Proceedings of the ASME 2020
International Design Engineering Technical Conferences
and Computers and Information in Engineering Conference
IDETC/CIE2020
August 16-19, 2020, St. Louis, MO, USA

IDETC2020-19246

PREDICTING AVERAGE PRODUCT LIFETIME USING PHYSICS-BASED GAUSSIAN PROCESS METHOD IN EARLY DESIGN STAGE

Xinpeng Wei, Daoru Han
Department of Mechanical and Aerospace Engineering
Missouri University of Science & Technology
Rolla, MO, USA

Xiaoping Du[1]
Department of Mechanical and Energy Engineering
Indiana University – Purdue University Indianapolis
Indianapolis, IN, USA

ABSTRACT

Average lifetime, or mean time to failure (MTTF), of a product is an important metric to measure the product reliability. Current methods of evaluating MTTF are mainly statistics or data based. They need lifetime testing on a number of products to get the lifetime samples, which are then used to estimate MTTF. The lifetime testing, however, is expensive in terms of both time and cost. The efficiency is also low because it cannot be effectively incorporated in the early design stage where many physics-based models are available. We propose to predict MTTF in the design stage by means of physics-based models. The advantage is that the design can be continually improved by changing design variables until reliability measures, including MTTF, are satisfied. Since the physics-based models are usually computationally demanding, we face a problem with both big data (on the model input side) and small data (on the model output side). We develop an adaptive supervised training method based on Gaussian process regression, and the method can then quickly predict MTTF with minimized number of calling the physics-based models. The effectiveness of the method is demonstrated by two examples.

1. INRODUCTION

In reliability engineering [1-5], the average lifetime, or mean time to failure (MTTF), is an important metric of product reliability [1, 6]. Statistics-based methods [7, 8] are widely used to estimate MTTF. The methods need lifetime testing on a number of products to obtain the lifetime samples, which are then used to estimate the average lifetime by statistical analysis. The methods are generally expensive in three aspects. First, lifetime testing is time-consuming when the actual product lifetime is very long such as years. Although the accelerated life testing [9] can reduce the testing time, the results may not reflect the reliability of the product in normal use conditions. Second, the budget of the testing is expensive and even unaffordable when the products themselves are expensive. Third, testing is performed or lifetime data are collected from field after the product was made. It is too late and more costly to fix reliability issues if MTTF is shorter than expected. It is desirable to predict MTTF during the early design stage.

Direct lifetime data, however, are rarely available during the design stage. Physics-based methods [10] then play an important role to deal with this problem. The methods use limit-state functions, which are computational models derived from physical principles, to predict the states of the components and subsystems of the product with respect to potential failure modes [11]. With the computational models for the failure modes, physics-based methods are much more efficient than the statistics-based methods. They can predict reliability performance for a given design. If the reliability measures, including MTTF, do not meet the design requirements, design variables will be changed until the reliability requirements are met. Physics-based methods are therefore a powerful tool to support design for reliability [12-16].

Physics-based methods were originally developed for structural reliability analysis [10]. In the last decades, many new physics-based reliability methods have been developed. These methods cover a wide range of applications, from component reliability to system reliability [10], and from time-independent reliability to time-dependent reliability [17-19] and time- and space-dependent reliability [20].

Computational models, such as a finite element analysis model [21], are usually computationally expensive. We usually know distributions of random input variables, and it is possible for us to generate many random samples for the input variables. In this sense we have big data. On the other hand, we can afford

[1] 723 W. Michigan Street, Indianapolis, IN 46202-5195, USA, Email: duxi@iu.edu

to run the computational models only a limited number of times, and then we have small data for the responses. For this reason, machine learning methodologies have been increasingly used for reliability analysis. For example, Gaussian process (GP) method for quantifying model structure uncertainty [22, 23]; the support vector machine (SVM) method for estimating rare event probabilities [11], and other methods for predicting component and system reliability [24].

In this study, we extend the physics-based methods to predict MTTF for a product. Since this task needs more calls of the computational model than a regular reliability analysis, we also rely on machine learning to maintain computational efficiency. Specifically, we employ the supervised machine learning method [25] and adaptively train a Gaussian process regression model [26] to approximate the computational function with respect to the basic random input variables. A learning function is developed to guide adding training points. Once the learning is finished, the MTTF of the product is obtained.

The problem statement is given in Section 2. The proposed method is discussed in Section 3. In Section 4, we extend the proposed method to deal with problems involving random processes. Three examples are provided in Section 5, followed by conclusions in Section 6.

2. PROBLEM STATEMENT

The computational function for reliability analysis is called a limit-state function, which is given by

$$Y = G(\mathbf{X}, t) \tag{1}$$

where $\mathbf{X} = (X_1, X_2 \ldots, X_N)^T$ are N basic input random variables and t is time. Note that the input of $G(\cdot)$ may also include random processes, which can be transformed into functions with respect to random variables and t. Thus Eq. (1) does not lose generality. Y is in general a random process. The product fails once its response Y takes a negative value.

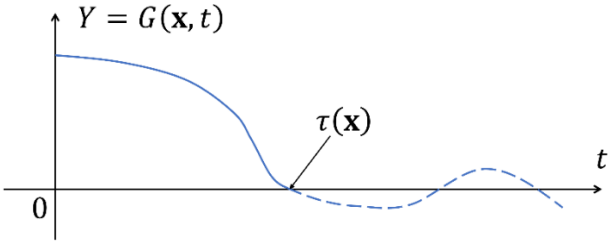


FIGURE 1: A SAMPLE OF THE LIMIT-STATE FUNCTION

Fig. 1 shows a sample of Y when $\mathbf{X}$ is fixed to a realization $\mathbf{x}$. When $t = \tau(\mathbf{x})$, Y takes a negative value at the first time, and hence $\tau(\mathbf{x})$ is called the first time to failure. If the product is non-repairable, $\tau(\mathbf{x})$ is the lifetime (given that $\mathbf{X} = \mathbf{x}$), and afterwards $Y(\mathbf{x}, t), t > \tau$ has no any physical meaning. Since $\tau(\mathbf{X})$ is dependent on the input random variables $\mathbf{X}$, it is also a random variable. The product's mean lifetime $\bar{\tau}$ or MTTF, is given by

$$\bar{\tau} = \mathrm{E}[\tau(\mathbf{X})] \tag{2}$$

where $\mathrm{E}(\cdot)$ represents an expectation.

