Bounds optimization of model response moments: a twin-engine Bayesian active learning method

Pengfei Wei · Fangqi Hong · Kok-Kwang Phoon · Michael Beer

Received: date / Accepted: date

Abstract The efficient propagation of imprecise probabilities through expensive simulators has emerged to be one of the great challenges for mixed uncertainty quantification in computational mechanics. An active learning method, named Collaborative and Adaptive Bayesian Optimization (CABO), is developed for tackling this challenge by combining Bayesian Probabilistic Optimization and Bayesian Probabilistic Integration. Two learning functions are introduced as engines for CABO, where one is introduced for realizing the adaptive optimization search in the epistemic uncertainty space, and the other one is developed for adaptive integration in the aleatory uncertainty space. These two engines work in a collaborative way to create optimal design points adaptively in the joint uncertainty space, by which a Gaussian process regression model is trained and updated to approach the bounds of model response moments with pre-specified error tolerances. The effectiveness of CABO is demonstrated using a numerical example and two engineering benchmarks.

Keywords Uncertainty quantification · Imprecise probabilities · Bayesian inference · Adaptive optimization · Gaussian process regression · Active learning

Pengfei Wei (Corresponding author)
School of Mechanics, Civil Engineering and Architecture, Northwestern Polytechnical University, Xi'an 710072, China.
Institute for Risk and Reliability, Leibniz Universität Hannover, Hannover 30167, Germany
E-mail: pengfeiwei@nwpu.edu.cn

Fangqi Hong
School of Mechanics, Civil Engineering and Architecture, Northwestern Polytechnical University, Xi'an 710072, China.
E-mail: fangqihong@mail.nwpu.edu.cn

Kok-Kwang Phoon
Department of Civil and Environmental Engineering, National University of Singapore, Singapore.
E-mail: kkphoon@nus.edu.sg

Michael Beer
Institute for Risk and Reliability, Leibniz Universität Hannover, Hannover 30167, Germany
Institute for Risk and Uncertainty, University of Liverpool, Liverpool L69 7ZF, UK
International Joint Research Center for Engineering Reliability and Stochastic Mechanics, Tongji University, Shanghai 200092, China
E-mail: beer@irz.uni-hannover.de

1 Introduction

Uncertainty quantification (UQ) is the process of quantitatively characterizing the different sources of uncertainties presented in computer simulations and real-world engineering applications, and the purpose is to produce reliable estimations of the output quantities of interest such as the risk and reliability of complex systems, and this way to make better data-informed decisions. In computational mechanics and related areas, the UQ of model responses has been of special interest due to the extensive uncertainties presented in structural parameters, initial/boundary conditions and environmental excitation (see e.g., [1,2]). Although the data sources can be of diverse variety, uncertainty can be generally classified into two categories, i.e., aleatory uncertainty and epistemic uncertainty [3],where the former one is also known as irreducible uncertainty, and caused by the intrinsic random natures of events or parameters, and the latter one is due to the lack of information, and can be generally reduced or eliminated by collecting more information. The aleatory uncertainty results in the intrinsic random nature of system responses, and thus the randomness of system failures, while the epistemic uncertainty prevents the analysts from learning the probability distribution of responses or the probabilities of failure events. Therefore, these two types of uncertainties need to be properly and separately characterized by mathematical models for decision making in the face of uncertainty [3]. There are several different sub-tasks for UQ such as uncertainty characterization [4,5], forward uncertainty propagation [6], backward model updating [7,8], uncertainty sensitivity analysis [9], etc., where the uncertainty characterization and propagation are the two most fundamental problems.

Uncertainty characterization aims at developing mathematical models for quantitively describing the uncertainties and statistical inference techniques for estimating the parameters of these models. It has been widely realized that the classical precise probability model is not sufficiently general to accommodate input variables, of which the information is imperfect (scarce,incomplete, and \ or imprecise) [3,4]. The non-probabilistic models [5], e.g., interval models and fuzzy set models, have been developed and widely investigated in the area of structural reliability [10,11,12,13]. However, it is also realized that these models are more suitable for modeling the epistemic uncertainty of the constant-but-unknown variables [14], due to their inability of separating the aleatory and epistemic uncertainties. For parameters or events where both kinds of uncertainties are present, the imprecise probabilities [4], such as probability box (p-box) [15], evidence theory [16], fuzzy probability [17], and second-order probability [18], as natural extensions of the classical probability theory, have received the most attention due to the capability of separating the two kinds of uncertainties with unified model frameworks. One can refer to Refs. [4,19,20] for comprehensive review and comparison of these models. Despite their appealing feature of separating the aleatory and epistemic uncertainty, the imprecise probabilities are still less widely applied than the classical precise probability model in real-world applications, mainly due to the demanding computational cost of uncertainty propagation. Thus, developing efficient numerical algorithms for propagating the imprecise probability models, with competitive efficiency as those for precise probability models, through computationally expensive simulators has been a research frontier of UQ, and received great attention. This paper concerns the propagation of the p-box model.

The tasks for uncertainty propagation of imprecise probabilities involve the estimation of model response moments, the probability of (rare) events, the density function of model response, etc., all of which are non-deterministic due to the imprecision presented in the probability distribution of model inputs. This topic has received extensive attention among the past decade. The most straightforward way for addressing this challenge commonly involves a double-loop procedure due to the natural two-layer model structure, and two different strategies have been developed following this scheme. The first strategy performs the outer-loop optimization in the space of epistemic distribution parameters, and then for each design point, calculates the responses of interest (e.g., failure probability and model response moments) using classical probabilistic analysis methods such as stochastic simulation [21,22]. Another strategy, termed as Interval Monte Carlo Simulation (IMCS), performs outer-loop random sampling to create interval samples, and then for each interval sample, computes the bound values of model responses in the inner loop by using either intrusive method (e.g., interval finite element analysis) or non-intrusive method (e.g., particle swarm optimization) [23,24,25,26]. Both of the above strategies have been applied to the 2014 NASA Langley UQ challenge problem [22,27]. These methods are less applicable for problems with computationally expensive simulators as the double-loop numerical procedures usually require an extensive number of model function calls. For alleviating the computational cost, surrogate model methods have also been developed [28,29,30,31], but aspects of optimal experiment design and the numerical errors associated with these surrogate models require further investigation.

For alleviating the computational burden caused by the double-loop schemes, some decoupling strategies have been developed. For example, the Extended Monte Carlo simulation (EMCS) is developed based on the weighting scheme, and shown to be efficient for low-dimensional imprecise probabilities inputs [32]. A similar algorithm based on the weighting scheme has also been developed in Ref. [33], with a special focus on deriving the optimal density for sampling. The operator norm theory was developed in Ref.[34] for efficiently estimating the first excursion probability of a linear structure subjected to imprecise stochastic load, but this method, in its current form, is not capable of incorporating the mixed uncertainties presented in structural parameters such as Young's modulus. Recently, a new methodology framework, named Non-intrusive Imprecise Stochastic Simulation (NISS), has been developed for efficiently propagating the imprecise probability models and the non-probabilistic models simultaneously [6,35]. The most appealing feature of this framework is that only one stochastic simulation is required for performing the analysis and the numerical errors are properly addressed. Till now, subset simulation and line sampling have both been adopted in this framework for analyzing the probabilities of rare events with p-

box inputs [36,37,38]. In this framework, the output of interests are the High-Dimensional Model Representation (HDMR) components of the probabilistic response functions, which provide a quantitative and visible representation of the relationship between the probabilistic responses (e.g., the model response moments) and the imprecise distribution parameters of the p-box inputs. In this work, the outputs of interest are the bounds of the model response moments.

The Bayesian numerical analysis [39], including the Bayesian Probabilistic Optimization (BPO, also known as Efficient Global Optimization, EGO) [40], the Bayesian Probabilistic Integration (BPI) [41,42], Bayesian PDE solution [43], etc., have emerged to be a frontier of scientific computation. The most attractive character is that the discretization error is regarded as a source of epistemic uncertainty, and is analytically or semi-analytically quantified by the variation of the posterior distribution. In our previous work, an improved NISS algorithm, named as Non-intrusive Imprecise Probabilistic Integration (NIPI), has been developed for learning the functional behavior of the model response moments with respect to the input epistemic parameters [44]. This method is strongly recommended when the analysts' target is to learn the details of the functional behavior of the probabilistic responses (e.g., model response moments) to the epistemic parameters visibly while the epistemic uncertainty is large. However, in some applications, the analysts may only concern the bounds of the probabilistic responses, regardless of the functional behavior. In this case, the required number of model function calls can be further reduced, and this is the main object of this work.

In this work, the BPI and BPO will be combined to develop a new Bayesian numerical method, called Collaborative and Adaptive Bayesian Optimization (CABO), for efficiently estimating the bounds of model response moments subjected to distributional p-box inputs. The method jointly performs adaptive BPO in the epistemic uncertainty space and adaptive BPI in the aleatory uncertainty space. Two learning functions, originally developed for solving the optimization and integration of deterministic functions respectively, are introduced as twin engines, and work in a collaborative way to adaptively produce optimal design points in the joint uncertainty space. A Gaussian Process Regression (GPR) model is then trained and updated in the joint uncertainty space, based on which, the bounds of model response moments are analytically derived using Bayesian inference with the discretization errors summarized by posterior variances. The developed CABO algorithm owns several appealing features, which make it efficient and robust for estimating the above-mentioned bounds. First, the adaptive BPO scheme in the epistemic space properly avoids waste of model function calls in the areas that are not informative for determining the bounds. Second, the global convergence of optimizing the bounds in the epistemic space is ensured by the learning function. Third, the spatial correlation information revealed by the GPR model is fully utilized for improving the accuracy of integration in the aleatory space. Forth, the discretization errors for estimating the bounds are explicitly summarized by using the posterior variance.

The remaining of this paper is organized as follows. Section 2 describes the mathematical formulation of the problem to be solved, followed by a brief review of the NIPI algorithm for estimating the probabilistic response functions in section 3. Section 4 presents the new developments, including the two learning functions and the CABO algorithm. In section 5, a numerical example and two engineering benchmarks are introduced for demonstrating the effectiveness of the CABO algorithm. Section 6 presents the conclusions.

2 Problem statement

Let $y = g(\boldsymbol{x})$ (abbreviated as g-function) denote the deterministic computer simulator of interest, where $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ indicates the n-dimensional random input vector, and y refers to the univariate model output. By saying "deterministic" we mean that, given a deterministic value of the input vector, the model output has a unique and deterministic value. We also assume that the simulator of concern is expensive to run. Based on the above setting, the probability distribution of the model output is uniquely determined by that of the input vector. However, in practical applications, the probability distribution of $\boldsymbol{x}$, which reflects the aleatory uncertainty of $\boldsymbol{x}$, cannot be precisely determined due to the lack of information. In this case, the distribution parameters can be imprecisely estimated as, e.g., confidence intervals, which characterize the epistemic uncertainty underlying the probability distribution of $\boldsymbol{x}$. We assume throughout this paper that the mixed uncertainty of $\boldsymbol{x}$ is characterized by distributional imprecise probability models, where the word "distributional" means that the distribution type is exactly known, but the values of the distributional parameters can not be precisely known. Let $f(\boldsymbol{x}|\boldsymbol{\theta})$ denote the joint probability density function (PDF), where $\boldsymbol{\theta} = (\theta_1, \theta_2, \cdots, \theta_d)$ is the d-dimensional non-deterministic distribution parameters, and its (epistemic) uncertainty is characterized by the hyper-rectangular $[\boldsymbol{\theta}_L, \boldsymbol{\theta}_U]$, where $\boldsymbol{\theta}_L = (\theta_{L1}, \theta_{L2}, \cdots, \theta_{Ld})$ is the lower bound vector, and $\boldsymbol{\theta}_U = (\theta_{U1}, \theta_{U2}, \cdots, \theta_{Ud})$ is the upper bound vector. One notes that the epistemic uncertainty on $\boldsymbol{\theta}$ can also be characterized by more complex models such as fuzzy sets [17] and subjective probability distribution [18], if the analysts expect to make a trade-off between the "risk" and "accuracy" for decision making. In the former case, the induced imprecise probability model is called fuzzy probability, while in the later case, it is called second-order probability model. In this paper, we only concern the p-box models.

With the above assumptions, any probabilistic characters of the model output y, e.g., the statistical moments and the cumulative distribution function (CDF), are functions of $\boldsymbol{\theta}$. In this paper, only the statistical moments of model output are concerned, and specifically, we take the response expectation function $m(\boldsymbol{\theta})$ as an example, which is formulated as:

$$m(\boldsymbol{\theta}) = \int_{\mathbb{R}^n} g(\boldsymbol{x}) f(\boldsymbol{x}|\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{x}. \tag{1}$$

In our previous papers, the estimation of the above moment function across the full support of $\boldsymbol{\theta}$ has been addressed by developing the NISS and NIPI frameworks [6,44]. However, in many applications, the analysts are more interested in the bounds $[m_L, m_U]$ of the output moments as they reflects how much the epistemic uncertainty underlies the response moments, and provide conservative estimations of system performances (e.g., the mean time to failure of a system). Intuitively, estimating the bounds of $m(\boldsymbol{\theta})$ can be computationally less expensive than estimating the function $m(\boldsymbol{\theta})$ across the whole support of $\boldsymbol{\theta}$, as for the former, the full information on the behavior of $m(\boldsymbol{\theta})$ may not be required. However, a special development is required to achieve this target, and this is the specific focus of this paper. The problem is then mathematically formulated as:

$$\begin{cases} m_L = \min\limits_{\boldsymbol{\theta} \in [\boldsymbol{\theta}_L, \boldsymbol{\theta}_U]} m(\boldsymbol{\theta}) \\ m_U = \max\limits_{\boldsymbol{\theta} \in [\boldsymbol{\theta}_L, \boldsymbol{\theta}_U]} m(\boldsymbol{\theta}) \end{cases}. \tag{2}$$

Commonly, once the moment function $m(\boldsymbol{\theta})$ is estimated, the calculation of the bounds $[m_L, m_U]$ can be quite straightforward. However, the accurate estimation of the function $m(\boldsymbol{\theta})$ across the full support $[\boldsymbol{\theta}_L, \boldsymbol{\theta}_U]$ can be computationally much more expensive than estimating the bounds $[m_L, m_U]$ since the former task requires samples of $\boldsymbol{\theta}$ distributed over the whole support $[\boldsymbol{\theta}_L, \boldsymbol{\theta}_U]$. Thus, to address this challenge, we develop a specific algorithm for efficiently estimating the bounds $[m_L, m_U]$ with high accuracy based on Bayesian numerical analysis. In the next section, we briefly review the NIPI method for estimating the moment function $m(\boldsymbol{\theta})$.

