

Active Learning Line Sampling for Rare Event Analysis

Jingwen Song^{a,b}, Pengfei Wei^{a,*b}, Marcos Valdebenito^c, Michael Beer^{b,d,e}

^a Northwestern Polytechnical University, West Youyi Road 127, Xi'an 710072, China

^b Institute for Risk and Reliability, Leibniz Universität Hannover, Callinstr. 34, Hannover 30167, Germany

^c Departamento de Obras Civiles, Universidad Tecnica Federico Santa Maria, Av. Espana 1680, Valparaiso, Chile

^d Institute for Risk and Uncertainty, University of Liverpool, Peach Street, L69 7ZF Liverpool, United Kingdom

^e International Joint Research Center for Engineering Reliability and Stochastic Mechanics, Tongji University, Shanghai 200092, China

Abstract: Line Sampling (LS) has been widely recognized as one of the most appealing stochastic simulation algorithms for rare event analysis, but when applying it to many real-world engineering problems, improvement of the algorithm with higher efficiency is still required. This paper aims to improve both the efficiency and accuracy of LS by active learning and Gaussian process regression (GPR). A new learning function is devised for informing the accuracy of the calculation of the intersection points between each line associated with LS and the failure surface. Then, an adaptive algorithm, with the learning function as an engine and a stopping criterion, is developed for adaptively training a GPR model to accurately estimate the intersection points for all lines in LS scheme, and the number of lines is actively increased if it is necessary for improving the accuracy of failure probability estimation. By introducing this adaptive GPR model, the number of required function calls has been largely reduced, and the accuracy for estimation of the intersection points has been largely improved, especially for highly nonlinear problems with extremely rare events. Numerical test examples and engineering applications show the superiority of the developed algorithm over the classical LS algorithm and some other active learning schemes.

Keywords: Rare Failure Event; Gaussian Process Regression; Line Sampling; Learning Function; Adaptive Experiment Design; Active learning

1 Introduction

Estimating the failure probability of complex structures has long been recognized as one of

* Corresponding author at School of Mechanics, Civil Engineering and Architecture, Northwestern Polytechnical University, Xi'an 710072, China, Email address: pengfeiwei@nwpu.edu.cn (Pengfei Wei)

the most important tasks in civil engineering, mechanical engineering, and related areas. The rapid development of computational power has allowed the simulation of more large-scale structural systems and more complex failure mechanisms, resulting in the requirement of more efficient and accurate computational methods for structural reliability analysis, especially when it comes to rare failure event analysis [1].

From the 60s of last century on, the probabilistic uncertainty propagation and the reliability analysis of structural systems have been coming into the view of the academic community, and plenty of classical computational methods with their own relative merits have been developed. These available methods can be generally grouped into (i) analytical approximation methods, (ii) probability-conservation based methods, (iii) stochastic simulation methods, and (iv) surrogate model method especially equipped by active learning and stochastic simulation.

Analytical approximation methods, including the first-order reliability method (FORM) [2], the second-order reliability method (SORM) [3], etc., aims at approximating the failure probability by statistical moments of the performance function (or limit state function) approximated by Taylor series expansion expended at the most probable points (MPPs). This group of methods requires gradient information of the performance function and is commonly only applicable for problems with continuous performance function of low nonlinearity (around the MPP).

Probability-preservation based methods, including the probability density evolution [4], the direct probability integral method [5], etc., propagate the probabilistic uncertainty from model inputs to outputs and also estimate the failure probability based on the principle of probability conservation. This group of methods commonly rely on experiment design that involves some low-discrepancy sequence techniques. Compared with the first group of methods, the latter is commonly computationally more expensive, but have wider applications, especially to dynamic problems.

Stochastic simulation, such as Monte Carlo simulation (MCS) and advanced MCS, are rooted in the classical probability theory, and the convergence and accuracy of the estimators are guaranteed by the central-limit theory and the law of larger numbers. For structural reliability analysis and especially rare event analysis, advanced MCS such as importance sampling (IS) [6][7], subset simulation (SS) [8][9], line sampling (LS) [10][11] and directional simulation (DS) [12] have been developed, and been comprehensively investigated from both theoretical and application aspects. These simulation methods have their advantages but also disadvantages. For example, SS is applicable for small failure probability estimation and high-dimensional problems, but the convergence is highly affected by the utilized Markov Chain Monte Carlo (MCMC) algorithms

^{[9][13]}, and the estimation errors also increase with respect to the number of introduced intermediate failure events. LS can be especially efficient for small failure probability estimation, but the efficiency and estimation accuracy highly rely on the important direction and the accuracy of calculating the intersection points along each line with the failure event; furthermore, for highly nonlinear problems, LS requires more lines and more evaluations of the system's response on each line, thus can be less efficient. Generally, the stochastic simulation methods provide rigorous treatments of numerical errors but are still computationally expensive for real-world structures with time-consuming simulators.

The requirement of highly efficient reliability analysis has motivated the development and application of surrogate model methods, especially those relying on active learning strategies. In particular, methods that combine the advantages of the Gaussian Process Regression (GPR) model (also called Kriging model) with stochastic simulation methods have received considerable attention. One of the pioneering developments in this direction is the AK-MCS (active learning Kriging driven by MCS) proposed by Echard et al. in Ref. [14]. This method makes full use of the convergence property of MCS, but avoids its high computational cost by actively learning the signs of the performance function for each MCS sample based on the property of GPR model. During the past years, this scheme has received a lot of attention, and many improved versions have been developed. There are two mainstreams of these new developments. The first line is focused on developing new learning functions for more effective learning. Some of the most well-known learning functions include the U-function ^[14], the expected improvement function (EIF) ^[15], the H-function ^[16], the least improvement function (LIF) ^[17], etc. Another line aims at combining the active learning scheme with advanced stochastic simulation to improve the applicability for small (typically less than 10^{-3}) or extremely small (less than 10^{-6}) failure probability estimation. Some of the representative developments in this direction include AK-IS methods that combine AK with (adaptive) IS method ^{[18]-[21]}, AK-SS, or AK-MCMC methods combining AK with SS method ^{[22]-[25]}, etc. Other developments based on AK-MCS also include the parallelization of the algorithm ^[26], the treatment of structural system reliability analysis ^{[27][28]}, etc. The combination of GPR with LS has also been presented in Refs. [29] and [30], but neither of these references considers an active training scheme, and specifically, in Ref. [29] a large number of performance function evaluations are required for calculating a correction coefficient introduced for addressing the model error. Theoretically, the proper combination of LS and active learning Kriging (named as adaptive GPR (AGPR) in this paper) has the potential to substantially reduce the required performance function calls for extremely small failure probability estimation since they are complementary to one another, however, the current

studies are still far from achieving this goal.

To make full use of the advantages of the AGPR model and LS method, we develop a new active learning scheme, which is named AGPR-LS, for efficiently estimating very small failure probabilities. A new active learning function is firstly developed for adaptively learning the intersection points between each line and the failure surface accurately, and which also serves as a stopping criterion. Then, based on this learning function, the adaptive learning scheme AGPR-LS is developed. Extensive numerical and engineering test cases show that the AGPR-LS algorithm is especially efficient and accurate for extremely rare event analysis.

The rest of this paper is organized as follows. Section 2 briefly reviews the classical LS method and highlights the aspects that could be improved by injecting the AGPR model. In section 3, the new learning function and the AGPR-LS algorithm are developed, followed by the case studies in section 4. Section 5 gives conclusions.

2 Review of Line Sampling

LS method, as a classical advanced MCS method, formulates a reliability problem as a group of conditional one-dimensional reliability estimations, and each one-dimensional problem is solved by searching along the line parallel to the important direction^{[10][29]}. The important direction is defined as a vector pointing from the origin to the most probable failure region in input space^{[10][11]}, and the performance of LS highly relies on the accuracy of specifying the important direction.

Assume that the n -dimensional input random variables are denoted by $\mathbf{x} = (x_1, x_2, \dots, x_n)$, and the performance function of a reliability problem is denoted as $y(\mathbf{x})$, where $y < 0$ indicates the failure of the structure. The classical LS method is established in the standard Gaussian space. However, in real-world applications, non-Gaussian input variables are ubiquitous, and these non-Gaussian input variables must be transformed into standard Gaussian variables. This can be realized by using an isoprobabilistic transformation such as Rosenblatt or Nataf transformation [31]. Here we briefly introduce the transformation for the independent case. Let $F(x_i)$ denote the cumulative distribution function (CDF) of any type of distribution, then the isoprobabilistic transformation is $z_i = \Phi^{-1}(F(x_i))$, where $\Phi^{-1}(\cdot)$ indicates the inverse CDF of the standard Gaussian variable z_i . Then the inverse transformation is given as $x_i = F^{-1}(\Phi(z_i))$. For the general case with dependent input variables, one can refer to Ref. [31] for details. For the general case, let $\mathbf{z} = T(\mathbf{x})$ denote the isoprobabilistic transformation (e.g., Rosenblatt transformation) of \mathbf{x} , and the inverse transformation is formulated as $\mathbf{x} = T^{-1}(\mathbf{z})$. Then the performance function with standard Gaussian arguments can be formulated as

$g(\mathbf{z}) = y(T^{-1}(\mathbf{z}))$. For simplification, all the subsequent work will be discussed in standard Gaussian space with the performance function expressed by $g(\mathbf{z})$.

The normalized important direction associated with $g(\mathbf{z})$ is denoted by \mathbf{e}_α . Once \mathbf{e}_α has been estimated, the standard Gaussian space can be orthogonally decomposed into a one-dimensional subspace and a $(n - 1)$ -dimensional subspace, and the input vector can be decomposed into two vectors:

$$\mathbf{z} = z^{\parallel} \mathbf{e}_\alpha + \mathbf{z}^{\perp} \quad (1)$$

, where z^{\parallel} is the one-dimensional standard Gaussian variable so that $z^{\parallel} \mathbf{e}_\alpha$ is parallel to \mathbf{e}_α , and \mathbf{z}^{\perp} is the $(n - 1)$ -dimensional standard Gaussian variables orthogonal to \mathbf{e}_α . For a given value of \mathbf{z} , the value of \mathbf{z}^{\perp} and z^{\parallel} can be calculated with the following expression

$$\begin{cases} z^{\parallel} = \langle \mathbf{e}_\alpha, \mathbf{z} \rangle \\ \mathbf{z}^{\perp} = \mathbf{z} - \langle \mathbf{e}_\alpha, \mathbf{z} \rangle \mathbf{e}_\alpha \end{cases} \quad (2)$$

, where $\langle \cdot, \cdot \rangle$ indicates inner product.