The task of this study is to predict $\bar{\tau}$ efficiently and accurately. Mathematically, $\tau(\mathbf{X})$ is the first (or minimum) root of the following equation

$$G(\mathbf{X}, t) = 0 \tag{3}$$

Finding the minimum root of Eq. (3), however, may be computationally expensive when the limit-state function $G(\mathbf{X}, t)$ is an expensive black-box function. Therefore, developing an accurate and efficient first-root finder is a challenge.

3. THE PROPOSED METHOD

3.1. Overview of the proposed method

The main idea of the proposed method is to adaptively train a Gaussian process model (or Kriging model [27]) $\hat{G}(\mathbf{X}, t)$ for $G(\mathbf{X}, t)$. With $\hat{G}(\mathbf{X}, t)$, we can obtain the surrogate model $\hat{\tau}(\mathbf{X})$ of $\tau(\mathbf{X})$ at the same time. Since $\hat{\tau}(\mathbf{X})$ is computationally cheap, we can calculate $\bar{\tau}$ using Monte Carlo simulation (MCS) [28].

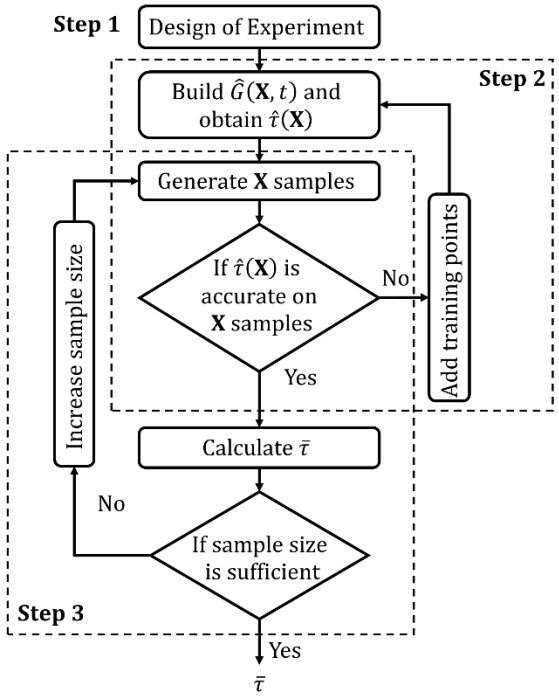


FIGURE 2: BRIEF FLOWCHART OF THE PROPOSED METHOD

Training $\hat{G}(\mathbf{X}, t)$ should be task-oriented in order to improve efficiency. We develop a learning function and a stopping criterion to fulfill the task-oriented training. Fig. 2 shows a brief flowchart of the proposed method. There are

mainly three steps. Step 1 is the design of experiments. It generates the initial training points for $\hat{G}(\mathbf{X},t)$. In Step 2, $\hat{G}(\mathbf{X},t)$ is adaptively refined by adding new training points. A learning function and a stopping criterion are developed to find the new training points and judge when to terminate the training. In Step 3, the sample size of $\mathbf{X}$, and hence of $\hat{\tau}(\mathbf{X})$, is adaptively enlarged until $\bar{\tau}$ is estimated with sufficiently high fidelity. The three steps are discussed in details in Subsections 3.2 through 3.4.

3.2. Design of experiments for initial surrogate model

The principle of the design of experiments for building a Kriging model is to spread the initial training points evenly. Commonly used sampling methods include random sampling, Latin hypercube sampling, and Hammersley sampling [29]. In this study, we employ the Hammersley sampling method because it has better uniformity properties over a multidimensional space [30]. Since the dimension of the entire input vector $(\mathbf{X},t)$ is $N+1$, the Hammersley sampling method generates initial training points in a hypercube $[0,1]^{N+1}$. To get initial training points of $\mathbf{X}$, we can simply use the inverse probability method to transform the training points from the hypercube space to the X-space. As for the initial training points of t, we treat t as if it was a uniform random variable and could also be transformed the interval $[0,1]$ to the time interval $[0,T]$. We assume that T is sufficiently large so that Eq. (3) has at least a root in $[0,T]$. The initial training points $\mathbf{x}^{\text{in}}$ of $\mathbf{X}=(X_1,X_2\ldots,X_N)^T$ are

$$\mathbf{x}^{\text{in}}=\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \ddots & x_N^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \ddots & x_N^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(n_{\text{in}})} & x_2^{(n_{\text{in}})} & \ddots & x_N^{(n_{\text{in}})} \end{bmatrix} \tag{4}$$

where n_{in} is the total number of initial training points. With $\mathbf{x}^{\text{in}}$ and the initial training points $\mathbf{t}^{\text{in}}$ of t, we then obtain initial training points $\mathbf{y}^{\text{in}}$ of Y by calling Eq. (1). Finally, we get the initial training set $(\mathbf{x}^{\text{trn}};\mathbf{t}^{\text{trn}};\mathbf{y}^{\text{trn}})=(\mathbf{x}^{\text{in}};\mathbf{t}^{\text{in}};\mathbf{y}^{\text{in}})$, where the superscript *trn* and *in* represents the general training points and initial training points, respectively.

3.3. Adaptive training

With the initial training points $(\mathbf{x}^{\text{in}};\mathbf{t}^{\text{in}};\mathbf{y}^{\text{in}})$, we can build an initial Kriging model $\hat{G}(\mathbf{X},t)$ to approximate $G(\mathbf{X},t)$. The initial $\hat{G}(\mathbf{X},t)$ is generally not accurate. The task of the adaptive training is to add training points to refine $\hat{G}(\mathbf{X},t)$ sequentially and adaptively. Specifically, a task-oriented learning function and stopping criterion are developed.

For numerical computation, $[0,T]$ is evenly discretized into m points $\mathbf{t}=(t_1,t_2,\ldots,t_m)^T$. Then $\tau(\mathbf{x})$ is approximated by

$$\hat{\tau}(\mathbf{x})=\min\{t\in\mathbf{t}\,|\,\hat{G}(\mathbf{x},t)\le 0\} \tag{5}$$

To estimate $\bar{\tau}$, we first randomly generate n_{s} samples $\mathbf{x}^{\text{s}}$ of $\mathbf{X}$. Then $\bar{\tau}$ is approximated by

$$\bar{\tau}=\frac{1}{n_{\text{s}}}\sum_{i=1}^{n_{\text{s}}}\hat{\tau}\left(\mathbf{x}^{(i)}\right) \tag{6}$$

where $\mathbf{x}^{(i)}$ is the i^{th} sample of $\mathbf{X}$. Eq. (6) can yield accurate $\bar{\tau}$ only when two conditions are satisfied. First, the sample size n_{s} is sufficiently large. How to determine n_{s} will be given in Subsection 4.3. Second, the model $\hat{\tau}(\mathbf{X})$ is accurate at all the samples $\mathbf{x}^{\text{s}}$. How to add training samples to refine $\hat{G}(\mathbf{x},t)$ so that the second condition is satisfied is the key of the adaptive training.