3 Non-intrusive Imprecise Probabilistic Integration

The NIPI method is originally developed for estimating the HDMR components of $m(\boldsymbol{\theta})$ in order to learn the functional behavior of $m(\boldsymbol{\theta})$ as well as the influence of each θ_i on the response moments visibly [44]. Here, we briefly review this method with the focus on the inference of the induced GPR model $\hat{m}(\boldsymbol{\theta})$ for approximating $m(\boldsymbol{\theta})$. In Ref. [44], the closed-form expressions for the posterior features of $m(\boldsymbol{\theta})$ are not reported, but they are important for understanding the core developments in this work. Thus, the following review will be focused on these closed-form expressions.

For NIPI, an auxiliary density $p(\boldsymbol{\theta})$ needs first to be attributed for $\boldsymbol{\theta}$ (see section 5 for details). After that, it is required to transform both $\boldsymbol{x}$ and $\boldsymbol{\theta}$ into standard normal variables. This can be realized by, e.g., Rosenblatt's transformation [45]. Denote the transformation of $\boldsymbol{\theta}$ as $\boldsymbol{\theta} = T(\boldsymbol{v})$ and the inverse transformation as $\boldsymbol{v} = T^{-1}(\boldsymbol{\theta})$, where $\boldsymbol{v}$ is the independent standard Gaussian vector of size d. Then, let $\boldsymbol{x} = S(\boldsymbol{u}|T(\boldsymbol{v}))$ denote the transformation for $\boldsymbol{x}$ conditional on $\boldsymbol{\theta} = T(\boldsymbol{v})$, and the corresponding inverse transformation is $\boldsymbol{u} = S^{-1}(\boldsymbol{x}|T(\boldsymbol{v}))$, where $\boldsymbol{u}$ is the independent standard Gaussian vector of size n. Then a g-function with arguments $\boldsymbol{w} = (\boldsymbol{u}, \boldsymbol{v})$ can be defined as:

$$\mathcal{G}(\boldsymbol{w}) = g(S(\boldsymbol{u}|T(\boldsymbol{v}))). \tag{3}$$

Let $\Pi_{1:n}(\cdot)$ indicate the n-dimensional integral to the first n input variables of the argument over independent Gaussian density weight, then the model output expectation to $\boldsymbol{v}$ can be formulated as:

$$\mathcal{M}(\boldsymbol{v}) = \Pi_{1:n}[\mathcal{G}(\boldsymbol{w})]. \tag{4}$$

Once $\mathcal{M}(\boldsymbol{v})$ has been learned, the model output expectation function to $\boldsymbol{\theta}$ is recovered as:

$$m(\boldsymbol{\theta}) = \mathcal{M}\left(T^{-1}(\boldsymbol{\theta})\right). \tag{5}$$

The basic idea of BPI is to first train a GPR model for the integrand, and then infer the resultant posterior probability distribution of the integral [41,42]. This method has two appealing features. First, the spatial correlation information which reflects the smoothness of the integrand is fully integrated for improving the accuracy of the integration. Second, the variation of the posterior distribution provides a rationale measure for the discretization errors. A Gaussian process model is assumed for $\mathcal{G}(\boldsymbol{w})$ as $\mathcal{GP}(b(\boldsymbol{w}), \kappa(\boldsymbol{w}, \boldsymbol{w}'))$, where $b(\boldsymbol{w})$ is the prior mean function, which can be assumed to be zero, constant or polynomial functions, and it reflects the prior information on the non-linearity of $\mathcal{G}(\boldsymbol{w})$, and $\kappa(\boldsymbol{w}, \boldsymbol{w}')$ is the prior covariance function (also called kernel function) which reflects the prior information on the spatial correlation of $\mathcal{G}(\boldsymbol{w})$ at any two points $\boldsymbol{w}$ and $\boldsymbol{w}'$. In this paper, the exponential squared kernel function is utilized, and it is formulated as [46,47]:

$$\kappa(\boldsymbol{w}, \boldsymbol{w}') = \sigma_0^2 \exp\left(-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{w}') \varSigma^{-1} (\boldsymbol{w}-\boldsymbol{w}')^\top\right), \tag{6}$$

where σ_0^2 is the hyper-parameter quantifying the magnitude of variance (prediction error) at each site, and $\varSigma = \text{diag}\left(\sigma_1^2, \ldots, \sigma_{n+d}^2\right)$ indicates the scale matrix, each component of which measures the strength of the correlation along the corresponding coordinate. With this kernel function, the posterior probability distribution of the integral over Gaussian density weight is still Gaussian, and closed-form expressions can be inferred for both posterior mean and variance for the integral, where the posterior variance summarizes the discretization errors. One notes that the other kernels such as Matérn kernel can also be used to provide more flexibility for approximating the integrand, however, in this case, the weight density should be changed to be uniform to ensure that both posterior mean and variance can be expressed in closed form. One can refer to Ref. [41] for more details. For real world applications, the best choice of kernel should be determined by the users based on their understanding on the behaviors of the physical systems under consideration. For the guidance of choosing proper kernel, one can refer to chapter 2 of Ref. [48].

Let $\mathcal{D} = (\mathcal{W}, \mathcal{Y})$ denote a set of labeled training data, where $\mathcal{W} = (\mathcal{U}, \mathcal{V})$ is the sample matrix of size $N \times (n+d)$, each row of which is a sample point of $\boldsymbol{w}$, $\mathcal{U}$ and $\mathcal{V}$ are the sample matrix of $\boldsymbol{u}$ and $\boldsymbol{v}$ respectively, and $\mathcal{Y}$ is the column vector of model output values, i.e., $\mathcal{Y} = \mathcal{G}(\mathcal{W})$. Based on the training data $\mathcal{D}$, the hyper-parameters involved in $b(\boldsymbol{w})$ and $\kappa(\boldsymbol{w}, \boldsymbol{w}')$ can be specified by using, e.g., the maximum likelihood method. Throughout this paper, the noise-free GPR model is utilized and the training is realized by using the function “fitrgp” in the Matlab “Statistics and Machine Learning Toolbox”.

With the g-function $\mathcal{G}(\boldsymbol{w})$ being characterized by a well-trained GPR model $\hat{\mathcal{G}}(\boldsymbol{w})$, the resulting posterior probability distribution of $\mathcal{M}(\boldsymbol{v})$ is also a GPR model, and the posterior mean and variance are formulated as [41,42]:

$$\begin{aligned} \mu_{\mathcal{M}}(\boldsymbol{v}) =& \mathbb{E}_{\mathcal{D}}\left[\hat{\mathcal{M}}(\boldsymbol{v})\right] \\ =& \varPi_{1:n}\left[b(\boldsymbol{w})\right] \\ &+ \varPi_{1:n}\left[\boldsymbol{\kappa}(\boldsymbol{w}, \mathcal{W})^\top\right] K^{-1}(\mathcal{Y} - \boldsymbol{b}(\mathcal{W})) \end{aligned} \tag{7}$$

, and

$$\begin{aligned} \sigma_{\mathcal{M}}^2(\boldsymbol{v}) =& \mathbb{V}_{\mathcal{D}}\left[\hat{\mathcal{M}}(\boldsymbol{v})\right] \\ =& \varPi_{1:n}\varPi'_{1:n}\left[\kappa(\boldsymbol{w}, (\boldsymbol{u}', \boldsymbol{v}))\right] \\ &- \varPi_{1:n}\left[\boldsymbol{\kappa}(\boldsymbol{w}, \mathcal{W})^\top\right] K^{-1} \varPi'_{1:n}\left[\boldsymbol{\kappa}((\boldsymbol{u}', \boldsymbol{v}), \mathcal{W})\right], \end{aligned} \tag{8}$$

respectively, where $\mathbb{E}_{\mathcal{D}}[\cdot]$ and $\mathbb{V}_{\mathcal{D}}[\cdot]$ refer to the posterior expectation and variance operators taken over the GPR model trained using $\mathcal{D}$, K indicates the covariance matrix of the training data and is computed as $K = \boldsymbol{\kappa}(\mathcal{W}, \mathcal{W})$, $\boldsymbol{\kappa}(\boldsymbol{w}, \mathcal{W})$ refers to the vector of covariance between any new point $\boldsymbol{w}$ and the training sample matrix $\mathcal{W}$, and $\varPi'_{1:n}[\cdot]$ indicates the integral with respect to $\boldsymbol{u}'$. In Eqs. (7) and (8), the closed-form expression of $\varPi_{1:n}[b(\boldsymbol{w})]$ is trivial, the integral $\varPi_{1:n}[\boldsymbol{\kappa}(\boldsymbol{w}, \mathcal{W})] = \varPi'_{1:n}[\boldsymbol{\kappa}((\boldsymbol{u}', \boldsymbol{v}), \mathcal{W})]$ is further derived as [42,49]:

$$\begin{aligned} \varPi_{1:n}\left[\boldsymbol{\kappa}(\boldsymbol{w}, \mathcal{W})\right] =& \sigma_0^2 \left|\varSigma_{\boldsymbol{u}}^{-1} + I\right|^{-1/2} \\ &\times \exp\left[\begin{array}{c} -\frac{1}{2}\text{vec}\left\{\text{diag}\left[\mathcal{U}(\varSigma_{\boldsymbol{u}} + I)^{-1}\mathcal{U}^\top\right]\right\} \\ -\frac{1}{2}(\boldsymbol{v} - \mathcal{V})\varSigma_{\boldsymbol{v}}^{-1}(\boldsymbol{v} - \mathcal{V})^\top \end{array}\right], \end{aligned} \tag{9}$$

where $\varSigma_{\boldsymbol{u}}$ indicates a $(n \times n)$-dimensional diagonal matrix with the elements being the squared lengths of $\boldsymbol{u}$, $\varSigma_{\boldsymbol{v}}$ is the $(d \times d)$-dimensional diagonal matrix consisting of the squared lengths of $\boldsymbol{v}$, and vec$\{$diag$[\cdot]\}$ refers to the vectorization operator performed on the diagonal elements of the argument. The term $\varPi_{1:n}\varPi'_{1:n}[\kappa(\boldsymbol{w}, (\boldsymbol{u}', \boldsymbol{v}))]$ in Eq. (8) indicates the integral of $\kappa(\boldsymbol{w}, (\boldsymbol{u}', \boldsymbol{v}))$ with respect to $\boldsymbol{u}$ and $\boldsymbol{u}'$ over standard Gaussian density weight, and its closed-form expression is formulated as [42,49]:

$$\varPi_{1:n}\varPi'_{1:n}\left[\kappa(\boldsymbol{w}, (\boldsymbol{u}', \boldsymbol{v}))\right] = \sigma_0^2 \left|2\varSigma_{\boldsymbol{u}}^{-1} + I\right|^{-1/2}. \tag{10}$$

From Eqs. (7) and (9), the posterior mean $\mu_{\mathcal{M}}(\boldsymbol{v})$ can be interpreted as the summation of the prior mean $\varPi_{1:n}[b(\boldsymbol{w})]$ and a linear combination of the kernels between $\boldsymbol{v}$ and the training data $\mathcal{V}$, where the second term can be regarded as a calibration (inferred from the training data) to the prior mean. Eq. (8) implies that the posterior variance $\sigma_{\mathcal{M}}^2(\boldsymbol{v})$ is generated by subtracting a positive term (the last term in Eq. (8)) from the prior variance $\Pi_{1:n}\Pi'_{1:n}\left[\kappa\left(\boldsymbol{w},\left(\boldsymbol{u}',\boldsymbol{v}\right)\right)\right]$, where the positive term being subtracted reflects the amount of (epistemic) uncertainty reduction on $\mathcal{M}\left(\boldsymbol{v}\right)$ that learned from the training data.

In Eqs. (7) and (8), by replacing $\boldsymbol{v}$ with $\boldsymbol{v}=T^{-1}\left(\boldsymbol{\theta}\right)$ (see Eq. (5)), we can generate the posterior mean $\mu_m\left(\boldsymbol{\theta}\right)$ and the posterior variance $\sigma_m^2\left(\boldsymbol{\theta}\right)$ of the GPR model $\hat{m}\left(\boldsymbol{\theta}\right)$, which is an approximation of the response expectation function $m\left(\boldsymbol{\theta}\right)$. Then it is straightforward to estimate the bounds of $m\left(\boldsymbol{\theta}\right)$ with, e.g., particle swarm optimization, by setting the posterior mean $\mu_m\left(\boldsymbol{\theta}\right)$ as the objective function. However, due to the epistemic uncertainty (measured by the posterior variance) presented in $\hat{m}\left(\boldsymbol{\theta}\right)$, the estimated bounds may not be reliable. One can increase the reliability of the estimates by increasing the training data size. However, this indiscriminate way of increasing data size without considering the epistemic uncertainty presented in $\hat{m}\left(\boldsymbol{\theta}\right)$ and the behavior of $m\left(\boldsymbol{\theta}\right)$ may result in a waste of computational resources since each simulator run can be expensive. For avoiding this, we develop the CABO algorithm in the next section for adaptively learning the bounds of the probabilistic response moments with high confidence. For simplicity, the response moment function is exemplified throughout the paper, and for higher-order moments, the technical details are exactly the same except that one needs to replace the integrand in Eq. (1) as, e.g., $g^2\left(\boldsymbol{x}\right)$, for which the GPR model needs to be trained. One can refer to our previous work in Ref. [44] for a detailed discussion.

