Reliability analysis of non-deterministic models using stochastic emulators

Anderson V Pires, M Moustapha, Stefano Marelli, Bruno Sudret

ETH zürich

Reliability analysis for non-deterministic limit-states using stochastic emulators

A. Pires, M. Moustapha, S. Marelli and B. Sudret

Chair of Risk, Safety and Uncertainty Quantification
Stefano-Franscini-Platz 5
CH-8093 Zürich

Risk, Safety & Uncertainty Quantification

Data Sheet

Journal:

Report Ref.: RSUQ-2024-011A

Arxiv Ref.: https://arxiv.org/abs/2412.13731 [stat.CO] [stat.ML] [stat.ME]

DOI:

Date submitted: December 16,2024

Date accepted:

Reliability analysis of non-deterministic models using stochastic emulators

Anderson V. Pires [*1], M. Moustapha [†1], Stefano Marelli[‡1], and Bruno Sudret[§1]

[1] *Chair of Risk, Safety and Uncertainty Quantification, ETH Zürich, Switzerland*

December 18, 2024

Abstract

Reliability analysis is a sub-field of uncertainty quantification that assesses the probability of a system performing as intended under various uncertainties. Traditionally, this analysis relies on deterministic models, where experiments are repeatable, *i.e.* they produce consistent outputs for a given set of inputs. However, real-world systems often exhibit stochastic behavior, leading to non-repeatable outcomes. These so-called stochastic simulators produce different outputs each time the model is run, even with fixed inputs.

This paper formally introduces reliability analysis for stochastic models and addresses it by using suitable surrogate models to lower its typically high computational cost. Specifically, we focus on the recently introduced generalized lambda models and stochastic polynomial chaos expansions. These emulators are designed to learn the inherent randomness of the simulator's response and enable efficient uncertainty quantification at a much lower cost than traditional Monte Carlo simulation.

We validate our methodology through three case studies. First, using an analytical function with a closed-form solution, we demonstrate that the emulators converge to the correct solution. Second, we present results obtained from the surrogates using a toy example of a simply supported beam. Finally, we apply the emulators to perform reliability analysis on a realistic wind turbine case study, where only a dataset of simulation results is available.

Keywords: Structural reliability – Reliability analysis – Stochastic simulators – Surrogate models

[*]apires@ethz.ch
[†]moustapha@ibk.baug.ethz.ch
[‡]marelli@ibk.baug.ethz.ch
[§]sudret@ethz.ch

1 Introduction

Structural reliability analysis aims to determine the probability that uncertainties in a real-world system lead to the failure of the considered system. This analysis typically utilizes computational models that serve as a virtual representation of the actual system. Although these models are often complex and expensive to evaluate, they are widely used because they allow for extensive testing of various scenarios without the need for physical prototypes or experimental setups.

The conditions applied to these virtual models are controlled by a set of input parameters described by a random vector $\boldsymbol{X} \in \mathcal{D}_{\boldsymbol{X}} \subset \mathbb{R}^{M_{\boldsymbol{X}}}$, which is characterized by the joint probability density function (PDF) $f_{\boldsymbol{X}}$. The state of the system, *i.e.*, whether it fails to meet its performance requirements or not, is determined by the limit-state function $g(\boldsymbol{x})$, typically associated with the computational model and prescribed conditions on its operation. Failure is associated to non-positive values of the limit-state function, *i.e.*, $g(\boldsymbol{x}) \leqslant 0$. By extension, the failure domain is defined as $\mathcal{D}_f = \{\boldsymbol{x} : g(\boldsymbol{x}) \leqslant 0\}$, and the boundary of this domain, where $g(\boldsymbol{x}) = 0$, defines the *limit-state surface*. This framework enables the definition of the probability of failure, P_f, as follows (Ditlevsen and Madsen, 1996; Lemaire, 2009; Melchers and Beck, 2018):

$$P_f = \mathbb{P}(g(\boldsymbol{X}) \leqslant 0) = \int_{\mathcal{D}_f} f_{\boldsymbol{X}}(\boldsymbol{x})\, d\boldsymbol{x}. \tag{1}$$

In this context, it is assumed that all sources of uncertainties are accounted for by the random variables and the limit-state function is deterministic, *i.e.*, evaluating it multiple times with the same input vector $\boldsymbol{x}_0$ yields the same response $g(\boldsymbol{x}_0)$. However, there are cases where this assumption does not hold.

In contrast to deterministic models, *stochastic simulators* yield different responses each time they are run with the same input parameters. This variability arises from additional sources of uncertainty that affects the computational model but are not explicitly included in the input parameters because it is either impossible or not desired. Common examples of stochastic simulators include epidemiological models, where disease transmission and recovery are treated as random events with specific probabilities (Britton, 2010; Binois et al., 2018; Zhu and Sudret, 2021). Another example is found in financial modeling, where stock prices are modeled using stochastic processes, such as the geometric Brownian motion, to capture market fluctuations and investor behavior (Heston, 1993; McNeil et al., 2015; Zhu and Sudret, 2023; Lüthen et al., 2023). Aero-servo-elastic simulations, which are commonly used in wind turbine engineering, present significant latent stochasticity because they rely on stochastic turbulent wind fields (Jonkman, 2009). Agent-based models (Wilensky, 2015) constitute another broad class of stochastic simulators widely adopted in engineering and applied sciences.

As a result of this implicit randomness, often referred to as *latent variability* or *noise*, stochastic simulators can produce both failed and non-failed outputs for different runs with the same set of

input parameters. In this case, failure does not depend uniquely on the input parameter $\boldsymbol{x}$, but also on the realization of the latent variables of the model.

Research on reliability analysis for problems involving stochastic simulators is still relatively sparse. Choe et al. (2015) first formalized the problem of performing reliability analysis in the presence of stochastic simulators, by explicitly accounting for latent variability in an extension of classical importance sampling (Melchers, 1989). A sequence of works have then followed, addressing different aspects of this complex problem: Choe et al. (2017) studied the asymptotic properties of the stochastic importance sampling estimator, proposing an asymptotically valid confidence interval. Cao and Choe (2019) introduced instead a cross-entropy-based solution scheme for stochastic importance sampling. Pan et al. (2020) introduced an adaptive stochastic importance sampling approach, while Li et al. (2021) proposed a nonparametric importance sampling method focused on a wind turbine case study. The latter quantifies the contributions of each environmental factor and their interactions while avoiding computational issues with data sparsity in rare event simulation. From another perspective, Zheng et al. (2022) utilized quantile regression techniques to develop a feedforward neural network surrogate model for stochastic systems. Upon training, they conducted reliability analysis via Monte Carlo simulation (MCS) by sampling across input variables and quantile levels. Hao et al. (2021) extended the well-known stochastic Kriging method (Ankenman et al., 2010) to handle unknown heteroskedastic Gaussian noise and perform reliability analysis. Recently, Gramstad et al. (2020) proposed the use of Gaussian process modeling, extreme value distributions and active learning techniques, to emulate parametrically the conditional response of a stochastic simulator, with the goal of assessing the short- and long- term reliability of wind turbines.