With the above decomposition, the failure probability p_f can be formulated as a double-loop integral, i.e.,

$$p_f = \int_{g(\mathbf{z}) \leq 0} \phi_n(\mathbf{z}) d\mathbf{z} = \int_{g(\mathbf{z}^{\perp} + z^{\parallel} \mathbf{e}_\alpha) \leq 0} \phi(z^{\parallel}) dz^{\parallel} \phi_{n-1}(\mathbf{z}^{\perp}) d\mathbf{z}^{\perp} \quad (3)$$

, with a $(n - 1)$ -dimensional integral of \mathbf{z}^{\perp} in the outer loop and one-dimensional integral of z^{\parallel} in the inner loop. Based on Eq.(3), the LS method involves first generating a set of N_z samples $(\mathbf{z}^{\perp(1)}, \dots, \mathbf{z}^{\perp(s)}, \dots, \mathbf{z}^{\perp(N_z)})$ in the $(n - 1)$ -dimensional subspace of \mathbf{z}^{\perp} based on Eq. (2), and then expressing the estimator of failure probability as:

$$\hat{p}_f = \frac{1}{N_z} \sum_{s=1}^{N_z} \int_{g(\mathbf{z}^{\perp(s)} + z^{\parallel} \mathbf{e}_\alpha) \leq 0} \phi(z^{\parallel}) dz^{\parallel}. \quad (4)$$

For estimating the failure probability based on Eq.(4), one only needs to estimate the N_z one-dimensional integrals, and this problem is schematically shown in Figure 1. As can be seen, given a fixed value $\mathbf{z}^{\perp(s)}$ of \mathbf{z}^{\perp} , $\mathbf{z}^{\perp(s)} + z^{\parallel} \mathbf{e}_\alpha$ varies along the line $l^{(s)}$ which is parallel to the important direction. The intersection point of this line with the failure surface is then denoted as $\mathbf{z}^{\perp(s)} + c^{(s)} \mathbf{e}_\alpha$. Clearly, if the value of z^{\parallel} exceeds $c^{(s)}$, then failure happens along this line, and since z^{\parallel} follows standard Gaussian distribution, the estimator in Eq.(4) can then be further derived as:

$$\hat{p}_f = \frac{1}{N_z} \sum_{s=1}^{N_z} \int_{c^{(s)}}^{+\infty} \phi(z^{\parallel}) dz^{\parallel} = \frac{1}{N_z} \sum_{s=1}^{N_z} \Phi(-c^{(s)}) \quad (5)$$

, and the variance of the estimator is

$$\mathbb{V}(\hat{p}_f) = \frac{1}{N_z(N_z - 1)} \sum_{s=1}^{N_z} (\Phi(-c^{(s)}) - \hat{p}_f)^2. \quad (6)$$

Therefore, the estimation of the failure probability is equivalent to the estimation of the intersection point for each line sample. With the estimator in Eq.(5) and the variance of the estimator in Eq.(6), the Coefficient Of Variation (COV) of the estimate can be computed by:

$$\text{COV}(\hat{p}_f) = \frac{\sqrt{\mathbb{V}(\hat{p}_f)}}{\hat{p}_f}. \quad (7)$$

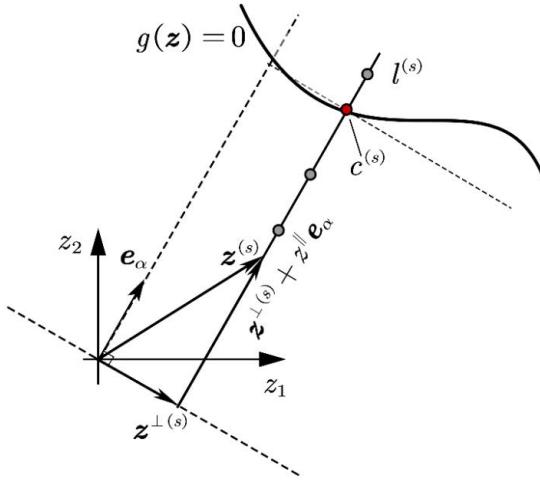


Figure 1 Geometric interpretation of LS in standard Gaussian space

Many numerical methods can be used for calculating the value of $c^{(s)}$ associated with the intersection point on each line^[35], and the most efficient way is to use the three-point-second-order (TPSO) polynomial interpolation method. This procedure involves first generating three values for z^{\parallel} , denoted as c_1 , c_2 , c_3 , and evaluating the performance function values at the three points on the s -th line, then the one-dimensional function $g(z^{\perp(s)} + z^{\parallel}e_{\alpha})$ can be approximated by TPSO polynomial interpolation, thus the value of $c^{(s)}$ is calculated by searching the root of this polynomial.

The above LS scheme has been widely known to be efficient for rare event analysis due to the high efficiency of one-dimensional searching in the most important direction. However, disadvantages also exist. For highly nonlinear problems, the TPSO method can be less effective for accurately estimating the intersection points, resulting in poor accuracy, and further, high nonlinearity also increases the number of required lines for generating sufficiently reliable failure probability estimations, which will largely increase the number of g -function calls. For rare event analysis, the proper selection of the three values c_1 , c_2 and c_3 is also a challenging problem because in most cases the distance of the intersection points from the origin is unknown, and improper selection of the three points will also result in a poor estimation of the failure probability. One can also increase the number of points on each line to improve the accuracy of

estimating the intersection points, but this will also increase the number of required g -function calls. Besides, improper selection of the important direction will also result in poor performance as more lines are required for identifying the whole important failure region. In the next section, we inject the adaptive GPR model into LS to tackle the above disadvantages.

3 The proposed method

3.1 Brief introduction of the GPR model

Before the development of AGPR-LS, it is necessary to briefly review the GPR model. One can refer to Ref. [32] for more details. Given the performance function $g(\mathbf{z})$, the GPR model (denoted as \mathcal{GP}) assumes that:

$$g(\mathbf{z}) \sim \mathcal{GP}(m(\mathbf{z}), c(\mathbf{z}, \mathbf{z}')) \quad (8)$$

, where $m(\mathbf{z})$ is the mean function which can be assumed to be zero, constant, linear, or any closed-form function, and $c(\mathbf{z}, \mathbf{z}')$ is the kernel function representing the covariance between two realizations \mathbf{z} and \mathbf{z}' . Many kinds of kernel functions have been developed for different situations, and one can refer to Ref. [32] for more information. The forms of the mean and kernel functions reflect part of our prior information on the GPR model. Assume we have a set of N_c training data (\mathbf{Z}, \mathbf{y}) , where \mathbf{Z} is a $(N_c \times n)$ matrix with each row being a sample of \mathbf{z} , and \mathbf{y} is a N_c -dimensional column-wise vector with the i -th value being the performance function evaluated at the i -th sample point of \mathbf{z} . Then, the maximum likelihood method can be utilized for estimating the values of the hyper-parameters included in the mean function $m(\mathbf{z})$ and the kernel function. Once these hyper-parameters have been computed, the posterior prediction $\hat{g}(\mathbf{z})$ of the GPR model at a new realization \mathbf{z} is also a Gaussian variable with expectation and variance given by:

$$\hat{\mu}_g(\mathbf{z}) = m(\mathbf{z}) + c(\mathbf{z}, \mathbf{Z})^\top C^{-1}(\mathbf{y} - m(\mathbf{Z})) \quad (9)$$

, and

$$\hat{\sigma}_g^2(\mathbf{z}) = c(\mathbf{z}, \mathbf{z}) - c(\mathbf{z}, \mathbf{Z})^\top C^{-1}c(\mathbf{z}, \mathbf{Z}) \quad (10)$$

, where $c(\mathbf{z}, \mathbf{Z})$ is a column-wise vector of functions with the i -th component being the covariance between \mathbf{z} and the i -th row of \mathbf{Z} , and C is a $(N_c \times N_c)$ -dimensional matrix with the (i, j) -th entry being the covariance between the i -th and j -th rows of \mathbf{Z} . The variance $\hat{\sigma}_g^2(\mathbf{z})$ actually measures the variation of prediction.

Eq.(9) reveals that the GPR model prediction equals to the mean function (prior knowledge on $g(\mathbf{z})$) plus a linear combination of the kernel function between the new site and the training data, where the second term reflects the information learned from the training data. Eq. (10) indicates that the variance of GPR model prediction equals the prior variance minus a term which reflects the reduction of epistemic uncertainty on the value of $g(\mathbf{z})$ learned from the

training data. The above interpretations indicate that, with more training data, the epistemic uncertainty on the prediction of any new sites will be reduced, and this property brings many more benefits for the algorithm to be developed. In the next subsection, we introduce a new learning function that serves as the engine of the proposed AGPR-LS algorithm.

3.2 Learning function

From the rationale of the GPR model, it is known that once the true performance function $g(\mathbf{z})$ is approximated by a GPR model $\hat{g}(\mathbf{z})$ with mean $\hat{\mu}_g(\mathbf{z})$ and variance $\hat{\sigma}_g^2(\mathbf{z})$, the prediction of the performance function at any new realization is a Gaussian variable. This property brings two benefits for LS. First, the distance $c^{(s)}$ w.r.t. the intersection point between the failure surface of $\hat{g}(\mathbf{z})$ and the s -th line can be easily computed by any numerical scheme due to the smoothness of $\hat{\mu}_g(\mathbf{z}^{\perp(s)} + z^{\parallel} \mathbf{e}_\alpha)$. Second, it can be used to judge whether the estimated value $c^{(s)}$ is accurate enough. For answering the second question, we develop a new definition of the learning function, which is expressed as:

$$\kappa(\mathbf{z}) = \int_{-\epsilon + \hat{\mu}_g(\mathbf{z})}^{\epsilon + \hat{\mu}_g(\mathbf{z})} \phi(\hat{g}(\mathbf{z}) | \hat{\mu}_g(\mathbf{z}), \hat{\sigma}_g^2(\mathbf{z})) d\hat{g}(\mathbf{z}) \quad (11)$$

, where $\phi(\cdot | \hat{\mu}_g(\mathbf{z}), \hat{\sigma}_g^2(\mathbf{z}))$ refers to the probability density function of Gaussian distribution with mean $\hat{\mu}_g(\mathbf{z})$ and variance $\hat{\sigma}_g^2(\mathbf{z})$, and ϵ is the error tolerance to control the width of integral interval, whose value should be close to zero. Generally, the learning function can be interpreted as the probability of the true value of $g(\mathbf{z})$ being included in the small interval $[-\epsilon + \hat{\mu}_g(\mathbf{z}), \epsilon + \hat{\mu}_g(\mathbf{z})]$.

For reliability analysis, specifically for the intersection point $\mathbf{z}^{\perp(s)} + c^{(s)} \mathbf{e}_\alpha$ of a given line, where the value of $\hat{\mu}_g(\mathbf{z})$ theoretically equals to zero, the learning function actually measures the probability that the g -function value at the true intersection point being included in the pre-specified narrow bounds $[-\epsilon, \epsilon]$. The larger this probability is, the more accurately this intersection point is estimated. The learning function is schematically interpreted in Figure 2. As can be seen in Figure 2(a), \mathbf{z}^{P_1} and \mathbf{z}^{P_2} are the intersection points of the same line $\mathbf{z}^{(s)} = \mathbf{z}^{\perp(s)} + z^{\parallel} \mathbf{e}_\alpha$ with the failure surfaces $\hat{g}_1(\mathbf{z}) = 0$ and $\hat{g}_2(\mathbf{z}) = 0$, and the two GPR models are both meta-models of the same limit state function. Figure 2(b) shows the corresponding probability density function of $\hat{g}(\mathbf{z})$ at the two points. Obviously, $\hat{g}_1(\mathbf{z}^{P_1})$ has a larger variation of prediction than $\hat{g}_2(\mathbf{z}^{P_2})$, thus its probability mass contained within the interval $[-\epsilon, \epsilon]$ is less than that of $\hat{g}_2(\mathbf{z}^{P_2})$; accordingly, the learning function value at \mathbf{z}^{P_1} is smaller than that evaluated at \mathbf{z}^{P_2} . Thus, a larger value of the learning function indicates a better estimation of the intersection point. It is easy to observe that $0 \leq \kappa(\mathbf{z}) \leq 1$, where $\kappa(\mathbf{z}) \approx 0$ indicates that the corresponding intersection point is poorly estimated, and $\kappa(\mathbf{z}) \approx 1$

reveals that the intersection point is accurately computed. Commonly, $\kappa(\mathbf{z}) \geq \kappa^* = 0.985$ provides satisfactory estimation, here κ^* denotes the learning function threshold. In the next subsection, we develop the AGPR-LS algorithm with the proposed learning function $\kappa(\mathbf{z})$.