Intuitively, $\hat{\tau}(\mathbf{X})$ is accurate as long as $\hat{G}(\mathbf{X},t)$ approximates $G(\mathbf{X},t)$ accurately. However, training $\hat{G}(\mathbf{X},t)$ in this way is not efficient and it disobeys the task-oriented rule. In fact, $t^*\in\mathbf{t}$ is an accurate solution to Eq. (5) as long as the signs of $\{G(\mathbf{x},t)|,t\in\mathbf{t},t\le t^*\}$ are predicted accurately. For example, if $\hat{G}(\mathbf{x},t)$ can accurately predict the signs of $G(\mathbf{x},t_j),j=1,2,3,4,5$ as $(+,+,+,+,-)$, then t_5 is definitely the accurate solution to Eq. (5). We do not need to care if $\hat{G}(\mathbf{x},t)$ predicts the specific values of $G(\mathbf{x},t_j),j=1,2,3,4,5$ or the signs of $G(\mathbf{x},t_j),j\ge 6$ accurately.

The well-known learning function U [31] is used to measure how accurate the sign at a point is predicted. It is given by

$$U(\mathbf{x},t)=\frac{|\hat{G}(\mathbf{x},t)|}{\sigma(\mathbf{x},t)} \tag{7}$$

where $\sigma(\mathbf{x},t)$ is the square root of the Kriging prediction error. The Kriging model predicts $G(\mathbf{x},t)$ as a normal variable $N(\hat{G}(\mathbf{x},t),\sigma^2(\mathbf{x},t))$. If $\hat{G}(\mathbf{x},t)>0$, the sign of $G(\mathbf{x},t)$ is predicted to be positive. The probability that $\hat{G}(\mathbf{x},t)>0$ is $\Phi\left(\frac{\hat{G}(\mathbf{x},t)}{\sigma(\mathbf{x},t)}\right)$, where $\Phi(\cdot)$ is the cumulative distribution function (CDF) of a standard normal variable. Therefore, the probability that the sign of $G(\mathbf{x},t)$ is predicted accurately is $\Phi\left(\frac{\hat{G}(\mathbf{x},t)}{\sigma(\mathbf{x},t)}\right)$. Similarly, if $\hat{G}(\mathbf{x},t)<0$, the sign of $G(\mathbf{x},t)$ is predicted to be negative, and the probability that the prediction is accurate is $\Phi\left(\frac{-\hat{G}(\mathbf{x},t)}{\sigma(\mathbf{x},t)}\right)$. Combining both cases, $\Phi\left(\frac{|\hat{G}(\mathbf{x},t)|}{\sigma(\mathbf{x},t)}\right)$ is the probability that the sign of $G(\mathbf{x},t)$ is accurately predicted. It is why $U(\mathbf{x},t)=\frac{|\hat{G}(\mathbf{x},t)|}{\sigma(\mathbf{x},t)}$ can measure how accurately the sign is predicted. (Note that $\Phi(\cdot)$ is an increasing function.)

To refine the Kriging model $\hat{G}(\mathbf{X},t)$, we should add training points where the accuracy is poor or U is small since a small U means that the chance of correctly predicting the sign of $G(\mathbf{x},t)$ is small. If $\mathbf{X}$ is fixed to $\mathbf{x}$, the next training point $(\mathbf{x},t^{\text{next}})$ is determined by

$$(\mathbf{x}, t^{\text{next}}) = \arg\min_{t\in\mathbf{t}, t\le\hat{\tau}(\mathbf{x})} U(\mathbf{x}, t) \tag{8}$$

Since there are n_s samples of $\mathbf{X}$, Eq. (8) determines n_s points. Among them, the point with minimal U is finally selected as the next training point $(\mathbf{x}^{\text{next}}, t^{\text{next}})$, which is determined by

$$(\mathbf{x}^{\text{next}}, t^{\text{next}}) = \arg\min_{\mathbf{x}\in\mathbf{x}^s} \left\{ \min_{t\in\mathbf{t}, t\le\hat{\tau}(\mathbf{x})} U(\mathbf{x}, t) \right\} \tag{9}$$

With the learning function given in Eq. (9), we can add training points to update $(\mathbf{x}^{\text{trn}}, \mathbf{t}^{\text{trn}}, \mathbf{y}^{\text{trn}})$ and $\hat{G}(\mathbf{X}, t)$ sequentially until a stopping criterion is satisfied.

The direct use of $U(\mathbf{x}, t)$ and hence Eq. (9), however, may result in duplicate training points. In other words, the next training point determined by Eq. (9) may be the one among $(\mathbf{x}^{\text{trn}}, \mathbf{t}^{\text{trn}}, \mathbf{y}^{\text{trn}})$. Once this happens, the adaptive training fails. Theoretically, because Kriging is an exact interpolator, if a point $(\mathbf{x}^*, t^*, y^*)$ is among the training set $(\mathbf{x}^{\text{trn}}, \mathbf{t}^{\text{trn}}, \mathbf{y}^{\text{trn}})$, the Kriging model $\hat{G}(\mathbf{X}, t)$ will predict $G(\mathbf{x}^*, t^*)$ exactly as y^*, i.e., $\hat{G}(\mathbf{x}^*, t^*) = y^*$ and $\sigma(\mathbf{x}^*, t^*) = 0$. As a result, $U(\mathbf{x}^*, t^*) = +\infty$, $(\mathbf{x}^*, t^*)$ will never be selected by Eq. (9) as the next training point, and the duplicate training points will never be encountered. However, due to the numerical error, $\sigma(\mathbf{x}^*, t^*)$ is not exactly zero but a small positive number. In this case, if $\hat{G}(\mathbf{x}^*, t^*)$ is smaller than $\sigma(\mathbf{x}^*, t^*)$, we will have $U(\mathbf{x}^*, t^*) < 1$, and Eq. (9) may select $(\mathbf{x}^*, t^*)$ as the next training point, leading to the duplicate training points.