In this paper, we propose instead the use of a class of stochastic emulators recently introduced by Zhu and Sudret (2021, 2023), as a natural tool for the model-agnostic solution of reliability analysis in the presence of stochastic models. These emulators are cheap-to-evaluate parametric models that can learn the behavior of stochastic simulators, including their inherent stochasticity, with reduced computational effort, thus enabling reliability analysis applications. We validate the proposed methodology on three case studies. The first is a standard reliability analysis problem modified with the introduction of latent variables in the limit-state function. The second study applies our approach to a simply supported beam, a classic problem in structural reliability. Analytical solutions are available in both cases, hence allowing us to easily validate our proposed methodology. The third case study focuses instead on a real-world wind turbine example, where only operational data is available.

This paper is organized as follows: Section 2 introduces the formal framework for reliability analysis in the context of stochastic simulators. Section 3 presents the stochastic emulators employed in our methodology, namely generalized lambda models and stochastic polynomial chaos expansions. Section 4 introduces reliability analysis using these emulators. Section 5 showcases the results obtained with our methodology when applied to the previously mentioned case studies. Finally, Section 6 presents our conclusions and provides an outlook on future

research on the topic.

2 Problem statement

2.1 Stochastic Simulators

A stochastic simulator $\mathcal{M}_S$ can be formally defined by the following mapping:

$$\begin{aligned} \mathcal{M}_S : D_{\boldsymbol{X}} \times \Omega &\rightarrow \mathbb{R}, \\ (\boldsymbol{x}, \omega) &\mapsto \mathcal{M}_S(\boldsymbol{x}, \omega), \end{aligned} \tag{2}$$

where $\boldsymbol{x} \in D_{\boldsymbol{X}} \subset \mathbb{R}^{M_{\boldsymbol{X}}}$ is the $M_{\boldsymbol{X}}$-dimensional vector of input parameters and ω is a random event in a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ that represents the inherent stochasticity. In practice, the latter is due to latent random variables $\boldsymbol{Z}(\omega)$ (defined over the same probability space, with support $D_{\boldsymbol{Z}} \subset \mathbb{R}^{M_{\boldsymbol{Z}}}$), which are internal parameters of the simulator that are not explicitly considered as input variables. In other words, a stochastic simulator is built from a *deterministic* model $\mathcal{M}$ as follows:

$$\mathcal{M}_S(\boldsymbol{x}, \omega) = \mathcal{M}(\boldsymbol{x}, \boldsymbol{Z}(\omega)). \tag{3}$$

For a given input vector $\boldsymbol{x}_0$, each run of the simulator corresponds to a different realization of the latent variables $\boldsymbol{Z}$, say $\boldsymbol{z}_0 \equiv \boldsymbol{Z}(\omega_0)$ and provides one particular realization of the output random variable $Y_{x_0} \equiv \mathcal{M}_S(x_0, \cdot)$. To characterize the distribution of the latter, *replications* should be generated, *i.e.* multiple runs of $\mathcal{M}_S$ with the same input vector $\boldsymbol{x}_0$ and different realizations $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_R\}$ of the latent variables. We will refer to the probability density function of Y_{x_0} as the *conditional distribution* of the response given $\boldsymbol{x}_0$.

In contrast, fixing the realization $\boldsymbol{z}_0$ and running the model for different inputs provides a *trajectory* of the simulator, *i.e.* a (standard) function $\mathcal{M}_S(\cdot, \omega_0) : D_{\boldsymbol{X}} \rightarrow \mathbb{R}$. This function can be obtained in practice by resetting the random seed of the random number generator which generates the realizations of $\boldsymbol{Z}$ for each $\boldsymbol{x}$. Statistical summaries of the output of a stochastic simulator, such as the mean value, variance or quantiles are also standard functions of the input parameters.

To illustrate the above definitions in a practical application, let us consider an aero-elastic simulator used to design wind turbines (see detailed example in Section 5.3). A wind field is characterized by macroscopic parameters, such as the average velocity over 10 minutes U, the turbulence intensity TI and the shear exponent α that describes the variation of wind speed with altitude. These three parameters correspond to the input vector $\boldsymbol{x}$. Then a three-dimensional spatio-temporal wind realization is generated over a time window $[0, T]$ (typicaly $T = 10$ minutes) using the power spectral density of the wind field and thousands of Gaussian random variables, which in turn form the latent vector $\boldsymbol{Z}$. For a fixed so-called climate $\boldsymbol{x}_0 = \{u_0, TI_0, \alpha_0\}$, an infinite number of 10-minute wind fields could be generated, which provide different structural responses of the wind turbine, *e.g.* maximal tip blade displacement, blade root bending moments,

etc. The latter are random variables whose distribution depend on $\boldsymbol{x}_0$, that may be further used in a reliability analysis, whose general problem statement is now introduced.

