seven sensors, the prognostic results of our proposed framework with LR, SVM, and NN are updated, and the comparisons are shown in Figure 11 and Figure 12. The result of Fang, Gebraeel and Paynabar (2017) is also included in the two figures as “FPCA” for comparison. For the LR, the prognostic result remains similar as before, and actually, there is a slight improvement for in-field units with moderate RULs. More improvement is observed for SVM and NN. In both figures, our proposed method performs consistently

better than the two benchmark methods. These results verify the effectiveness of our proposed method in sensor selection and prognostic analysis.

## 5. Conclusion

In condition monitoring, multiple sensors are widely used to collect multiple signals from the same unit simultaneously, so as to improve the estimation of the degradation status and predict the RUL more accurately. However, most existing studies on degradation modeling focus on analyzing a single sensor signal, and the literature still lacks a generic multisensor degradation model that is tailored for degradation process, is flexible enough to explore different relationships between the underlying degradation status and the sensor signals, is suitable for asynchronous sensor signals, and is capable of screening out non-informative sensor signals automatically. In this article, we aim to fill the gap in the literature and propose a generic framework for multisensor degradation modeling, which can be viewed as an extension of the degradation model from one-dimensional space to multi-dimensional space. There are two primary tasks involved in the proposed framework: (i) modeling the underlying paths of multiple sensor signals; and (ii) estimating the failure surface in the multi-dimensional space. Although the first task can be accomplished by modeling the underlying path of each sensor signal with an extension that considers the correlation among signals, the second task has not been investigated before and leads to great challenges, since the failure surface may be unknown, complex and random in practice.

Our innovative idea is to transform the multisensor degradation modeling problem into a supervised classification problem. Since a variety of classifiers can be used to define the relation between the underlying signal paths and the degradation status (fail or not), the proposed framework gains great flexibility in defining the failure surface in the multi-dimensional space. The HI-based methods in the literature turn out to be only a special case of the proposed framework. In addition, the proposed framework is also capable of performing sensor selection by incorporating a variable selection algorithm and can deal with asynchronous multiple sensor signals. We develop classifier estimation methods, where existing software packages can be directly utilized for easy implementation. A series of simulation studies are conducted to evaluate the proposed method from different aspects and provide guidelines for choosing the tuning parameters. A case study on the degradation of aircraft engines is also conducted, and the results are compared with existing benchmarks, which shows better prognostic performance of the proposed framework.

In the future, there are several topics worth investigating. First, this paper takes samples with equal space as in Equation (3). However, this strategy may be further improved, since measurements close to failure are usually more critical than others. Therefore, an improved sampling strategy is in need to ensure the optimal balance between the prognostic performance of the framework and

computational efficiency. Second, the proposed framework involves several approximations and tuning parameters. In addition to the numerical studies described in this article, it is desired that theoretical analyses are conducted to evaluate how the approximations and tuning parameters affect the accuracy of RUL prediction in the proposed framework. Third, although the proposed framework allows a simultaneous modeling of multiple sensor signals through defining the prior  $p(\Gamma_i)$ , the literature on the construction of the prior  $p(\Gamma_i)$  in a unified manner is still sparse. It is of great interest to develop a systematic approach for specifying  $p(\Gamma_i)$  and analyze the effect of  $p(\Gamma_i)$  on the prognostic performance of the proposed method.

## Funding

This work was supported in part by the office of Naval Research under Grant N00014-17-1-2261, in part by the Department of Energy under award number DE-NE0008805, and in part by the National Science Foundation of China under award numbers 71771004, 71571003, and 71690232.

## Notes on contributors

**Changyue Song** received a B.S. degree in industrial engineering from Tsinghua University in 2012 and an M.S. degree in industrial engineering from Peking University in 2015. Currently he is a Ph.D. student at the Department of Industrial and Systems Engineering, University of Wisconsin-Madison. His research interests are focused on statistical modeling and improvement of complex systems.

**Kaibo Liu** received a B.S. degree in industrial engineering and engineering management from the Hong Kong University of Science and Technology in 2009, and an M.S. degree in statistics and Ph.D. degree in industrial engineering from the Georgia Institute of Technology in 2011 and 2013, respectively. Currently, he is an assistant professor at the Department of Industrial and Systems Engineering, University of Wisconsin-Madison. His research interests are focused on data fusion for process modeling, monitoring, diagnosis and prognostics and decision making. Dr. Liu is a member of IEEE, ASQ, INFORMS and IISE.

