Physics-Based Gaussian Process Method for Predicting Average Product Lifetime in Design Stage

Xinpeng Wei
Graduate Research Assistant
Department of Mechanical and Aerospace Engineering
Missouri University of Science and Technology
400 West 13th Street, Rolla, MO 65409-0500, USA
E-mail: weixinp@mst.edu

Daoru Han
Assistant professor
Department of Mechanical and Aerospace Engineering
Missouri University of Science and Technology
400 West 13th Street, Rolla, MO 65409-0500, USA
E-mail: handao@mst.edu

Xiaoping Du
Professor
Department of Mechanical and Energy Engineering
Indiana University – Purdue University Indianapolis
723 W. Michigan Street, Indianapolis, IN 46202-5195, USA
E-mail: duxi@iu.edu

ABSTRACT

The average lifetime or the mean time to failure (MTTF) of a product is an important metric to measure the product reliability. Current methods of evaluating the MTTF are mainly based on statistics or data. They need lifetime testing on a number of products to get the lifetime samples, which are then used to estimate the 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 the MTTF in the design stage by means of a

This is the author's manuscript of the article published in final edited form as:

Wei, X., Han, D., & Du, X. (2021). Physics-Based Gaussian Process Method for Predicting Average Product Lifetime in Design Stage. Journal of Computing and Information Science in Engineering, 21(4). https://doi.org/10.1115/1.4049509

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

physics-based Gaussian process method. 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). The proposed adaptive supervised training method with the Gaussian process regression can quickly predict the MTTF with a reduced number of physical model calls. The proposed method can enable continually improved design by changing design variables until reliability measures, including the MTTF, are satisfied. The effectiveness of the method is demonstrated by three examples.

Keywords: Average Lifetime, Mean Time to Failure, Gaussian Process Model, Adaptive Training, Supervised Learning

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

1. Introduction

In reliability engineering [1-5], the average lifetime, or the mean time to failure (MTTF), is an important metric of product reliability [1, 6]. Statistics-based methods [7, 8] are widely used to estimate the MTTF. The methods need lifetime testing [9] on many 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 [10-12] can reduce the testing time, the results may not reflect the reliability of the product in normal use conditions. Second, the cost of the testing is usually high. Third, the testing is performed and lifetime data are collected after the products have been made. It is too late and more costly to fix reliability issues if the MTTF is shorter than expected. It is desirable to predict the MTTF during the early design stage.

Direct lifetime data, however, are rarely available during the design stage. Physics-based methods [13-17] 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 [15]. 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 metrics, including the MTTF, do not meet the design requirements, design variables will be updated until the reliability

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

requirements are satisfied. Physics-based methods are therefore a powerful tool to support design for reliability [18-22].

Physics-based methods were originally developed for structural reliability analysis [13, 16, 17]. In the last decades, many new physics-based reliability methods have been developed. These methods cover a wide range of applications, from component reliability [23-28] to system reliability [29-33], and from time-independent reliability to time-dependent reliability [34-41] and time- and space-dependent reliability [42, 43].

Computational models, such as a finite element analysis model [44], 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 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 (ML) methodologies have been increasingly used for reliability analysis. Typical ML methods for reliability analysis include Gaussian process (GP) based methods [28, 31, 34, 45, 46], support vector machine (SVM) based methods [47, 48], and neural network based methods [49-51].

In this study, we extend the physics-based methods to predict the MTTF of a product. Since this task needs more calls of the computational model than a regular reliability analysis, we also rely on ML to maintain computational efficiency. Specifically, we employ the supervised machine learning method [52] and adaptively train a GP to approximate the computational function with respect to the basic random input variables and random processes. Once the learning is finished, the MTTF of the product is obtained.

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

The rest of this paper is organized as follows. The problem statement is given in Section 2. A brief review of GP is provided in Section 3. The proposed method is discussed in Section 4. Three examples are illustrated in Section 5, followed by conclusions in Section 6.

2. Problem Statement

The computational function for reliability analysis is called a limit-state function 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 of additional random variables and t. Thus Eq. (1) does not lose generality. Y is in general a random process. The product fails once the response Y becomes negative.

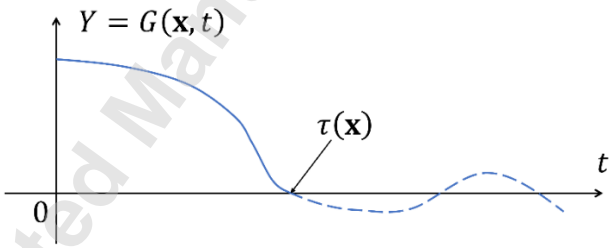

Fig. 1. A sample path of the limit-state function

Fig. 1 shows a sample path (trajectory) of Y when $\mathbf{X}$ is fixed to a realization $\mathbf{x}$. When $t = \tau(\mathbf{x})$, Y takes a negative value for the first time, and hence $\tau(\mathbf{x})$ is called first time to failure (FTTF). 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 physical meaning. A special scenario is $G(\mathbf{x}, 0) < 0$, which means that the product fails at $t = 0$ and hence $\tau(\mathbf{x}) = 0$. Combining the two scenarios, $\tau(\mathbf{x})$ is defined as

$$\tau(\mathbf{x}) = \begin{cases} 0, \text{if } G(\mathbf{x}, 0) < 0 \\ \text{minimum root of } G(\mathbf{x}, t) = 0, \text{if } G(\mathbf{x}, 0) \geq 0 \end{cases} \tag{2}$$

Since $\tau(\mathbf{X})$ is dependent on the input random variables $\mathbf{X}$, it is also a random variable. The product's MTTF $\bar{\tau}$ is the mean value of $\tau(\mathbf{X})$ and is given by [6]

$$\bar{\tau} = \int_0^{+\infty} \tau f_\tau(\tau)\, \mathrm{d}\tau \tag{3}$$

where $f_\tau(\tau)$ is the probability density function (PDF) of $\tau(\mathbf{X})$.

For statistics-based methods, the lifetime testing is used to obtain n_{test} samples of τ and then estimate $\bar{\tau}$ through the following equation

$$\bar{\tau} \approx \frac{1}{n_{\text{test}}} \sum_{k=1}^{n_{\text{test}}} \tau_k \tag{4}$$

where τ_k is the k^{th} sample of τ. As discussed in Section 1, statistics-based methods are generally expensive and time-consuming due to the lifetime testing.