4 Collaborative and Adaptive Bayesian Optimization

The estimation of the bounds of the model response moments involves a double-loop process. In the outer loop, a numerical optimization procedure is required for estimating the moment bounds by searching the global minima and maxima of the response moments in the epistemic uncertainty space, and then for each fixed design site of the distribution parameters, a numerical integration procedure is performed in the inner loop to estimate the corresponding deterministic value of model response moment. The basic idea of CABO is illustrated in Figure 1. It involves first training a GPR model in the joint space of aleatory and epistemic uncertainties, and deriving the induced marginal GPR model in the aleatory space and that in the epistemic space respectively. Then, based on the marginal GPR model in the epistemic space, solve the outer-loop optimization by BPO, and based on the marginal GPR model in the aleatory space, compute the inner-loop integral by BPI. In the BPO scenario, a learning function $\mathcal{L}^{\mathrm{AEI}}\left(\boldsymbol{\theta}\right)$, named as Argumented Expected Improvement (AEI) function [50,51], is utilized as the first engine for adaptively generating the design point $\boldsymbol{\theta}^+$ (or $\boldsymbol{v}^+$) for the imprecise distribution parameters $\boldsymbol{\theta}$ to acquire the global optimization points that maximize (or minimize) the response moment, while in the BPI scenario, another learning function $\mathcal{L}^{\mathrm{PVC}}\left(\boldsymbol{u},\boldsymbol{v}^+\right)$, called Posterior Variance Contribution (PVC) function [42], is used as an engine for adaptively generating experiment design point $\boldsymbol{x}^+$ (or $\boldsymbol{u}^+$) for the input random variables $\boldsymbol{x}$ in order to accelerate the convergence of integration. Thus, these two engines work in a collaborative and sequential manner to produce the optimal design point $\boldsymbol{w}^+=\left(\boldsymbol{u}^+,\boldsymbol{v}^+\right)$ in the joint space. Before presenting the CABO algorithm, the two engines need to be introduced. One should note that both AEI and PVC are closed-form functions explicitly and uniquely formulated with the training data and the hyper-parameters of the GPR model, thus any computations with calling these two engines do not introduce new g-function calls.

4.1 Engine 1: Argumented Expected Improvement Function

The AEI function used in the outer-loop BPO is developed for searching the training point $\boldsymbol{\theta}^+$ (or correspondingly $\boldsymbol{v}^+$) in the epistemic space, by adding which to the training data, the most expected improvement for searching the minimum (or maximum) value of $m\left(\boldsymbol{\theta}\right)$ can be achieved. We take the estimation of the lower bound m_L of $m\left(\boldsymbol{\theta}\right)$ subjected to $\boldsymbol{\theta}\in\left[\boldsymbol{\theta}_L,\boldsymbol{\theta}_U\right]$ as an example to illustrate the developed method.

Suppose now, based on the current training data $\mathcal{D}$, the GPR model $\hat{\mathcal{G}}\left(\boldsymbol{w}\right)$ for approximating $\mathcal{G}\left(\boldsymbol{w}\right)$ is trained in the joint space, from which a marginal GPR model $\hat{m}\left(\boldsymbol{\theta}\right)\sim\mathcal{GP}\left(\mu_m\left(\boldsymbol{\theta}\right),\sigma_m^2\left(\boldsymbol{\theta}\right)\right)$ in the epistemic space for approximating $m\left(\boldsymbol{\theta}\right)$ is inferred by using Eqs. (7) and (8). Let

$$\boldsymbol{\theta}^*=\arg\min_{\boldsymbol{\theta}\in\left[\boldsymbol{\theta}_L,\boldsymbol{\theta}_U\right]}\left[\mu_m\left(\boldsymbol{\theta}\right)+\alpha\sigma_m\left(\boldsymbol{\theta}\right)\right] \tag{11}$$

denote the current optimal solution, where α is a parameter reflecting the degree of risk aversion, and one can refer to Ref. [50] for more details. In this paper, α is assumed to be 1. Then we need to find a new point $\boldsymbol{\theta}$, by adding which to the training data set $\mathcal{D}$, the maximum reduction of the objective function can be achieved. For this purpose, we introduce the noise-free version of the AEI function, originally developed in Ref. [50] for global optimization, as follow:

$$\mathcal{L}^{\mathrm{AEI}}\left(\boldsymbol{\theta}\right)=\mathbb{E}_{\mathcal{D}}\left[\max\left(\mu_m\left(\boldsymbol{\theta}^*\right)-\hat{m}\left(\boldsymbol{\theta}\right),0\right)\right]. \tag{12}$$

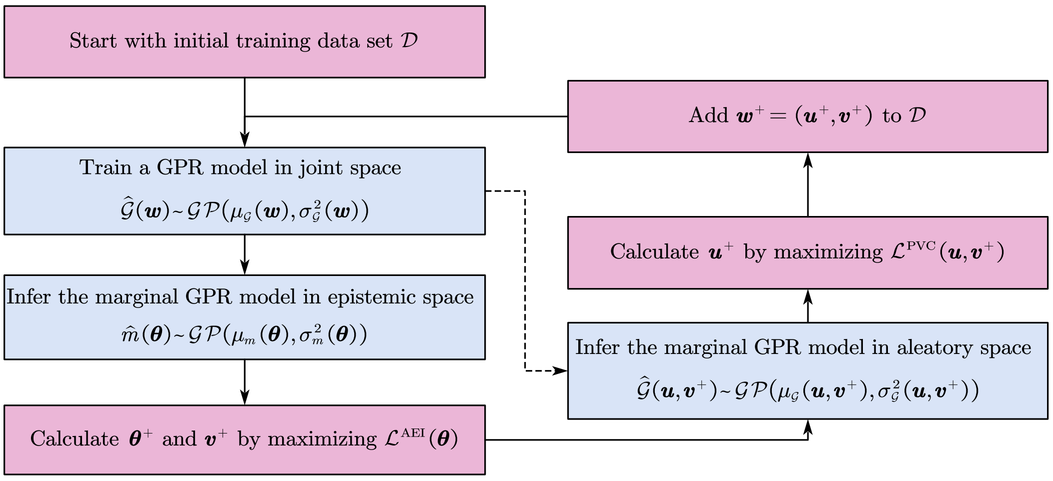
Start with initial training data set $\mathcal{D}$
Add $\boldsymbol{w}^+ = (\boldsymbol{u}^+, \boldsymbol{v}^+)$ to $\mathcal{D}$
Train a GPR model in joint space $\hat{\mathcal{G}}(\boldsymbol{w}) \sim \mathcal{GP}(\mu_{\mathcal{G}}(\boldsymbol{w}), \sigma^2_{\mathcal{G}}(\boldsymbol{w}))$
Calculate $\boldsymbol{u}^+$ by maximizing $\mathcal{L}^{\mathrm{PVC}}(\boldsymbol{u}, \boldsymbol{v}^+)$
Infer the marginal GPR model in epistemic space $\hat{m}(\boldsymbol{\theta}) \sim \mathcal{GP}(\mu_m(\boldsymbol{\theta}), \sigma^2_m(\boldsymbol{\theta}))$
Infer the marginal GPR model in aleatory space $\hat{\mathcal{G}}(\boldsymbol{u}, \boldsymbol{v}^+) \sim \mathcal{GP}(\mu_{\mathcal{G}}(\boldsymbol{u}, \boldsymbol{v}^+), \sigma^2_{\mathcal{G}}(\boldsymbol{u}, \boldsymbol{v}^+))$
Calculate $\boldsymbol{\theta}^+$ and $\boldsymbol{v}^+$ by maximizing $\mathcal{L}^{\mathrm{AEI}}(\boldsymbol{\theta})$

Fig. 1 Illustration of the basic rationale of CABO, where the solid lines show the rough algorithm flow, and the dashed line only indicate the information flow.

The AEI function measures how much the minimum value of $\hat{m}(\boldsymbol{\theta})$ can be reduced compared to $\mu_m(\boldsymbol{\theta}^*)$ if the true value of $m(\boldsymbol{\theta})$ at the point $\boldsymbol{\theta}$ is learned. The use of the maximum operator in the above definition ensures the non-negativity of the AEI function. The closed-form expression of the AEI function is formulated as [50]:

$$\mathcal{L}^{\mathrm{AEI}}(\boldsymbol{\theta}) = (\mu_m(\boldsymbol{\theta}^*) - \mu_m(\boldsymbol{\theta}))\,\Phi\left(\frac{\mu_m(\boldsymbol{\theta}^*) - \mu_m(\boldsymbol{\theta})}{\sigma_m(\boldsymbol{\theta})}\right) + \sigma_m(\boldsymbol{\theta})\,\phi\left(\frac{\mu_m(\boldsymbol{\theta}^*) - \mu_m(\boldsymbol{\theta})}{\sigma_m(\boldsymbol{\theta})}\right), \tag{13}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ are the univariate CDF and PDF of standard Gaussian distribution respectively.

Eq. (13) indicates that, even in the case $\boldsymbol{\theta} = \boldsymbol{\theta}^*$, the value of $\mathcal{L}^{\mathrm{AEI}}(\boldsymbol{\theta})$ can be non-zero. This property allows specifying the next training data as $\boldsymbol{\theta}^*$. This is an appealing property since it can sufficiently reduce the prediction epistemic uncertainty at the current optimal point, and the current design may also have a high probability of being the optimal design. If we replace $\mu_m(\boldsymbol{\theta}^*)$ in Eq. (12) with $\hat{m}(\boldsymbol{\theta}^*)$ so as to account the prediction uncertainty at $\boldsymbol{\theta}^*$, as it is suggested in Ref. [51], the AEI function will equal to zero if the next design is equal to $\boldsymbol{\theta}^*$, indicating that no replicate is allowed. This is not a desired property. Thus, given the definition in Eq. (12), the $\mathcal{L}^{\mathrm{AEI}}(\boldsymbol{\theta})$ does not equal to zero even when the next design is specified to be $\boldsymbol{\theta}^*$, as revealed by Eq. (13).

Searching $\boldsymbol{\theta}^*$ by solving the optimization problem in Eq. (11) across the full support of $\boldsymbol{\theta}$ is usually time-consuming, yet unnecessary. As suggested by Huang et al. [50], $\boldsymbol{\theta}^*$ can be generated by minimizing $\mu_m(\boldsymbol{\theta}) + \alpha\sigma_m(\boldsymbol{\theta})$ over the previously observed locations of $\boldsymbol{\theta}$. The next design site $\boldsymbol{\theta}^+$ to be added to the training set $\mathcal{D}$ is specified by maximizing the AEI function, i.e.,

$$\boldsymbol{\theta}^+ = \arg\max_{\boldsymbol{\theta} \in [\boldsymbol{\theta}_L, \boldsymbol{\theta}_U]} \mathcal{L}^{\mathrm{AEI}}(\boldsymbol{\theta}). \tag{14}$$

The AEI function is found to be multimodal in most cases, thus global optimization algorithms such as particle swarm is recommended for solving Eq. (14). The stopping criteria is given as $\max_{\boldsymbol{\theta}} \mathcal{L}^{\mathrm{AEI}}(\boldsymbol{\theta}) < \Delta^{\mathrm{BPO}}$, or the normalized one:

$$\frac{\max_{\boldsymbol{\theta}} \mathcal{L}^{\mathrm{AEI}}(\boldsymbol{\theta})}{(\max \mathcal{G}(\boldsymbol{w}_i) - \min \mathcal{G}(\boldsymbol{w}_i))_{\boldsymbol{w}_i \in \mathcal{D}}} < \Delta^{\mathrm{BPO}}, \tag{15}$$

if the magnitude of the g-function value is large, where Δ^{BPO} is a user-defined error tolerance. The value of this error tolerance can be pre-specified based on the user's requirement for the accuracy of locating the maxima and minima. For example, for the normalized stopping criteria, it can be specified as $10^{-3} \sim 10^{-2}$. In the next subsection, the strategy for deriving the next design point $\boldsymbol{u}^+$ in the aleatory space, accompanied by $\boldsymbol{v}^+$, is developed.

4.2 Engine 2: Posterior Variance Contribution Function

The PVC function aims at specifying the design point of $\boldsymbol{u}$, by adding which to the training data set together with $\boldsymbol{\theta}^+$, the most reduction of the posterior variance $\sigma^2_m(\boldsymbol{\theta}^+)$ at the site $\boldsymbol{\theta}^+$ can be achieved. This is equivalent to identify a design point of $\boldsymbol{u}$ to minimize the posterior variance $\sigma^2_{\mathcal{M}}(\boldsymbol{v}^+)$ of the Gaussian random variable $\hat{\mathcal{M}}(\boldsymbol{v}^+) \sim \mathcal{GP}(\mu_{\mathcal{M}}(\boldsymbol{v}^+), \sigma^2_{\mathcal{M}}(\boldsymbol{v}^+))$, where $\boldsymbol{v}^+ = T^{-1}(\boldsymbol{\theta}^+)$.

Based on the GPR model $\hat{\mathcal{G}}(\boldsymbol{w})$ trained in the joint space, a marginal GPR model in the aleatory space is inferred as $\hat{\mathcal{G}}(\boldsymbol{u},\boldsymbol{v}^+) \sim \mathcal{GP}\left(\mu_{\mathcal{G}}(\boldsymbol{u},\boldsymbol{v}^+), \sigma^2_{\mathcal{G}}(\boldsymbol{u},\boldsymbol{v}^+)\right)$, whose posterior covariance between two points $\boldsymbol{u}$ and $\boldsymbol{u}'$ is denoted as $\mathrm{cov}_{\mathcal{G}}\left((\boldsymbol{u},\boldsymbol{v}^+),(\boldsymbol{u}',\boldsymbol{v}^+)\right)$. Then, from Eq. (8), the posterior variance $\sigma^2_{\mathcal{M}}(\boldsymbol{v})$ can be reformulated as:

$$\sigma^2_{\mathcal{M}}(\boldsymbol{v}^+) = \Pi_{1:n}\left[h(\boldsymbol{u},\boldsymbol{v}^+)\right], \tag{16}$$

where

$$\begin{aligned} &h(\boldsymbol{u},\boldsymbol{v}^+) \\ &= \Pi'_{1:n}\left[\mathrm{cov}_{\mathcal{G}}\left((\boldsymbol{u},\boldsymbol{v}^+),(\boldsymbol{u}',\boldsymbol{v}^+)\right)\right] \\ &= \Pi'_{1:n}\left[\kappa\left((\boldsymbol{u},\boldsymbol{v}^+),(\boldsymbol{u}',\boldsymbol{v}^+)\right)\right] \\ &- \boldsymbol{\kappa}\left((\boldsymbol{u},\boldsymbol{v}^+),\mathcal{W}\right)^\top K^{-1}\Pi'_{1:n}\left[\boldsymbol{\kappa}\left((\boldsymbol{u}',\boldsymbol{v}^+),\mathcal{W}\right)\right]. \end{aligned} \tag{17}$$