**Xi Zhang** received a B.S. degree in mechanical engineering and automation from Shanghai Jiaotong University, Shanghai, China, in 2006, and a Ph.D. degree in industrial engineering from the University of South Florida, Tampa, 2010. He is an associate professor with the Department of Industrial Engineering and Management, Peking University, Beijing. His research interests are physical-statistical modeling and analysis for process monitoring, diagnosis and optimization in complex dynamic systems. Dr. Zhang is a member of IEEE, INFORMS, IISE and ASQ.

## ORCID

Changyue Song  <http://orcid.org/0000-0001-6015-0981>

Xi Zhang  <http://orcid.org/0000-0003-3415-5345>

## References

Bae, S.J., Kuo, W. and Kvam, P.H. (2007) Degradation models and implied lifetime distributions. *Reliability Engineering and System Safety*, **92**(5), 601–608.  
 Bae, S.J. and Kvam, P.H. (2004) A nonlinear random-coefficients model for degradation testing. *Technometrics*, **46**(4), 460–469.

Baraldi, P., Mangili, F. and Zio, E. (2012) A Kalman filter-based ensemble approach with application to turbine creep prognostics. *IEEE Transactions on Reliability*, **61**(4), 966–977.  
 Brotherton, T., Grabill, P., Wroblewski, D., Friend, R., Sotomayer, B. and Berry, J. (2002) A testbed for data fusion for engine diagnostics and prognostics, in *Proceedings of IEEE Aerospace Conference 2002*, IEEE Press, Piscataway, NJ, pp. 3029–3042.  
 Chen, N. and Tsui, K.L. (2013) Condition monitoring and remaining useful life prediction using degradation signals: Revisited. *IIE Transactions*, **45**(9), 939–952.  
 Fang, X., Gebraeel, N. and Paynabar, K. (2017) Scalable prognostic models for large-scale condition monitoring applications. *IIE Transactions*, **49**(7), 698–710.  
 Fang, X., Paynabar, K. and Gebraeel, N. (2017) Multistream sensor fusion-based prognostics model for systems with single failure modes. *Reliability Engineering & System Safety*, **159**, 322–331.  
 Gebraeel, N.Z. (2006) Sensor-updated residual life distributions for components with exponential degradation patterns. *IEEE Transactions on Automation Science and Engineering*, **3**(4), 382–393.  
 Gebraeel, N.Z., Lawley, M.A., Li, R. and Ryan, J.K. (2005) Residual-life distributions from component degradation signals: A Bayesian approach. *IIE Transactions*, **37**(6), 543–557.  
 Guyon, I. and Elisseeff, A. (2003) An introduction to variable and feature selection. *Journal of Machine Learning Research*, **3**, 1157–1182.  
 Hall, D.L. and Llinas, J. (1997) An introduction to multisensor data fusion. *Proceedings of the IEEE*, **85**, 6–23.  
 Hastie, T., Tibshirani, R. and Friedman, J.H. (2009) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, second edition, Springer, New York, NY.  
 Hu, C., Youn, B.D., Wang, P. and Yoon, J.T. (2012) Ensemble of data-driven prognostic algorithms for robust prediction of remaining useful life. *Reliability Engineering and System Safety*, **103**, 120–135.  
 Japkowicz, N. (2000) The class imbalance problem: significance and strategies, in *Proceedings of the 2000 International Conference on Artificial Intelligence: Special Track on Inductive Learning*, Las Vegas, Nevada.  
 Jardine, A.K., Lin, D. and Banjevic, D. (2006) A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, **20**(7), 1483–1510.  
 Kim, M., Song, C. and Liu, K. (2019) A generic health index approach for multisensor degradation modeling and sensor selection. *IEEE Transactions on Automation Science and Engineering*, in press, DOI: 10.1109/TASE.2018.2890608  
 Knight, K. and Fu, W. (2000) Asymptotics for LASSO-type estimators. *The Annals of Statistics*, **28**(5), 1356–1378.  
 Liu, K., Chehade, A. and Song, C. (2017) Optimize the signal quality of the composite health index via data fusion for degradation modeling and prognostic analysis. *IEEE Transactions on Automation Science and Engineering*, **14**(3), 1504–1514.  
 Liu, K., Gebraeel, N.Z. and Shi, J. (2013) A data-level fusion model for developing composite health indices for degradation modeling and prognostic analysis. *IEEE Transactions on Automation Science and Engineering*, **10**(3), 652–664.  
 Liu, K. and Huang, S. (2016) Integration of data fusion methodology and degradation modeling process to improve prognostics. *IEEE Transactions on Automation Science and Engineering*, **13**(1), 344–354.  
 Loutas, T.H., Roulias, D. and Georgoulas, G. (2013) Remaining useful life estimation in rolling bearings utilizing data-driven probabilistic e-support vectors regression. *IEEE Transactions on Reliability*, **62**(4), 821–832.  
 Lu, C.J. and Meeker, W.O. (1993) Using degradation measures to estimate a time-to-failure distribution. *Technometrics*, **35**(2), 161–174.  
 Meeker, W.Q. and Escobar, L.A. (1998) *Statistical Methods for Reliability Data*, Wiley, New York, NY.  
 Nelson, W. (1990) *Accelerated Testing Statistical Models, Test Plans and Data Analysis*, Wiley, New York, NY.  
 Pinheiro, J.C. and Bates, D.M. (1995) Approximations to the log-likelihood function in the nonlinear mixed-effects model. *Journal of Computational and Graphical Statistics*, **4**(1), 12–35.