For physics-based methods, the main challenge of estimating $\bar{\tau}$ using Eq. (3) is that it is difficult to obtain $f_\tau(\tau)$ on the interval $[0,+\infty)$ or $[0, T]$ where T is a sufficiently large value. The most straightforward method is the Monte Carlo simulation (MCS) [53]. It samples $\mathbf{X}$ with sufficiently large sample size n_{MCS} and then finds corresponding n_{MCS} samples of $\tau(\mathbf{X})$ using Eq. (2). It finally estimates $\bar{\tau}$ through the following equation

$$\bar{\tau} \approx \frac{1}{n_{\text{MCS}}} \sum_{k=1}^{n_{\text{MCS}}} \tau_k \tag{5}$$

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

where τ_k is the k^{th} sample of τ. It is noted that in physics-based method, the samples of τ are obtained through Eq. (2), which is based on the numerical model $G(\mathbf{X}, t)$, instead of through lifetime testing. However, even without lifetime testing, it is still computationally expensive to solve Eq. (2) n_{MCS} times to obtain the n_{MCS} samples of τ, with the premise that $G(\mathbf{X}, t)$ is generally an expensive black-box function in practical engineering problems.

Another way to derive MTTF is through the time-dependent reliability $R(t)$ [34-39], which is defined as

$$R(t) = \Pr\{G(\mathbf{X}, t') > 0, \forall t' \in [0, t]\} \tag{6}$$

Then MTTF is given by [6]

$$\bar{\tau} = \int_0^T R(t)\,\mathrm{d}t \tag{7}$$

which is equivalent to Eq. (3). Estimating MTTF through Eq. (7), however, is also a challenge, because we need to know the time-dependent reliability $R(t)$ on interval $[0, T]$. In the last decades, many time-dependent reliability methods have been proposed. A simple yet well-known time-dependent reliability analysis method is the equivalent Gaussian process method [37, 39, 54]. The main idea of the equivalent Gaussian process method is to convert the limit-state function $G(\mathbf{X}, t)$ into an equivalent Gaussian process using the first-order reliability method (FORM) [37, 39, 54]. Then Gaussian integral methods are employed to calculate $R(t)$. This method works well for some engineering problems. However, calculating $R(t)$ on a large-span interval $[0, T]$ efficiently is still a challenge.

The objective of this study is to predict $\bar{\tau}$ efficiently and accurately. The proposed method avoids using the brute-force MCS and the expensive methods based on the complex time-dependent reliability analysis.

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

3. Introduction to Gaussian Process Model

Before presenting the proposed method, we briefly introduce GP [45] (or Kriging model [46]) and the learning function U [28], on which the proposed method is based.

3.1. Gaussian process model

A GP makes regression to a function $F(\mathbf{x})$ from a training sample set, or a design of experiment (DoE). The main idea of GP is to treat $F(\mathbf{x})$ as a realization of a Gaussian process $\mathcal{F}(\mathbf{x})$ given by

$$\mathcal{F}(\mathbf{x}) = \mathbf{f}(\mathbf{x})^T\boldsymbol{\beta} + Z(\mathbf{x}) \tag{8}$$

where $\mathbf{f}(\mathbf{x})$ is a vector of regression functions whose coefficients are assembled in vector $\boldsymbol{\beta}$, and $Z(\mathbf{x})$ is a stationary Gaussian process with zero mean and covariance given by

$$\mathrm{Cov}\left[Z(\mathbf{x}_i), Z(\mathbf{x}_j)\right] = \sigma_Z^2 r(\mathbf{x}_i, \mathbf{x}_j) \tag{9}$$

where σ_Z^2 is the variance of $Z(\mathbf{x})$, and $r(\cdot,\cdot)$ is the correlation function. A widely used model of the correlation function is called Gaussian model, or squared exponential model [46], and is given by

$$r(\mathbf{x}_i, \mathbf{x}_j) = \prod_{d=1}^{D} \exp\left[-\theta_d\left(x_{id} - x_{jd}\right)^2\right] \tag{10}$$

where D is the dimension of $\mathbf{x}$, x_{id} is the d^{th} component of $\mathbf{x}_i$, x_{jd} is the d^{th} component of $\mathbf{x}_j$, and θ_d is a parameter indicating the correlation in dimension d. The output of the GP is a Gaussian variable

$$\hat{F}(\mathbf{x}) \sim N\left(\mu_{\hat{F}}(\mathbf{x}), \sigma_{\hat{F}}^2(\mathbf{x})\right) \tag{11}$$

where $\mu_{\hat{F}}(\mathbf{x})$ is the GP prediction, and $\sigma^2_{\hat{F}}(\mathbf{x})$ is called mean squared error [46]. Given a DoE, $\mu_{\hat{F}}(\mathbf{x})$ and $\sigma^2_{\hat{F}}(\mathbf{x})$ are determined using the maximum likelihood method [55]. More details about GP are available in [45, 46].

3.2. Learning function *U*

In engineering problems where only the sign of $F(\mathbf{x})$ is of interest, such as reliability analysis where only the sign of the limit-state function is important, we need to measure how certain the sign of $F(\mathbf{x})$ has been predicted by $\text{sign}[\mu_{\hat{F}}(\mathbf{x})]$, the sign of $\mu_{\hat{F}}(\mathbf{x})$. If $\mu_{\hat{F}}(\mathbf{x}) > 0$, then the probability that $F(\mathbf{x}) > 0$ is $\Phi\left(\frac{\mu_{\hat{F}}(\mathbf{x})}{\sigma_{\hat{F}}(\mathbf{x})}\right)$, where $\Phi(\cdot)$ is the cumulative distribution function of a standard normal variable. Similarly, if $\mu_{\hat{F}}(\mathbf{x}) < 0$, then the probability that $F(\mathbf{x}) < 0$ is $\Phi\left(-\frac{\mu_{\hat{F}}(\mathbf{x})}{\sigma_{\hat{F}}(\mathbf{x})}\right)$. Combining the two cases, the probability that the sign of $F(\mathbf{x})$ has been correctly predicted by $\text{sign}[\mu_{\hat{F}}(\mathbf{x})]$ is $\Phi\left(\frac{|\mu_{\hat{F}}(\mathbf{x})|}{\sigma_{\hat{F}}(\mathbf{x})}\right)$. $\frac{|\mu_{\hat{F}}(\mathbf{x})|}{\sigma_{\hat{F}}(\mathbf{x})}$, which is monotonic to $\Phi\left(\frac{|\mu_{\hat{F}}(\mathbf{x})|}{\sigma_{\hat{F}}(\mathbf{x})}\right)$ and known as the learning function U [28], is widely used to determine whether $\text{sign}[F(\mathbf{x})]$ is predicted correctly.