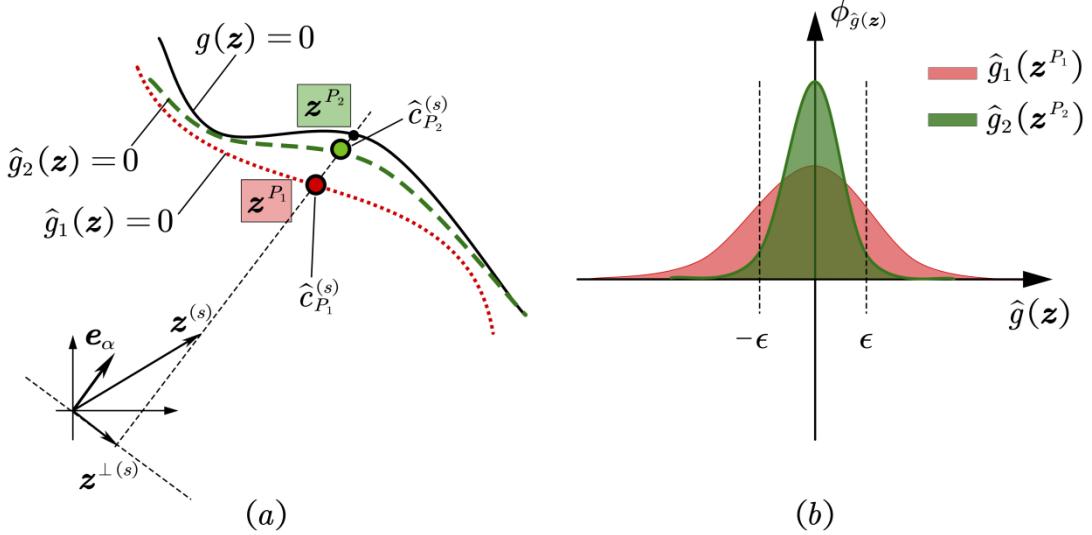


Figure 2 Schematic interpretation of the learning function $\kappa(\mathbf{z})$

3.3 The AGPR-LS algorithm

The basic idea of the AGPR-LS algorithm is then adaptively learning the correct intersection points for each line of LS based on the GPR model, which is actively updated by including the most informative points identified by the learning function $\kappa(\mathbf{z})$. The flowchart of the algorithm is represented in Figure 3. The detailed procedure is also described as follows.

◆ Step 1: Initialization

The algorithm is started by setting the total number N of candidate lines, the number N_0 of initial lines for training the initial GPR model, the threshold κ^* and the error tolerance ϵ . Then, generate N samples $\mathbf{Z}_k = (\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots, \mathbf{z}^{(N)})$ so as to create N lines along the important direction \mathbf{e}_α by using, e.g., Latin-hypercube sampling. Then randomly select N_0 lines from those N lines, and estimate the intersection point for each of these N_0 lines by using TPSO polynomial interpolation that is also mentioned in section 2; the found intersection points are expressed by $\mathbf{z}^{(s)\perp} + c^{(s)}\mathbf{e}_\alpha (s=1, \dots, N_0)$. This procedure introduces $3N_0$ training data points, which are added to the training data set \mathbf{S} . After that, evaluate the g -function of the N_0 intersection points, and also add them into the training data set. In practical applications, the important direction generally cannot be derived analytically, and numerical procedures such as FORM need to be used for calculating it numerically [2]. This numerical procedure also introduces N_{e_α} extra g -function calls, and it is recommended to also add these

data points into the training data set \mathbf{S} . Let N_c denote the training sample size of \mathbf{S} , so that the training sample size after initialization will become $N_c = N_{\epsilon_\alpha} + 3N_0 + N_0$. The number of lines N can be set to be the same as in the classical LS algorithm. Commonly, higher nonlinearity and/or larger span failure regions require a larger number of candidate lines. N_0 can be set to be a small value less than ten, e.g., 4.

◆ **Step 2: Train or update the GPR model**

Train or update the GPR model $\hat{g}(\mathbf{z})$ by using the training data set \mathbf{S} . In this step, one needs to specify the mean function $m(\mathbf{z})$ and the kernel function $c(\mathbf{z}, \mathbf{z}')$. Commonly, if the nonlinearity of the performance function is not high, zero or constant mean function is recommended. However, if the nonlinearity is high, linear or quadratic polynomial mean function is recommended. For the kernel function, the squared exponential kernel is utilized in this work. The function “fitrgp” in the Matlab Statistic and Machine Learning Toolbox is utilized in this work for training the GPR model.

◆ **Step 3: Learning from the GPR model**

The GPR model trained in **Step 2** provides a pair of quantities, i.e., $\mu_g(\mathbf{z})$ and $\sigma_g^2(\mathbf{z})$, for any realization \mathbf{z} . Compute the intersection point $\mathbf{z}^{(s)\perp} + c^{(s)}\mathbf{e}_\alpha$ for each line (including the N_0 initial lines) by solving the univariate equation $\mu_g(\mathbf{z}^{(s)\perp} + z^\# \mathbf{e}_\alpha) = 0$. During this procedure, it may happen that, for some lines, no zero point can be found, indicating a large GRP prediction error in this line. One can simply set the corresponding value of $c^{(s)}$ as the average values of c for other lines, but this point is definitely not an estimated intersection point. Then, for each line, compute the learning function value $\kappa^{(s)}$ for the intersection point $\mathbf{z}^{(s)\perp} + c^{(s)}\mathbf{e}_\alpha$ by modifying the learning function of Eq. (11) as:

$$\kappa^{(s)} = \int_{-\epsilon}^{\epsilon} \phi(y|0, \sigma_g^2(\mathbf{z}^{(s)\perp} + c^{(s)}\mathbf{e}_\alpha)) dy. \quad (12)$$

Find the minimum value $\kappa_{\min} = \min_{s=1}^N (\kappa^{(s)})$. If $\kappa_{\min} < \kappa^*$, find the intersection point with the minimum value of learning function, compute the corresponding g -function value, and add this point to the training data set \mathbf{S} , let $N_c = N_c + 1$, and go back to **Step 2**; else go to **Step 4**.

◆ **Step 4: Estimation and Iteration**

Estimate the failure probability p_f with the intersection point computed for each line in **Step 3** by Eqs. (5) and (6). If the COV estimated by Eq. (7) is higher than a pre-specified tolerance, say 0.05, then create N^a more lines. Let $N = N + N^a$, and go back to **Step 3**; otherwise, end the algorithm. N^a can be set to be 50 or more.

■

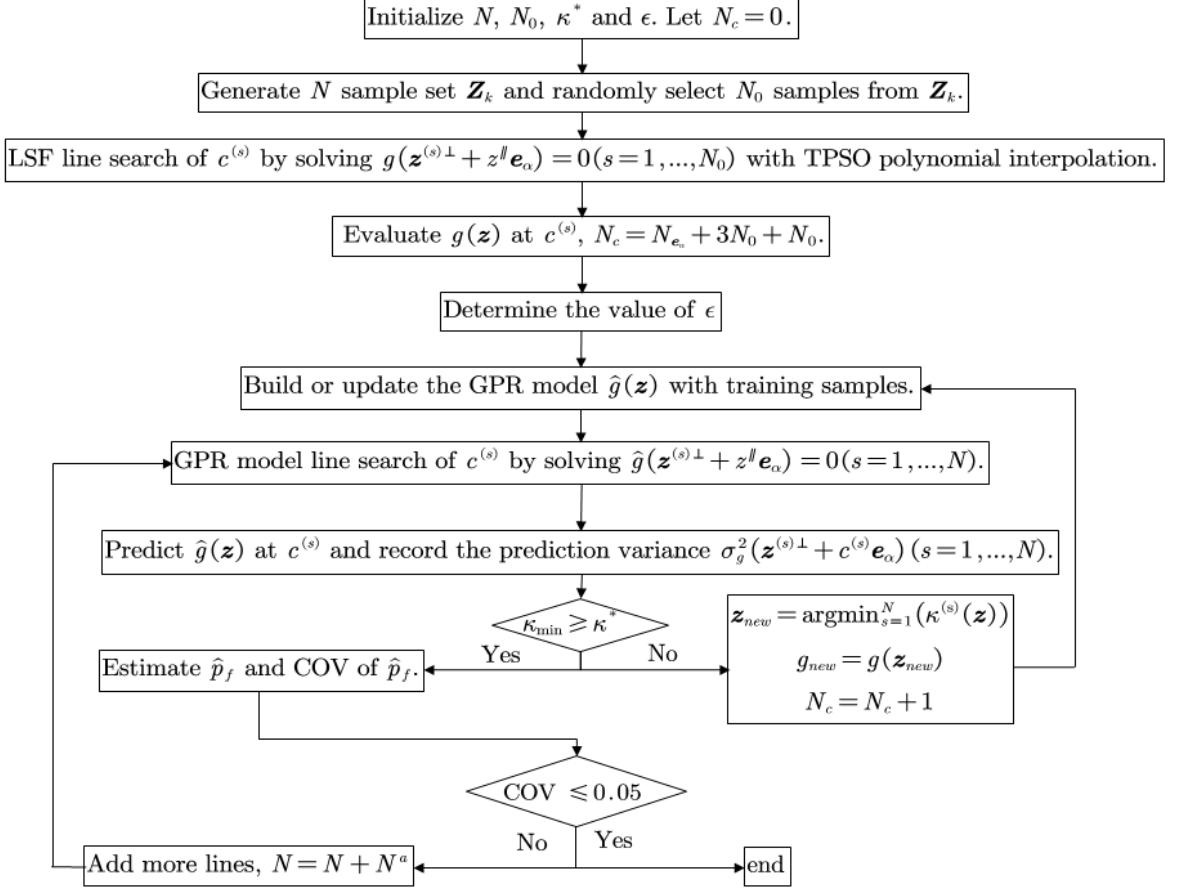


Figure 3 Flowchart of the AGPR-LS algorithm

In **step 1**, the value of the error tolerance ϵ should be carefully treated. Since the GPR prediction at each calculated intersection point equals to zero, the next point being selected by the learning function in Eq. (12) is always the one with the largest value of prediction variance if it is an intersection point, or the one on a line whose intersection point is not available by solving $\mu_g(z^{(s)\perp} + z^l e_\alpha) = 0$. The value of ϵ does not affect the training data to be added in each iteration. However, this value definitely affects the stopping criteria in **step 3**. A larger value of ϵ results in faster convergence but also poorer accuracy of each intersection point, while smaller value requires more training data, leading to higher computational cost. Therefore, a proper tradeoff should be made for ϵ . Based on our experience, it is suggested to set ϵ as p times the average absolute values of g -function at the intersection points of the initial N_0 lines estimated by TPSO interpolation, where $p = 10 \sim 100$. Another choice of ϵ is suggested as $(0.01 \sim 0.10)\sigma_y$, where σ_y is the standard deviation of the g -function. This value can also be updated at each iteration based on the intersection points which are being accurately estimated by the trained GPR model.

It is found in the last step that, increasing the number of lines will not increase the required

number of g -function calls too much. As for most newly added lines, the well-fitted GPR model can produce accurate estimations of the intersection points. In the case that for some newly added lines, the intersection points are not accurately estimated, the active learning function $\kappa(\mathbf{z})$ can commonly improve those estimations to required accuracy level with only a small number of training data (thus g -function calls) being added. Thus, compared with the classical LS algorithm, the AGPR-LS is more applicable to highly nonlinear performance function, and also the case where the important direction is not accurately specified. Besides, for rare event analysis, searching the intersection point based on the fitted GPR model can be much easier and more efficient due to the smoothness of the GPR predictor.