- Saha, B., Goebel, K. and Christophersen, J. (2009) Comparison of prognostic algorithms for estimating remaining useful life of batteries. *Transactions of the Institute of Measurement and Control*, **31**(3-4), 293–308.
- Sarkar, S., Jin, X. and Ray, A. (2011) Data-driven fault detection in aircraft engines with noisy sensor measurements. *Journal of Engineering for Gas Turbines and Power*, **133**(8), 081602.
- Saxena, A. and Goebel, K. (2008) Turbofan engine degradation simulation data set, NASA Ames Research Center, Moffett Field, CA. Available at <https://ti.arc.nasa.gov/tech/dash/groups/pcoc/prognostic-data-repository/>. Accessed 30 April 2018.
- Saxena, A., Goebel, K., Simon, D. and Eklund, N. (2008) Damage propagation modeling for aircraft engine run-to-failure simulation, in *Proceedings of the International Conference of Prognostics and Health Management 2008*, IEEE, Denver, CO, pp. 1–9.
- Si, X.S., Wang, W., Hu, C.H. and Zhou, D.H. (2011) Remaining useful life estimation – A review on the statistical data driven approaches. *European Journal of Operation Research*, **213**(1), 1–14.
- Song, C., and Liu, K. (2018) Statistical degradation modeling and prognostics of multiple sensor signals via data fusion: A composite health index approach. *IISE Transactions*, **50**(10), 853–867.
- Song, C., Liu, K. and Zhang, X. (2018) Integration of data-level fusion model and kernel methods for degradation modeling and prognostic analysis. *IEEE Transactions on Reliability*, **67**(2), 640–650.
- Tian, Z. (2012) An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring. *Journal of Intelligent Manufacturing*, **23**(2), 227–237.
- Tibshirani, R. (1996) Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society, Series B (Methodological)*, **58**(1), 267–288.
- Weiss, G.M. and Provost, F. (2001) The effect of class distribution on classifier learning: An empirical study. Technical Report ML-TR-44, Department of Computer Science, Rutgers University, Piscataway, NJ.
- Xu, Z., Ji, Y. and Zhou, D. (2008) Real-time reliability prediction for a dynamic system based on the hidden degradation process identification. *IEEE Transactions on Reliability*, **57**(2), 230–242.
- Ye, Z.S. and Xie, M. (2015) Stochastic modeling and analysis of degradation for highly reliable products. *Applied Stochastic Models in Business and Industry*, **31**(1), 16–32.
- Yu, H. (2006) Designing an accelerated degradation experiment with a reciprocal Weibull degradation rate. *Journal of Statistical Planning and Inference*, **136**(1), 282–297.
- Zhou, R., Serban, N. and Gebraeel, N. (2014) Degradation-based residual life prediction under different environments. *The Annals of Applied Statistics*, **8**(3), 1671–1689.
- Zhou, R., Serban, N., Gebraeel, N. and Müller, H.G. (2014) A functional time warping approach to modeling and monitoring truncated degradation signals. *Technometrics*, **56**(1), 67–77.
- Zou, H. (2006) The adaptive LASSO and its oracle properties. *Journal of American Statistical Association*, **101**(476), 1418–1429.