Eq. (16) reveals that the posterior variance $\sigma^2_{\mathcal{M}}(\boldsymbol{v}^+)$ is equal to the integral of $h(\boldsymbol{u},\boldsymbol{v}^+)$ over the standard Gaussian density, thus $h(\boldsymbol{u},\boldsymbol{v}^+)$ actually measures the contribution of the prediction error of the resulting GPR model $\hat{\mathcal{G}}(\boldsymbol{u},\boldsymbol{v}^+)$ at the point $\boldsymbol{u}$ to the posterior variance $\sigma^2_{\mathcal{M}}(\boldsymbol{v}^+)$. Eq. (17) implies that $h(\boldsymbol{u},\boldsymbol{v}^+)$ is defined by integrating the posterior covariance with respect to $\boldsymbol{u}'$ over the whole aleatory space, indicating $h(\boldsymbol{u},\boldsymbol{v}^+)$ summarizes the spatial correlation information of the node $\boldsymbol{u}$ with all the other points in the aleatory space. Inspired by this fact, the PVC learning function for adaptive BPI has been developed in our previous work, and it is formulated as[42]:

$$\mathcal{L}^{\mathrm{PVC}}(\boldsymbol{u},\boldsymbol{v}^+) = \boldsymbol{\phi}_n(\boldsymbol{u})\,h(\boldsymbol{u},\boldsymbol{v}^+), \tag{18}$$

where $\boldsymbol{\phi}_n(\cdot)$ is the joint PDF of n-dimensional standard Gaussian distribution. By adding the point of $\boldsymbol{u}$ with the maximum PVC value to the training data set, it is expected to achieve the maximum reduction of the posterior variance $\sigma^2_{\mathcal{M}}(\boldsymbol{v}^+)$, and thus also the most improvement of the accuracy of estimating the inner-loop integral $\mathcal{M}(\boldsymbol{v}^+)$. From Eqs. (17) and (18), it is seen that derivation of the closed-form expression for the PVC function $\mathcal{L}^{\mathrm{PVC}}(\boldsymbol{u},\boldsymbol{v}^+)$ requires the closed-form expressions for the two integrals $\Pi'_{1:n}\left[\kappa\left((\boldsymbol{u},\boldsymbol{v}^+),(\boldsymbol{u}',\boldsymbol{v}^+)\right)\right]$ and $\Pi'_{1:n}\left[\boldsymbol{\kappa}\left((\boldsymbol{u}',\boldsymbol{v}^+),\mathcal{W}\right)\right]$, where the second integral can be generated as Eq. (9) by replacing $\boldsymbol{v}$ with $\boldsymbol{v}^+$, and the first integral is formulated as [42]:

$$\begin{aligned} &\Pi'_{1:n}\left[\kappa\left((\boldsymbol{u},\boldsymbol{v}^+),(\boldsymbol{u}',\boldsymbol{v}^+)\right)\right] \\ &= \sigma_0^2\left|\Sigma_{\boldsymbol{u}}^{-1}+I\right|^{-1/2}\exp\left[-\frac{1}{2}\boldsymbol{u}\left(\Sigma_{\boldsymbol{u}}+I\right)^{-1}\boldsymbol{u}^\top\right]. \end{aligned} \tag{19}$$

In summary, the next design point $\boldsymbol{u}^+$ of $\boldsymbol{u}$ can be computed by maximizing the closed-form expression of the PVC function. It is found that, in most cases, the PVC function is also multimodal, thus global optimization algorithms such as particle swarm are recommended. Similar to the BPO procedure, a stopping threshold Δ^{BPI} should be pre-specified for controlling the accuracy of integration. Usually, it is recommended to take the value between $1 \sim 5\%$, depending on the users' requirement on integration accuracy.

4.3 Twin-engine Collaborative and Adaptive Bayesian Optimization

With the two learning functions as twin engines, the CABO algorithm is devised, and the flowchart is shown in Figure 2. We take the computation of the lower bound of the response expectation as an example, the procedure is described as follows in detail.

Step 1: Initialization.

Specify the initial sample size N_0 (e.g., 20), and then create the corresponding initial training data set $\mathcal{D} = (\mathcal{W},\mathcal{Y})$ of size N_0. Record the number of g-function calls as $N = N_0$. The initial training sample set $\mathcal{W} = (\mathcal{U},\mathcal{V})$ can be generated by the following two steps. First, generate a random sample $\boldsymbol{\theta}^{(i)}$ following the auxiliary distribution $p(\boldsymbol{\theta})$, and then transform this point to a standard normal sample point by $\boldsymbol{v}^{(i)} = T^{-1}\left(\boldsymbol{\theta}^{(i)}\right)$. Second, create a random sample point $\boldsymbol{x}^{(i)}$ by the probability distribution $f\left(\boldsymbol{x}\middle|\boldsymbol{\theta}^{(i)}\right)$, and then transform it into a standard normal sample point using $\boldsymbol{u}^{(i)} = S^{-1}\left(\boldsymbol{x}^{(i)}|\boldsymbol{\theta}^{(i)}\right)$. With the above procedure, we generate a joint sample point $\boldsymbol{w}^{(i)} = \left(\boldsymbol{u}^{(i)},\boldsymbol{v}^{(i)}\right)$, and the corresponding value of the model response can be computed as $y^{(i)} = g\left(\boldsymbol{x}^{(i)}\right)$. In this paper, random sampling is realized using Latin-Hypercube Sampling (LHS) design [52].

Step 2: Train the GPR model.

Train a GPR model $\hat{\mathcal{G}}(\boldsymbol{w})$ with $\mathcal{D}$, and derive the marginal GPR model $\hat{m}(\boldsymbol{\theta}) \sim \mathcal{GP}\left(\mu_m(\boldsymbol{\theta}),\sigma^2_m(\boldsymbol{\theta})\right)$ in the epistemic space by Eqs. (7) and (8). Since the computer simulator is deterministic, thus the noise-free GPR model is utilized. For problems with low nonlinearity, zero or constant mean function is suggested, while for the highly nonlinear problems, the linear or higher-order polynomial bases are recommended.

Step 3: Run BPO.

This step involves first computing the current optimal site $\boldsymbol{\theta}^*$ by Eq. (11). For reducing the computational cost in this step, it is suggested to search $\boldsymbol{\theta}^*$ across the training data of $\boldsymbol{\theta}$ which is included in $[\boldsymbol{\theta}_L,\boldsymbol{\theta}_U]$, instead of performing optimization in the whole support $[\boldsymbol{\theta}_L,\boldsymbol{\theta}_U]$. Then, the closed-form expression of the AEI function can be generated as Eq. (13). By solving the optimization problem in Eq. (14), we can obtain

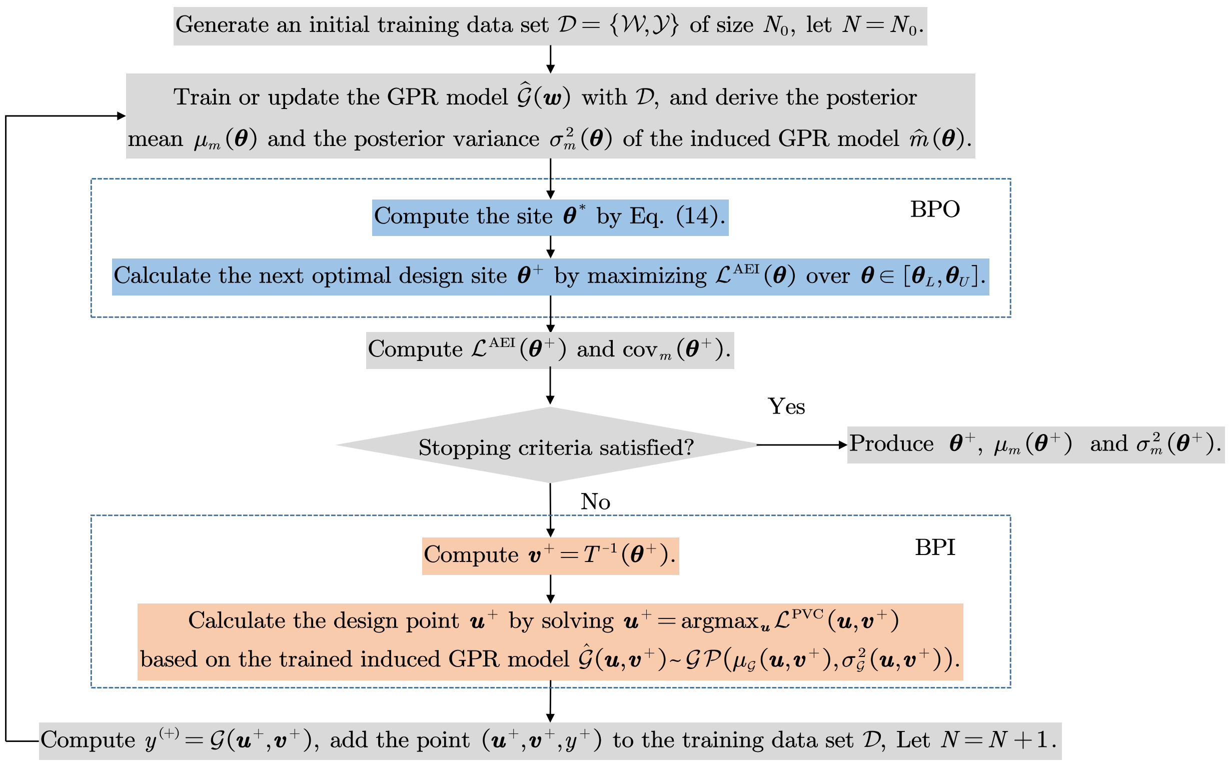
Generate an initial training data set $\mathcal{D}=\{\mathcal{W},\mathcal{Y}\}$ of size N_0, let $N=N_0$.
Train or update the GPR model $\hat{\mathcal{G}}(\boldsymbol{w})$ with $\mathcal{D}$, and derive the posterior mean $\mu_m(\boldsymbol{\theta})$ and the posterior variance $\sigma_m^2(\boldsymbol{\theta})$ of the induced GPR model $\hat{m}(\boldsymbol{\theta})$.
BPO
Compute the site $\boldsymbol{\theta}^*$ by Eq. (14).
Calculate the next optimal design site $\boldsymbol{\theta}^+$ by maximizing $\mathcal{L}^{\mathrm{AEI}}(\boldsymbol{\theta})$ over $\boldsymbol{\theta}\in[\boldsymbol{\theta}_L,\boldsymbol{\theta}_U]$.
Compute $\mathcal{L}^{\mathrm{AEI}}(\boldsymbol{\theta}^+)$ and $\mathrm{cov}_m(\boldsymbol{\theta}^+)$.
Stopping criteria satisfied?
Yes
Produce $\boldsymbol{\theta}^+$, $\mu_m(\boldsymbol{\theta}^+)$ and $\sigma_m^2(\boldsymbol{\theta}^+)$.
No
BPI
Compute $\boldsymbol{v}^+=T^{-1}(\boldsymbol{\theta}^+)$.
Calculate the design point $\boldsymbol{u}^+$ by solving $\boldsymbol{u}^+=\mathrm{argmax}_{\boldsymbol{u}}\mathcal{L}^{\mathrm{PVC}}(\boldsymbol{u},\boldsymbol{v}^+)$ based on the trained induced GPR model $\hat{\mathcal{G}}(\boldsymbol{u},\boldsymbol{v}^+)\sim\mathcal{GP}(\mu_{\mathcal{G}}(\boldsymbol{u},\boldsymbol{v}^+),\sigma_{\mathcal{G}}^2(\boldsymbol{u},\boldsymbol{v}^+))$.
Compute $y^{(+)}=\mathcal{G}(\boldsymbol{u}^+,\boldsymbol{v}^+)$, add the point $(\boldsymbol{u}^+,\boldsymbol{v}^+,y^+)$ to the training data set $\mathcal{D}$, Let $N=N+1$.

Fig. 2 Flowchart of the twin-engine CABO algorithm.

the next best design site $\boldsymbol{\theta}^+$. Since the AEI function is mostly multimodal (have multiple local maximas) and the next design point is the global maxima, it is suggested to use global optimization algorithms such as the particle swarm solver, which can be realized by the 'particleswarm' function in the Matlab Global Optimization Toolbox. With this step, we generate the next design point $\boldsymbol{\theta}^+$, the corresponding maximum AEI value $\mathcal{L}^{\mathrm{AEI}}\left(\boldsymbol{\theta}^+\right)$, the corresponding GPR prediction $\mu_m\left(\boldsymbol{\theta}^+\right)$ and the corresponding posterior variance $\sigma_m^2\left(\boldsymbol{\theta}^+\right)$. The posterior Coefficient Of Variation (COV) of the GPR prediction is then computed by $\mathrm{cov}_m\left(\boldsymbol{\theta}^+\right) = \mu_m\left(\boldsymbol{\theta}^+\right) / \sigma_m\left(\boldsymbol{\theta}^+\right)$.

Step 4: Judge the stopping criteria.

The stopping criteria for BPO is given as $\mathcal{L}^{\mathrm{AEI}}\left(\boldsymbol{\theta}^+\right) < \Delta^{\mathrm{BPO}}$ or the normalized one in Eq. (15), and that for BPI is $\mathrm{cov}_m\left(\boldsymbol{\theta}^+\right) < \Delta^{\mathrm{BPI}}$, where Δ^{BPO} and Δ^{BPI} are both small thresholds. Here we suggest using a delayed judgment, which means to stop the algorithm only when the above two stopping criteria are simultaneously satisfied for several (e.g., two) times in successive steps. If the stopping criteria are satisfied, end the algorithm; if not, go to **Step 5**.

Step 5: Perform BPI.

The design point $\boldsymbol{v}^+$ corresponding to $\boldsymbol{\theta}^+$ is computed by $\boldsymbol{v}^+ = T^{-1}\left(\boldsymbol{\theta}^+\right)$, and then the closed-form expression of the PVC function can be generated by Eq. (18). The next design point $\boldsymbol{u}^+$ in the aleatory space is then computed by maximizing the PVC function. Similarly, the PVC function is often multimodal, and the particle swarm solver is recommended.

Step 6: Enrich the training data set.

With the above steps, we obtain the next design point $\boldsymbol{w}^+ = (\boldsymbol{u}^+, \boldsymbol{v}^+)$ in the joint space. Computing the corresponding g-function value y^+, the training data set $\mathcal{D}$ is enriched with $(\boldsymbol{u}^+, \boldsymbol{v}^+, y^+)$. Let $N = N + 1$, go to **Step 2**.