2.2 Reliability analysis on stochastic simulators

Following the formalism of Choe et al. (2015), the probability of failure associated with a stochastic limit-state function g_s is defined as:

$$P_f = \mathbb{P}\left(g_s\left(\boldsymbol{X}, \omega\right) \leqslant 0\right). \tag{4}$$

Introducing the underlying deterministic function g and the latent variables $\boldsymbol{Z}$, Eq. (4) can be written as:

$$P_f = \mathbb{P}\left(g\left(\boldsymbol{X}, \boldsymbol{Z}\right) \leqslant 0\right). \tag{5}$$

Assuming that the random variables $\boldsymbol{X}$ and $\boldsymbol{Z}$ are independent and that the failure domain is defined in the joint $(\boldsymbol{X}, \boldsymbol{Z})$ space, *i.e.*, $\mathcal{D}_f = \{(\boldsymbol{x}, \boldsymbol{z}) : g\left(\boldsymbol{x}, \boldsymbol{z}\right) \leqslant 0\}$, this equation can be expanded into:

$$\begin{aligned} P_f &= \int\int_{\{(\boldsymbol{x},\boldsymbol{z}):g(\boldsymbol{x},\boldsymbol{z})\leqslant 0\}} f_{\boldsymbol{X}}\left(\boldsymbol{x}\right) f_{\boldsymbol{Z}}\left(\boldsymbol{z}\right) \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{z} \\ &= \int_{\mathbb{R}^{M_{\boldsymbol{X}}}} \int_{\mathbb{R}^{M_{\boldsymbol{Z}}}} \mathbb{1}_{\mathcal{D}_f}(\boldsymbol{x}, \boldsymbol{z}) f_{\boldsymbol{X}}\left(\boldsymbol{x}\right) f_{\boldsymbol{Z}}\left(\boldsymbol{z}\right) \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{z}, \end{aligned} \tag{6}$$

where $\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{x}, \boldsymbol{z})$ is the indicator function of the failure domain, equal to 1 if the system fails, *i.e.*, $g\left(\boldsymbol{x}, \boldsymbol{z}\right) \leqslant 0$, and 0 otherwise. This definition integrates the uncertainty arising from both the input parameters and latent variables.

Furthermore, by manipulating Eq. (6), we can derive equivalent definitions of P_f based on the two perspectives discussed in Sec. 2.1. For a given $\boldsymbol{x}_0$, the conditional failure probability is defined as:

$$\mathbb{P}_{\boldsymbol{Z}}\left(g\left(\boldsymbol{X}, \boldsymbol{Z}\right) \leqslant 0 \mid \boldsymbol{X} = \boldsymbol{x_0}\right) = \int_{\mathbb{R}^{M_{\boldsymbol{Z}}}} \mathbb{1}_{\mathcal{D}_f}(\boldsymbol{x_0}, \boldsymbol{z}) f_{\boldsymbol{Z}}\left(\boldsymbol{z}\right) \mathrm{d}\boldsymbol{z}. \tag{7}$$

For different values of $\boldsymbol{x}$, we then obtain a deterministic *conditional failure probability function*, which we denote by $s\left(\boldsymbol{x}\right)$:

$$s(\boldsymbol{x}) = \mathbb{P}_{\boldsymbol{Z}}\left(g\left(\boldsymbol{X}, \boldsymbol{Z}\right) \leqslant 0 \mid \boldsymbol{X} = \boldsymbol{x}\right). \tag{8}$$

This function reflects the probability of failing due to the internal stochasticity of the limit-state function alone.

By reformulating Eq. (6), we can express P_f as a function of $s\left(\boldsymbol{x}\right)$:

$$\begin{aligned} P_f &= \int_{\mathbb{R}^{M_{\boldsymbol{X}}}} \left(\int_{\mathbb{R}^{M_{\boldsymbol{Z}}}} \mathbb{1}_{\mathcal{D}_f}(\boldsymbol{x}, \boldsymbol{z}) f_{\boldsymbol{Z}}\left(\boldsymbol{z}\right) \mathrm{d}\boldsymbol{z} \right) f_{\boldsymbol{X}}\left(\boldsymbol{x}\right) \mathrm{d}\boldsymbol{x} \\ &= \int_{\mathbb{R}^{M_{\boldsymbol{X}}}} s\left(\boldsymbol{x}\right) f_{\boldsymbol{X}}\left(\boldsymbol{x}\right) \mathrm{d}\boldsymbol{x} \\ &\equiv \mathbb{E}_{\boldsymbol{X}}\left[s(\boldsymbol{X})\right]. \end{aligned} \tag{9}$$

This allows us to decompose the estimation of the probability of failure into a two-step process, namely the characterization of the function s (with semi-analytical equations), and the computation of its expected value (see details in Sec. 4).

Alternatively, when the latent stochasticity is fixed, we can express P_f as:

$$P_f = \mathbb{E}_{\boldsymbol{Z}} \left[\mathbb{P}_{\boldsymbol{X}} \left(g\left(\boldsymbol{X}, \boldsymbol{Z}\right) \leqslant 0 \mid \boldsymbol{Z} = \boldsymbol{z} \right) \right]. \tag{10}$$

The inner term in Eq. (10) represents the probability of failure due to uncertainty in the input parameter for a fixed realization of the stochasticity. In other words, P_f can be computed as the expected value (over many random seeds) of the probabilities of failure associated to each resulting trajectory.

2.3 Estimating P_f

Similarly to the deterministic case, solving Eq. (4) analytically is generally not feasible. An alternative approach is to estimate P_f using Monte Carlo simulation. This method provides an unbiased estimation of P_f because each time the limit-state function is evaluated for a given $\boldsymbol{x}$, a realization of ω, that is of $\boldsymbol{Z}$, is also implicitly used. In this sense, MCS is performed on both $\boldsymbol{X}$ and $\boldsymbol{Z}$, however only $\boldsymbol{X}$ is explicitly sampled, while $\boldsymbol{Z}$ is sampled implicitly through a mechanism internal to the stochastic simulator for each evaluation. Another important aspect is that MCS does not suffer from the curse of dimensionality. Consequently, the presence of stochasticity does not increase the complexity of the problem, and the number of limit-state function evaluations needed only depends on the desired accuracy when estimating P_f, just as in the deterministic case.

Monte Carlo simulation is, however, often intractable in practice as it requires numerous evaluations of the possibly expensive-to-evaluate limit-state function. Variance-reduction techniques have been developed to address this issue, *e.g.*, importance sampling (Melchers, 1989), subset simulation (Au and Beck, 2001; Papaioannou et al., 2016), line sampling (Koutsourelakis et al., 2004; De Angelis et al., 2014), or cross-entropy importance sampling (Kurtz and Song, 2013; Geyer et al., 2019). These methods aim to provide more accurate estimates of the probability of failure at a lower computational cost compared to crude MCS. While well-established for deterministic simulators, stochastic-aware variance-reduction techniques are not yet widespread, with the exception of the family of stochastic importance sampling mentioned earlier (Choe et al., 2015, 2016, 2017).