4. The Proposed Method

4.1. Overview of the proposed method

The main idea of the proposed method is to adaptively train a GP $\hat{G}(\mathbf{X}, t)$ for $G(\mathbf{X}, t)$. With $\hat{G}(\mathbf{X}, t)$, we can obtain the surrogate model $\hat{\tau}(\mathbf{X})$ of $\tau(\mathbf{X})$. Since $\hat{\tau}(\mathbf{X})$ is computationally cheap, we can calculate $\bar{\tau}$ using MCS.

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

Training $\hat{G}(\mathbf{X}, t)$ should be task-oriented in order to improve efficiency. We employ 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 initial 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. The learning function and stopping criterion are used to find the new training points and determine 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 detail in Subsections 4.2 through 4.4.

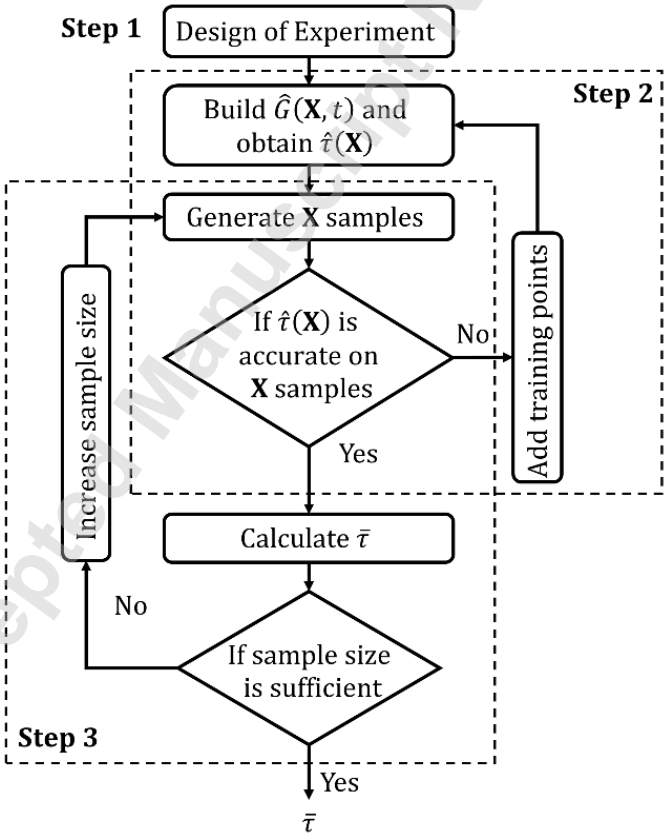

Fig. 2. Brief flowchart of the proposed method

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

4.2. Design of experiments for initial GP

The principle of the design of experiments for building a GP is to spread the initial training points evenly. Commonly used sampling methods include random sampling, Latin hypercube sampling, and Hammersley sampling [56]. In this study, we employ the Hammersley sampling method because it has better uniformity properties over a multidimensional space [57]. 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 from 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

$$\mathbf{x}^{\text{in}} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \ddots & x_N^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \ddots & x_N^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(n_{\text{in}})} & x_2^{(n_{\text{in}})} & \ddots & x_N^{(n_{\text{in}})} \end{bmatrix} \tag{12}$$

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 evaluating Eq. (1) n_{in} times. Finally, we get the initial training set $(\mathbf{x}^{\text{trn}}, \mathbf{t}^{\text{trn}}, \mathbf{y}^{\text{trn}}) = \left(\mathbf{x}^{\text{in}}, \mathbf{t}^{\text{in}}, \mathbf{y}^{\text{in}}\right)$, where the superscript *trn* and *in* represents the general training points and initial training points, respectively.

4.3. Adaptive training

With the initial training points $(\mathbf{x}^{\text{in}}, \mathbf{t}^{\text{in}}, \mathbf{y}^{\text{in}})$, we can build an initial GP $\hat{G}(\mathbf{X}, t)$ to approximate $G(\mathbf{X}, t)$. The initial $\hat{G}(\mathbf{X}, t)$ is usually 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 employed.

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

$$\hat{\tau}(\mathbf{x}) = \min\{t \in \mathbf{t} | \mu_{\hat{G}}(\mathbf{x}, t) \leq 0\} \tag{13}$$

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

$$\bar{\tau} = \frac{1}{n_{\text{s}}} \sum_{i=1}^{n_{\text{s}}} \hat{\tau}\left(\mathbf{x}^{(i)}\right) \tag{14}$$

where $\mathbf{x}^{(i)}$ is the i^{th} random sample of $\mathbf{X}$. Eq. (14) can yield accurate $\bar{\tau}$ when two conditions are satisfied. First, the sample size n_{s} is sufficiently large. The determination of n_{s} will be given in Subsection 4.4. Second, the model $\hat{\tau}(\mathbf{X})$ is accurate for 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 $\mu_{\hat{G}}(\mathbf{X}, t)$ 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. (2) as long as the signs of $\{G(\mathbf{x}, t)|, t \in \mathbf{t}, t \leq t^*\}$ 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 the accurate solution to Eq. (2). We do not need to care if $\hat{G}(\mathbf{x}, t)$ predicts the specific values of

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

$G(\mathbf{x}, t_j), j = 1,2,3,4,5$ or the signs of $G(\mathbf{x}, t_j), j \geq 6$ accurately. Note that in this example the exact solution to Eq. (2) should be within the interval $[t_4, t_5]$, but we can still treat t_5 as the solution without losing significant accuracy as long as m is sufficiently large.