Another problem caused by U is that added training points may cluster together [19]. It will make the correlation matrix of the Kriging model ill-conditioned. If this happens, some of the clustered training points will have negligible effect on the refinement of $\hat{G}(\mathbf{X}, t)$, and the adaptive training may not converge. Hu and Mahadevan [19] proposed to disqualify those points to be candidate training points if they are highly correlated with any one of the existing training points. Specifically, the candidate training points are shrunk from point set $\mathbf{x}^s \times \mathbf{t}$ to $\left\{ (\mathbf{x}, t) \in \mathbf{x}^s \times \mathbf{t} \middle| \max_{(\mathbf{x}', t')\in(\mathbf{x}^{\text{trn}}, \mathbf{t}^{\text{trn}})} \rho[(\mathbf{x}, t), (\mathbf{x}', t')] < \eta \right\}$, where $\rho(\cdot,\cdot)$ is the correlation coefficient used in Kriging model to describe the correlation of two points, and η is a hyperparameter. It guarantees that the candidate training points are sufficiently far away from the current training points, and thereby that the newly selected training point will not overlap or cluster with any one of the current training points.

We employ this method and then the learning function proposed in Eq. (9) is updated to

$$(\mathbf{x}^{\text{next}}, t^{\text{next}}) = \arg\min_{t\le\hat{\tau}(\mathbf{x}), (\mathbf{x}, t)\in\mathbf{C}} U(\mathbf{x}, t) \tag{10}$$

where

$$\mathbf{C} = \left\{ (\mathbf{x}, t) \in \mathbf{x}^s \times \mathbf{t} \middle| \max_{(\mathbf{x}', t')\in(\mathbf{x}^{\text{trn}}, \mathbf{t}^{\text{trn}})} \rho[(\mathbf{x}, t), (\mathbf{x}', t')] < \eta \right\}.$$

In addition to the learning function, the other important component of the adaptive training is the stopping criterion. Since the learning function can add training points iteratively to update $\hat{G}(\mathbf{X}, t)$, and hence $\hat{\tau}(\mathbf{x})$ in Eq. (5), a stopping criterion is necessary to terminate the iteration. Once the model $\hat{\tau}(\mathbf{X})$ is accurate on all the samples $\mathbf{x}^s$, we no longer add new training points. Therefore, the iteration ends if the following condition is satisfied

$$W > w \tag{11}$$

where $W = \min_{t\le\hat{\tau}(\mathbf{x}), (\mathbf{x}, t)\in\mathbf{C}} U(\mathbf{x}, t)$, and w is a hyperparameter and is recommended to set to 2. Generally, the larger is w, the more accurate will $\bar{\tau}$ be. Larger w, however, will lower the efficiency, so the selection of w needs a tradeoff. There is no rigorous theory to determine the best w, and we recommend 2 based on both our experience from many experiments and [31].

3.4. Adaptive sample size

Since the random sampling method is used to estimate $\bar{\tau}$ through Eq. (6), it is desirable to select a good sample size n_s. We use an initial sample size n_0 and then adaptively increase the sample size until $\bar{\tau}$ is obtained with a sufficiently high fidelity [32].

Since $\tau(\mathbf{X})$ is a random variable, the sample size needed to estimate its mean value $\bar{\tau}$ is dependent on its standard deviation σ_τ. With the sample size n_s, the deviation coefficient Γ of $\bar{\tau}$ is given by

$$\Gamma = \frac{\sigma_\tau}{\bar{\tau}\sqrt{n_s}} \tag{12}$$

where $\bar{\tau}$ is estimated by Eq. (6) and σ_τ is estimated by

$$\sigma_\tau = \sqrt{\frac{1}{n_s - 1}\sum_{i=1}^{n_s} [\hat{\tau}(\mathbf{x}^{(i)}) - \bar{\tau}]^2} \tag{13}$$

Eq. (12) shows that the larger is n_s, the smaller will Γ we have. A smaller Γ means that $\bar{\tau}$ is more accurately estimated by Eq. (6). $\bar{\tau}$ is said to be accurate if the following condition is satisfied

$$\Gamma \le \gamma \tag{14}$$

where γ is a threshold, which usually takes a small positive number, such as 0.005.

If the current n_s cannot satisfy Eq. (14), we should increase it. Combining Eq. (12) and Eq. (14), we have

$$n_s \ge \left(\frac{\sigma_\tau}{\bar{\tau}\gamma}\right)^2 \tag{15}$$

It means that at least a sample size of $\left(\frac{\sigma_\tau}{\bar{\tau}\gamma}\right)^2$ is necessary to guarantee Eq. (14). Let $n_1 = \text{ceil}\left[\left(\frac{\sigma_\tau}{\bar{\tau}\gamma}\right)^2\right]$, where ceil(·)

represents the operation to get the nearest larger integer. Then the number n_{add} by which n_{s} should be increased is given by

$$n_{\text{add}} = n_1 - n_{\text{s}} \tag{16}$$

However, when $\hat{G}(\mathbf{X}, t)$ is too rough at the first several adaptive training iterations, both $\bar{\tau}$ and σ_τ may have poor accuracy, and n_{add} given in Eq. (16) may be misleading. To deal with this problem, we set a threshold $\tilde{n}_{\text{add}}$ for n_{add}. Then Eq. (16) is updated to

$$n_{\text{add}} = \begin{cases} \tilde{n}_{\text{add}}, \text{if } n_1 - n_{\text{s}} > \tilde{n}_{\text{add}} \\ n_1 - n_{\text{s}}, \text{otherwise} \end{cases} \tag{17}$$

3.5. Implementation

In this subsection, we give the detailed procedure of the proposed method. The full flowchart is shown in Fig. 3. The total number n_{e} of function evaluations of $G(\mathbf{X}, t)$ is used to measure the main computational cost of the proposed method.