Another commonly used approach to reduce the costs associated with MCS is the introduction of surrogate models. These are inexpensive mathematical approximations that mimic the behavior of complex, costly-to-evaluate functions. These models are trained on an *experimental design*, which is a set of points obtained from full-scale simulations. Many surrogate models are available and well-established for deterministic functions. In the context of reliability analysis (Teixeira et al., 2021; Moustapha et al., 2022), Kriging is arguably the most widely adopted surrogate

model. Other methods, such as polynomial chaos expansion (Marelli and Sudret, 2018) or support vector machines (Bourinet et al., 2011) have also been employed. However, these approaches cannot be directly applied to stochastic simulators, as they do not account for latent stochasticity.

With the growing interest in stochastic simulators, traditional deterministic methods have been extended to handle stochastic cases. For example, Ankenman et al. (2010) extended the traditional Kriging method to stochastic systems. However, this approach assumes that the conditional distributions are Gaussian, which is often not realistic. Generalized linear models and generalized additive models, introduced by McCullagh and Nelder (1989) and Hastie and Tibshirani (1990), respectively, relax this Gaussian assumption by allowing the conditional distributions to belong to a broader exponential family. The primary limitation of these approaches is that they focus on approximating only the mean function of the simulator.

In an effort to model both the mean and variance, Iooss and Ribatet (2009) proposed a joint modeling approach, which was further refined by Marrel et al. (2012). Despite these advancements, such methods remain limited to capturing the first two moments. To approximate the full PDF of the response of the simulators, Moutoussamy et al. (2015) introduced the use of kernel density estimation. For the specific purpose of performing reliability analysis on extreme value-type problems, typical of wind turbine design, Gramstad et al. (2020) combine instead Kriging and active learning to emulate the conditional distribution of maximum loads, as a function of statistical summaries of short and long term environmental conditions. More recently, Zhu and Sudret (2020, 2021) proposed the generalized lambda models (GLaM), a method that uses generalized lambda distributions to approximate conditional distributions. Similarly, Zhu and Sudret (2023) introduced stochastic polynomial chaos expansions (SPCE), which map a latent random variable to target conditional distributions through polynomial chaos expansions. Finally, Lüthen et al. (2023) developed spectral surrogate models that rely on Karhunen-Loève expansions for problems where the random seed can be controlled within the computational model.

These newly introduced emulators address some of the limitations of the other existing methods. Specifically, Zhu and Sudret (2021, 2023) focus on emulating the PDF of conditional distributions, with minimal assumptions regarding the shape of those distributions. In contrast, Lüthen et al. (2023) propose an emulator specifically targeted to problems where trajectories (with controlled random seeds) are available. In this paper, we leverage the methods developed by Zhu and Sudret (2021, 2023) to conduct reliability analysis on stochastic simulators.

3 Stochastic emulators

3.1 Generalized lambda models

Generalized lambda models (GLaM) are a type of surrogates designed for stochastic simulators. The primary objective of GLaM is to mimic the probability density function of the random

variable $Y_{\boldsymbol{x}} \equiv Y \mid \boldsymbol{X} = \boldsymbol{x}$. This is done by fitting a *generalized lambda distribution* (GLD) to the conditional response $Y_{\boldsymbol{x}}$. GLDs can approximate most common unimodal distributions, such as Gaussian, Students' t-, or Weibull distributions. They are defined by their quantile function $Q(u)$, which is a monotonically increasing function defined on $u \in [0, 1]$. GLaM specifically relies on the Freimer-Kollia-Mudholkar-Lin GLD family (Freimer et al., 1988), which reads:

$$Q(u; \boldsymbol{\lambda}) = \lambda_1 + \frac{1}{\lambda_2}\left(\frac{u^{\lambda_3} - 1}{\lambda_3} - \frac{(1-u)^{\lambda_4} - 1}{\lambda_4}\right), \tag{11}$$

where λ_1 represents the location parameter, $\lambda_2 > 0$ is the scale parameter, λ_3, and λ_4 are the shape parameters. For a deeper explanation of how these parameters affect the resulting GLD, the reader is referred to Zhu and Sudret (2020, 2021).

The rationale behind the GLaM emulator is that for every $\boldsymbol{x}$, there exists a set of parameters $\{\lambda_i(\boldsymbol{x}), 1, \ldots, 4\}$ that can be used to generate a distribution that approximates the response of the original simulator, *i.e.*:

$$Y_x \overset{\mathrm{d}}{\approx} \tilde{Y}_x \sim \mathrm{GLD}\left(\lambda_1(\boldsymbol{x}), \lambda_2(\boldsymbol{x}), \lambda_3(\boldsymbol{x}), \lambda_4(\boldsymbol{x})\right). \tag{12}$$

The symbol $\overset{\mathrm{d}}{\approx}$ indicates that the two sides are approximately equal in distribution.

Each component of $\boldsymbol{\lambda}(\boldsymbol{x})$ consists of a deterministic function and, as a consequence, can be modeled by polynomial chaos expansions (Lüthen et al., 2021). Since $\lambda_2(\boldsymbol{x})$ must be positive, its associated expansion is constructed in the logarithmic space. Thus the polynomial chaos expansions read:

$$\begin{aligned} \lambda_l(\boldsymbol{x}) &\approx \lambda_l^{\mathrm{PC}}(\boldsymbol{x}; \boldsymbol{c}) = \sum_{\boldsymbol{\alpha} \in \mathcal{A}_l} c_{l,\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\boldsymbol{x}), \quad l = 1, 3, 4, \\ \lambda_2(\boldsymbol{x}) &\approx \lambda_2^{\mathrm{PC}}(\boldsymbol{x}; \boldsymbol{c}) = \exp\left(\sum_{\boldsymbol{\alpha} \in \mathcal{A}_2} c_{2,\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\boldsymbol{x})\right), \end{aligned} \tag{13}$$

where $\psi_{\boldsymbol{\alpha}}(\boldsymbol{x})$ are a set of polynomials, orthonormal with respect to the input distribution $f_{\boldsymbol{X}}$, $\boldsymbol{\alpha}$ are multi-indices, which identify the degree of the multivariate polynomials, $\boldsymbol{c} = \{c_{l,\boldsymbol{\alpha}} : l = 1, \ldots, 4, \boldsymbol{\alpha} \in \mathcal{A}_l\}$ denote the corresponding coefficients and $\mathcal{A} = \{\mathcal{A}_l : l = 1, \ldots, 4\}$ represents the truncation set, which defines the basis functions. For a more detailed explanation on the construction of PCEs and their properties, interested readers are referred to Lüthen et al. (2021).