The learning function U proposed in [28] is used to measure how accurate the sign at a point is predicted. It is given by

$$U(\mathbf{x}, t) = \frac{|\mu_{\hat{G}}(\mathbf{x}, t)|}{\sigma_{\hat{G}}(\mathbf{x}, t)} \tag{15}$$

To refine $\hat{G}(\mathbf{X}, t)$, 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

$$(\mathbf{x}, t^{\text{next}}) = \underset{t \in \mathbf{t}, t \leq \hat{\tau}(\mathbf{x})}{\arg\min}\, U(\mathbf{x}, t) \tag{16}$$

Since there are n_s samples of $\mathbf{X}$, Eq. (16) 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

$$(\mathbf{x}^{\text{next}}, t^{\text{next}}) = \underset{\mathbf{x} \in \mathbf{x}^s, t \in \mathbf{t}, t \leq \hat{\tau}(\mathbf{x})}{\arg\min}\, U(\mathbf{x}, t) \tag{17}$$

With the learning function given in Eq. (17), 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 same learning function in Eq. (17) is also used in [40, 41] for time-dependent reliability analysis with the first passage time.

The direct use of $U(\mathbf{x}, t)$ and hence Eq. (17), however, may result in duplicate training points. In other words, the next training point determined by Eq. (17) may be among $(\mathbf{x}^{\text{trn}}, \mathbf{t}^{\text{trn}}, \mathbf{y}^{\text{trn}})$. Once this happens, the adaptive training fails. Theoretically, because GP

is an exact interpolator, if a point $(\mathbf{x}^*, t^*, y^*)$ is among the training set $(\mathbf{x}^{\text{trn}}, \mathbf{t}^{\text{trn}}, \mathbf{y}^{\text{trn}})$, $\hat{G}(\mathbf{X}, t)$ will predict $G(\mathbf{x}^*, t^*)$ exactly as y^*, i.e., $\mu_{\hat{G}}(\mathbf{x}^*, t^*) = y^*$ and $\sigma_{\hat{G}}(\mathbf{x}^*, t^*) = 0$. As a result, $U(\mathbf{x}^*, t^*) = +\infty$, $(\mathbf{x}^*, t^*)$ will never be selected by Eq. (17) as the next training point, and the duplicate training points will never be encountered. However, due to the numerical error, $\sigma_{\hat{G}}(\mathbf{x}^*, t^*)$ is not exactly zero but a small positive number. In this case, if $\mu_{\hat{G}}(\mathbf{x}^*, t^*)$ is smaller than $\sigma_{\hat{G}}(\mathbf{x}^*, t^*)$, we will have $U(\mathbf{x}^*, t^*) < 1$, and Eq. (17) may select $(\mathbf{x}^*, t^*)$ as the next training point, leading to the duplicate training points.

Another problem caused by U is that the adaptively added training points may cluster together [34]. It will make the correlation matrix of GP 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 [34] 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 $\mathbf{x}^{\text{s}} \times \mathbf{t}$ to $\left\{(\mathbf{x}, t) \in \mathbf{x}^{\text{s}} \times \mathbf{t} \middle| \max_{(\mathbf{x}', t') \in (\mathbf{x}^{\text{trn}}, \mathbf{t}^{\text{trn}})} r[(\mathbf{x}, t), (\mathbf{x}', t')] < \eta \right\}$, where $r(\cdot, \cdot)$ is the correlation function used in GP 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 update the learning function in Eq. (17) to the following one

$$(\mathbf{x}^{\text{next}}, t^{\text{next}}) = \arg \min_{t \le \hat{\tau}(\mathbf{x}), (\mathbf{x}, t) \in \mathbf{C}} U(\mathbf{x}, t) \tag{18}$$

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

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

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. (13), 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

$$W > w \tag{19}$$

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 [28].

4.4. Adaptive sample size

Since the random sampling method is used to estimate $\bar{\tau}$ through Eq. (14), 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 [58].

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

$$\Gamma = \frac{\sigma_\tau}{\bar{\tau}\sqrt{n_s}} \tag{20}$$

where $\bar{\tau}$ is estimated by Eq. (14) and σ_τ is estimated by

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

$$\sigma_\tau = \sqrt{\frac{1}{n_\mathrm{s}-1}\sum_{i=1}^{n_\mathrm{s}}[\hat{t}(\mathbf{x}^{(i)}) - \bar{\tau}]^2} \tag{21}$$

Eq. (20) shows that the larger is n_s, the smaller Γ will we have. A smaller Γ means that $\bar{\tau}$ is more accurately estimated by Eq. (14). $\bar{\tau}$ is said to be accurate if the following condition is satisfied

$$\Gamma \leq \gamma \tag{22}$$

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

If the current n_s cannot satisfy Eq. (22), we should increase it. Combining Eq. (20) and Eq. (22), we have

$$n_\mathrm{s} \geq \left(\frac{\sigma_\tau}{\bar{\tau}\gamma}\right)^2 \tag{23}$$

It means that at least a sample size of $\left(\frac{\sigma_\tau}{\bar{\tau}\gamma}\right)^2$ is necessary to guarantee Eq. (22). Let $n_1 = \mathrm{ceil}\left[\left(\frac{\sigma_\tau}{\bar{\tau}\gamma}\right)^2\right]$, where ceil$(\cdot)$ represents the operation to get the nearest larger integer. Then the number n_add by which n_s should be increased is given by

$$n_\mathrm{add} = n_1 - n_\mathrm{s} \tag{24}$$

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. (24) may be misleading. To deal with this issue, we set a threshold $\tilde{n}_\mathrm{add}$ for n_add. Then Eq. (24) is updated to

$$n_\mathrm{add} = \begin{cases} \tilde{n}_\mathrm{add}, \text{if } n_1 - n_\mathrm{s} > \tilde{n}_\mathrm{add} \\ n_1 - n_\mathrm{s}, \text{otherwise} \end{cases} \tag{25}$$

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

Since it is cheap to compute samples of $\hat{\tau}(\mathbf{X})$, $\tilde{n}_{\text{add}}$ is not really a key hyperparameter of the proposed method, and generally it is good to set $\tilde{n}_{\text{add}}$ to $1{,}000$, according to our experience from many experiments.

4.5. Implementation

The detailed flowchart of the proposed method 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, with the assumption that $G(\mathbf{X}, t)$ is a computationally expensive black-box function in practical engineering problems.