The AGPR-LS algorithm also has more appealing advantages over the advanced AK-MCS algorithms. The classical AK-MCS algorithm is known to be not effective for rare event analysis due to the large size of the required sample pool. Many improved algorithms such as the AK-MCMC have been developed [24][25]. As will be illustrated in the test examples, the AK-MCMC algorithm needs to approximate a set of intermediate failure surfaces adaptively, which will cost a considerable number of g -function calls. However, due to the high efficiency of the one-dimensional line search, the AGPR-LS method can be much more efficient for identifying the failure surface, especially when the failure probability is extremely small (less than 10^{-6}). Besides, all the AK-MCS and advanced AK-MCS algorithms require a large sample pool (with e.g., 10^5 samples) especially for extremely small failure probability, making the implementation inefficient. The AGPR-LS algorithm avoids this shortcoming since only a much small line pool (commonly with several hundreds of lines) is required.

However, the AGPR-LS algorithm also has its limits. The high efficiency of line searching is based on the specified important direction. In most applications, the failure region is mainly concentrated in one direction, and the proposed algorithm can be extremely efficient. However, if multiple important directions exist, the algorithm can be less effective for approaching the whole failure region.

4 Case studies

4.1 A two-dimensional numerical example

A two-dimensional toy example is considered with limit state function:

$$y = g(\mathbf{x}) = 1 - \frac{(x_1 - 1)^2}{a^2} - \frac{(x_2 - 1)^3}{b^2} + d \sin(cx_1) \quad (13)$$

, where a and b are constants used for determining the magnitude of p_f , c and d are also constants used to justify the nonlinearity of the limit state function. x_1 and x_2 are two independent random input variables, both of which follow standard Gaussian distribution. The

important direction for this example is assumed to be known precisely, and given in Table 1.

Next, we consider three cases for this example. The first case is utilized for demonstrating the robustness of the proposed AGPR-LS algorithm given different important directions, the second case is used for demonstrating its performance for extremely small failure probability, and the third case is designed for investigating its performance for highly nonlinear problems.

For implementing the classical LS in case 1 and case 2, the intersection point for each line is calculated by the three-point interpolation, thus the total number of function calls is $N_c = 3N_{line}$; while for case 3, the intersection point for each line is computed by the four-point interpolation due to the high nonlinearity, thus the total number of function calls is $N_c = 4N_{line}$. For all three cases, the classical LS algorithm is implemented by the COSSAN software [34].

For implementing the AGPR-LS algorithm, four initial training lines are created in the same way with the classical LS algorithm, and for each line, the three-point interpolation is utilized for calculating the intersection points. Thus, the total number of initial training samples is sixteen.

◆ **Case 1: $a = 6$, $b = 7$ and $d = 0$**

The reference result is computed by LS and IS, as given in Table 1. We implement the LS algorithm by setting the line size as 10, 100, and 1000 respectively, and the corresponding results are reported in Table 1. As can be seen, although the mean estimates of the three runs are all near to the reference solution, the COVs with 10 and 100 lines are both higher than 20%, indicating that the accuracy is not acceptable. When the line size is increased to 1000, the COV drops to 2.7%, indicating the convergence of the LS algorithm.

For implementing the AGPR-LS algorithm, the important direction is set to $\mathbf{e}_\alpha = (0, 1)$ and $\mathbf{e}_\alpha = (-0.3420, 0.9397)$ to demonstrate the insensitivity of the algorithm to the accuracy of important direction. For both runs, the stopping criteria are set to be $\text{COV}(\hat{p}_f) \leq 5\%$.

The training process of AGPR-LS for case 1 with the important direction $\mathbf{e}_\alpha = (0, 1)$ is schematically shown in Figure 4. As can be seen, four lines are first generated randomly, and for each line, the three-point second-order interpolation is utilized for calculating the intersection point with the limit state function. The above procedure introduces sixteen input-output samples for training the initial GPR model. By setting the parameters as $\epsilon = 0.01$ and $\kappa^* = 0.985$, 596 more lines are generated, but only four more training samples are added sequentially based on the learning function $\kappa(\mathbf{x})$. Based on the 20 training samples, the intersection points for all the 600 lines are accurately estimated, and the failure probability is then calculated based on the LS estimators, and the results are shown in the second row of Table 1. The reference results generated by another adaptive learning method AK-MCMC developed in Ref. [24] are also listed for comparison. As can be seen, results generated by all the methods

are in good agreement, and the COV of the estimate by AGPR-LS is quite small (approximately 4.5%), indicating that the failure probability estimation by AGPR-LS for this case is accurate, robust and efficient.

We then change the important direction to $\mathbf{e}_\alpha = (-0.3420, 0.9397)$ to test the sensitivity of the performance of AGPR-LS to the important direction. The training process is shown in Figure 5, and the results are given in Table 1. It is shown that, although the utilized important direction is distinct from the most informative one, the AGPR-LS algorithm still produces a correct and robust estimation, and the total number of g -function calls is still 20. It is also shown that the total number of required lines has increased to 1500, indicating when the important direction is not the most information one, more lines are required. However, this does not result in a significant increment of the computational cost since the required number of required training samples is still 20. This indicates that, for this case, the AGRP-LS method is not very sensitive to the important direction.

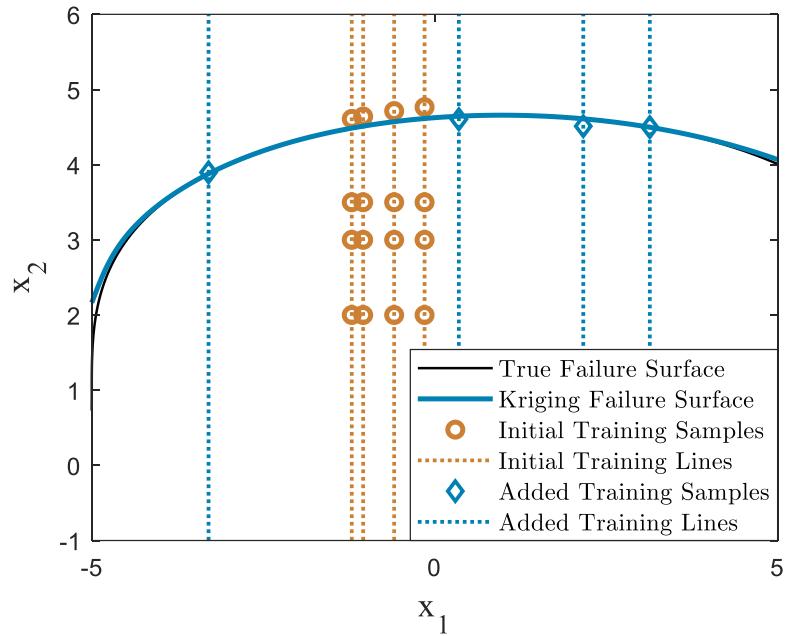


Figure 4 Results of AGPR-LS for case 1 of the toy example with important direction being

$$\mathbf{e}_\alpha = (0, 1).$$

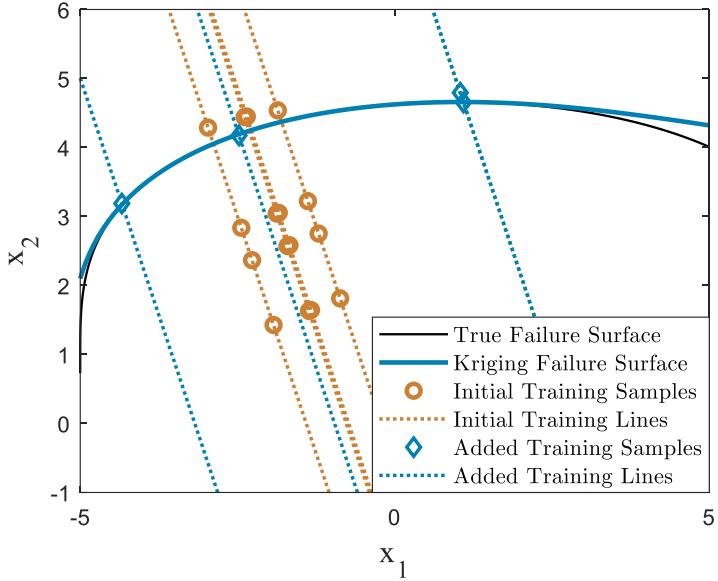


Figure 5 Results for case 1 of the toy example generated by AGPR-LS by setting the important direction as $\mathbf{e}_\alpha = (-0.3420, 0.9397)$.

◆ **Case 2:** $a = 9$, $b = 11$ and $d = 0$

With this setting, we aim at testing the performance of the AGPR-LS algorithm for analyzing the extremely rare failure events. In this case, the important direction is set to be $\mathbf{e}_\alpha = (0, 1)$. The classical LS algorithm is still implemented using COSSAN with 10, 100, and 1000 lines, respectively, and the results are reported in Table 1. As can be seen, with the line size less than 100, it is impossible to create a robust estimate with COV less than 10%.

We then implement the AGPR-LS algorithm by setting the stopping criteria as $\text{COV}(\hat{p}_f) \leq 5\%$, and we use four initial training lines (thus sixteen initial training samples) to start the AGPR-LS algorithm. The details of the training process are illustrated in Figure 6, and the estimation results are listed in Table 1, together with the estimations by AK-MCMC, LS, and IS for comparison.

It is seen that, although the failure probability is extremely small (with the order of magnitude being 10^{-9}), the AGPR-LS algorithm can still give an accurate and robust estimation, with the same number of g -function calls as in case 1. This means that estimating a smaller failure probability does not necessarily increase computational cost, attributed to the high efficiency of line search. It is also found that the COV of the estimation, in this case, is even smaller than that in case 1, although the line size (250) is less than that in case 1, indicating that the AGPR-LS method can be especially useful for extremely rare event analysis. Figure 6 shows that the intersection points between the initial four lines and limit state function computed by three-points second-order interpolations are not as accurate as those in case 1. However, during

the adaptive training process, the intersection points for all lines (including the four initial training lines) are adaptively updated, and the final intersection points for all lines are much accurately calculated. This indicates that, in the classical LS method, the inaccuracy of estimating the intersection points will result in an extra numerical error, however, by injecting the active learning procedure into LS, this shortcoming can be largely alleviated.

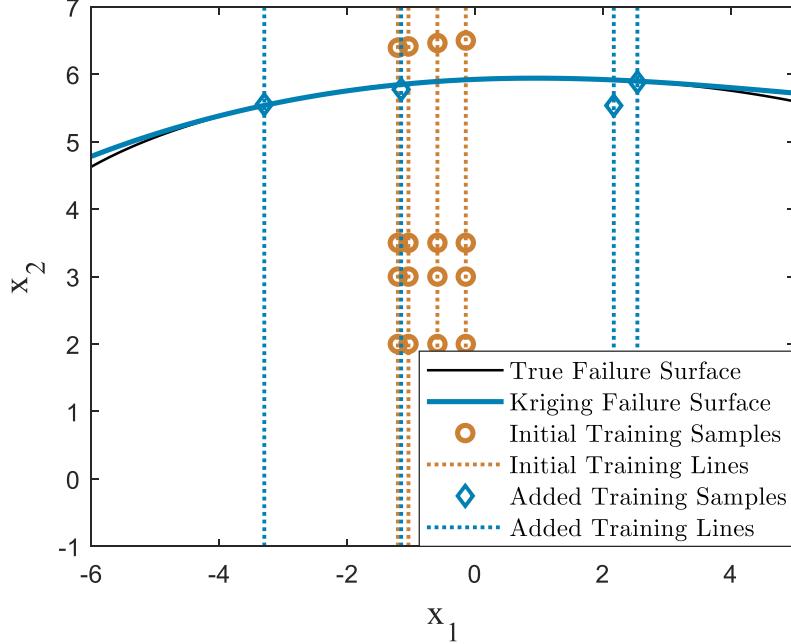


Figure 6 Training process of APGR-LS algorithm for case 2 of the toy example.