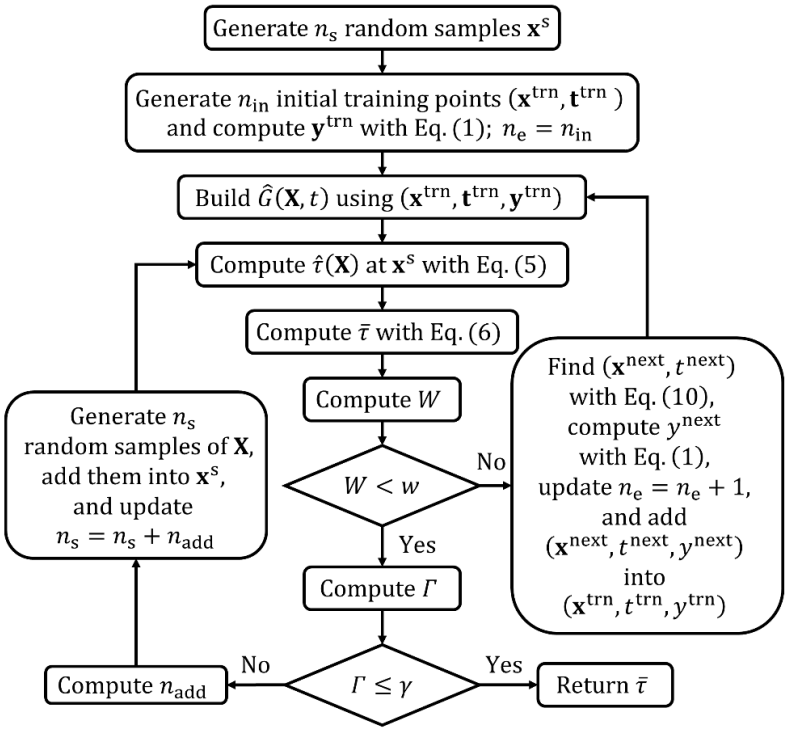


FIGURE 3: DETAILED FLOWCHART OF THE PROPOSED METHOD

4. EXTENSION TO PROBLEMS WITH INPUT RANDOM PROCESSES

When the limit-state function $G(\cdot)$ has input random processes, it is straightforward to employ the series expansion methods of the random processes, so that the above implementation of the proposed method can still work.

Let $\mathbf{H}(t)$ represents a vector of random processes, then the limit-state function is given by

$$Y = G(\mathbf{X}, \mathbf{H}(t), t) \tag{18}$$

To easily present the idea, we assume there is only one random process $H(t)$. Widely used series expansions for random fields include, the Karhunen-Loeve series expansion (K-L), the orthogonal series expansion (OSE), and the expansion optimal linear estimation method (EOLE) [33]. Since t is discretized into $\mathbf{t}$, the autocorrelation coefficient function of $H(t)$ is discretized into the autocorrelation coefficient matrix $\mathbf{M}_H$ with dimension $m \times m$. Then the EOLE expansion $H(\boldsymbol{\xi}, t)$ of $H(t)$ is given by

$$H(\boldsymbol{\xi}, t) = \mu_H(t) + \sigma_H(t) \sum_{k=1}^{m} \frac{\xi_k}{\sqrt{\lambda_k}} \mathbf{V}_k \mathbf{M}_H(:, k), t \in \mathbf{t} \tag{19}$$

where $\mu_H(t)$ is the mean value function of $H(t)$, $\sigma_H(t)$ is the standard deviation function of $H(t)$, $\xi_k, k = 1,2,\ldots,m$ are m independent standard Gaussian variables, λ_k is the k-th eigenvalue of $\mathbf{M}_H$, $\mathbf{V}_k$ is the k-th (row) eigenvector of $\mathbf{M}_H$, and $\mathbf{M}_H(:, k)$ is the k-th column of $\mathbf{M}_H$. Note that the eigenvalues are sorted from the largest to the smallest. Usually only the first m' $(m' \leq m)$ eigenvalues are significant. Therefore, Eq. (19) is practically truncated, and only the first m' orders are kept:

$$H(\boldsymbol{\xi},t)=\mu_H(t)+\sigma_H(t)\sum_{k=1}^{m'}\frac{\xi_k}{\sqrt{\lambda_k}}\mathbf{V}_k\mathbf{M}_H(:,k),t\in\mathbf{t} \quad (20)$$

With the truncated expansion in Eq. (20), Eq. (18) is rewritten as

$$Y=G(\mathbf{X},H(\boldsymbol{\xi},t),t) \quad (21)$$

or equivalently as

$$Y=G\left(\widetilde{\mathbf{X}},t\right) \quad (22)$$

where $\widetilde{\mathbf{X}}=(\boldsymbol{\xi},\mathbf{X})$. Eq. (22) shares the same format with Eq. (1) and hence the implementation given in Subsection 4.4 also works.

The direct implementation this way, however, may suffer from the curse of dimensionality. Since there are many random variables, i.e. $\boldsymbol{\xi}$, in the series expansion $H(\boldsymbol{\xi},t)$, the dimension of $\boldsymbol{\xi}$ and hence that of $G\left(\widetilde{\mathbf{X}},t\right)$ is high. As a result, the dimension of the Kriging model $\hat{G}\left(\widetilde{\mathbf{X}},t\right)$ is also high. The high dimensionality has as least two drawbacks. First, it is not computationally cheap anymore, losing its expected advantages. Second, more training points are needed to train the Kriging model. To overcome the drawbacks, we build a Kriging model $\hat{G}(\mathbf{X},H,t)$ with respect to $\mathbf{X}$, H, and t. Note that the entire random process H is treated as only one variable for $\hat{G}(\mathbf{X},H,t)$. Then the surrogate model $\hat{G}\left(\widetilde{\mathbf{X}},t\right)$ with respect to $\widetilde{\mathbf{X}}$ and t is obtained through

$$\hat{G}\left(\widetilde{\mathbf{X}},t\right)=\hat{G}[\mathbf{X},H(\boldsymbol{\xi},t),t] \quad (23)$$

Since the truncated series expansion $H(\boldsymbol{\xi},\mathbf{Z})$ in Eq. (20) has a simple closed-form expression, if $\hat{G}(\mathbf{X},H,t)$ is accurate and efficient, so will be $\hat{G}\left(\widetilde{\mathbf{X}},t\right)$ in Eq. **Error! Reference source not found.**. Since the dimension of $\hat{G}(\mathbf{X},H,t)$ is $(m'-1)$ lower than that of $\hat{G}\left(\widetilde{\mathbf{X}},t\right)$, it is more efficient to train $\hat{G}(\mathbf{X},H,t)$. To build $\hat{G}(\mathbf{X},H,t)$, we need the training points $\mathbf{h}^{\mathrm{trn}}$ of H. $\mathbf{h}^{\mathrm{trn}}$ can be obtained simply by substituting $(\boldsymbol{\xi}^{\mathrm{trn}},\mathbf{t}^{\mathrm{trn}})$ into Eq. (20). Similarly, when $\left(\tilde{\mathbf{x}}^{(\mathrm{next})},t^{(\mathrm{next})}\right)$ is determined by Eq. (10), the next training point $h^{(\mathrm{next})}$ of H is obtained by substituting $\left(\boldsymbol{\xi}^{(\mathrm{next})},\mathbf{z}^{(\mathrm{next})}\right)$ into Eq. (20). Note that $\tilde{\mathbf{x}}^{(\mathrm{next})}=\left(\boldsymbol{\xi}^{(\mathrm{next})},\mathbf{x}^{(\mathrm{next})}\right)$. When multiple input random processes are involved, the procedure of building and updating the surrogate model $\hat{G}$ is similar.