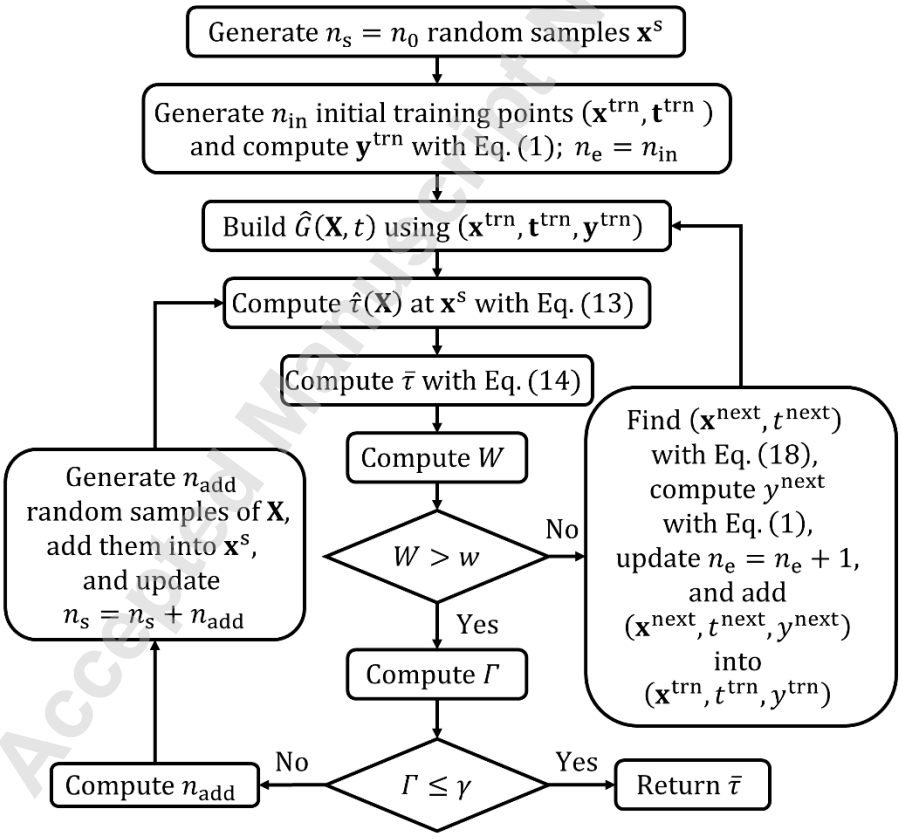

Fig. 3. Detailed flowchart of the proposed method

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

4.6. 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

$$Y = G(\mathbf{X}, \mathbf{H}(t), t) \tag{26}$$

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) [59]. 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

$$H(\boldsymbol{\xi}, t) = \mu_H(t) + \sigma_H(t) \sum_{k=1}^{m} \frac{\xi_k}{\sqrt{\lambda_k}} \mathbf{V}_k \mathbf{M}_H(:, k), t \in \mathbf{t} \tag{27}$$

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. (27) is practically truncated, and only the first m' orders are kept:

$$H(\boldsymbol{\xi}, t) = \mu_H(t) + \sigma_H(t) \sum_{k=1}^{m'} \frac{\xi_k}{\sqrt{\lambda_k}} \mathbf{V}_k \mathbf{M}_H(:, k), t \in \mathbf{t} \tag{28}$$

With the truncated expansion in Eq. (28), Eq. (26) is rewritten as

$$Y = G(\mathbf{X}, H(\boldsymbol{\xi}, t), t) \tag{29}$$

or equivalently as

$$Y = G\left(\widetilde{\mathbf{X}}, t\right) \tag{30}$$

where $\widetilde{\mathbf{X}} = (\boldsymbol{\xi}, \mathbf{X})$. Eq. (30) shares the same format with Eq. (1) and hence the implementation given in Subsection 4.5 works as well.

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 $\hat{G}\left(\widetilde{\mathbf{X}}, t\right)$ 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 GP. To overcome the drawbacks, we build a GP $\hat{G}(\mathbf{X}, H, t)$ with respect to $\mathbf{X}$, H, and t [34, 58]. 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

$$\hat{G}\left(\widetilde{\mathbf{X}}, t\right) = \hat{G}[\mathbf{X}, H(\boldsymbol{\xi}, t), t] \tag{31}$$

Since the truncated series expansion $H(\boldsymbol{\xi}, \mathbf{Z})$ in Eq. (28) 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. (31). 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

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

simply by substituting $(\boldsymbol{\xi}^{\text{trn}}, \mathbf{t}^{\text{trn}})$ into Eq. (28). Similarly, when $\left(\tilde{\mathbf{x}}^{(\text{next})}, t^{(\text{next})}\right)$ is determined by Eq. (18), the next training point $h^{(\text{next})}$ of H is obtained by substituting $\left(\boldsymbol{\xi}^{(\text{next})}, \mathbf{z}^{(\text{next})}\right)$ into Eq. (28). Note that $\tilde{\mathbf{x}}^{(\text{next})} = \left(\boldsymbol{\xi}^{(\text{next})}, \mathbf{x}^{(\text{next})}\right)$**.** When multiple input random processes are involved, the procedure of building and updating the surrogate model $\hat{G}$ is similar.

4.7. Application in design

The proposed method predicts MTTF and can be used in design optimization. A design process may involve a number of iterations where new designs are iteratively generated. At each new design point, the proposed method is performed to predict MTTF if MTTF is one of the design requirements. From the analysis of the proposed method, design engineers know if the MTTF requirement is met. If the predicted MTTF is lower than expected, they can modify and generate a new design. By repeating this design process, engineers reach their final design that satisfies all the requirements, including the MTTF requirement. If optimization is the design tool used for the design, the MTTF could be treated as a design constraint. It is convenient to integrate the proposed method and the powerful Bayesian optimization [60] to perform design optimization since the GP method used in the proposed method is also employed in the Bayesian optimization. How to effectively do so is worth a further investigation in the future work.

4.8. Discussions

The advantage of the new physics-based method over statistics-based methods is its high efficiency and low cost. The main reason is that the statistics-based methods need the

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

lifetime testing while the physics-based method estimates MTTF using a numerical model. The other advantage is that MTTF can be predicted early during the design stage and can therefore be incorporated in the design optimization process. In addition, the adaptive training strategy embedded in the method improves the efficiency further.

The proposed method, however, has two limitations. First, it is based on the premise that the failure mode can be modeled by a limit-state function. It will not work if the limit-state function is not available or if the failure mode is hidden or unidentifiable during the early design. Second, the proposed method can only handle single failure modes. Extending it to multiple failure modes needs a further investigation.