■

Although the global algorithms such as particle swarm solver need to be performed in step 3 and step 5, the target functions are both in closed form, and thus no g-function call is required. It is only in step 6 that one needs to compute the g-function value at the design point $\boldsymbol{w}^+$ collaboratively produced by the two engines, thus the algorithm is of high efficiency for expensive computer simulators.

With the above procedure, the posterior mean and posterior variance of the lower bound m_L can be computed, where the posterior variance summarize the discretization errors for computing the bound. For computing the upper bound, one needs only to set the initial training data set as the one obtained for computing the lower bound and the target function of the optimization problem as minus $m(\boldsymbol{\theta})$, then restart the above procedure. In the next section, we introduce one illustrative example and two real-world engineering examples for illustrating and demonstrating the advantages of the CABO method.

5 Test examples

As has been mentioned above, for implementing the CABO algorithm, an auxiliary probability density $p(\boldsymbol{\theta})$ needs to be attributed to $\boldsymbol{\theta}$. Following our experience in Ref. [44], it is assumed that $\boldsymbol{\theta}$ follow a uniform distribution with relaxed support $[\boldsymbol{\theta}_l - \epsilon_l, \boldsymbol{\theta}_u + \epsilon_u]$. The values of ϵ_l and ϵ_u can be specified to satisfy the following two criteria, $-2.2 \leqslant \Phi^{-1}\left(U\left(\boldsymbol{\theta}_l | \boldsymbol{\theta}_l - \epsilon_l, \boldsymbol{\theta}_u + \epsilon_u\right)\right) \leqslant -1.5$ and $1.5 \leqslant \Phi^{-1}\left(U\left(\boldsymbol{\theta}_u | \boldsymbol{\theta}_l - \epsilon_l, \boldsymbol{\theta}_u + \epsilon_u\right)\right) \leqslant 2.2$, where $U\left(\cdot | \boldsymbol{\theta}_l - \epsilon_l, \boldsymbol{\theta}_u + \epsilon_u\right)$ indicates the CDF of uniform distribution with support $[\boldsymbol{\theta}_l - \epsilon_l, \boldsymbol{\theta}_u + \epsilon_u]$ and $\Phi^{-1}(\cdot)$ refers to the inverse CDF of standard normal distribution.

5.1 An illustrative example

The g-function is formulated as:

$$g(x_1, x_2) = \sum_{i=1}^{4} c_i \exp\left[-\alpha_{i1}(x_1 - \beta_{i1})^2 - \alpha_{i2}(x_2 - \beta_{i2})^2\right], \tag{20}$$

where $\boldsymbol{\alpha} = \begin{pmatrix} 2 & 3 & 1 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix}^\top$, $\boldsymbol{\beta} = \begin{pmatrix} -0.5 & 0.5 & -0.5 & 0.5 \\ -0.5 & -0.5 & 0.5 & 0.5 \end{pmatrix}^\top$, $\boldsymbol{c} = \begin{pmatrix} 1 & -1.5 & -1.5 & 2 \end{pmatrix}^\top$, x_1 and x_2 are independent normal random variables. The task is to estimate the bounds $[m_L, m_U]$ of the model response expectation. We consider two cases. For case 1, it is assumed that $x_1 \sim \mathcal{N}(\theta_1, 0.1^2)$ and $x_2 \sim \mathcal{N}(\theta_2, 0.1^2)$, where the imprecise parameters θ_1 and θ_2 are both bounded by $[-1.5, 1.5]$. For case 2, it is assumed that $x_1 \sim \mathcal{N}(\theta_1, \theta_3^2)$ and $x_2 \sim \mathcal{N}(\theta_2, \theta_4^2)$, where the mean parameters θ_1 and θ_2 are still bounded by [-1.5, 1.5], the STandard Deviation (STD) parameters θ_3 and θ_4 are also assumed to be imprecise and bounded by [0.05, 0.2]. For this example, the linear basis function is assumed for the prior GPR means.

♦ **Results and discussions for case 1**

For this case, the closed-form expression of the response expectation function $m(\theta_1, \theta_2)$ is available, which is shown in Figure 3, together with the true extreme points. As can be seen, there exist two local minima and two local maxima, but there are only one global minima and one global maxima, which determine the bounds to be estimated. The purpose of this example is, on the one hand, to illustrate the details of the training process of CABO, and on the other hand, to demonstrate the global convergence of CABO for nonlinear problems with multiple pairs of local extrema.

The auxiliary probability distributions for θ_1 and θ_2 are both set to be a uniform distribution with support $[-1.65, 1.65]$, and then 40 joint samples are generated with LHS design to initialize the CABO algorithm. The initial 40 training samples for $\boldsymbol{\theta}$ are shown in Figure 4. The stopping criteria for BPO and BPI are set to be $\max_{\boldsymbol{\theta}} \mathcal{L}^{\mathrm{AEI}}(\boldsymbol{\theta}) < 2 \times 10^{-3}$ and $\mathrm{cov}_m\left(\boldsymbol{\theta}^{(+)}\right) < 1\%$, respectively. The CABO algorithm stops only when the above stopping criteria are satisfied simultaneously twice in succession. With the above setting, the CABO algorithm adaptively produces 34 more training points for estimating the global minimum, and then 27 more training points for calculating the global maximum. The adaptively added training points in the epistemic space are shown in Figure 4, together with the minimum and maximum points. Thus, the total number of g-function calls is $N_{\mathrm{call}} = 40 + 34 + 27 = 101$. The results of lower and upper bounds estimated by CABO are reported in Table 1, together with the analytical results and the results generated by IMCS (see Ref. [23]) for comparison. For implementing the IMCS method, the size of interval samples is set to be $N = 100$, and for each interval sample, the particle swarm algorithm is utilized for computing the corresponding bounds of model output, and then the resultant interval samples of model response are used for computing the bounds of response expectation [23]. The total number of g-function calls consumed by IMCS is 6.192×10^4, which is much higher than that of CABO. One notes that the estimated bounds by IMCS is usually more conservative than the real bounds. The real bounds of the model response expectation are determined and only determined by all the Gaussian distributions bounded by the p-box model. However, by creating interval samples with IMCS, a default assumption, that all the possible CDFs (including those non-Gaussian ones) bounded by $\Phi\left(x_i | -1.5, 0.1^2\right)$ and $\Phi\left(x_i | 1.5, 0.1^2\right)$ can be realized, has been made, which may result in underestimate of the lower bound and overestimate of the upper bound [35].

It is seen that CABO requires much less g-function calls than IMCS, and the computed interval is tighter than that estimated by IMCS, and has a better agreement with the analytical results. One can also see from Table 1 that the bounds generated by CABO is [-1.343, 1.337], which show a translation to the analytical bounds [-1.354, 1.327], although the difference is very small.

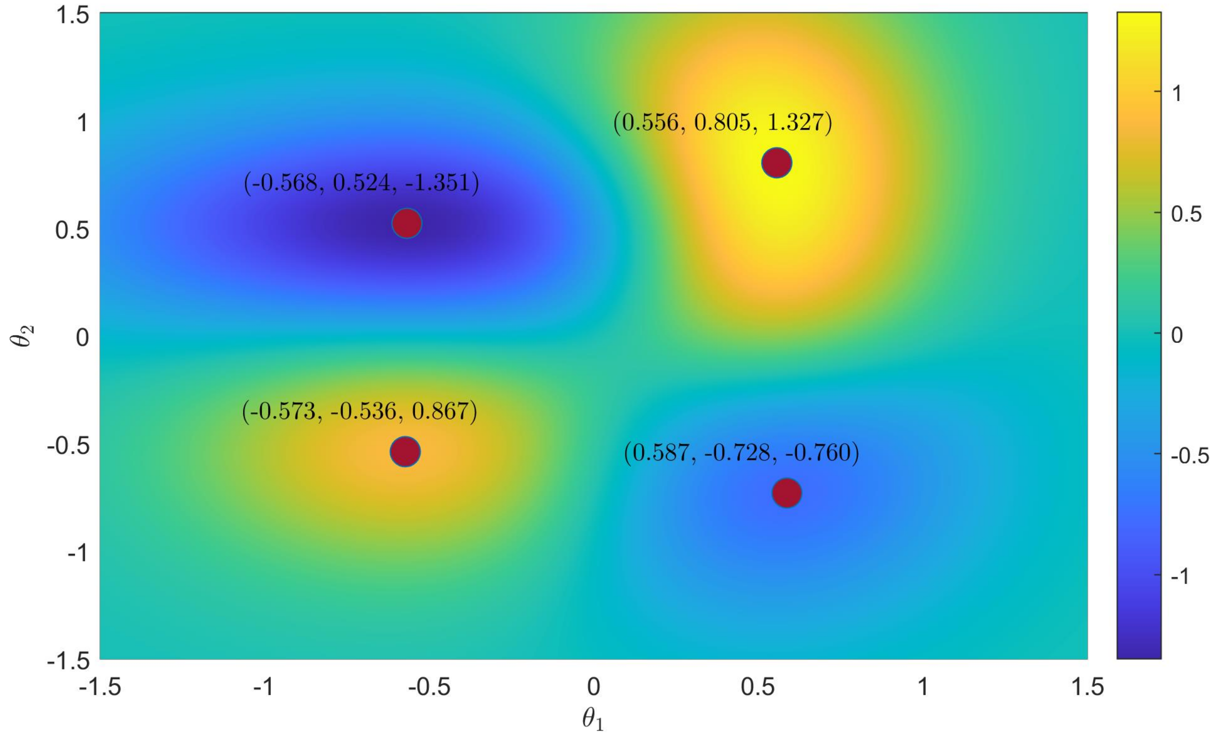

Fig. 3 The analytical response expectation function and the corresponding four extreme points for case 1 of the illustrative example.

Since the posterior COVs of the estimates are very small (around 0.1%), this must be caused by the error of finding the extreme points by BPO. This can be easily verified by the difference between the extreme points generated by CABO and those generated analytically. One can further improve the accuracy for computing the extreme points by setting the threshold $\varDelta^{\mathrm{BPO}}$ to be a smaller value. For example, as we set $\varDelta^{\mathrm{BPO}}$ as 2×10^{-4}, the generated bounds are [-1.348, 1.327], which match better with the analytical results, but the total number of g-functions is increased from 101 to 149.

Figure 4 shows that both the global maximum and minimum points are successfully and accurately identified by the CABO algorithm, and the results in Table 1 also imply that the posterior COVs of the estimated lower and upper bounds are both much less than the pre-specified threshold 1%, demonstrating the high accuracy and robustness of the results. It can be seen from Figure 4 that, the adaptively produced training data of $\boldsymbol{\theta}$ for estimating the lower bound are partly located around the boundary of the support of $\boldsymbol{\theta}$, which is due to the large prediction error of the initial trained GPR model in these areas. Setting the probability distribution support of $\boldsymbol{\theta}$ larger than its real support allows initial training points to be sampled outside the real support (see Figure 4), which helps to reduce the GPR prediction error around the boundary. This is one of the reasons why we suggest to set the support of the auxiliary distribution of $\boldsymbol{\theta}$ a little bit larger than its real support.

It is also shown in Figure 4 that the adaptively added training points for lower bound are located around the two local pairs of minimum. This is fair since more training data are required for identifying the global minimum point. It can also be seen that, during the training process for the lower bound, there is no training point around the two local maximum points. This is because, although the initial GPR model shows large prediction error in these areas, the (subjective) probability that these areas contain the minimum point is extremely low, thus no g-function call is wasted in these areas. This fact makes the CABO algorithm extremely efficient for estimating the bounds of the output expectation without the necessity to approximate the output response moment function with high accuracy across the whole support of $\boldsymbol{\theta}$. This advantage benefits from the cooperativity and adaptivity of the two engines. Based on the 40+34 training samples, the CABO algorithm automatically produces 27 more training points to accurately identify the global maxima. As can be

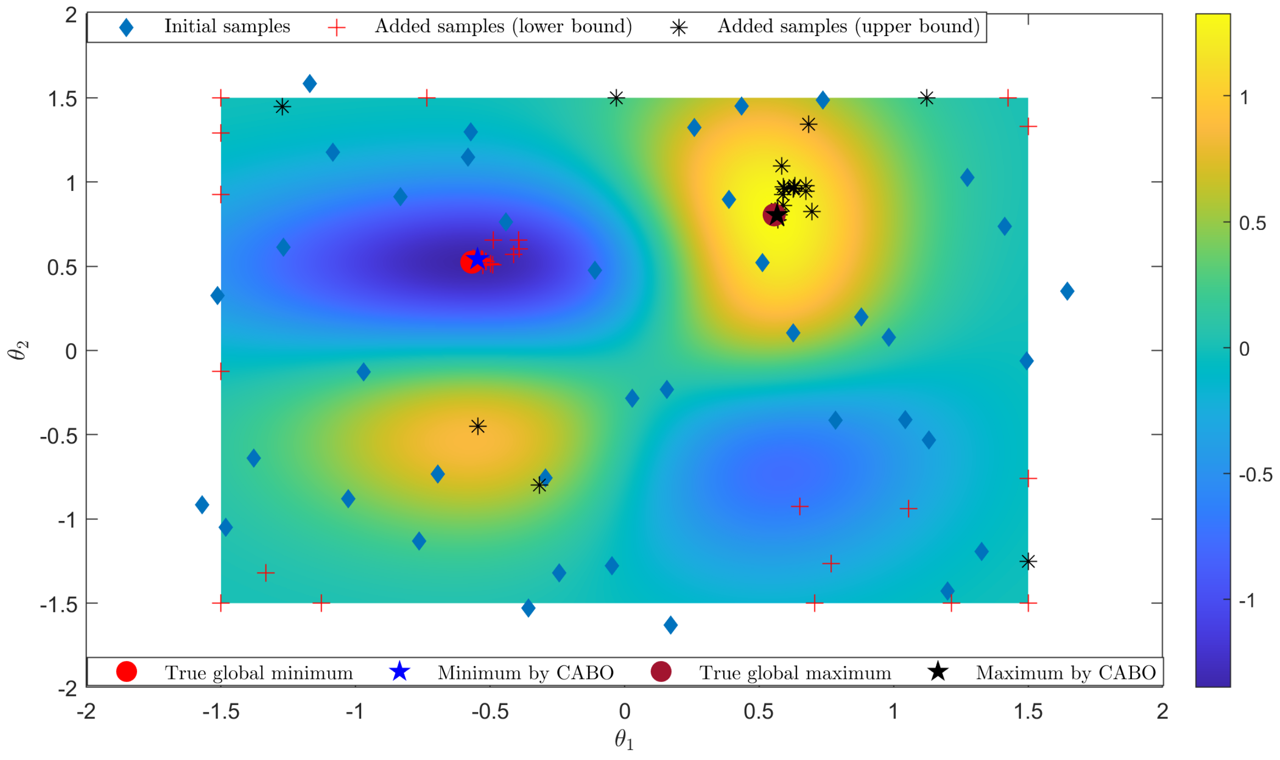

Fig. 4 The training details (in the epistemic space) of the CABO algorithm for estimating the lower, and then, upper bounds, where the colormap refers to the true function of $m(\theta_1, \theta_2)$.