There are two approaches to constructing the GLaM emulator. The first approach, introduced by Zhu and Sudret (2020), relies on replications, *i.e.*, for each point $\boldsymbol{x}^{(i)}$, there are R evaluations of the simulator: $\mathcal{Y}^{(i)} = \left\{y^{(i,1)}, y^{(i,2)}, \ldots, y^{(i,R)}\right\}$. The training then involves two steps. First, the local GLD parameters $\boldsymbol{\lambda}\left(\boldsymbol{x}^{(i)}\right)$ for each $\boldsymbol{x}^{(i)}$ are inferred from the dataset $\mathcal{Y}^{(i)}$, using either the method of moments or maximum likelihood estimation. In the second step, PCE approximation of the lambda functions, *i.e.* the mapping $\boldsymbol{x} \mapsto \boldsymbol{\lambda}(\boldsymbol{x})$, are constructed using the experimental design $\left\{\left(\boldsymbol{x}^{(i)}, \boldsymbol{\lambda}\left(\boldsymbol{x}^{(i)}\right)\right), i = 1, \ldots, N\right\}$.

A second, generally more efficient approach was introduced by Zhu and Sudret (2021) and eliminates the need for replications. Given an experimental design $\mathcal{X} = \left\{\boldsymbol{x}^{(1)} \ldots, \boldsymbol{x}^{(N)}\right\}$ and its

associated model responses $\mathcal{Y} = \left\{y^{(1)} \ldots, y^{(N)}\right\}$, the associated maximum likelihood optimization problem is directly cast as a function of the PCE coefficients $\boldsymbol{c}$:

$$\widehat{c} = \arg\max_{\boldsymbol{c}\in\mathcal{C}} \mathrm{L}(\boldsymbol{c}), \tag{14}$$

where

$$\mathrm{L}(\boldsymbol{c}) = \sum_{i=1}^{N} \log\left(f_Y\left(y^{(i)}; \boldsymbol{\lambda}^{\mathrm{PC}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{c}\right)\right)\right). \tag{15}$$

Here, the set $\mathcal{C}$ defines the search space for $\boldsymbol{c}$ and f_Y represents the PDF of the generalized lambda distribution and reads:

$$f_Y(y;\boldsymbol{\lambda}) = \frac{1}{Q'(u;\boldsymbol{\lambda})} = \frac{\lambda_2}{u^{\lambda_3-1} + (1-u)^{\lambda_4-1}} \qquad \text{with } u = Q^{-1}(y;\boldsymbol{\lambda}). \tag{16}$$

A closed-form solution of Q^{-1} is generally not available, therefore the PDF is obtained numerically. For further details on the estimator in Eq. (15), the reader is referred to Zhu and Sudret (2020, 2021).

A detailed analysis of the impact of replications on experimental design is provided in Zhu and Sudret (2021). A straightforward advantage of not using replications is that the available computational budget can be allocated to a more space-filling experimental design, ensuring information is distributed across the entire domain rather than concentrated at replicated points. Furthermore, it eliminates the constraints associated to predefined experimental designs, allowing datasets collected for other purposes to be used to train the emulator.

3.2 Stochastic polynomial chaos expansion

Stochastic Polynomial Chaos Expansion (SPCE), introduced by Zhu and Sudret (2023), is another stochastic emulator that enables modeling the output distribution of $Y_{\boldsymbol{x}}$. An artificial latent variable Z is introduced to represent the stochasticity of the simulator within the polynomial chaos expansions framework. This approach capitalizes on the probability integral transform (PIT), a statistical technique that allows transforming a continuous random variable into another, enabling the mapping of a known variable Z to the desired output distribution. Typically, Z is chosen to follow a standard uniform or Gaussian distribution.

Let the cumulative distribution function (CDF) associated to $Y_{\boldsymbol{x}}$ be denoted by $F_{Y|\boldsymbol{X}}(y \mid \boldsymbol{x})$. The PIT allows us to express a relationship between $Y_{\boldsymbol{x}}$ and an artificial latent variable Z, as follows:

$$Y_{\boldsymbol{x}} \overset{\mathrm{d}}{=} F^{-1}_{Y|\boldsymbol{X}}\left(F_Z(Z) \mid \boldsymbol{x}\right), \tag{17}$$

where F_Z is the CDF of the latent variable Z. The symbol $\overset{\mathrm{d}}{=}$ indicates that the two sides are equal in distribution.

Eq. (17) implies that $Y_{\boldsymbol{x}}$ can be represented as a deterministic function of both the input $\boldsymbol{x}$ and the latent variable Z. This deterministic mapping is then captured using a PCE in the

$(\boldsymbol{X}, Z)$ space:

$$F^{-1}_{Y|\boldsymbol{X}}\left(F_Z(Z) \mid \boldsymbol{X} = \boldsymbol{x}\right) \overset{\mathrm{d}}{=} \sum_{\boldsymbol{\alpha} \in \mathbb{N}^{M_{\boldsymbol{X}}+1}} c_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\boldsymbol{x}, Z). \tag{18}$$

For a given $\boldsymbol{x}$, the expansion above is a function of the latent variable Z, making the response $Y_{\boldsymbol{x}}$ a random variable. Considering a truncation scheme $\mathcal{A}$, we have:

$$Y_x \overset{\mathrm{d}}{\approx} \tilde{Y}_x = \sum_{\alpha \in \mathcal{A}} c_\alpha \psi_{\boldsymbol{\alpha}}(\boldsymbol{x}, Z). \tag{19}$$

Relying on Eq. (19) as-is may lead to issues such as singularities in the probability density functions. To address this, Zhu and Sudret (2023) introduced an additive noise component $\epsilon \sim \mathcal{N}(0, \sigma^2)$, which acts, in practice, as a regularization term. The resulting SPCE emulator is then expressed as:

$$Y_x \overset{\mathrm{d}}{\approx} \tilde{Y}_x = \sum_{\alpha \in \mathcal{A}} c_\alpha \psi_\alpha(\boldsymbol{x}, Z) + \epsilon. \tag{20}$$

The PDF of $\tilde{Y}_{\boldsymbol{x}}$ results from the convolution of that of the PCE with the Gaussian noise. Assuming the noise variance σ^2 is known, the corresponding PDF can be cast as:

$$f_{\tilde{Y}_{\boldsymbol{x}}}(y) = \int_{\mathcal{D}_Z} \frac{1}{\sigma} \varphi\left(\frac{y - \sum_{\boldsymbol{\alpha} \in \mathcal{A}} c_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\boldsymbol{x}, z)}{\sigma}\right) f_Z(z) \mathrm{d}z, \tag{21}$$

where φ represents the standard normal PDF.