5. Examples

In this section, we use three examples to illustrate the proposed method. The first one is an artificial 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 where there are five input random processes. The limit-state functions of all the three examples are computationally inexpensive so that the brute-force method MCS is affordable for the validation purpose. The results of MCS are treated as accurate solutions for the accuracy comparison. The proposed method, however, is not limited to those simple problems. It also works for any failure mode with a complexed limit-state function as long as the limit-state function can be built for the MTTF analysis. In Example 2, we also compare the proposed method with a time-dependent reliability based method

(TRBM). Specifically, in the TRBM, we convert the limit-state function into an equivalent Gaussian process [37, 39, 54], then use MCS to calculate $R(t)$ on $[0,T]$, and finally estimate MTTF using Eq. (7).

All the three examples share the same values of the following parameters: $m = 100$, $w = 2$, $\eta = 0.95$, $\gamma = 0.005$, and $\tilde{n}_{\text{add}} = 1{,}000$. MCS 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. All the three methods share the same discretization of $t \in [0,T]$.

5.1. Example 1: An artificial math example

In this example, the limit-state function is given by

$$Y = \exp(-0.05t)\cos(0.25t + X),\ t \in [0{,}40] \tag{32}$$

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

$$\mathbf{M} = \begin{bmatrix} 0 & 0.5 \\ 0.2 & 0.25 \\ 0.4 & 0.75 \\ 0.6 & 0.125 \\ 0.8 & 0.625 \end{bmatrix} \tag{33}$$

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

Eq. (1) has been evaluated 5 times so far, and therefore currently $n_{\mathrm{e}} = 5$. With the training points $(\mathbf{x}^{\mathrm{trn}}, \mathbf{t}^{\mathrm{trn}}, \mathbf{y}^{\mathrm{trn}}) = \left(\mathbf{x}^{\mathrm{in}}, \mathbf{t}^{\mathrm{in}}, \mathbf{y}^{\mathrm{in}}\right)$, $\hat{G}(X,t)$ is built. Then more and more training points determined by the learning function in Eq. (18) are added one by one into the training set $(\mathbf{x}^{\mathrm{trn}}, \mathbf{t}^{\mathrm{trn}}, \mathbf{y}^{\mathrm{trn}})$ 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, 6 training points are added, and n_{e} is finally updated to $5 + 6 = 11$. n_{s} is finally increased to 2,632.

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, $G(X,t) = 0$ has three roots. However, we need only the minimum one. In other words, we need the GP to accurately predict only the first contour. With the proposed learning function in Eq. (18), almost all adaptive training points are added near the first contour. It helps the GP efficiently find the first root, i.e., $\tau(X)$, without wasting computational effort in improving the GP in unimportant area. This is an expected good property of the proposed task-oriented adaptive training.

Results are given in Table 1. The MTTF estimated by the proposed method is 4.48, and that estimated by MCS is 4.49. The relative error is -0.2%, showing the high accuracy of the proposed method. In addition, the proposed method evaluates the limit-state function 11 times, far less than 10^7 times by MCS, showing the high efficiency of the proposed method.

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

Table 1. Results of Example 1

Methods	Proposed	MCS
$\bar{\tau}$	4.48	4.49
Relative error	-0.2%	-
n_e	11	10^7

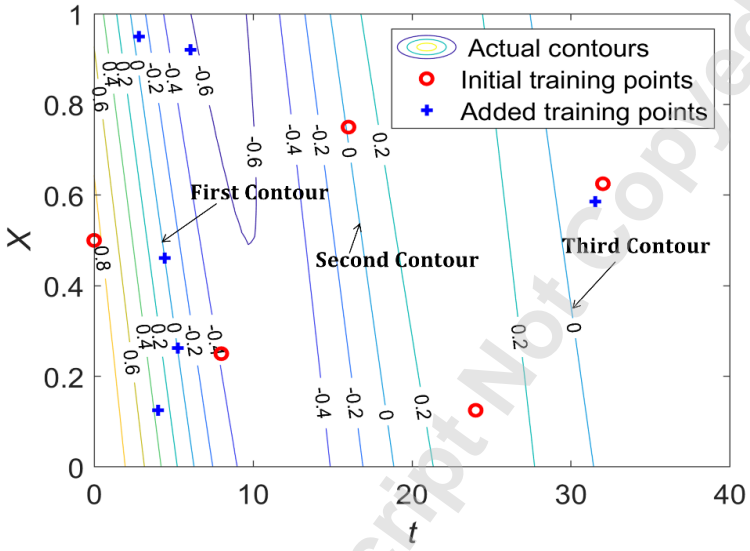

Fig. 4. Contours and training points

5.2. Example 2: A simply supported beam

This example is modified from an example in [54]. 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

$$a = a_0 \exp(-0.02t) \tag{34}$$

and

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

$$b = b_0 \exp(-0.02t) \tag{35}$$

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

$$\rho(t_1, t_2) = \exp\left[-\left(\frac{t_1 - t_2}{5}\right)^2\right] \tag{36}$$

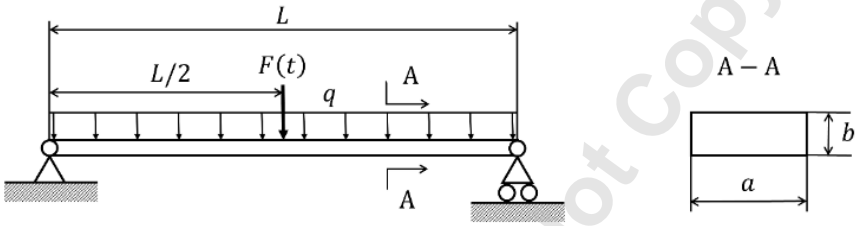

Fig. 5. A simply supported beam [54]

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

$$Y = -0.25F(t)L - 0.125qL^2 - 0.125\rho a_0 b_0 L^2 + 0.25(a_0 - 2kt)(a_0 - 2kt)^2\sigma \tag{37}$$

where σ is the ultimate strength, $\rho = 78.5$ kg/m^3 is the density of the beam, $L = 5$ m is the length of the beam, and $\mathrm{t} \in [0, 20]$ 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 $F(t)$.