Table 1 Results for case 1 of the illustrative example.

Methods	Bounds	Extreme points	Mean estimates	COVs (%)	N_{call}[1]
Analytical	m_L	(-0.568, 0.524)	-1.351	–	–
	m_U	(0.556, 0.805)	1.327	–	–
CABO	m_L	(-0.547, 0.542)	-1.343	0.110	40+34=74
	m_U	(0.567, 0.804)	1.337	0.112	74+27=101
IMCS ($N = 100$)	m_L	–	-1.433	< 0.010	6.192×10^4
	m_U	–	1.412	< 0.010	

[1] N_{call} indicates the total number of g-function calls. For the lower bound m_L computed by CABO, the first number indicates the number of initial design points, while the second number implies the extra number of calls for searching the lower bound. For the upper bound m_U, the first number indicates the total number of calls consumed for computing m_L, and the second number refers to the extra number of calls for estimating m_U.

shown, those 27 training points are mostly located around the two local maximum points, and none of them is located around the two local minimum points. This further improves the high efficiency and high robustness of the algorithm, as no g-function call will be wasted when it is not necessary. It is also shown that there are more training points around the global optimal points, and this is due to the necessity of improving the integration accuracy around the optimal points. This is finally realized with a small number of training points in the aleatory uncertainty space adaptively identified by the PVC function. The above facts have comprehensively demonstrated the superiority of the twin-engine CABO algorithm.

The training samples in the aleatory space are shown in Figure 5, together with the colormap plot of the g-

function. It is shown that the distribution patterns of the adaptively produced training samples for the lower and upper bounds are both quite similar to those produced in the epistemic space shown in Figure 4. Considering that the variations of the two input variables are both very small, the design point of $\boldsymbol{x}$ accompanied by $\boldsymbol{\theta}^{+}$ is specified for minimizing the posterior variance of the integral $m\left(\boldsymbol{\theta}^{+}\right)$. This can be further illustrated by the following facts. As shown in Figure 4, most of the adaptively specified integration points $\boldsymbol{x}$ for estimating the lower bound m_L are located around the area point [-0.5, 0.5]. Obviously, these points are designed for reducing the posterior variance of the integral $m\left(\boldsymbol{\theta}\right)$ at those design points of $\boldsymbol{\theta}$ around the global minimum point. A similar phenomenon can be found for the adaptively produced training samples for the upper bound of $m\left(\boldsymbol{\theta}\right)$.

Next, we investigate the prediction errors of the GPR models trained for estimating the lower and upper bounds. With the 40+34 training data, a GPR model $\hat{m}\left(\boldsymbol{\theta}\right)$ is obtained. The absolute difference between the posterior mean of $\hat{m}\left(\boldsymbol{\theta}\right)$ and the analytical $m\left(\boldsymbol{\theta}\right)$ is shown by the top-left graph of Figure 6, and the posterior STD of $\hat{m}\left(\boldsymbol{\theta}\right)$ is shown in the top-right graph. Both graphs show that, when the global minimum point is identified, the GPR prediction error around this point is small, but that in the other areas (e.g., around the two local maximum points) which are not informative for the lower bound, the prediction error can be large. The left two graphs in Figure 6 show the absolute difference and posterior STD of the GPR model trained with all the 101 training points, and it is shown that the prediction error around the global maximum point is extremely small. The above phenomenon highlights the fact CABO does not seek to minimize the global error of the GPR surrogate model $\hat{m}\left(\boldsymbol{\theta}\right)$, but instead, makes a good trade-off between accuracy and efficiency for estimating the bounds of $m\left(\boldsymbol{\theta}\right)$. This is an appealing feature when the target is to learn the bounds, instead of the global behavior of $m\left(\boldsymbol{\theta}\right)$, especially when the epistemic uncertainty is large.

⧫ Results and discussions for case 2

In this case, the dimension of the epistemic space is four, and it is used to test the performance of CABO for the situations where both the mean and STD parameters are imprecise. The stopping criteria is set to be the same as that in case 1. The support of the auxiliary distribution for θ_1 and θ_2 is set to be [-1.65, 1.65], and that for θ_3 and θ_4 is set to be [0.04, 0.21]. Forty training points randomly generated by LHS design are utilized for initializing the CABO algorithm, and 100 interval samples are generated for implementing the IMCS samples. The results are reported in Table 2, together with the analytical results for comparison. It is shown that for both lower and upper bounds, the results computed by CABO match well with the analytical results, and the posterior COVs are less than 1%, indicating the higher accuracy and robustness of CABO. The bounds estimated by IMCS are also in good agreement with the true results, but are a little bit more conservative. The total number of g-function calls consumed by IMCS is 4.42×104, while that of CABO is 143, indicating the high efficiency of CABO.

5.2 NASA Langley UQ challenge models

Next, we consider the NASA 2014 UQ challenge problem, which simulates the dynamics of remotely operated twin-jet aircraft named Generic Transport Model. One can refer to Refs. [14] and [35] for the details of this challenge. It is composed of five fixed-discipline models and eight cross-discipline models, where each cross-discipline model forms a failure mode. In this paper, we only concern the fixed-discipline models. As the fifth fixed discipline model involves only one input variable, it is not of interest here. The remaining four fixed-discipline models are then described by:

$$y_i = g_i\left(p_{5(i-1)+1}, p_{5(i-1)+2}, \cdots, p_{5(i-1)+5}\right), \tag{21}$$

where y_i $(i = 1, 2, 3, 4)$ indicate the model outputs of the four models, each of which has five input variables $p_{5(i-1)+1} \sim p_{5(i-1)+5}$. In the original setting, some of the input variables, e.g., p_2 and p_6, are constant-but-unknown variables, and are modeled by independent intervals. In this paper, we don't concern the interval models, thus these variables are assumed to be fixed. The details of the twenty input variables are described in Table 3. For each of the four models, there are five input variables, three of which are characterized by p-box. For the first model, the three p-box models introduce seven imprecise distribution parameters, while for each of the other three models, six imprecise distribution parameters are involved. The target is then to estimate the bounds of response expectations for these four fixed-discipline models, which are termed as $[m_{Li}, m_{Ui}]$ $(i = 1, 2, 3, 4)$ respectively. For all the runs in this example, constant basis function is assumed for the GPR means.

We implement CABO for the four fixed discipline models by setting $N_0 = 20$, $\Delta^{\mathrm{BPO}} = 5 \times 10^{-4}$ and $\Delta^{\mathrm{BPI}} = 0.05$. Actually, for all the four runs, it is found that, when the stopping criteria for BPO is satisfied, the stopping criteria for BPI is always satisfied. This case also happens in the first illustrative example. This is why the posterior COVs are always much smaller

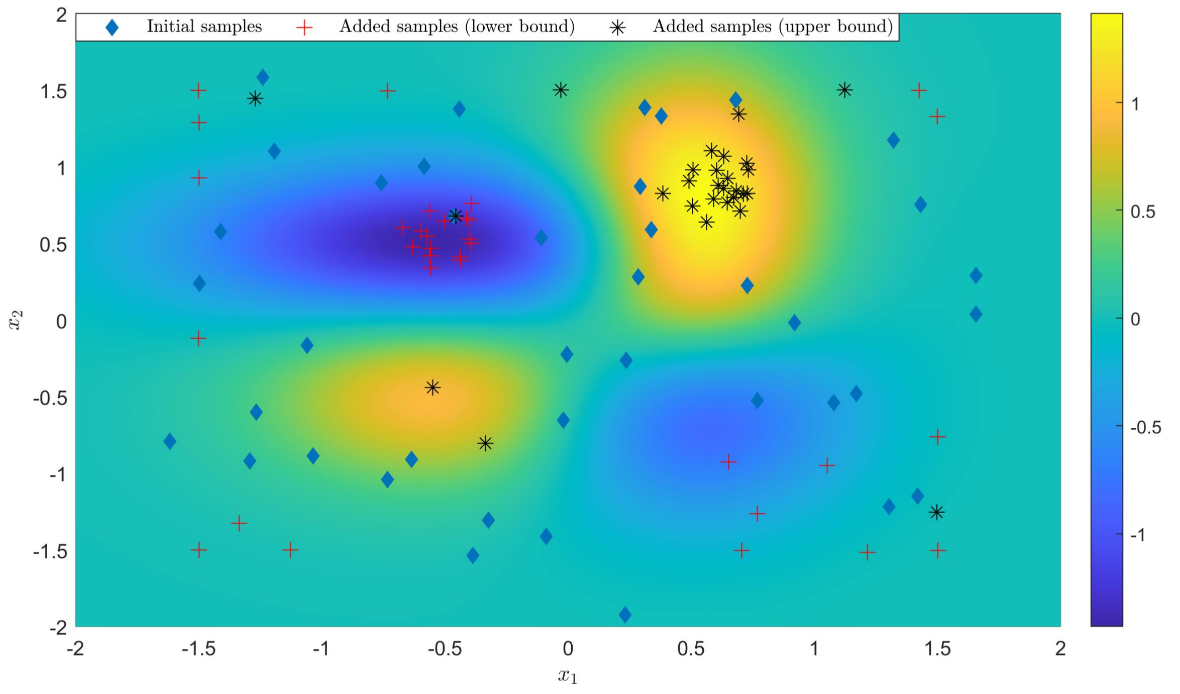

Fig. 5 Training details (in the aleatory space) of the CABO algorithm for estimating the lower, and then upper bounds, where the heat map indicates the true function $g(x_1, x_2)$.

Table 2 Results for case 2 of the illustrative example.

Methods	Bounds	Extreme points	Mean estimates	COVs (%)	N_{call}
Analytical	m_L	(-0.558, 0.521, 0.05, 0.05)	-1.411	–	–
	m_U	(0.552, 0.802, 0.05, 0.05)	1.390	–	–
CABO	m_L	(-0.579, 0.527, 0.05, 0.05)	-1.417	0.16	40+51=91
	m_U	(0.552, 0.806, 0.05, 0.05)	1.407	0.04	91+52=143
IMCS ($N = 100$)	m_L	–	-1.433	< 0.010	4.42×10^4
	m_U	–	1.412	< 0.010	

than the pre-specified threshold. The results of CABO for the four models are reported in Table 4-7, where the reference solutions are computed by using the local NISS method without HDMR decomposition and the IMCS method. The sample size of NISS is set to be 10^5 to ensure the high accuracy. For implementing IMCS, the particle swarm algorithm is utilized for computing the bounds of response value corresponding to each set of input interval samples.

We first discuss the results of the first models reported in Table 4. The IMCS is implemented by setting the size of interval samples as 100 and 1000, and the corresponding numbers of g-function calls are 5.334×10^4 and 5.7078×10^5 respectively due to the expensive inner loop particle swarm optimization. It can be seen that the CABO algorithm, initialized with 20 training points, accurately estimate the lower bound m_{L1} with 19 adaptively produced design points in the joint space, and then, with 7 more design points, the algorithm also successfully computes the upper bound m_{U1} with high accuracy. The posterior COVs of both bounds are less than 1%, indicating the high robustness of the mean estimates. The total number of model calls is then 46, implying the high efficiency of CABO. The mean estimates by IMCS with 100 interval samples are acceptable, but the COVs are higher than 5%. When the size of interval samples increases to 1000, much smaller COVs are achieved. These results show that, in terms of efficiency, CABO is much more promising than NISS and IMCS.

From the third columns of Table 4, it can be learned that for some epistemic parameters such as θ_5 and θ_6,

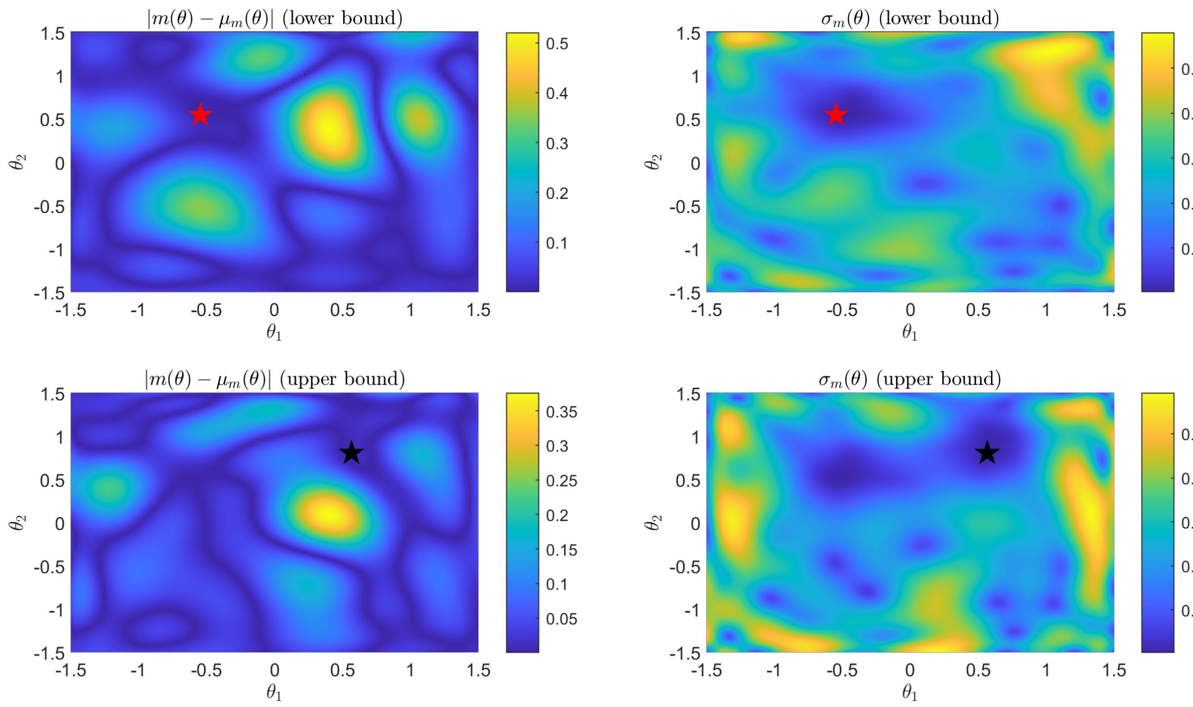

Fig. 6 The errors of the trained GPR models, where the left figures show the absolute error of the posterior means of $\hat{m}(\theta_1, \theta_2)$ when the lower and upper bounds are specified, respectively, and the right figures show the posterior STDs of the corresponding GPR models.

the extreme points computed by CABO are not consistent with the reference solution by NISS, this is caused by the insensitivity of the model response expectation to these parameters. For illustrating this, we estimate the first-order HDMR component functions of $m(\boldsymbol{\theta})$ by the NIPI method in Ref. [44] with 200 training samples generated by LHS design, and the posterior means and the posterior 99% confidence intervals are shown in Figure 7. It is seen that, among the seven epistemic parameters, θ_1 and θ_4 have the largest influence on the model response expectation, and for these two variables, the extreme points for both bounds are accurately specified by CABO, as shown in Table 4.