## Appendix

### More simulation results

Here we describe the detailed results of three more simulation studies regarding the tuning parameters  $N$ ,  $\delta$ , and  $K$ . In the first simulation, we fix  $\delta = 0.2$  and set  $N$  to be different values to see how the estimated coefficient of the classifier changes, where LR and SVM are considered as the classifiers. A set of randomly generated 100 historical units is used for classifier estimation. We use the estimated coefficient  $\hat{\beta}^{500}$  with  $N = 500$  as the baseline, and calculate the relative absolute difference  $\Delta_j^N$  for each entry  $j$  of the estimated coefficient  $\hat{\beta}^N$  as  $\Delta_j^N = |\hat{\beta}_j^N - \hat{\beta}_j^{500}| / \hat{\beta}_j^{500}$ . The results for LR and SVM are shown in Figure A1 and Figure A2, respectively. Similar to Figure 6 and Figure 7, we do not show the entry corresponding to Sensor 4. The figures indicate that the relative absolute difference decreases as  $N$  increases for both classifiers. Also, we can see that the relative absolute difference

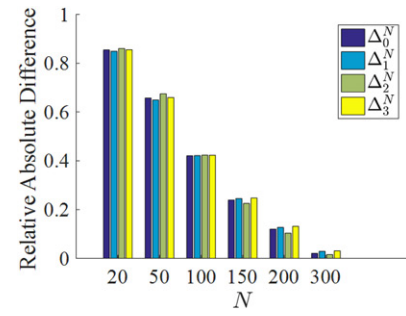


Figure A1. Sensitivity of the estimated coefficient of LR to  $N$ .

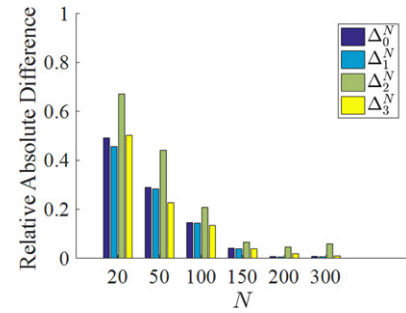


Figure A2. Sensitivity of the estimated coefficient of SVM to  $N$ .

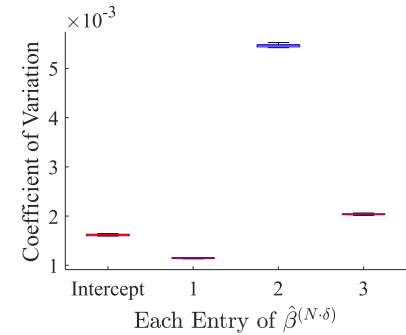


Figure A3. Coefficient of variation for each entry of the estimated coefficient of LR.

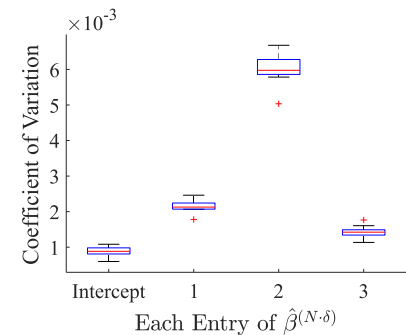


Figure A4. Coefficient of variation for each entry of the estimated coefficient of SVM.

of SVM is usually much smaller than LR. For SVM, the relative absolute difference becomes stable from  $N = 200$ , i.e., the estimated coefficient of SVM only changes little if we further increases  $N$  from 200. On the other hand, for LR, significant decrease can still be observed