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

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. (36)
q	450 N/m	50 N/m	Gaussian	N/A

Results are given in Table 3. The MTTF evaluated by the proposed method is 11.61 years, with a relative error of -0.4%. In addition, the proposed method needs 23 limit-state function evaluations, which is much cheaper than MCS. In this example we also compare the proposed method with TRBM. TRBM also obtains accurate MTTF, 11.55 years, with a relative error of -0.9%. TRBM, however, costs 1728 function evaluations, much less efficient than the proposed method. A main reason for the low efficiency is the computation in calculating the time-dependent reliability. For this reason, we do not recommend estimating MTTF through time-dependent reliability analysis methods. Another reason is that TRBM does not employ a surrogate model as the proposed method does.

Table 3. Results of Example 2

Methods	Proposed	TRBM	MCS
$\bar{\tau}$	11.61 yr	11.55 yr	11.66 yr
Relative error	-0.4%	-0.9%	-
n_{e}	23	1728	10^7

5.3. Example 3: A 52-bar space truss

This example is modified from an example in [61, 62]. Shown in Fig. 6 is a 52-bar space truss with 21 nodes. To distinguish the node numbers and the bar numbers, we add a decimal point after all node numbers in Fig. 6. 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, which is a lognormal random variable with mean and standard deviation being 25,000 ksi and 25 ksi, respectively. Nodes 1~5 are subjected to external loads $F_1(t)$~$F_5(t)$, all in the $-z$ direction. The five loads are modeled as Gaussian processes. They are independent of each other with the following autocorrelation coefficient function:

$$\rho(t_1, t_2) = \exp\left[-\left(\frac{t_1 - t_2}{5}\right)^2\right] \tag{38}$$

where $t_1, t_2 \in [0, 10]$ yr. $F_2(t)$~$F_5(t)$ are all stationary processes whose mean and standard deviation are 50 kip and 1 kip, respectively. $F_1(t)$ is nonstationary, with mean value $\mu_1(t)$ and standard deviation $\sigma_1(t)$ given by

$$\mu_1(t) = 50\exp(0.02t) \text{ kip} \tag{39}$$

and

$$\sigma_1(t) = \exp(0.02t) \text{ kip} \tag{40}$$

where $t \in [0, 10]$ yr.

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

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

$$Y(t) = \delta_0 - \delta(E, \mathbf{F}) \tag{41}$$

where $\mathbf{F} = [F_1, F_2, F_3, F_4, F_5]$ is the vector of 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 10 and 1,000, respectively. We use six random variables in the EOLE expansion of each random load. Results are given in Table 4. The mean lifetime evaluated by the proposed method is 4.79 years with a relative error of 0.8%. Besides, the proposed method costs 56 limit-state function evaluations and is much more efficient than MCS.

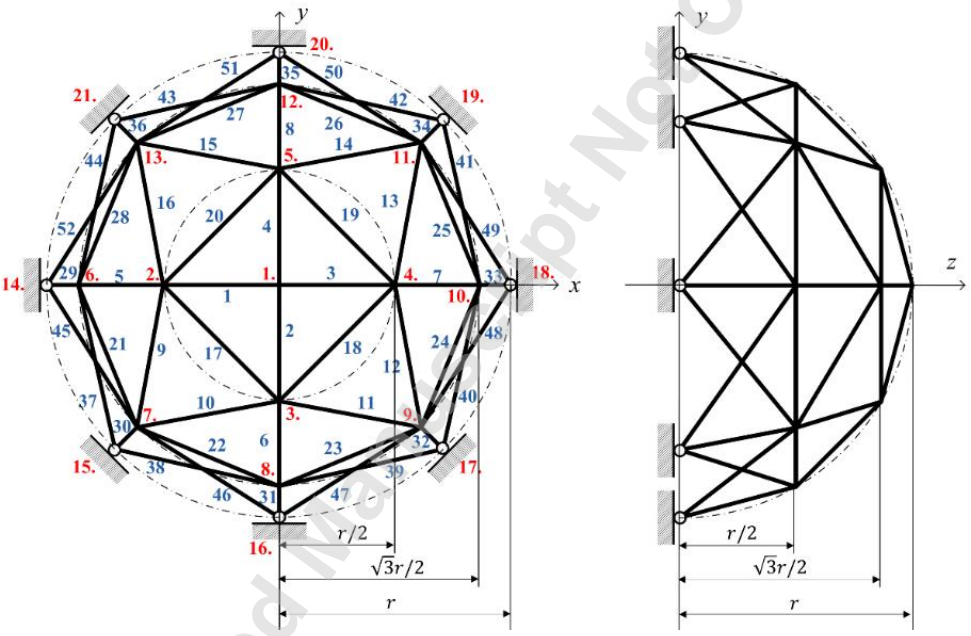

Fig. 6. A 52-bar truss: top view (left) and left view (right) [61, 62]

Table 4. Results of Example 3

Methods	Proposed	MCS
$\bar{\tau}$	4.79 yr	4.75 yr
Relative error	0.8%	-
n_{e}	56	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 a 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 statistics-based methods.

This study also demonstrates that ML is a powerful tool to assist the prediction of the MTTF. The results indicate that the proposed Gaussian process based adaptive training is effective to predict the MTTF. 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, the 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 ML can play a more significant role. Our future work will include developing physics-based ML 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).

Reference

[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] Epstein, B., and Sobel, M., 1953, "Life testing," Journal of the American Statistical Association, 48(263), pp. 486-502.
[10] Viertl, R., 1988, Statistical methods in accelerated life testing, Vandenhoeck & Ruprecht.
[11] Zhang, X. P., Shang, J. Z., Chen, X., Zhang, C. H., and Wang, Y. S., 2014, "Statistical inference of accelerated life testing with dependent competing failures based on copula theory," IEEE Transactions on Reliability, 63(3), pp. 764-780.
[12] Hu, Z., Mahadevan, S. J. Q., and International, R. E., 2016, "Accelerated life testing (ALT) design based on computational reliability analysis," 32(7), pp. 2217-2232.
[13] Ditlevsen, O., and Madsen, H. O., 1996, Structural reliability methods, Wiley, New York.
[14] Hu, Z., and Du, X., "A physics-based reliability method for components adopted in new series systems," Proc. 2016 Annual Reliability and Maintainability Symposium (RAMS), IEEE, pp. 1-7.
[15] 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).
[16] Melchers, R. E., and Beck, A. T., 2018, Structural reliability analysis and prediction, John wiley & sons.
[17] Thoft-Cristensen, P., and Baker, M. J., 2012, Structural reliability theory and its applications, Springer Science & Business Media.
[18] 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.