Given a replication-free experimental design, as defined in Sec. 3.1, the PCE coefficients $\boldsymbol{c}$ can be computed using maximum likelihood estimation. Because the noise variance is, in general, not known in advance, tuning this parameter is still needed. Zhu and Sudret (2023) proposes using cross-validation for this task. A more detailed explanation regarding the fitting process can be found in Zhu and Sudret (2023).

SPCE does not require replications in the experimental design, making all the advantages discussed in Sec. 3.1 also applicable. Moreover, unlike GLaM, SPCE does not impose a specific parametric form on the simulator response. Instead, it offers a flexible model capable of approximating a wide range of distributions, including bimodal ones. Another notable advantage of SPCE is that it represents stochasticity using a single latent variable. Regardless of the number of latent variables $\boldsymbol{Z}$ used to represent the inherent stochasticity of the model (See Eq. (3)), the emulator consistently reduces it to a single latent variable.

4 Proposed methodology for reliability analysis

4.1 Monte Carlo estimators of the probability of failure

Once the emulators are trained, the probability of failure P_f can be estimated using MCS. Since P_f can be defined in multiple ways, as in Eqs. (6), (9) and (10), it is of interest to determine

which definition yields the estimator with the minimum variance. For this purpose, we compare the variances of the MCS estimators derived from Eqs. (6) and (9). This comparison assumes that $s(\boldsymbol{x})$ is known, which is generally not the case. However, in the context of the emulators under consideration, such a function can be inexpensively computed, as described in Sec. 4.2 and Sec. 4.3.

The MCS estimator of Eq. (5) follows the conventional MCS framework and reads:

$$\overline{P_f} = \frac{1}{N_{\mathrm{MCS}}} \sum_{i=1}^{N_{\mathrm{MCS}}} \mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(i)}, \boldsymbol{Z}^{(i)}\right), \tag{22}$$

where $\left\{\boldsymbol{X}^{(i)},\, i = 1, \ldots, N_{\mathrm{MCS}}\right\}$ are i.i.d copies of the input vector $\boldsymbol{X} \sim f_{\boldsymbol{X}}$ and $\left\{\boldsymbol{Z}^{(i)},\, i = 1, \ldots, N_{\mathrm{MCS}}\right\}$ are i.i.d copies of the latent variables of the simulator.

The variance of this estimator is computed as follows:

$$\begin{aligned} \mathrm{Var}\left[\overline{P_f}\right] &= \mathrm{Var}\left[\frac{1}{N_{\mathrm{MCS}}} \sum_{i=1}^{N_{\mathrm{MCS}}} \mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(i)}, \boldsymbol{Z}^{(i)}\right)\right], \\ &= \frac{1}{N_{\mathrm{MCS}}} \mathrm{Var}\left[\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X}, \boldsymbol{Z})\right] = \frac{\overline{P_f}\left(1 - \overline{P_f}\right)}{N_{\mathrm{MCS}}}. \end{aligned} \tag{23}$$

Similarly, the MCS estimator derived from Eq. (9) reads:

$$\widehat{P_f} = \frac{1}{N_{\mathrm{MCS}}} \sum_{i=1}^{N_{\mathrm{MCS}}} s\left(\boldsymbol{X}^{(i)}\right). \tag{24}$$

Its variance reads:

$$\begin{aligned} \mathrm{Var}\left[\widehat{P_f}\right] &= \mathrm{Var}\left[\frac{1}{N_{\mathrm{MCS}}} \sum_{i=1}^{N_{\mathrm{MCS}}} s\left(\boldsymbol{X}^{(i)}\right)\right], \\ &= \frac{1}{N_{\mathrm{MCS}}} \mathrm{Var}\left[s(\boldsymbol{X})\right]. \end{aligned} \tag{25}$$

Thus, for a fixed N_{MCS}, comparing the variance of the estimators reduces to comparing $\mathrm{Var}\left[\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X}, \boldsymbol{Z})\right]$ and $\mathrm{Var}\left[s(\boldsymbol{X})\right]$. From the law of total variance, we have:

$$\begin{aligned} \mathrm{Var}\left[\mathbb{1}_{\mathcal{D}_f}(\mathbf{X}, \mathbf{Z})\right] &= \mathbb{E}_{\mathbf{X}}\left[\mathrm{Var}_{\mathbf{Z}}(\mathbb{1}_{\mathcal{D}_f}(\mathbf{X}, \mathbf{Z}) \mid \mathbf{X})\right] + \mathrm{Var}_{\mathbf{X}}\left(\mathbb{E}_{\mathbf{Z}}[\mathbb{1}_{\mathcal{D}_f}(\mathbf{X}, \mathbf{Z}) \mid \mathbf{X}]\right), \\ &= \mathbb{E}_{\mathbf{X}}\left[\mathrm{Var}_{\mathbf{Z}}(\mathbb{1}_{\mathcal{D}_f}(\mathbf{X}, \mathbf{Z}) \mid \mathbf{X})\right] + \mathrm{Var}(s(\mathbf{X})), \\ &= \mathbb{E}_{\mathbf{X}}\left[s\left(\boldsymbol{X}\right)\left(1 - s\left(\boldsymbol{X}\right)\right)\right] + \mathrm{Var}(s(\mathbf{X})) \end{aligned} \tag{26}$$

This implies that, for a given computational budget, $\mathrm{Var}\left[\overline{P_f}\right]$ will always exceed $\mathrm{Var}\left[\widehat{P_f}\right]$, as the term $\mathbb{E}_{\mathbf{X}}\left[s\left(\boldsymbol{X}\right)\left(1 - s\left(\boldsymbol{X}\right)\right)\right]$ is always non-negative. This contribution represents the part of the variance due to sampling explicitly in the $\boldsymbol{Z}$-space, which does not exist when the conditional failure probability function $s(\boldsymbol{x})$ can be computed without sampling.

Although $s(\boldsymbol{x})$ is typically unknown, the presented emulators allow us to exploit the latent space to numerically integrate the uncertainty associated with $\boldsymbol{Z}$. It is important to note that this benefit only applies when the conditional failure probability function is computationally tractable, as is the case for SPCE and GLaM. We show in Appendix A that in cases where $s(\boldsymbol{x})$ is unknown and has to be estimated by MCS, the estimator $\overline{P_f}$ is preferable.