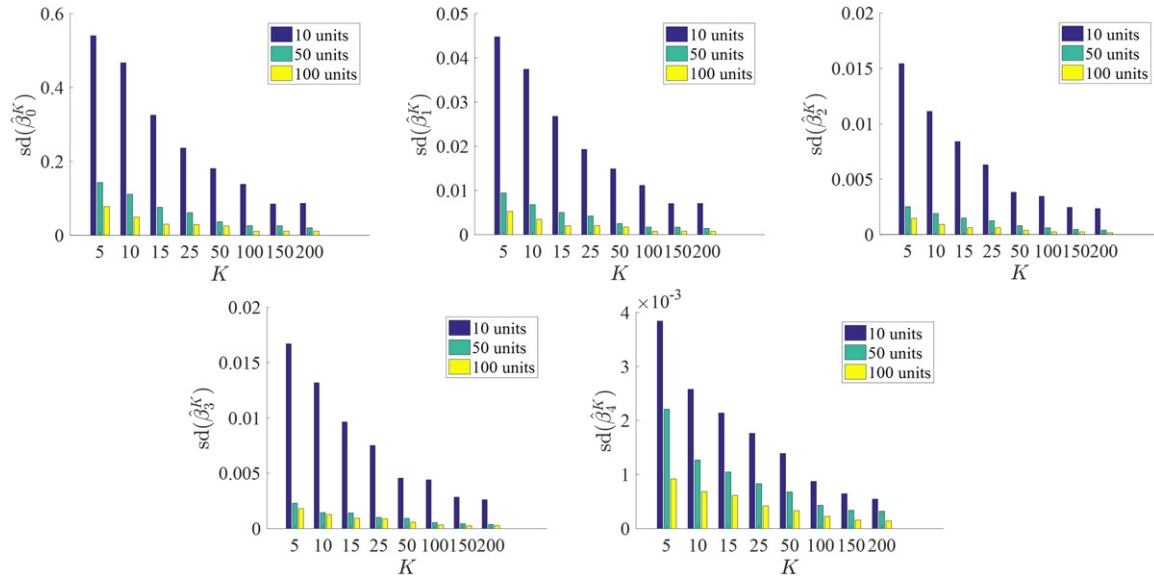


Figure A5. Effect of  $K$  with different number of historical units.

from  $N = 200$  to  $N = 300$ . Therefore, LR seems more sensitive to  $N$  in this case.

In the second simulation, we fix  $N \cdot \delta = 40$  and choose a series of pairs of  $(N, \delta)$  from  $N = 20$  to  $N = 500$ . For each pair of  $(N, \delta)$ , we estimate the coefficient  $\hat{\beta}^{(N, \delta)}$  of the classifier. Then for each entry of  $\hat{\beta}^{(N, \delta)}$ , we calculate the Coefficient of Variation (CV) across all pairs of  $(N, \delta)$ , which is defined as the standard deviation divided by the average. A small value of CV indicates small difference between  $\hat{\beta}^{(N, \delta)}$  with different pairs of  $(N, \delta)$ . We repeat this procedure for several times to obtain the average and standard deviation of CV. The boxplot of CV for LR and SVM are shown in Figure A3 and Figure A4, respectively, which shows that only very small difference exists for different pairs of  $(N, \delta)$  for both classifiers. Combined with the previous simulation, we conclude that the value of  $N \cdot \delta$  is very important for the estimation performance of our proposed method.

The last simulation that we consider is the effect of  $K$  with different number of historical units  $m$ . Specifically, for  $m$  randomly generated historical units, we repeat the classifier estimation procedure of the proposed framework using Monte Carlo method for 50 times. Each time we randomly draw a different set of  $K$  random samples  $\Gamma_i^{(k)}$  from the posterior distribution  $p(\Gamma_i | L_i)$  and estimate the coefficient of the LR classifier to be  $\hat{\beta}^K$ . In this way, we can calculate the standard deviation for each entry of  $\hat{\beta}^K$  across the 50 repetitions. If the value of  $K$  is large enough, the estimated  $\hat{\beta}^K$  should be stable in different repetitions and thus the standard deviation should be small. The result is shown in Figure A5. Generally, the standard deviation decreases as  $K$  increases, which agrees with our intuition. Furthermore, the standard deviation becomes smaller with more historical units. Therefore, with more historical units available, a smaller value of  $K$  would be enough for an accurate approximation for the Monte Carlo method.