[19] 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.
[20] 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.
[21] 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.
[22] 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.
[23] Wang, C., Zhang, H., and Li, Q., 2019, Moment-based evaluation of structural reliability.
[24] Shi, Y., Zhenzhou, L., Chen, S., and Xu, L., 2018, A reliability analysis method based on analytical expressions of the first four moments of the surrogate model of the performance function.
[25] Hu, Z., and Du, X., 2018, "Saddlepoint approximation reliability method for quadratic functions in normal variables," Structural Safety, 71, pp. 24-32.
[26] Sun, Z., Wang, J., Li, R., and Tong, C., 2017, "LIF: A new Kriging based learning function and its application to structural reliability analysis," Reliability Engineering & System Safety, 157, pp. 152-165.
[27] Peijuan, Z., Ming, W. C., Zhouhong, Z., and Liqi, W., 2017, "A new active learning method based on the learning function U of the AK-MCS reliability analysis method," Engineering Structures, 148, pp. 185-194.
[28] 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.
[29] Du, X., 2010, "System reliability analysis with saddlepoint approximation," Structural and Multidisciplinary Optimization, 42(2), pp. 193-208.
[30] Yun, W., Zhenzhou, L., Zhou, Y., and Jiang, X., 2018, AK-SYSi: an improved adaptive Kriging model for system reliability analysis with multiple failure modes by a refined U learning function.
[31] Wu, H., Zhu, Z., and Du, X., 2020, "System Reliability Analysis With Autocorrelated Kriging Predictions," Journal of Mechanical Design, 142(10).
[32] Song, J., and Kang, W.-H., 2009, "System reliability and sensitivity under statistical dependence by matrix-based system reliability method," Structural Safety, 31(2), pp. 148-156.
[33] Youn, B. D., and Wang, P., 2009, "Complementary intersection method for system reliability analysis," Journal of Mechanical Design, 131(4), p. 041004.
[34] 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.
[35] Jiang, C., Wei, X. P., Huang, Z. L., and Liu, J., 2017, "An outcrossing rate model and tts efficient calculation for time-dependent system reliability analysis," Journal of Mechanical Design, 139(4), p. 041402.

[36] Zhang, D., Han, X., Jiang, C., Liu, J., and Li, Q., 2017, "Time-dependent reliability analysis through response surface method," Journal of Mechanical Design, 139(4).
[37] Gong, C., and Frangopol, D. M., 2019, "An efficient time-dependent reliability method," Structural Safety, 81, p. 101864.
[38] Shi, Y., Lu, Z., Xu, L., and Chen, S., 2019, "An adaptive multiple-Kriging-surrogate method for time-dependent reliability analysis," Applied Mathematical Modelling, 70, pp. 545-571.
[39] Jiang, C., Wei, X. P., Wu, B., and Huang, Z. L., 2018, "An improved TRPD method for time-variant reliability analysis," Structural and Multidisciplinary Optimization.
[40] Hu, Z., and Mahadevan, S., 2016, "Resilience assessment based on time-dependent system reliability analysis," Journal of Mechanical Design, 138(11), p. 111404.
[41] Hu, Y., Lu, Z., Wei, N., and Zhou, C., 2020, "A single-loop Kriging surrogate model method by considering the first failure instant for time-dependent reliability analysis and safety lifetime analysis," 145, p. 106963.
[42] 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.
[43] Shi, Y., Lu, Z., Zhang, K., and Wei, Y., 2017, "Reliability analysis for structures with multiple temporal and spatial parameters based on the effective first-crossing point," Journal of Mechanical Design, 139(12), pp. 121403-121403.
[44] Zienkiewicz, O. C., Taylor, R. L., Nithiarasu, P., and Zhu, J., 1977, The finite element method, McGraw-hill, London.
[45] Williams, C. K., and Rasmussen, C. E., 2006, Gaussian processes for machine learning, MIT Press Cambridge, MA.
[46] Lophaven, S. N., Nielsen, H. B., and Søndergaard, J., 2002, DACE: a Matlab kriging toolbox, Citeseer.
[47] Rocco, C. M., and Moreno, J. A., 2002, "Fast Monte Carlo reliability evaluation using support vector machine," Reliability Engineering & System Safety, 76(3), pp. 237-243.
[48] Pan, Q., and Dias, D., 2017, "An efficient reliability method combining adaptive support vector machine and Monte Carlo simulation," structural safety, 67, pp. 85-95.
[49] Cheng, J., Li, Q. S., and Xiao, R., 2008, "A new artificial neural network-based response surface method for structural reliability analysis," Probabilistic Engineering Mechanics, 23(1), pp. 51-63.
[50] Chojaczyk, A. A., Teixeira, A. P., Neves, L. C., Cardoso, J. B., and Soares, C. G., 2015, "Review and application of Artificial Neural Networks models in reliability analysis of steel structures," Structural Safety, 52(3), pp. 78-89.
[51] Dai, H., Zhang, H., and Wang, W., 2015, "A Multiwavelet Neural Network‐Based Response Surface Method for Structural Reliability Analysis," Computer-Aided Civil and Infrastructure Engineering, 30(2), pp. 151-162.
[52] 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.
[53] Mooney, C. Z., 1997, Monte carlo simulation, Sage Publications, Thousand Oaks.
[54] 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.
[55] Eliason, S. R., 1993, Maximum likelihood estimation: Logic and practice, Sage.

[56] 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.
[57] 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.
[58] 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).
[59] 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.
[60] Lizotte, D. J., 2008, Practical bayesian optimization, University of Alberta.
[61] 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.
[62] Gomes, H. M., 2011, "Truss optimization with dynamic constraints using a particle swarm algorithm," Expert Systems with Applications, 38(1), pp. 957-968.

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

Table Caption List

Table 1	Results of Example 1
Table 2	Variables of Example 2
Table 3	Results of Example 2
Table 4	Results of Example 3

Journal of Computing and Information Science in Engineering. Received August 27, 2020;
Accepted manuscript posted December 05, 2020. doi:10.1115/1.4049509

Figure Caption List

Fig. 1	A sample path of the limit-state function
Fig. 2	Brief flowchart of the proposed method
Fig. 3	Detailed flowchart of the proposed method
Fig. 4	Contours and training points
Fig. 5	A simply supported beam [54]
Fig. 6	A 52-bar truss: top view (left) and left view (right) [61, 62]