◆ **Case 3:** $a = 9$, $b = 10$, $c = 4$ and $d = 0.5$.

With this setting, the failure probability is still very small, but the nonlinearity of the limit state function is much higher than that of the former two cases (see Figure 7 for the true limit state function). The classical LS algorithm is implemented using COSSAN with line sizes varying, and the results are listed in Table 1. As can be seen, for this highly nonlinear problem, even when the line size touches 1000, the COV is still higher than 5%, which is much higher than those in case 1 and case 2. This is unquestionably caused by the high nonlinearity of the g -function. This phenomenon indicates that, for highly nonlinear problems, the classical LS algorithm requires many more lines to achieve acceptable accuracy. As will be shown later, this can be largely alleviated by the AGPR-LS algorithm.

The stopping criteria of the AGPR-LS algorithm is still set to be $\text{COV}(\hat{p}_f) \leq 5\%$, and the important direction is set to be $e_\alpha = (0, 1)$. The training process is then shown in Figure 7. For this highly nonlinear limit state function, 41 more samples are adaptively added to accurately estimate the intersection points for all the candidate lines, thus the total number of g -function

calls is 57, which is still much smaller than that of AK-MCMC algorithm, which is 173, as shown in Table 1. This indicates that, for even highly nonlinear problems, the AGPR-LS algorithm is much more efficient than the AK-MCMC algorithm. This is because, for small failure probability, many g -function calls need to be performed for approximating a set of intermediate failure surfaces, and this number can be large when the nonlinearity of the limit state function is high; however, the AGPR-LS algorithm can approach the true failure surface very efficiently along each line without the requirement of approximating any intermediate failure surface, no matter how far the failure surface is, thus can be extremely effective. It can be seen from Table 1 that, both the estimations of AGPR-LS and AK-MCMC algorithms are accurate when compared with the reference solutions computed by LS and IS algorithms, but the estimation of AGPR-LS is a little bit better than that of AK-MCMC. In terms of efficiency, the AGPR-LS algorithm consumes much fewer g -function calls than AK-MCMC.

Compared with case 1 and case 2, the required number of training samples has increased, but it is still small. This increment is caused by the necessity of capturing highly nonlinear behavior along the failure surface. From Figure 7, it is also seen that, on some lines, more than one training sample is added, this is also due to the high nonlinearity of the limit state function along these lines. However, as long as the limit state function is continuous along this line, this active learning mechanism driven by the learning function can always approach the real intersection points within the allowed error range.

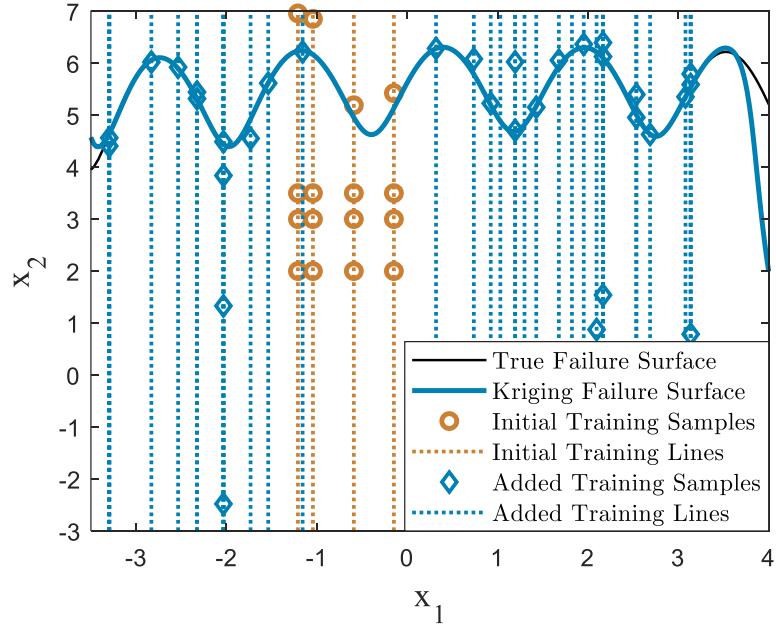


Figure 7 Learning details of AGPR-LS algorithm for case 3 of the toy example

Table 1 Reliability analysis results of the toy example

	Methods	Parameter settings	N_{lines}	p_f	COV(%)	N_c	
Case 1	AGPR-LS	$\mathbf{e}_\alpha = (0, 1)$ $\epsilon = 0.01, \kappa^* = 0.985,$	600	2.596×10^{-6}	4.5	20	
		$\mathbf{e}_\alpha = (-0.3420, 0.9397)$ $\epsilon = 0.01, \kappa^* = 0.985$	1.5×10^3	2.740×10^{-6}	4.7	20	
	AK-MCMC	—	—	2.581×10^{-6}	7.3	47	
	LS	$\mathbf{e}_\alpha = (0, 1)$	10	3.606×10^{-6}	21.5	30	
			100	3.295×10^{-6}	20.9	300	
			10^3	2.728×10^{-6}	2.7	3×10^3	
	IS	MPP = (0, 4)	—	2.646×10^{-6}	3.1	10^4	
	Case 2	AGPR-LS	$\mathbf{e}_\alpha = (0, 1)$ $\epsilon = 0.01, \kappa^* = 0.985$	250	1.891×10^{-9}	3.8	20
			—	—	1.649×10^{-9}	7.7	150
		LS	$\mathbf{e}_\alpha = (0, 1)$	10	2.319×10^{-9}	19.1	30
				100	2.305×10^{-9}	12.9	300
				10^3	2.033×10^{-9}	1.7	3×10^3
		IS	MPP = (0, 5)	—	2.027×10^{-9}	3.6	10^4
Case 3	AGPR-LS	$\mathbf{e}_\alpha = (0, 1)$ $\epsilon = 0.01, \kappa^* = 0.985,$	1.9×10^3	3.520×10^{-7}	4.8	57	
			—	—	3.141×10^{-7}	6.7	173
	LS	$\mathbf{e}_\alpha = (0, 1)$	10	5.331×10^{-7}	36.5	40	
			100	2.159×10^{-7}	21.2	400	
			10^3	3.515×10^{-7}	6.8	4×10^3	
	IS	MPP = (0, 5)	—	3.560×10^{-7}	5.7	10^4	

4.2 Dynamic response of a nonlinear oscillator

Consider a nonlinear undamped single degree of freedom system, shown in Figure 8, which is adapted from Ref.[14]. The limit state function is formulated as:

$$g(c_1, c_2, m, r, t_1, F_1) = 3r - \left| \frac{2F_1}{m\omega_0^2} \sin\left(\frac{\omega_0^2 t_1}{2}\right) \right| \quad (14)$$

, where $\omega_0 = \sqrt{(c_1 + c_2)/m}$. The six input variables are all assumed to follow Gaussian distribution with distribution parameters shown in Table 2.

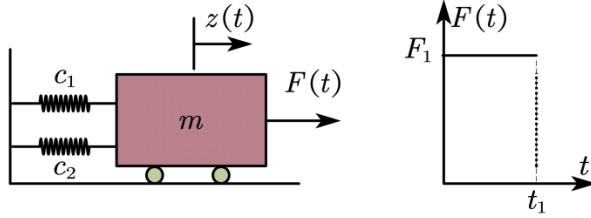


Figure 8 A nonlinear oscillator

Table 2 Probability distributions of the six input variables of the nonlinear oscillator

Variables	Distribution	Mean	COV
m	Gaussian	1	0.05
c_1	Gaussian	1	0.1
c_2	Gaussian	0.1	0.1
r	Gaussian	0.5	0.1
F_1	Gaussian	0.45	1/6
t_1	Gaussian	1	0.2

The results of the failure probability estimated by AGPR-LS, AK-MCMC, LS, and IS are listed in Table 3. The most probable point (MPP) is estimated by the FORM method to be (-0.4405, -1.2432, -0.1243, -4.0363, 2.6542, 2.3750), and the total number of function calls is seventeen. Then IS procedure is implemented by moving the sampling center from the mean point to the MPP. The important direction for AGRP-LS and LS is then derived from the MPP as (-0.0794, -0.2241, -0.0224, -0.7282, 0.4787, 0.4285).

The LS algorithm is implemented using COSSAN by setting the line size as 10, 100, and 500 respectively, and for each line, five points are used for estimating the intersection points. As can be seen, with ten lines, the accuracy is not acceptable as the COV is higher than 20%. The accuracy of results generated with 100 lines is acceptable for engineering computation, but the COV is still too high for academic research. With 500 lines, the COV is below 5%, and the estimate can be regarded as the reference solution.

For running the AGPR-LS algorithm, four initial training lines (thus sixteen initial training samples) are randomly generated. One notes that, in this example, the parameter κ^* is still set to be 0.985, while the parameter ϵ is set to be 0.005, which is different from the last example. This is because the level of magnitude of the response in this example is smaller than the last example. For implementing the AK-MCMC algorithm in Ref. [24], the size N of the sample pool for each intermediate failure surface is set to be 10^5 , the initial training sample size N_0 is set to be 12, and the intermediate probability p_0 is set to be 0.01.

Table 3 shows that the results produced by the four methods are in good agreement. Compared with the AK-MCMC algorithm, the AGPR-LS demanded only 77 g -function calls, which is much less than that of the AK-MCMC algorithm. However, the AGRP-LS algorithm gives a better estimate since the COV of the estimation is much smaller than that of the AK-MCMC algorithm. This indicates that for this example with extremely small failure probability, the AGPR-LS method outperforms AK-MCMC. The AGPR-LS results are also competitive with those generated by the classical LS algorithm with 500 lines due to the same level of COV, but the computational cost is much lower.

For illustrating the learning process of AGPR-LS, we plot the minimum value of the learning function $\kappa(\mathbf{x})$ at each iteration step in Figure 9. As can be seen, with more training samples added, the minimum value of $\kappa(\mathbf{x})$ over all lines tends to increase, but this is not always the case at each step. With the minimum value adaptively approaching one, it is believed that the intersection point for each line is accurately calculated, resulting in an accurate estimation of failure probability as long as the number of lines is enough.