The results for the remaining three fixed discipline models are reported in Table 5, Table 6, and Table 7 respectively. It can be easily concluded that, for all these three models, both lower and upper bounds are accurately and robustly estimated by CABO, and the maximum number of model calls is 64, showing the high efficiency of the CABO algorithm. From the third column of each table, it can be seen that the specified extreme points for each dimension by CABO are not always consistent with the reference solutions, which is definitely caused by the unimportance of these parameters. For influential epistemic parameters, the specified extreme points are always identical. For simplicity, we don't give more details.

5.3 Thermal stress analysis of jet engine turbine blade

We consider a jet engine turbine blade model appears in the Matlab Partial Differential Equation (PDE) Toolbox. The mesh model is shown in the top left corner of Figure 8. This model concern the maximum von Mises stress and the maximum displacement of the tip when the pressure temperature loads are applied simultaneously. The pressure loads on the pressure and suction sides of the blade are denoted as p_1 and p_2 respectively. The material of the blade is assumed to be nickel-based alloy (NIMONIC 90). Four material parameters, i.e., Young's modulus E, the Poisson's ratio ν, the coefficient α of thermal expansion and the thermal conductivity κ, as well as the two pressure loads, are assumed to be random input variables following independent normal distributions with imprecise distribution parameters shown in Table 8. The temperature boundary on each face is assumed to be constant, and one can refer to the Matlab PDE toolbox for details. By fixing the six random input variables at the middle point of their mean parameters, the deterministic simulations are carried out for structural analysis with the consid-

Table 3 Description of input variables of the four fixed discipline models in the NASA UQ challenge.

Inputs	Category[2]	Uncertainty characterization models
p_1	III	Unimodal Beta, $\theta_1 = \mu_1 \in [0.6783, 0.7097]$, $\theta_2^2 = \sigma_1^2 \in [0.0387, 0.0397]$
p_2	II	$p_2 = 0.99$
p_3	I	Uniform, $[0, 1]$
p_4,p_5	III	Normal, $\theta_3 = \mu_4 \in [3.4493, 4.5812]$, $\theta_4 = \mu_5 \in [-1.5306, -0.9106]$, $\theta_5^2 = \sigma_4^2 \in [0.4190, 2.7209]$, $\theta_6^2 = \sigma_5^2 \in [0.2157, 0.6914]$, $\theta_7 = \rho \in [-0.4370, 0.7008]$
p_6	II	$p_6 = 0.5$
p_7	III	Beta, $\theta_8 = a \in [0.982, 3.537]$, $\theta_9 = b \in [0.619, 1.080]$
p_8	III	Beta, $\theta_{10} = a \in [7.450, 14.093]$, $\theta_{11} = b \in [4.285, 7.864]$
p_9	I	Uniform, $[0, 1]$
p_{10}	III	Beta, $\theta_{12} = a \in [1.520, 4.513]$, $\theta_{13} = b \in [1.536, 4.750]$
p_{11}	I	Uniform, $[0, 1]$
p_{12}	II	$p_{12} = 0.5$
p_{13}	III	Beta, $\theta_{14} = a \in [0.412, 0.737]$, $\theta_{15} = b \in [1.000, 2.068]$
p_{14}	III	Beta, $\theta_{16} = a \in [0.931, 2.169]$, $\theta_{17} = b \in [1.000, 2.407]$
p_{15}	III	Beta, $\theta_{18} = a \in [5.435, 7.095]$, $\theta_{19} = b \in [5.287, 6.945]$
p_{16}	II	$p_{16} = 0.5$
p_{17}	III	Beta, $\theta_{20} = a \in [1.060, 1.662]$, $\theta_{21} = b \in [1.000, 1.488]$
p_{18}	III	Beta, $\theta_{22} = a \in [1.000, 4.266]$, $\theta_{23} = b \in [0.553, 1.000]$
p_{19}	I	Uniform, $[0, 1]$
p_{20}	III	Beta, $\theta_{24} = a \in [7.530, 13.492]$, $\theta_{25} = b \in [4.711, 8.148]$

[2] Category I indicates the random variables with precise distribution parameters; Category II refers to the constant variables with precisely known values; Category III denotes the random variables characterized by p-box models.

eration of only pressure loads, and for thermal analysis applying only the temperature loads, as well as combined analysis with the consideration of both kinds of loads. The resultant von Mises stress nephograms are shown in Figure 8. The model output of interest in this application is the maximum von Mises stress, which happens in the constrained root, as shown by the last nephogram in Figure 8. This quantity is of great importance for predicting the fatigue life of the blade. The six normal random inputs variables as well as their imprecise distribution parameters, denoted as $\theta_1 \sim \theta_{12}$, are described in Table 8. Thus, the dimension of design parameters for BPO is twelve, and that for the BPI is six.

We initialize the CABO algorithm by setting the thresholds $\varDelta^{\mathrm{BPO}}$ and $\varDelta^{\mathrm{BPI}}$ as 2×10^{-3} and 0.05 respectively and initialize the training data set with 20 random samples generated by LHS design in the joint space. The prior mean of the GPR model is assumed to be constant. For BPO, the normalized stopping criteria in Eq. (15) is utilized. The results are then reported in Table 9. As can be seen, initialized by 20 training points, CABO adaptively produces 44 more samples for estimating the lower bound m_L before the stopping criteria is satisfied. The lower bound is estimated to be 667.961 MPa with the posterior COV being 0.210%, indicating the high accuracy. Then, with one more training data being added, the stopping criteria for estimating the upper bound m_U is reached, and the posterior mean and posterior COV of the upper bound are computed to be 1048.930 MPa and 0.657% respectively.

6 Conclusions

This paper has introduced the Bayesian numerical analysis for bounds estimation of response moments for computer simulators with the input random variables characterized by distributional p-box models, and the CABO algorithm has been developed by combining the BPI analysis and BPO analysis. It is shown that, by

Table 4 Results for the first model of the NASA UQ challenge.

Methods	Boundss	Extreme points ($\theta_1 \sim \theta_7$)	Mean estimates	COVs (%)	N_{call}
CABO	m_{L1}	(0.7097, 0.1993, 4.5812, -1.5306, 0.6473, 0.8315, 0.7008)	0.1991	0.46	$20 + 19 + 7 = 46$
	m_{U1}	(0.6783, 0.1967, 3.4493, -0.9106, 1.6495, 0.7441,-0.4370)	0.2408	0.37	
NISS	m_{L1}	(0.7097, 0.1967, 4.5812, -1.5306, 1.6495, 0.4644, 0.7008)	0.1967	0.83	10^5
	m_{U1}	(0.6783, 0.1993, 3.4493, -0.9106, 1.2197, 0.4644, -0.4370)	0.2415	0.64	
IMCS ($N = 10^2$)	m_{L1}	–	0.1801	7.52	5.334×10^4
	m_{U1}	–	0.2597	5.21	
IMCS ($N = 10^3$)	m_{L1}	–	0.1810	2.43	5.7078×10^5
	m_{U1}	–	0.2600	1.69	

Table 5 Results for the second model of the NASA UQ challenge.

Methods	Boundss	Extreme points ($\theta_8 \sim \theta_{13}$)	Mean estimates	COVs (%)	N_{call}
CABO	m_{L2}	(3.5370, 0.6190, 7.4500, 7.8640, 1.5200, 1.5360)	0.8979	0.10	$20 + 22 + 22 = 64$
	m_{U2}	(0.9820, 1.0800, 7.4500, 6.2616, 4.5130, 4.7500)	1.0399	0.09	
NISS	m_{L2}	(3.5370, 0.6190, 7.4500, 7.8640, 1.5200, 4.7500)	0.8829	2.08	10^5
	m_{U2}	(0.9820, 1.0800, 13.5998, 7.8640, 4.5130, 1.5360)	1.0595	3.72	
IMCS ($N = 10^2$)	m_{L2}	–	0.8886	1.15	3.321×10^4
	m_{U2}	–	1.0429	0.98	

training a GPR model in the joint space of aleatory and epistemic uncertainties, the induced marginal random field model in the aleatory space by fixing the epistemic parameters and the one in the epistemic space generated by integrating out the aleatory variables are both Gaussian. Based on this property, the introduced AEI learning function is found to be effective for searching the global optima in the epistemic space, and the PVC learning function is shown to be efficient for reducing the discretization error (quantified by posterior variance) of the numerical integration in the aleatory space. With these two learning functions as twin engines, CABO realizes the adaptive BPO in the epistemic space and the adaptive BPI in the aleatory space in a collaborative way. The outcomes of CABO include the lower and upper bounds of the model response moments as well as their posterior variances. The results of the illustrative example and the two engineering applications have demonstrated the efficiency and accuracy of the proposed methods. The CABO algorithm is developed in this work by taking the p-box model as an example, but it should be mentioned that it is also applicable to other distributional imprecise probability models such as the fuzzy probability model, by which it is expected to provide richer information for decision making [17].

One notes that the calculation of the maximum values of both AEI and PVC functions by the particle swarm algorithm may take some time, but both steps do not need g-function calls. For high-dimensional problems, as the Euclid distance used in the GPR kernels is less informative, the CABO algorithm may not be ap-

Table 6 Results for the third model of the NASA UQ challenge.

Methods	Boundss	Extreme points ($\theta_{14} \sim \theta_{19}$)	Mean estimates	COVs (%)	N_{call}
CABO	m_{L3}	(0.4120, 2.0680, 2.1690, 1.0000, 5.4350, 6.9450)	1.0522	0.04	$20 + 17 + 24 = 61$
	m_{U3}	(0.7353, 2.0680, 0.9310, 2.4070, 7.0950, 5.2870)	1.0771	0.08	
NISS	m_{L3}	(0.4120, 2.0680, 2.1690, 1.0000, 5.4350, 6.9450)	1.0534	0.26	2×10^5
	m_{U3}	(0.4120, 2.0680, 0.9310, 2.4070, 7.0950, 5.2870)	1.0782	0.21	
IMCS ($N = 10^2$)	m_{L3}	–	1.0489	0.25	3.663×10^4
	m_{U3}	–	1.0789	0.25	

Table 7 Results for the forth model of the NASA UQ challenge.

Methods	Boundss	Extreme points ($\theta_{20} \sim \theta_{25}$)	Mean estimates	COVs (%)	N_{call}
CABO	m_{L4}	(1.0600, 1.0000, 4.2660, 0.5530, 13.4920, 8.1480)	0.9399	0.08	$20 + 11 + 9 = 40$
	m_{U4}	(1.6620, 1.0000, 1.0000, 1.0000, 13.4920, 4.7110)	0.9844	0.09	
NISS	m_{L4}	(1.0600, 1.4880, 4.2660, 0.5530, 9.2329, 8.1480)	0.9358	0.22	2×10^5
	m_{U4}	(1.6620, 1.0000, 1.0000, 1.0000, 13.4920, 6.8395)	0.9791	0.79	
IMCS ($N = 10^2$)	m_{L4}	–	0.9345	0.15	3.330×10^4
	m_{U4}	–	0.9889	0.14	

plicable and some dimension reduction techniques need to be properly embedded (for example, see Ref. [53]). The high dimension of either $\boldsymbol{x}$ or $\boldsymbol{\theta}$ will also result in low efficiency of searching the global maxima of the two learning functions although they are both in closed form. If one call of the g-function takes much more time than the optimization search, this method is still competitive. However, if the optimization search is less efficient than calling the g-function, the method may partly lose its high efficiency. Thus, CABO in its current form is recommended for expensive computer simulators with relatively low dimensional aleatory variables and epistemic parameters. Improvement of CABO for high-dimensional problems will be specifically treated in our future work. Besides, the generalization of CABO for estimating the bounds of probabilities of rare events as well as the bounds of the distribution function of model outputs will also be conducted in the future.

Acknowledgements This work is supported by the National Natural Science Foundation of China (NSFC 51905430) and the Mobility Programme 2020 of Sino-German Center. The first author is also supported by the Alexander von Humboldt Foundation of Germany and the Top International University Visiting Program for Outstanding Young Scholars of Northwestern Polytechnical University. The third author extends his appreciation to the Institute for Risk and Reliability, Leibniz University, and the Humboldt award from the Alexander von Humboldt Foundation for providing the support to complete this paper. The authors would also like to thank Assist. Prof. Dr. Jingwen Song for the extensive and helpful discussions.

Conflict of interest

The authors declare that they have no conflict of interest.