4.2 Computing $s(\boldsymbol{x})$ from GLaM emulator

For the GLaM model, computing the conditional probability of failure $s(\boldsymbol{x})$ is straightforward. Once trained, the PCE models provide the parameters $\boldsymbol{\lambda}(\boldsymbol{x})$, and the associated quantile function of the generalized lambda distribution is obtained according to Eq. (11). The computation of $s(\boldsymbol{x})$ is then given by:

$$s(\boldsymbol{x}) = Q^{-1}(0; \boldsymbol{\lambda}(\boldsymbol{x})). \tag{27}$$

Since the CDF of the generalized lambda distribution does not have a closed-form solution, a numerical inversion of the quantile function is required. Due to the monotonic property of the quantile function, this inversion is computationally efficient.

4.3 Computing $s(\boldsymbol{x})$ from SPCE emulator

For SPCE, the CDF associated with a predicted $\tilde{Y}_{\boldsymbol{x}}$, denoted as $F_{\tilde{Y}_{\boldsymbol{x}}}(y)$, is obtained by integrating the PDF introduced in Eq. (21). Since this PDF is a one-dimensional integral, it can be evaluated using Gaussian quadrature as follows:

$$\begin{aligned} f_{\tilde{Y}_{\boldsymbol{x}}}(y) &= \int_{\mathcal{D}_Z} \frac{1}{\sigma}\varphi\left(\frac{y - \sum_{\boldsymbol{\alpha}\in\mathcal{A}} c_{\boldsymbol{\alpha}}\psi_{\boldsymbol{\alpha}}(\boldsymbol{x}, z)}{\sigma}\right) f_Z(z)\mathrm{d}z \\ &\approx \sum_{j=1}^{N_Q} \frac{w^{(j)}}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y - \sum_{\boldsymbol{\alpha}\in\mathcal{A}} c_{\boldsymbol{\alpha}}\psi_{\boldsymbol{\alpha}}\left(\boldsymbol{x}, z^{(j)}\right)\right)^2}{2\sigma^2}\right), \end{aligned} \tag{28}$$

where N_Q is the number of integration points, $z^{(j)}$ is the j-th integration point, and $w^{(j)}$ is the corresponding weight, both associated to the weight function f_Z, which is in practice the uniform or Gaussian PDF, for which the integration points and weights are well-known.

The approximation introduced by the numerical integration scheme implies that $f_{\tilde{Y}_{\boldsymbol{x}}}(y)$ is approximated by a mixture of N_Q Gaussian distributions $\left\{\mathcal{N}\left(\sum_{\boldsymbol{\alpha}\in\mathcal{A}} c_{\boldsymbol{\alpha}}\psi_{\boldsymbol{\alpha}}\left(\boldsymbol{x}, z^{(j)}\right), \sigma^2\right), j = 1 \dots N_Q\right\}$,

with corresponding weights $w^{(j)}$. The CDF of this approximation is straightforward and reads:

$$\begin{aligned} F_{\tilde{Y}_{\boldsymbol{x}}}(y) &\approx \int_{-\infty}^{y} \sum_{j=1}^{N_Q} \frac{w^{(j)}}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y - \sum_{\boldsymbol{\alpha}\in\mathcal{A}} c_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}\left(\boldsymbol{x}, z^{(j)}\right)\right)^2}{2\sigma^2} \right) dy \\ &\approx \sum_{j=1}^{N_Q} w^{(j)} \int_{-\infty}^{y} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y - \sum_{\boldsymbol{\alpha}\in\mathcal{A}} c_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}\left(\boldsymbol{x}, z^{(j)}\right)\right)^2}{2\sigma^2} \right) dy \\ &\approx \sum_{j=1}^{N_Q} w^{(j)} \Phi\left(\frac{y - \sum_{\boldsymbol{\alpha}\in\mathcal{A}} c_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}\left(\boldsymbol{x}, z^{(j)}\right)}{\sigma} \right), \end{aligned} \tag{29}$$

where Φ is the standard Gaussian CDF.

Finally, the conditional failure probability function is given by $F_{\tilde{Y}_{\boldsymbol{x}}}(0)$, that is:

$$s\left(\boldsymbol{x}\right) \approx \sum_{j=1}^{N_Q} w^{(j)} \Phi\left(-\frac{\sum_{\boldsymbol{\alpha}\in\mathcal{A}} c_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}\left(\boldsymbol{x}, z^{(j)}\right)}{\sigma} \right). \tag{30}$$

Once the SPCE model is trained, computing $\sum_{\boldsymbol{\alpha}\in\mathcal{A}} c_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\boldsymbol{x}, z)$ is inexpensive. The accuracy of this approach strongly depends on the number of quadrature points used. In this paper, we used $N_Q = 100$, which, based on our tests, results in a negligible numerical error.

5 Results

All simulations presented in this section were conducted using the Generalized Lambda Models (Lüthen et al., 2024a) and Stochastic Polynomial Chaos Expansions (Lüthen et al., 2024b) modules from the UQLab software for uncertainty quantification (Marelli and Sudret, 2014).

5.1 Stochastic $R - S$ function

We first showcase the usage of the presented stochastic emulators using a modified stochastic $R - S$ example. The limit-state function reads:

$$g(\boldsymbol{X}, \boldsymbol{Z}) = \frac{R}{Z_1} - S \cdot Z_2, \tag{31}$$

where $\boldsymbol{X} = \{R, S\}$ are the random variables representing the resistance and demand, as in the classical $R - S$ problem, and $\boldsymbol{Z} = \{Z_1, Z_2\}$ are two latent variables introduced to represent the inner stochasticity of the model. The distribution of the associated random variables, their moments and their associated parameters are described in Table 1.

Table 1: Moments, distributions, and parameters of the considered variables for the $R - S$ problem.