Table 3 Reliability analysis results of the nonlinear oscillator

Methods	Parameter Settings	N_{lines}	p_f ($\times 10^{-8}$)	COV (%)	N_c
AGPR-LS	$\epsilon = 0.005, \kappa^* = 0.985$	300	1.530	4.1	$17+60=77$
AK-MCMC	$N = 10^5, N_0 = 12, p_0 = 0.01$	—	1.493	9.9	155
LS	—	10	2.370	27.8	$17+50=67$
		100	1.889	10.4	$17+500=517$
		500	1.775	4.6	$17+2.5 \times 10^3=2517$
IS	—	—	1.512	2.7	$17+10^4$

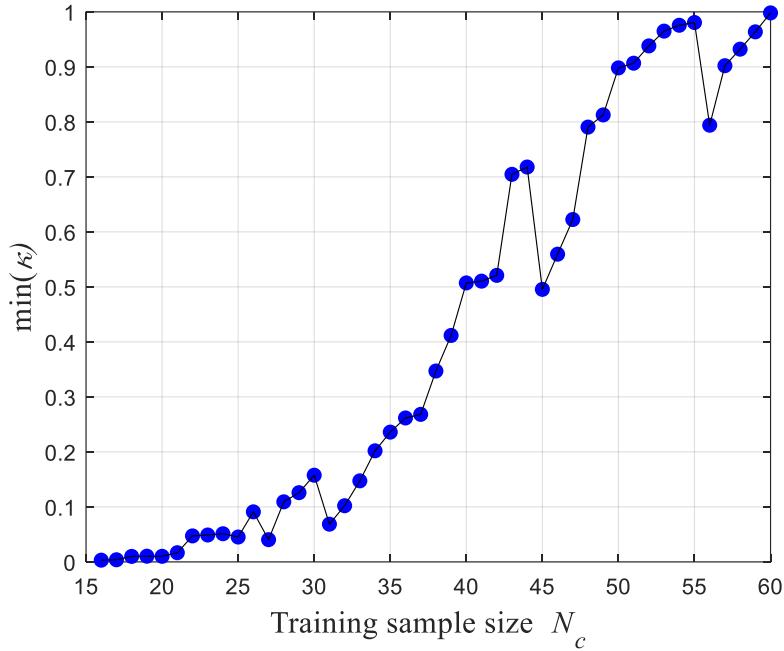


Figure 9 Plots of the minimum value of the learning function against the learning step for the nonlinear oscillator example

4.3 Confined seepage model

A steady state of confined seepage below a dam discussed in Ref.[35] is considered, and the elevation of the dam is shown in Figure 10. The water flows from the upstream side (segment AB) towards the downstream side (segment CD) through the two permeable layers, silty gravel and silty sand, and an impermeable layer is below these two permeable layers. It is assumed that there is no water flow on any of the boundaries except for the segments AB and CD. In Figure 10, the water height h_D in the upstream side of the dam is modeled as a random variable with uniform distribution $U(7[m], 10[m])$, the hydraulic head h_W over the impermeable layer is $h_W = h_D + 20[m]$. The permeability of the two permeable layers are assumed to be anisotropic and modeled as random variables following lognormal distribution, the horizontal and vertical permeabilities are denoted by $k_{xx,i}$ and $k_{yy,i}$ ($i=1$ for sand layer, $i=2$ for gravel layer). The distribution parameters of the permeability of the two soil layers as well as the water height are provided in Table 4. The governing partial differential equation of the seepage problem is

$$k_{xx,i} \frac{\partial^2 h_W}{\partial x^2} + k_{yy,i} \frac{\partial^2 h_W}{\partial y^2} = 0, \quad i=1, 2. \quad (15)$$

The boundary conditions are the hydraulic head over segments AB and CD. A finite element mesh with 3413 nodes and 1628 quadratic triangular elements is established to solve the governing equation. The seepage q at the downstream side can be calculated by

$$q = - \int_{CD} k_{yy,2} \frac{\partial h_w}{\partial y} dx. \quad (16)$$

Note that the unit of q is the volume over time over distance [$L/h/m$]. Commonly, we expect the seepage to be small enough for ensuring a safe state of the dam, so the failure event of interest is defined when seepage q exceeds a prescribed threshold 50 [$L/h/m$], and the limit state function is $g(\mathbf{x}) = 50 - q(\mathbf{x})$.

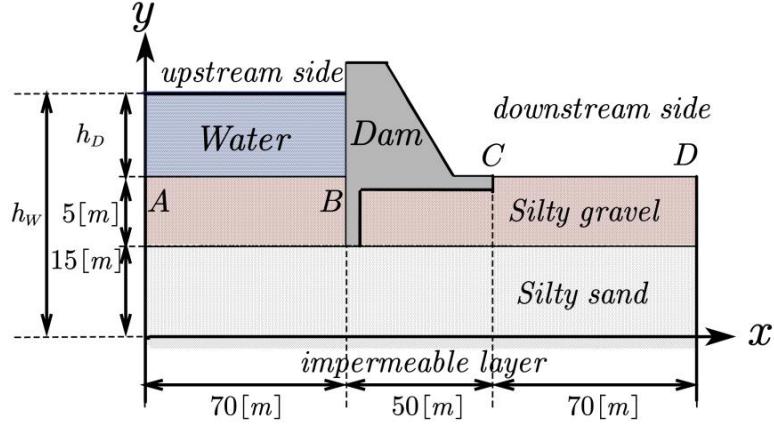


Figure 10 Elevation of the dam in confined seepage model

Table 4 Distribution parameters of input variables for confined seepage model

Variables	Description	Distribution type	Parameter1	Parameter2
$k_{xx,1} [10^{-7}\text{m/s}]$	Horizontal permeability of silty sand soil layer	lognormal	Mean=5	COV=1
$k_{yy,1} [10^{-7}\text{m/s}]$	Vertical permeability of silty sand soil layer	lognormal	Mean=2	COV=1
$k_{xx,2} [10^{-6}\text{m/s}]$	Horizontal permeability of silty gravel soil layer	lognormal	Mean=5	COV=1
$k_{yy,2} [10^{-6}\text{m/s}]$	Vertical permeability of silty gravel soil layer	lognormal	Mean=2	COV=1
$h_D [\text{m}]$	water height in upstream side of dam	uniform	$a_{h_D} = 7$	$b_{h_D} = 10$

We first calculate the MPP by FORM, and the result is (3.1257, 1.5715, 1.0808, 0.9211, 0.8865), thus the important direction can be specified as (0.8059, 0.4052, 0.2787, 0.2375, 0.2286) by normalizing the vector from the origin to MPP. The total number of function calls in FORM is 30. Then we implement the AGPR-LS algorithm with four lines and thus $N_0 = 16$ initial

training points. The algorithm parameters are set to be $\epsilon = 0.1$ and $\kappa^* = 0.985$. The results are then reported in Table 5, together with the reference results computed by AK-MCMC, LS, and IS respectively, where IS is implemented by shifting the sampling center to the MPP. The LS is implemented by setting the line size to 10, 100, and 200, and it is shown that the COV of the estimate generated with 10 lines is over 20%, thus it is not acceptable. However, the results generated with 100 or more lines are robust and accurate, and can be served as reference solutions. As can be seen, the failure probability estimated by AGRP-LS is a little bit better than that calculated by AK-MCMC, when compared with the reference solutions computed by IS and LS. However, the AGRP-LS demands only 80 g -function calls, which is much less than that consumed by AK-MCMC. This indicates that, for this example, both the AGRP-LS and AK-MCMC algorithms work well, but the AGRP-LS algorithm is much more efficient than AK-MCMC.

Similarly, the minimum value of $\kappa(\mathbf{x})$ against the iteration step is schematically shown in Figure 11. A similar phenomenon as seen in Figure 9 is found here, that is, the minimum value of $\kappa(\mathbf{x})$ across all lines decreases rapidly with the increase of training samples identified by the learning function, and finally with only 50 training points, the AGPR-LS algorithm produces accurate estimations for the intersection points of all lines, and also accurate estimation of the failure probability.

Table 5 Reliability analysis results for the confined seepage model

Methods	Parameter Settings	N_{lines}	$p_f (\times 10^{-5})$	COV (%)	N_c
AGPR-LS	$\epsilon = 0.1, \kappa^* = 0.985$	200	2.811	4.6	$30+50=80$
AK-MCMC	$N = 1e5, N_0 = 12, p_0 = 0.01$	—	2.465	4.9	337
LS	—	10	1.696	25.6	$30+30=60$
		100	3.006	7.8	$30+300=330$
		200	2.933	4.2	$30+600=630$
IS	—	—	2.846	1.8	$30+5\times 10^4$

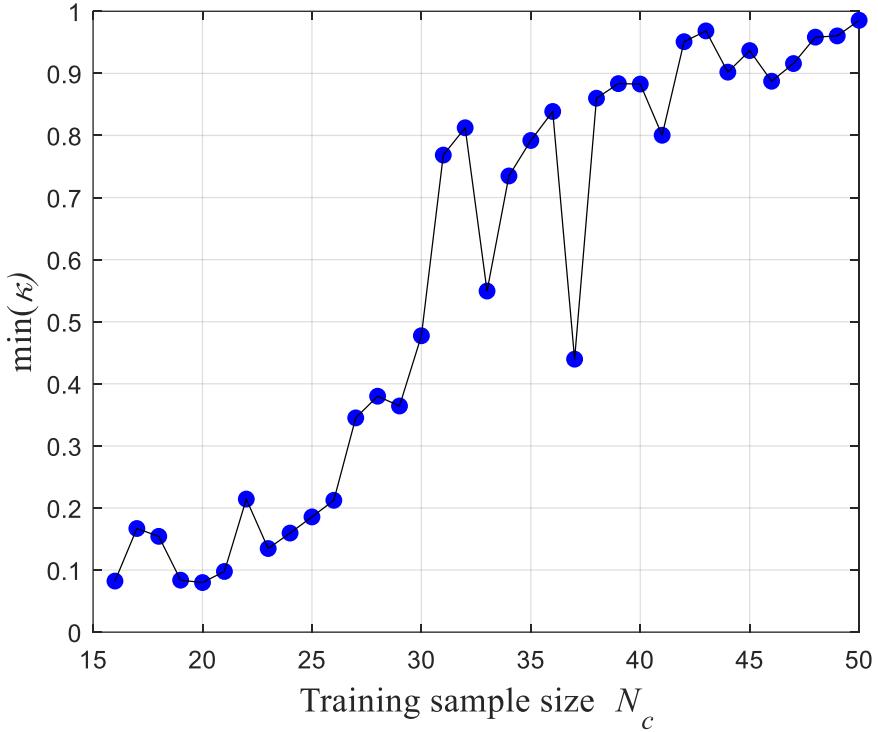


Figure 11 Plots of the minimum value of the learning function at each iteration step for the seepage model

4.4 A two-dimensional wing flutter model

A two-dimensional wing flutter model adapted from Refs. [25] and [36] is introduced here. As shown in Figure 12, the mass of the wing is denoted by m , the point G denotes the center-of-mass of the wing, E is the location of stiffness center. Let h and α denote the vertical and rotational displacements, respectively. K_h and K_α are the stiffness of the vertical spring and the torsional spring both of which are fixed at the stiffness center. The chord length of the wing is $2b$, the variable a refers to the dimensionless distance between the midpoint of the chord and the stiffness center, and the variable x_α refers to the dimensionless distance between the center-of-mass G and the stiffness center E . The phugoid mode frequency of the wing is $\omega_h = \sqrt{K_h/m}$, the pitching mode frequency is $\omega_\alpha = \sqrt{K_\alpha/m}$, the radius of the rotation of the wing towards G is expressed as r_α . The equation governing the vibration of the two-dimensional wing is derived as:

$$\begin{cases} \frac{d^2}{dt^2}\left(\frac{h}{b}\right) + x_\alpha \frac{d^2}{dt^2}\alpha + \omega_h^2 \frac{h}{b} = \frac{Q_h}{mb} \\ x_\alpha \frac{d^2}{dt^2}\left(\frac{h}{b}\right) + r_\alpha^2 \frac{d^2}{dt^2}\alpha + r_\alpha^2 \omega_\alpha^2 \alpha = \frac{Q_\alpha}{mb^2}. \end{cases} \quad (17)$$

Let $\mathbf{q} = (h/b, \alpha)^T$ express the general displacement, and $d\tau = \omega_\alpha dt$ denotes the dimensionless

time, then the above governing equation can be rewritten as

$$\mathbf{M} \frac{d^2 \mathbf{q}}{d\tau^2} + \mathbf{K} \mathbf{q} = \mathbf{F}$$

, where

$$\mathbf{M} = \begin{bmatrix} 1 & x_\alpha \\ x_\alpha & r_\alpha^2 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} (\omega_h/\omega_\alpha) & 0 \\ 0 & r_\alpha^2 \end{bmatrix} \quad (18)$$

, and \mathbf{F} is the generalized aerodynamic force expressed as

$$\mathbf{F} = \frac{V^2}{\omega_\alpha^2 b^2} \frac{\pi \rho b^2}{m} \frac{1}{\pi} \begin{pmatrix} C_L \\ 2C_{M_E} \end{pmatrix} = \frac{1}{\pi} V_f^{*2} \begin{pmatrix} C_L \\ 2C_{M_E} \end{pmatrix} \quad (19)$$

, C_L and C_{M_E} are the aerodynamic force coefficient of the wing and aerodynamic moment coefficients towards the stiffness center E , respectively. Assume that the mass ratio is $\mu = m/\pi \rho b^2$, then $V_f^* = V/(\omega_\alpha b \sqrt{\mu})$ expresses the dimensionless flutter critical speed. Theo Dawson unsteady aerodynamic model is used to derive the aerodynamic force of the wing, and then the above flutter model is solved with V-g method, one can find more details about V-g method in subsection 3.7 of Ref.[36].