5. EXAMPLES

In this section, we use three examples to illustrate the proposed method. The first one is a math example with only one input random variable. It is designed to graphically show the procedure of the proposed method. The second one is an engineering example with both input random variables and a random process. The third one is an engineering example where the limit-state function is a black box using the finite element method (FEM) and there are many input random variables.

All the three examples share the same values of the following parameters: $m=100$, $w=2$, $\eta=0.95$, $\gamma=0.005$, and $\tilde{n}_{\mathrm{add}}=1{,}000$. MCS is also used to evaluate MTTF; it calls the original limit-state function in Eq. (1) directly to get samples of $\tau(\mathbf{X})$, and hence the mean lifetime $\bar{\tau}$. The sample size n_{MCS} of MCS is set to 10^5. The results of MCS are treated as the exact ones for the accuracy comparison. Both the proposed method and MCS share the same discretization of $t\in[0,T]$.

5.1. Example 1: A math example

The limit-state function is given by

$$Y=\exp(-0.05t)\cos(0.25t+X),\ t\in[0{,}40] \quad (24)$$

where X is a standard uniform variable. With the Hammersley sampling method, we get $n_{\mathrm{in}}=5$ initial training points in $[0,1]^2$. They are assembled in a matrix $\mathbf{M}$

$$\mathbf{M}=\begin{bmatrix}0 & 0.5\\ 0.2 & 0.25\\ 0.4 & 0.75\\ 0.6 & 0.125\\ 0.8 & 0.625\end{bmatrix} \quad (25)$$

The first column of $\mathbf{M}$ is mapped to the interval $[0,T]$ of t and then we get the initial training points $\mathbf{t}^{\mathrm{in}}=(0,8,16,24,32)^T$ of t. The second column of $\mathbf{M}$ is mapped to the interval $[0,1]$ of X and then we get the initial training points $\mathbf{x}^{\mathrm{in}}=(0.5,0.25,0.75,0.125,0.625)^T$ of X. Substituting the five training points $(\mathbf{x}^{\mathrm{in}},\mathbf{t}^{\mathrm{in}})$ into Eq. (1), we get the 5 training points $\mathbf{y}^{\mathrm{in}}=(0.8776,-0.4211,0.0169,0.2974,-0.1407)^T$ of Y.

Eq. (1) has been evaluated 5 times so far, and therefore currently $n_{\mathrm{e}}=5$. With the training points $(\mathbf{x}^{\mathrm{trn}},\mathbf{t}^{\mathrm{trn}},\mathbf{y}^{\mathrm{trn}})^T=(\mathbf{x}^{\mathrm{in}},\mathbf{t}^{\mathrm{in}},\mathbf{y}^{\mathrm{in}})^T$, the Kriging model $\hat{G}(X,t)$ is built. Then more and more training points determined by the learning function in Eq. (10) are added one by one into the training set $(\mathbf{x}^{\mathrm{trn}},\mathbf{t}^{\mathrm{trn}},\mathbf{y}^{\mathrm{trn}})^T$ to refine $\hat{G}(X,t)$. The sample size n_{s} is also increased adaptively from the initial value $n_0=1{,}000$. After the algorithm converges, 8 training points are added and n_{e} is finally updated to $5+8=13$. n_{s} is finally increased to 2,701.

Fig. 4 shows the actual contours of the limit-state function, as well as the training points. There are three contours indicating $Y=0$. For each value of X, Eq. (3) has three roots. However, we only need the first root. In other words, we only need the Kriging model to accurately predict the first contour. Thanks to the proposed learning function in Eq. (10), almost all adaptive training points are added near the first contour. It helps the Kriging model efficiently find the first root, i.e., $\tau(X)$, without putting unnecessary computational effort in improving unimportant area of the Kriging model. This is an expected good property of the task-oriented adaptive training.

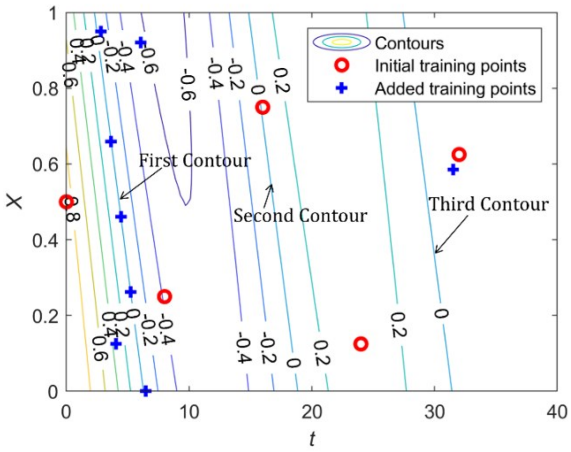


FIGURE 4: CONTOURS AND TRAINING POINTS

TABLE 1: RESULTS OF EXAMPLE 1

Methods	Proposed	MCS
$\bar{\tau}$	4.51	4.49
Relative error	0.6%	-
n_e	13	10^7

Results are given in Table 1. MTTF estimated by the proposed method is 4.51, and that estimated by MCS is 4.49. The relative error is 0.6%, showing the high accuracy of the proposed method. In addition, the proposed method only evaluated the limit-state function 13 times, far less than 10^7 times by MCS, showing the high efficiency of the proposed method.

5.2. Example 2: A simply supported beam

This example is modified from an example in [34]. Shown in Fig. 5 is a simply supported beam subjected to two random loads. The cross-section A-A is rectangular with width a and height b. Due to corrosion, both a and b decrease with time t and are given by

$$a = a_0 \exp(-0.02t) \tag{26}$$

and

$$b = b_0 \exp(-0.02t) \tag{27}$$

where a_0 and b_0 are their initial values. A stationary random process load $F(t)$ acts at the midpoint of the beam. The beam is also subjected to a constant weight load and a load q, which is uniformly distributed on the top surface of the beam. The autocorrelation coefficient functions of $F(t)$ is given by

$$\rho(t_1, t_2) = \exp\left[-\left(\frac{t_1 - t_2}{5}\right)^2\right] \tag{28}$$