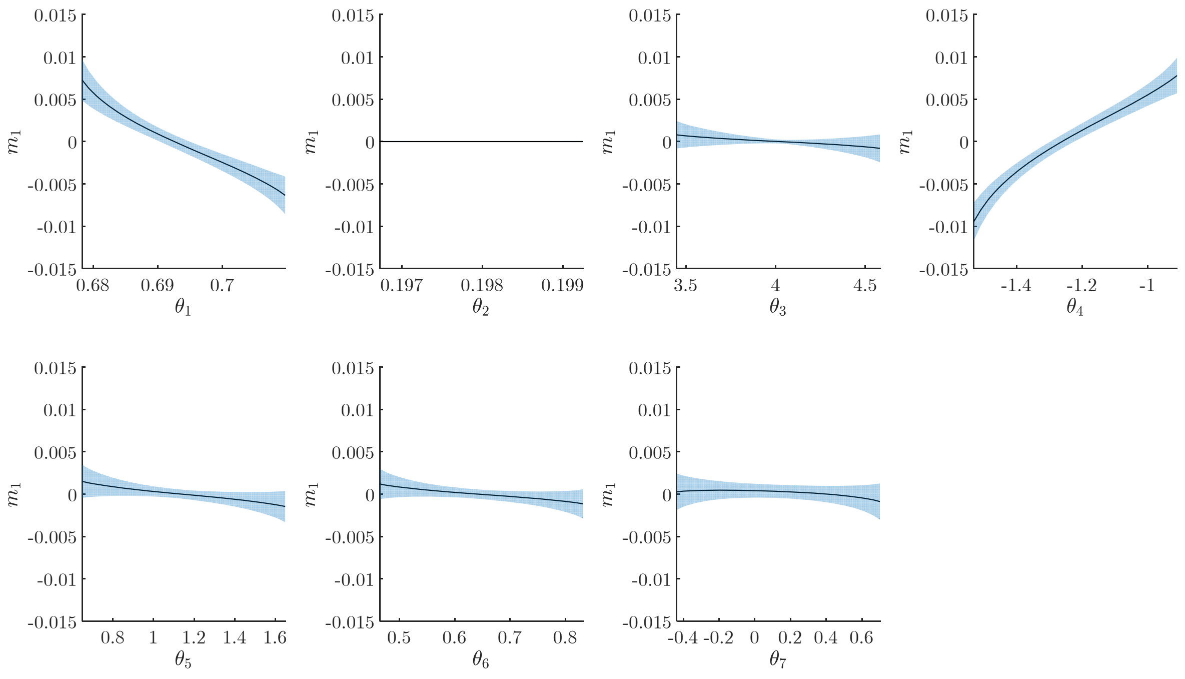

Fig. 7 The first-order HDMR component function of the model response expectation function of the first model of the NASA UQ challenge computed by NIPI with 200 training data, where the black solid lines indicate the posterior means and the blue boxes show the 99% posterior confidence intervals.

Table 8 Description of the normal random variables and their imprecise distribution parameters of the turbine blade model.

Variables	Description	Unit	Means	STDs
E	Young's modulus	Pa	$\theta_1 \in 227 \times 10^9 \times [0.9, 1.1]$	$\theta_7 \in 227 \times 10^9 \times [0.01,\ 0.05]$
α	coefficient of thermal expansion	1/K	$\theta_2 \in 12.7 \times 10^{-6} \times [0.9, 1.1]$	$\theta_8 \in 12.7 \times 10^{-6} \times [0.01, 0.05]$
ν	Poisson's ratio	–	$\theta_3 \in 0.27 \times [0.9,\ 1.1]$	$\theta_9 \in 0.27 \times [0.01,\ 0.05]$
κ	thermal conductivity	W/m/K	$\theta_4 \in 11.5 \times [0.9,\ 1.1]$	$\theta_{10} \in 11.5 \times [0.01,\ 0.05]$
p_1	Pressure load	Pa	$\theta_5 \in 5 \times 10^5 \times [0.9,\ 1.1]$	$\theta_{11} \in 5 \times 10^5 \times [0.01,\ 0.05]$
p_2	Pressure load	Pa	$\theta_6 \in 4.5 \times 10^5 \times [0.9,\ 1.1]$	$\theta_{12} \in 4.5 \times 10^5 \times [0.01,\ 0.05]$

References

1. S. Zein, A. Laurent, D. Dumas, Computational Mechanics **63**(6), 1083 (2019)
2. C. Ding, R.R. Deokar, X. Cui, G. Li, Y. Cai, K.K. Tamma, Computational Mechanics **63**(3), 521 (2019)
3. A. Der Kiureghian, O. Ditlevsen, Structural Safety **31**(2), 105 (2009)
4. M. Beer, S. Ferson, V. Kreinovich, Mechanical Systems and Signal Processing **37**(1-2), 4 (2013)
5. M. Faes, D. Moens, Archives of Computational Methods in Engineering pp. 1–39 (2019)
6. P. Wei, J. Song, S. Bi, M. Broggi, M. Beer, Z. Lu, Z. Yue, Mechanical Systems and Signal Processing **124**, 349 (2019)
7. I. Behmanesh, B. Moaveni, G. Lombaert, C. Papadimitriou, Mechanical Systems and Signal Processing **64**, 360 (2015)
8. Z. Hu, S. Mahadevan, D. Ao, Computational Mechanics **61**(1), 237 (2018)
9. P. Wei, Z. Lu, J. Song, Reliability Engineering & System Safety **142**, 399 (2015)
10. W. Gao, D. Wu, C. Song, F. Tin-Loi, X. Li, Finite Elements in Analysis and Design **47**(7), 643 (2011)
11. J.E. Hurtado, D.A. Alvarez, Computer Methods in Applied Mechanics and Engineering **225**, 74 (2012)
12. R. Kang, Q. Zhang, Z. Zeng, E. Zio, X. Li, Chinese Journal of Aeronautics **29**(3), 571 (2016)
13. J.E. Hurtado, Probabilistic Engineering Mechanics **32**, 80 (2013)
14. L.G. Crespo, S.P. Kenny, D.P. Giesy, in *16th AIAA Non-Deterministic Approaches Conference* (2014), p. 1347
15. L.G. Crespo, S.P. Kenny, D.P. Giesy, Mechanical Systems and Signal Processing **37**(1-2), 121 (2013)
16. H. Agarwal, J.E. Renaud, E.L. Preston, D. Padmanabhan, Reliability Engineering & System Safety **85**(1-3), 281 (2004)

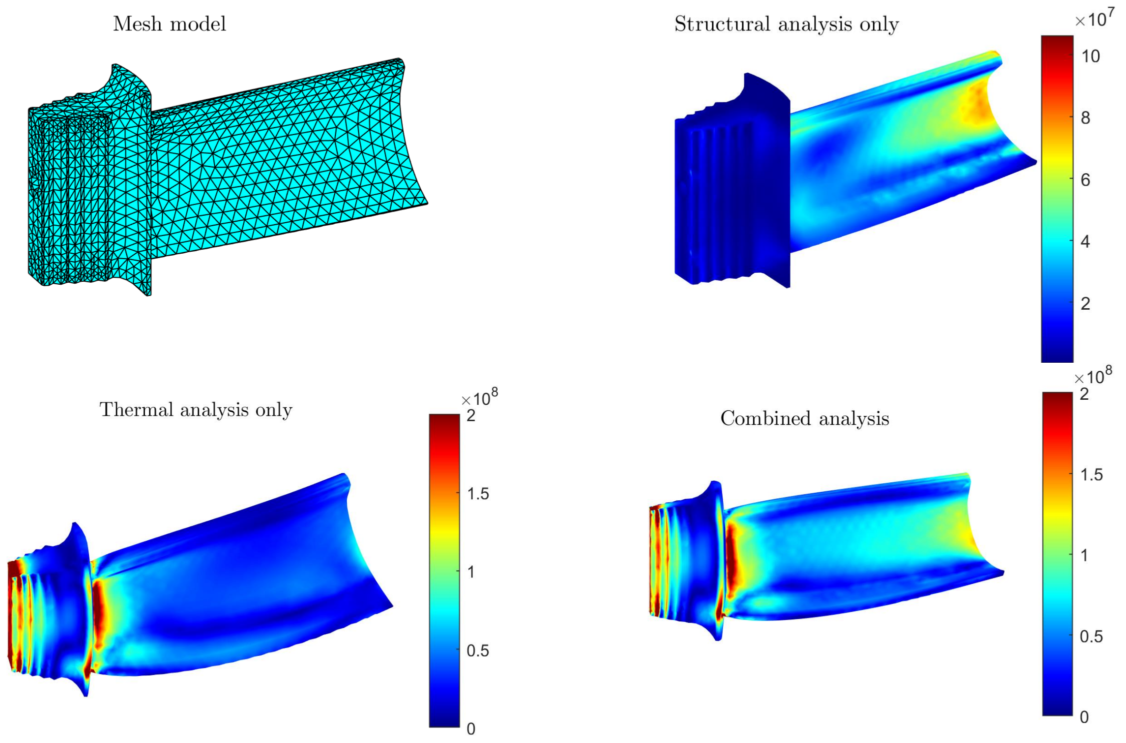

Fig. 8 Mesh model of the turbine blade (left) and analysis results, where the top right corner shows the von Mises tress nephogram of structural analysis only, and the lower-left corner displays the stress of thermal analysis only, and the last nephogram refers to that of the combined structural and thermal analysis.

17. M. Stein, M. Beer, V. Kreinovich, Information Sciences **245**, 96 (2013)
18. S. Sankararaman, S. Mahadevan, Mechanical Systems and Signal Processing **37**(1-2), 182 (2013)
19. J.C. Helton, J.D. Johnson, W.L. Oberkampf, C.J. Sallaberry, International Journal of General Systems **39**(6), 605 (2010)
20. I. Papaioannou, M. Daub, M. Drieschner, F. Duddeck, M. Ehre, L. Eichner, M. Eigel, M. Götz, W. Graf, L. Grasedyck, et al., GAMM-Mitteilungen **42**(2), e201900009 (2019)
21. M. de Angelis, E. Patelli, M. Beer, Structural Safety **52**, 170 (2015)
22. N. Pedroni, E. Zio, Journal of Aerospace Information Systems **12**(1), 73 (2015)
23. H. Zhang, R.L. Mullen, R.L. Muhanna, Structural Safety **32**(3), 183 (2010)
24. H. Zhang, H. Dai, M. Beer, W. Wang, Mechanical Systems and Signal Processing **37**(1-2), 137 (2013)
25. D.A. Alvarez, F. Uribe, J.E. Hurtado, Mechanical Systems and Signal Processing **100**, 782 (2018)
26. W. Gao, D. Wu, K. Gao, X. Chen, F. Tin-Loi, Applied Mathematical Modelling **55**, 49 (2018)
27. E. Patelli, D.A. Alvarez, M. Broggi, M.d. Angelis, Journal of Aerospace Information Systems **12**(1), 140 (2015)
28. R. Schöbi, B. Sudret, Probabilistic Engineering Mechanics **48**, 27 (2017)
29. R. Schöbi, B. Sudret, Journal of Computational Physics **339**, 307 (2017)
30. A. Sofi, G. Muscolino, F. Giunta, Probabilistic Engineering Mechanics **60**, 103020 (2020)
31. X. Yang, Y. Liu, Y. Zhang, Z. Yue, Acta Mechanica **226**(5), 1341 (2015)
32. P. Wei, Z. Lu, J. Song, AIAA Journal **52**(4), 867 (2014)
33. J. Zhang, M.D. Shields, Mechanical Systems and Signal Processing **98**, 465 (2018)
34. M.G. Faes, M.A. Valdebenito, D. Moens, M. Beer, Computers & Structures **239**, 106320 (2020)
35. J. Song, P. Wei, M. Valdebenito, S. Bi, M. Broggi, M. Beer, Z. Lei, Mechanical Systems and Signal Processing **134**, 106316 (2019)
36. P. Wei, J. Song, S. Bi, M. Broggi, M. Beer, Z. Lu, Z. Yue, Mechanical Systems and Signal Processing **126**, 227 (2019)
37. J. Song, M. Valdebenito, P. Wei, M. Beer, Z. Lu, Structural Safety **84**, 101936 (2020)
38. J. Song, P. Wei, M. Valdebenito, M. Beer, Computer Methods in Applied Mechanics and Engineering **372**, 113344 (2020)
39. J. Cockayne, C.J. Oates, T.J. Sullivan, M. Girolami, SIAM Review **61**(4), 756 (2019)
40. P. Hennig, C.J. Schuler, The Journal of Machine Learning Research **13**(1), 1809 (2012)
41. F.X. Briol, C.J. Oates, M. Girolami, M.A. Osborne, D. Sejdinovic, et al., Statistical Science **34**(1), 1 (2019)
42. P. Wei, X. Zhang, M. Beer, Computer Methods in Applied Mechanics and Engineering **365**, 113035 (2020)
43. O.A. Chkrebtii, D.A. Campbell, B. Calderhead, M.A. Girolami, et al., Bayesian Analysis **11**(4), 1239 (2016)
44. P. Wei, F. Liu, M. Valdebenito, M. Beer, Mechanical System and Signal Processing **149**, 107219 (2021)
45. R. Lebrun, A. Dutfoy, Probabilistic Engineering Mechanics **24**(4), 577 (2009)

Table 9 Results for the turbine blade model.

Lower bound m_L	Design points ($\theta_1 \sim \theta_6$)	$(204.3\times10^9, 11.430\times10^{-6}, 0.297, 10.350, 5.500\times10^5, 4.121\times10^5)$
	Design points ($\theta_7 \sim \theta_{12}$)	$(11.350\times10^9, 1.270\times10^{-7}, 0.0135, 0.5750, 5.00\times10^3, 4.500\times10^3)$
	Posterior mean (MPa)	667.961
	Posterior COV (%)	0.210
Upper bound m_U	Design points ($\theta_1 \sim \theta_6$)	$(249.7\times10^9, 13.970\times10^{-6}, 0.243, 12.650, 4.500\times10^5, 4.190\times10^5)$
	Design points ($\theta_7 \sim \theta_{12}$)	$(6.218\times10^9, 5.5353\times10^{-7}, 0.0027, 0.1150, 5.00\times10^3, 4.5\times10^3)$
	Posterior mean (MPa)	1048.930
	Posterior COV (%)	0.657
N_{call}		20+44+1=65

46. C.E. Rasmussen, C. Williams, MIT press **39**, 40 (2006)
47. B. Schölkopf, A.J. Smola, F. Bach, et al., *Learning with kernels: support vector machines, regularization, optimization, and beyond* (MIT press, 2002)
48. D. Duvenaud, (2014)
49. C.E. Rasmussen, Z. Ghahramani, In Advances in neural information processing systems, **15**, 505 (2003)
50. D. Huang, T.T. Allen, W.I. Notz, N. Zeng, Journal of Global Optimization **34**(3), 441 (2006)
51. B.J. Williams, T.J. Santner, W.I. Notz, Statistica Sinica pp. 1133–1152 (2000)
52. M.D. Shields, J. Zhang, Reliability Engineering & System Safety **148**, 96 (2016)
53. R. Tripathy, I. Bilionis, M. Gonzalez, Journal of Computational Physics **321**, 191 (2016)