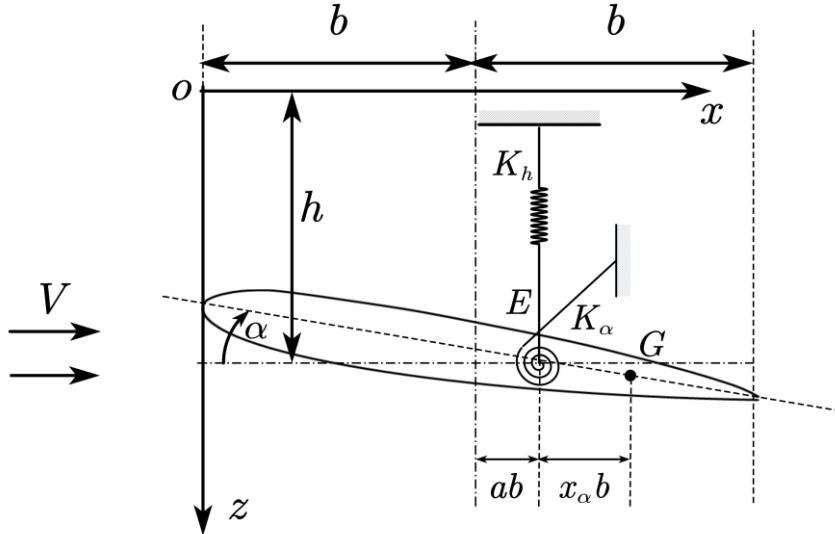


Figure 12 A two-dimensional wing flutter model

The flutter will happen if the critical speed V_f^* is smaller than the threshold 0.4414, thus the performance function is defined as $g = V_f^* - 0.4414$. The six inputs variables, i.e. μ , r_α , ω_h , a , ω_α and x_α , are assumed to follow truncated Gaussian distribution with distribution parameters listed in Table 6 and truncated support $[m_i - 5\sigma_i, m_i + 5\sigma_i]$, where m_i and σ_i are the mean and standard deviation of each input respectively.

Table 6 Distribution parameters of the input variables in the wing flutter model

Variables	Description	Mean	COV
μ	Mass ratio	20	0.0425
r_α	Dimensionless radius of rotation	0.5	0.0425
ω_h	Phugoid mode frequency	30	0.0255
a	Dimensionless distance between midpoint of the chord and E	-0.4	0.0255
ω_α	Pitching mode frequency	50	0.0255
x_α	Dimensionless distance between E and G	0.2	0.0255

The MPP and important direction are first calculated by the FORM method, and the total number of function calls is 18. Then the AGPR-LS is implemented with six initial lines (thus 24 initial training samples) by setting $\epsilon = 0.0005$ and $\kappa^* = 0.985$, and the results are reported in Table 7, with the training process being schematically illustrated by the evolution of learning function values shown in Figure 13. The reference solutions computed by AK-MCMC, LS, and IS are also reported in Table 7 for comparison, where the LS algorithm is implemented by setting the line size as 10, 100, and 200 respectively, and for each line, five points are utilized for calculating the intersection point. As can be seen, both AGPR-LS and AK-MCMC algorithms produce satisfactory results, but still, the AGPR-LS algorithm is much more efficient than AK-MCMC, as revealed by the total number of g -function calls. It is also shown in Table 7 that, the AGPR-LS with totally 120 g -function calls produces the estimate with the same level of accuracy as the classical LS with 200 lines (thus $18+10^3$ g -function calls), indicating the superiority of the AGPR-LS algorithm to the classical LS algorithm.

Table 7 Reliability analysis results of the two-dimensional wing flutter model

Methods	Parameter Settings	N_{lines}	$p_f (\times 10^{-4})$	COV (%)	N_c
AGPR-LS	$\epsilon = 0.0005$, $\kappa^* = 0.985$	200	9.332	2.7	$18+102=120$
AK-MCMC	$N = 3e4$, $N_0 = 12$, $p_0 = 0.01$	—	9.409	6.0	346
LS	—	10	8.390	18.8	$18+50=68$
		100	10.552	7.1	$18+500=518$
		200	9.493	3.6	$18+10^3=1018$
IS	—	—	9.298	2.1	$18+10^4$

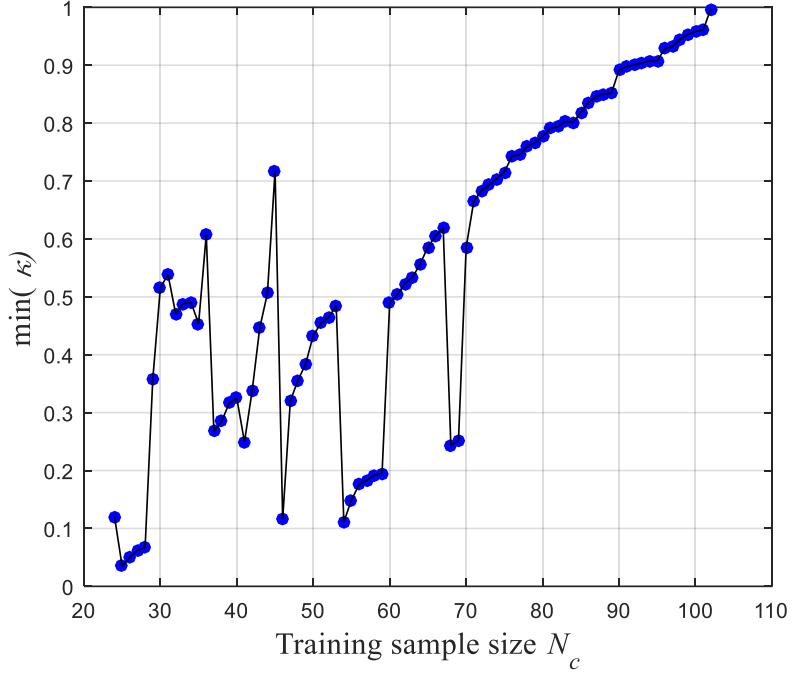


Figure 13 Plot of the minimum value of the learning function with respect to the training step for the wing flutter model

4.5 Transmission tower

For demonstrating the performance of the AGPR-LS algorithm for high-dimensional problems, we consider an electricity transmission tower structure shown in Figure 14, which is adapted from Refs. [37] and [38]. The finite element model is established with Matlab. This structure consists of 80 bars, all of which behave within the linear elastic range. Four static loads are applied in the top nodes. All these four loads are assumed to be deterministic with magnitude $F=200$ [kN], and they are all applied in the direction $[\sin(\pi/3), \cos(\pi/3), 0]$. There are twenty corner bars whose cross-section areas (A_1^c, \dots, A_{20}^c) and Young's modulus (E_1^c, \dots, E_{20}^c) are assumed to be random input variables, and for the rest 60 bars, both the cross-section areas Young's modulus are assumed to be deterministic with magnitudes 4.35×10^{-3} [m^2] and 2.1×10^{11} [Pa] respectively. For the twenty corner bars, both A_i^c and E_i^c follow lognormal distribution with mean values being 7.45×10^{-3} [m^2] and 2.1×10^{11} [Pa] respectively. The COVs of all these 40 input random variables are assumed to be 0.1. The failure event is defined as the displacement of node A at the top of the tower exceeding 0.072 [m].

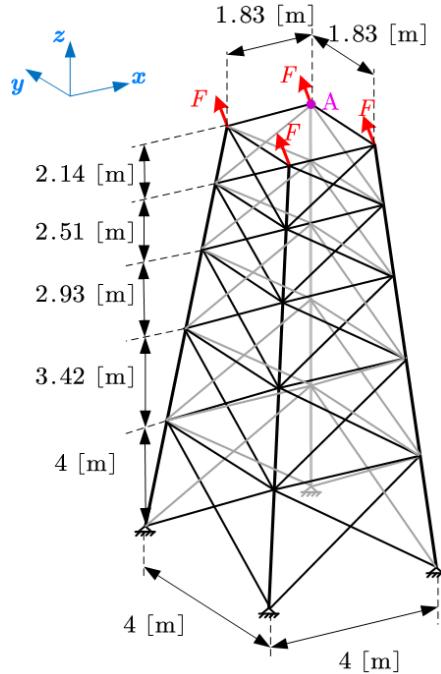


Figure 14 A transmission tower structure

All the 40 lognormal random variables are first transformed into independent standard Gaussian variables by using the isoprobabilistic transformation, and then the MPP is calculated in the standard Gaussian space by using FORM, and 18 g-function calls are consumed. This MPP is then utilized for implementing the simulation. The LS and IS algorithms are implemented for providing reference solutions, as shown in Table 8. One notes that with the IS algorithm, only when the sample size being very large (e.g., 2×10^5), the COV of the estimate is less than 5%. The LS algorithm is implemented by setting the line size as 10, 100, and 200 respectively, and for each line, five points are utilized for calculating the intersection points with spline interpolation. It is shown that the accuracy of the result with 10 lines is not acceptable due to the large COV. When 200 lines are used, the COV of the estimate is less than 5%, and the result can be served as a reference solution.

The AGPR-LS algorithm is then implemented with three initial lines, and for each line, three points are used for calculating the intersection points, thus the initial training sample size is 12. The results are then reported in Table 8. As can be seen, the AGPR-LS algorithm consumes totally 231 g-function calls to produce the estimate of the same level of accuracy with the classical LS algorithm with 200 lines (18+1000 g-function calls), indicating that even for this high-dimensional problem, the AGPR-LS algorithm outperforms the classical LS algorithm.

An interesting phenomenon appears in the implementation of the AGPR-LS algorithm for this high-dimensional problem. During the training process, especially in the first several dozens

of iterations, it happens that for some lines, the intersection points defined by $\mu_g(\mathbf{z}^{(s)\perp} + z^{\parallel} \mathbf{e}_\alpha) = 0$ (see step 3 in subsection 3.3) do not exist. For this case, we set the corresponding $c^{(s)}$ values as the average value of c across other lines computed in the previous iteration to improve the robustness of the algorithm. Interestingly, this phenomenon rarely happens in low-dimensional problems. The reason behind it is that, with the increment of the input dimension, the distance between lines tend to be larger, indicating weaker correlation strength between lines. For the lines which are far from the training data, the GPR prediction errors can be large, making it sometimes intractable to solve the univariate equation $\mu_g(\mathbf{z}^{(s)\perp} + z^{\parallel} \mathbf{e}_\alpha) = 0$. This is also why we need more training samples, and thus g -function calls, for this high-dimensional problem than that for the several previous low-dimensional problems. However, as indicated, the AGPR-LS algorithm is still much more efficient than the classical LS algorithm if the target is to generate estimates with the same level of COV.