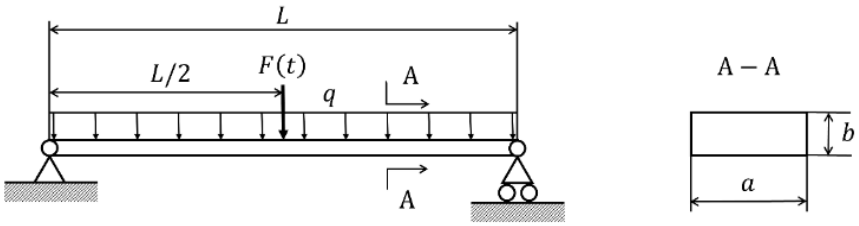


FIGURE 5: A SIMPLY SUPPORTED BEAM

A failure occurs once the stress exceeds the ultimate strength. The limit-state function is given by

$$Y = -0.25F(t)L - 0.125qL^2 - 0.125\rho a_0 b_0 L^2 + 0.25(a_0 - 2kt)(a_0 - 2kt)^2\sigma \tag{29}$$

where σ is the ultimate strength, $\rho = 78.5\ \text{kg/m}^3$ is the density of the beam, $L = 5$ m is the length of the beam, and $t \in [0, 20]$ yr. Table 2 gives all random variables. n_{in} and n_0 are set to 10 and 1,000, respectively. We use six random variables for the EOLE expansion of $F(t)$.

Results are given in Table 3. The beam's mean lifetime evaluated by the proposed method is 11.64 years, with a relative error of -0.1%. In addition, the proposed method only costed 89 limit-state function evaluations, which is way cheaper than MCS.

5.3. Example 3: A 52-bar space truss

This example is modified from an example in [35]. Shown in Fig. 6 is a 52-bar space truss with 21 nodes. All the nodes are located on the surface of an imaginary hemisphere whose radius is $r = 240$ in. The cross-sectional areas of Bars 1~8 and 29~36 are 2 in^2. The cross-sectional areas of Bars 9~16 and other bars are 1.2 in^2 and 0.6 in^2, respectively. The Young's modulus of all bars is E. To distinguish the node numbers and the bar numbers, we add a decimal point after all node numbers in Fig. 6. Nodes 1~13 are subjected to external loads F_1~F_{13}, all in the $-z$ direction. E and F_1~F_{13} are random variables, whose

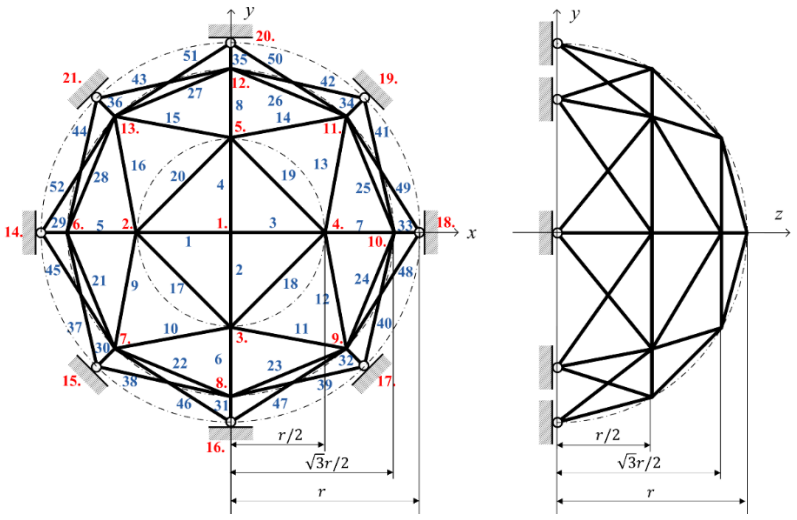
distribution information is given in Table 4. Note that the mean value of F_1 is a function of time $t \in [0, 10]$ yr.

TABLE 2: VARIABLES OF EXAMPLE 2

Variable	Mean	Standard deviation	Distribution	Autocorrelation
a_0	0.2 m	0.002 m	Gaussian	N/A
b_0	0.04 m	0.004 m	Gaussian	N/A
σ	0.24 GPa	0.0024 GPa	Gaussian	N/A
$F(t)$	5,000 N	500 N	Stationary Gaussian process	Eq. (28)
q	450 N/m	50 N/m	Gaussian	N/A

TABLE 3: RESULTS OF EXAMPLE 2

Methods	Proposed	MCS
$\bar{\tau}$	11.64 yr	11.66 yr
Relative error	-0.1%	-
n_e	89	10^7



(A) TOP VIEW (B) LEFT VIEW

FIGURE 6: A 52-BAR SPACE TRUSS

TABLE 4: VARIABLES OF EXAMPLE 3

Variable	Mean	Standard deviation	Distribution
E	2.5×10^4 ksi	25 ksi	Gaussian
F_1	$40\exp(0.05t)$ kip	0.4 kip	Lognormal
$F_2 \sim F_5$	50 kip	0.5 kip	Lognormal
$F_6 \sim F_{13}$	60 kip	0.6 kip	Lognormal

A failure occurs when the displacement δ of Node 1 along $-z$ direction exceeds the threshold $\delta_0 = 1.3$ in. The limit-state function is given by

$$Y(t) = \delta_0 - \delta(E, \mathbf{F}) \tag{30}$$

where $\mathbf{F} = [F_1, F_2, F_3, \dots, F_{13}]$ is the vector all loads. $\delta(E, \mathbf{F})$ is calculated by FEM and the linear bar element is used.

n_{in} and n_0 are set to 30 and 100, respectively. Results are given in Table 5. The mean lifetime evaluated by the proposed method is 6.65 years, with a relative error of -0.7%. In addition, the proposed method only costs 107 limit-state function evaluations, which is much cheaper than MCS.

TABLE 5: RESULTS OF EXAMPLE 3

Methods	Proposed	MCS
$\bar{\tau}$	6.65	6.69 yr
Relative error	-0.7%	-
n_{e}	107	10^7

6. CONCLUSIONS

The mean time to failure (MTTF) is an important measure of product reliability. This study demonstrates that MTTF can be predicted computationally by physics-based method. If a failure mode of the product is well understood and can be modelled mathematically, a limit-state function is available, and the physics-based method can then be used. It is in general much more efficient and cheaper than the statistics-based methods.

This study also demonstrates that machine learning is a powerful tool to assist the prediction of MTTF, which requires a large number of calls of the limit-state function. The results indicate that the proposed Gaussian process based adaptive training is effective to predict MTTF. The key of the learning algorithm is the learning function that is especially designed for the adaptive training. Three examples have shown the high accuracy and efficiency of the proposed method.

The proposed method can only accommodate one failure mode. If there are multiple failure modes, MTTF will depend on the limit-state functions of the failure modes and their relationships, for instance, whether they are in parallel or in series, and this will involve time-dependent system reliability analysis, where machine learning can play a more significant role. Our future work will include developing physics-based machine learning algorithms for multiple Gaussian process responses so that multiple limit-state functions can be handled.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under grant CMMI 1923799 (formerly 1727329).

REFERENCES

[1] O'Connor, P., and Kleyner, A., 2012, Practical reliability engineering, John Wiley & Sons.
[2] Henley, E. J., and Kumamoto, H., 1981, Reliability engineering and risk assessment, Prentice-Hall Englewood Cliffs (NJ).
[3] Birolini, A., 2013, Reliability engineering: theory and practice, Springer Science & Business Media.
[4] Zio, E., 2009, "Reliability engineering: Old problems and new challenges," Reliability Engineering & System Safety, 94(2), pp. 125-141.
[5] Pham, H., 2006, Handbook of reliability engineering, Springer Science & Business Media.
[6] Rausand, M., and Høyland, A., 2004, System reliability theory: models, statistical methods, and applications, John Wiley & Sons.
[7] Meeker, W. Q., and Escobar, L. A., 2014, Statistical methods for reliability data, John Wiley & Sons.
[8] Lawless, J., 1983, "Statistical methods in reliability," Technometrics, 25(4), pp. 305-316.
[9] Viertl, R., 1988, Statistical methods in accelerated life testing, Vandenhoeck & Ruprecht.
[10] Ditlevsen, O., and Madsen, H. O., 1996, Structural reliability methods, Wiley, New York.
[11] Hu, Z., and Du, X., 2018, "Integration of Statistics-and Physics-Based Methods—A Feasibility Study on Accurate System Reliability Prediction," Journal of Mechanical Design, 140(7).
[12] Wang, Z., and Wang, P., 2012, "A Nested Extreme Response Surface Approach for Time-Dependent Reliability-Based Design Optimization," Journal of Mechanical Design, 134(12), pp. 121007-121014.
[13] Du, X., and Chen, W., 2004, "Sequential optimization and reliability assessment method for efficient probabilistic design," Journal of Mechanical Design, 126(2), pp. 225-233.
[14] Dubourg, V., Sudret, B., and Bourinet, J.-M., 2011, "Reliability-based design optimization using kriging surrogates and subset simulation," Structural and Multidisciplinary Optimization, 44(5), pp. 673-690.
[15] Liu, X., Wu, Y., Wang, B., Ding, J., and Jie, H., 2017, "An adaptive local range sampling method for reliability-based design optimization using support vector machine and Kriging model," Structural and Multidisciplinary Optimization, 55(6), pp. 2285-2304.
[16] Papadrakakis, M., and Lagaros, N. D., 2002, "Reliability-based structural optimization using neural networks and Monte Carlo simulation," Computer Methods in Applied Mechanics and Engineering, 191(32), pp. 3491-3507.
[17] Hu, Z., and Du, X., 2013, "Time-dependent reliability analysis with joint upcrossing rates," Structural and Multidisciplinary Optimization, 48(5), pp. 893-907.
[18] Andrieu-Renaud, C., Sudret, B., and Lemaire, M., 2004, "The PHI2 method: a way to compute time-variant reliability," Reliability Engineering & System Safety, 84(1), pp. 75-86.

[19] Hu, Z., and Mahadevan, S., 2016, "A single-loop kriging surrogate modeling for time-dependent reliability analysis," Journal of Mechanical Design, 138(6), p. 061406.
[20] Wei, X., and Du, X., 2019, "Uncertainty Analysis for Time- and Space-Dependent Responses With Random Variables," Journal of Mechanical Design, 141(2), p. 021402.
[21] Zienkiewicz, O. C., Taylor, R. L., Nithiarasu, P., and Zhu, J., 1977, The finite element method, McGraw-hill, London.
[22] Jiang, Z., Li, W., Apley, D. W., and Chen, W., 2015, "A spatial-random-process based multidisciplinary system uncertainty propagation approach with model uncertainty," Journal of Mechanical Design, 137(10).
[23] Xi, Z., 2019, "Model-based reliability analysis with both model uncertainty and parameter uncertainty," Journal of Mechanical Design, 141(5).
[24] Stern, R. E., Song, J., and Work, D. B., 2017, "Accelerated Monte Carlo system reliability analysis through machine-learning-based surrogate models of network connectivity," Reliability Engineering & System Safety, 164, pp. 1-9.
[25] Kotsiantis, S. B., Zaharakis, I., and Pintelas, P., 2007, "Supervised machine learning: A review of classification techniques," Emerging Artificial Intelligence Applications in Computer Engineering, 160, pp. 3-24.
[26] Williams, C. K., and Rasmussen, C. E., 2006, Gaussian processes for machine learning, MIT Press Cambridge, MA.
[27] Lophaven, S. N., Nielsen, H. B., and Søndergaard, J., 2002, DACE: a Matlab kriging toolbox, Citeseer.
[28] Mooney, C. Z., 1997, Monte carlo simulation, Sage Publications, Thousand Oaks.
[29] Chen, W., Tsui, K.-L., Allen, J. K., and Mistree, F., 1995, "Integration of the response surface methodology with the compromise decision support problem in developing a general robust design procedure," ASME Design Engineering Technical Conference, pp. 485-492.
[30] Hosder, S., Walters, R., and Balch, M., 2007, "Efficient sampling for non-intrusive polynomial chaos applications with multiple uncertain input variables," Proc. 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, p. 1939.
[31] Echard, B., Gayton, N., and Lemaire, M., 2011, "AK-MCS: an active learning reliability method combining Kriging and Monte Carlo simulation," Structural Safety, 33(2), pp. 145-154.
[32] Wei, X., and Du, X., 2020, "Robustness Metric for Robust Design Optimization Under Time-and Space-Dependent Uncertainty Through Metamodeling," Journal of Mechanical Design, 142(3).
[33] Sudret, B., and Der Kiureghian, A., 2000, Stochastic finite element methods and reliability: a state-of-the-art report, Department of Civil and Environmental Engineering, University of California Berkeley.
[34] Hu, Z., and Du, X., 2015, "First order reliability method for time-variant problems using series expansions," Structural and Multidisciplinary Optimization, 51(1), pp. 1-21.
[35] Zhang, Z., Jiang, C., Han, X., and Ruan, X., 2019, "A high-precision probabilistic uncertainty propagation method for problems involving multimodal distributions," Mechanical Sysems & Signal Processing, 126, pp. 21-41.