Table 8 Reliability results of the transmission tower

Methods	Parameter Settings	N_{lines}	$p_f (\times 10^{-9})$	COV (%)	N_c
AGPR-LS	$\epsilon = 0.1$, $\kappa^* = 0.980$	200	5.297	4.5	$18+213=231$
LS	—	10	3.995	25.9	$18+50=68$
		100	5.576	8.0	$18+500=518$
		200	5.009	4.9	$18+10^3=1018$
IS	—	—	5.458	4.5	$18+2\times 10^5$

4.6 Final remarks

With the above five test examples, we have shown the high performance of the AGRP-LS algorithm. The results have proved that, with the introduction of the adaptive learning procedure, the AGPR-LS algorithm has the potential to outperform classical LS algorithm for problems with extremely rare failure events, nonlinear performance function, and high-dimensional inputs. The reason behind this improvement is that the AGPR-LS algorithm, on the one hand, takes full advantage of the high efficiency of the one-dimensional line search of the classical LS algorithm and, on the other hand, makes the best use of the spatial correlation information among lines and training samples to improve the speed and accuracy of calculating the intersection point for each line.

One notes that there are also other improved LS schemes being developed, and one of the most related developments is the metamodel LS (MLS) developed in Ref. [29], which improves the classical LS by combining it with the GPR without adaptive learning. We make a simple comparison of AGPR-LS with the MLS by using the second test example (a parallel system) of

Ref. [29] and their results (Table 6 of Ref. [29]). The performance function is highly nonlinear. It is reported in that paper that, the results with MLS and LS are 2.42×10^{-4} (with COV being 3.52%) and 2.45×10^{-4} (with COV being 4.00%) respectively, and the corresponding total numbers of g -function calls are 762 and 2905 respectively. We then implement AGPR-LS algorithm to achieve the same level of estimation accuracy, and the mean estimate and the corresponding COV are 2.42×10^{-4} and 3.58% respectively, while the total number of g -function calls is only 42, indicating that for this highly nonlinear problem, the AGPR-LS algorithm is much more efficient than both the MLS and LS algorithms. This high efficiency benefits from the adaptive learning scheme. The combination of the GPR model, the active learning scheme, and LS has largely improved the efficiency and robustness of the LS algorithm for different types of problems.

5 Conclusions and discussions

The LS algorithm is one of the most competitive stochastic simulation algorithms for small failure probability estimation. However, it is mostly applied to problems with moderately nonlinear performance functions, and the correct identification of the important direction is extremely important for the efficient implementation of the algorithm. The reason is that, for highly nonlinear performance function, many more lines are required for accurately estimating the failure probability. Besides, for highly non-linear performance functions, more g -function calls are required for accurately calculating the intersection point for each line. All the above elements may lead to a considerable increment of g -function calls. However, compared with the other stochastic simulation algorithms such as SS, the LS can be especially efficient due to the high searching efficiency along lines, each of which is equivalent to solving a one-dimensional nonlinear equation.

The AGPR-LS algorithms developed in this paper has tackled the above disadvantages, but keeping the high efficiency of one-dimensional searching. The devised learning function $\kappa(\mathbf{x})$ is proven to be especially effective for improving the accuracy of calculating the intersection point for each line, and the induced AGPR-LS algorithm is shown to be extremely efficient for extremely small failure probability estimation, and also less sensitive to the specified important directions and the nonlinearity of performance function as more lines can be added without largely increasing the number of performance function evaluations. Compared with the other active learning algorithms such as AK-MCMC, due to the high efficiency of one-dimensional search, the AGPR-LS algorithm is more efficient especially for rare events since the line search allows approaching the failure surface very easily. Besides, the introduction of a small line pool in the AGPR-LS algorithm, instead of the large sample pool as used in the AK-MCS and

advanced AK-MCS methods, makes it even more efficient for numerical implementation. However, for problems with multiple important directions and/or failure modes and/or failure domains, the proposed algorithm is still less effective, and needs to be improved in future work.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (NSFC 51905430) and ANID (Agency for Research and Development, Chile) under its program FONDECYT, grant number 1180271. The first author is supported by the program of China Scholarships Council (CSC). The second and third authors are both supported by the Alexander von Humboldt Foundation of Germany. The second author is also supported by the Top International University Visiting Program for Outstanding Young Scholars of Northwestern Polytechnical University. We would also like to thank our colleague Dr. Matteo Broggi for his strong support on COSSAN.

References

- [1] Zio E. "Reliability engineering: Old problems and new challenges." *Reliability Engineering & System Safety* 94.2 (2009): 125-141.
- [2] Hasofer AM. "An Exact and Invariant First Order Reliability Format." *Journal of Engineering Mechanics, Proc. ASCE* 100.1 (1974): 111-121.
- [3] Der Kiureghian A, Lin H Z, Hwang SJ. "Second-order reliability approximations." *Journal of Engineering mechanics* 113.8 (1987): 1208-1225.
- [4] Li J, Chen J, Sun W, et al. "Advances of the probability density evolution method for nonlinear stochastic systems." *Probabilistic Engineering Mechanics* 28 (2012): 132-142.
- [5] Chen G, Yang D. "Direct probability integral method for stochastic response analysis of static and dynamic structural systems." *Computer Methods in Applied Mechanics and Engineering* 357 (2019): 112612.
- [6] Au SK, Beck JL. "A new adaptive importance sampling scheme for reliability calculations." *Structural safety* 21.2 (1999): 135-158.
- [7] Beaurepaire P, Jensen H A, Schuëller G I, et al. "Reliability-based optimization using bridge importance sampling." *Probabilistic Engineering Mechanics* 34 (2013): 48-57.
- [8] Au SK, Beck JL. "Estimation of small failure probabilities in high dimensions by subset simulation." *Probabilistic engineering mechanics* 16.4 (2001): 263-277.
- [9] Papaioannou I, Betz W, Zwirglmaier K, et al. "MCMC algorithms for subset simulation." *Probabilistic Engineering Mechanics* 41 (2015): 89-103.

- [10] Schuëller GI, Pradlwarter HJ, Koutsourelakis PS. "A critical appraisal of reliability estimation procedures for high dimensions." *Probabilistic engineering mechanics* 19.4 (2004): 463-474.
- [11] de Angelis M, Patelli E, Beer M. "Advanced line sampling for efficient robust reliability analysis." *Structural safety* 52 (2015): 170-182.
- [12] Melchers RE. "Structural system reliability assessment using directional simulation." *Structural Safety* 16.1-2 (1994): 23-37.
- [13] Wang Z, Broccardo M, Song J. "Hamiltonian Monte Carlo methods for Subset Simulation in reliability analysis." *Structural Safety* 76 (2019): 51-67.
- [14] Echard B, Gayton N, Lemaire M. "AK-MCS: an active learning reliability method combining Kriging and Monte Carlo simulation." *Structural Safety* 33.2 (2011): 145-154.
- [15] Bichon BJ, Eldred MS, Swiler LP, et al. "Efficient global reliability analysis for nonlinear implicit performance functions." *AIAA Journal* 46.10 (2008): 2459-2468.
- [16] Lv Z, Lu Z, Wang P. "A new learning function for Kriging and its applications to solve reliability problems in engineering." *Computers & Mathematics with Applications* 70.5 (2015): 1182-1197.
- [17] Sun Z, Wang J, Li R, et al. "LIF: A new Kriging based learning function and its application to structural reliability analysis." *Reliability Engineering & System Safety* 157 (2017): 152-165.
- [18] Echard B, Gayton N, Lemaire M, et al. "A combined importance sampling and kriging reliability method for small failure probabilities with time-demanding numerical models." *Reliability Engineering & System Safety* 111 (2013): 232-240.
- [19] Dubourg V, Sudret B, Deheeger F. "Metamodel-based importance sampling for structural reliability analysis." *Probabilistic Engineering Mechanics* 33 (2013): 47-57.
- [20] Cadini F, Santos F, Zio E. "An improved adaptive kriging-based importance technique for sampling multiple failure regions of low probability." *Reliability Engineering & System Safety* 131 (2014): 109-117.
- [21] Yun W, Lu Z, Jiang X. "An efficient reliability analysis method combining adaptive Kriging and modified importance sampling for small failure probability." *Structural and Multidisciplinary Optimization* 58.4 (2018): 1383-1393.
- [22] Huang X, Chen J, Zhu H. "Assessing small failure probabilities by AK-SS: an active learning method combining Kriging and subset simulation." *Structural Safety* 59 (2016): 86-95.
- [23] Zhang J, Xiao M, Gao L. "An active learning reliability method combining kriging

- constructed with exploration and exploitation of failure region and subset simulation." Reliability Engineering & System Safety 188 (2019): 90-102.
- [24] Wei P, Tang C, Yang Y. "Structural reliability and reliability sensitivity analysis of extremely rare failure events by combining sampling and surrogate model methods." Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability (2019), 233(6): 943-957
 - [25] Wei P, Song J, Bi S, et al. "Non-intrusive stochastic analysis with parameterized imprecise probability models: II. Reliability and rare events analysis." Mechanical Systems and Signal Processing 126 (2019): 227-247.
 - [26] Lelièvre N, Beaurepaire P, Matstrand C, et al. "AK-MCSi: a Kriging-based method to deal with small failure probabilities and time-consuming models." Structural Safety 73 (2018): 1-11.
 - [27] Fauriat W, Gayton N. "AK-SYS: an adaptation of the AK-MCS method for system reliability." Reliability Engineering & System Safety 123 (2014): 137-144.
 - [28] Wei PF, Liu FC, Tang CH. "Reliability and reliability-based importance analysis of structural systems using multiple response Gaussian process model." Reliability Engineering & System Safety 175 (2018): 183-195.
 - [29] Depina I, Le TMH, Fenton G, et al. "Reliability analysis with metamodel line sampling." Structural Safety 60 (2016): 1-15.
 - [30] Zhang X, Lu Z, Yun W, et al "Line sampling-based local and global reliability sensitivity analysis." Structural and Multidisciplinary Optimization (2019): 1-15.
 - [31] Lebrun R, Dutfoy A. "Do Rosenblatt and Nataf isoprobabilistic transformations really differ?." Probabilistic Engineering Mechanics 24.4 (2009): 577-584.
 - [32] Rasmussen CE, Williams CKI. "Gaussian processes for machine learning". Cambridge: The MIT Press, 2006.
 - [33] Pradlwarter HJ, Pellissetti MF, Schenk CA, et al. "Realistic and efficient reliability estimation for aerospace structures." Computer Methods in Applied Mechanics and Engineering 194.12-16 (2005): 1597-1617.
 - [34] Patelli E. "COSSAN: A Multidisciplinary Software Suite for Uncertainty Quantification and Risk Management", in Handbook of Uncertainty Quantification Ghanem, R.; Higdon, D. & Owhadi, H. (Eds.) Springer International Publishing, 2016, 1-69
 - [35] Valdebenito MA, Jensen HA, Hernandez H B, et al. "Sensitivity estimation of failure probability applying line sampling." Reliability Engineering & System Safety, 2018, 171: 99-111.

- [36] Ye ZY, Zhang WW, Shi AM. "Fundamentals of fluid-structure coupling and its application.", Harbin, China: Harbin Institute of Technology Press, 2010.
- [37] Haukaas T, Der Kiureghian A. "Strategies for finding the design point in non-linear finite element reliability analysis." Probabilistic Engineering Mechanics 21.2 (2006): 133-147.
- [38] Song J, Valdebenito M, Wei P, et al. "Non-intrusive imprecise stochastic simulation by line sampling." Structural Safety 84 (2020): 101936.