Variable	Distribution	Mean	Std. Deviation	λ	ζ
R	Lognormal	5.0	0.8	1.5968	0.1590
S	Lognormal	2.0	0.6	0.6501	0.2936
Z_1	Lognormal	1.0	0.028	-0.0004	0.0280
Z_2	Lognormal	1.0	0.096	-0.0046	0.0958

Given the problem setup, an analytical solution can be obtained by remarking that the failure domain $\mathcal{D}_f = \{(\boldsymbol{x}, \boldsymbol{z}) : g(\boldsymbol{x}, \boldsymbol{z}) \leq 0\}$ also corresponds to the negative values of $\tilde{g}(\boldsymbol{x}, \boldsymbol{z}) = \ln\left(\frac{R}{Z_1}\right) - \ln(S \cdot Z_2) = \ln R - \ln S - \ln Z_1 - \ln Z_2$. Since all input variables have a lognormal distribution, $\tilde{g}$ is a linear combination of Gaussian variables, and P_f can be computed analytically:

$$P_f = \Phi\left(\frac{\lambda_R - \lambda_{Z_1} - \lambda_S - \lambda_{Z_2}}{\sqrt{\zeta_R^2 + \zeta_{Z_1}^2 + \zeta_S^2 + \zeta_{Z_2}^2}}\right) = 3.154 \times 10^{-3}. \tag{32}$$

It is also possible to obtain an analytical solution for the conditional failure probability function $s(\boldsymbol{x})$ defined in Eq. (8). In this case, we condition the limit-state function on $\boldsymbol{x} = (r, s)^T$, while $\boldsymbol{Z}$ is random. This leads to the following function:

$$s(\boldsymbol{x}) = s(r, s) = \Phi\left(\frac{\ln r - \lambda_{Z_1} - \ln s - \lambda_{Z_2}}{\sqrt{\zeta_{Z_1}^2 + \zeta_{Z_2}^2}}\right). \tag{33}$$

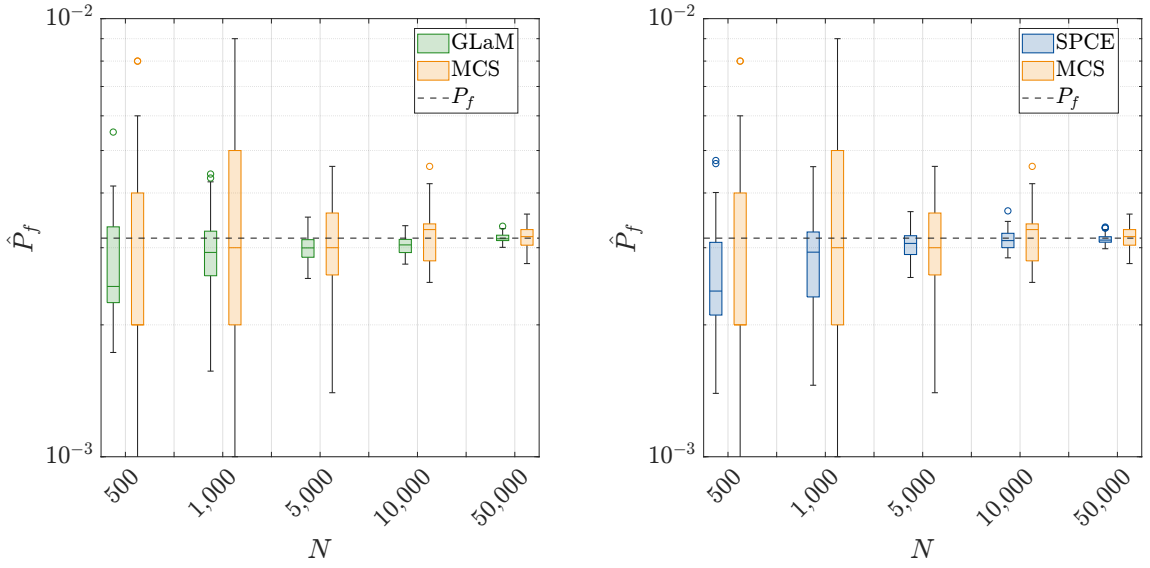
To demonstrate the performance of the emulators in estimating P_f and $s(\boldsymbol{x})$, we trained both models using various experimental design sizes, sampled via space-filling Latin Hypercube Sampling (LHS, Olsson et al., 2003), with sizes $N \in \{500; 1{,}000; 5{,}000; 10{,}000; 50{,}000\}$. The polynomial chaos expansions used in surrogating the parameters $\lambda_1(\boldsymbol{x})$ $\lambda_2(\boldsymbol{x})$ of the GLaM model have degree adaptivity ranging from $p \in [0, 3]$, while the expansions related to parameters $\lambda_3(\boldsymbol{x})$ and $\lambda_4(\boldsymbol{x})$ vary from $p \in [0, 2]$. The SPCE emulators have degree adaptivity ranging from $p \in [0, 4]$. For both emulators, the hyperbolic truncation q-norm ranges in $q \in [0.7, 1]$ with constant increments of 0.1.

The probability of failure $\widehat{P_f}$ (i.e., the average of the conditional failure probabilities) is estimated for each emulator using Monte Carlo simulation with a fixed sample size of $N_{\text{MCS}} = 10^6$. Moreover, to compare the performance of the emulators against direct MCS, we also estimate $\overline{P_f}$ using the same sample size of full-scale simulations that were used to train the emulators, *i.e.* $N \in \{500; 1{,}000; 5{,}000; 10{,}000; 50{,}000\}$. We depict the results in box plots to capture statistical variability, as each experimental setup is repeated 50 times. The width of the box plots represents the interquartile range, the horizontal line is the median and responses deviating by more than ± 2.7 standard deviations from the mean are considered outliers and plotted as circles.

Figure 1 depicts the box plots of the probabilities of failure estimated by the emulators, with GLaM in green and SPCE in blue, alongside the crude MCS estimates in orange. The black dashed line represents the analytical probability of failure P_f from Eq. (32). The emulators provide accurate estimates of P_f, even in a small data regime when $N \leqslant 1{,}000$ samples. As the size of the experimental design increases, the bias on the estimations of the emulators disappears, and the emulators converge to P_f. Additionally, the box plots resulting from the emulators are consistently narrower than those from direct MCS. This allows us to conclude that the emulators produce a lower variance in their estimations with the same number of full-scale simulations. In other words, it demonstrates the variance-reduction property of the emulators. These results suggest that the presented stochastic emulators are a viable option for performing reliability analysis.

The current implementation of this methodology still requires numerous evaluations of the limit-state function. However, active learning approaches, as shown in the literature for deterministic models (Teixeira et al., 2021; Moustapha et al., 2022), offer a way to significantly reduce the cost of training the emulators. By iteratively enriching the experimental design on regions where a more accurate surrogate model is needed, these methods can drastically lower the number of simulations required. As a result, precise estimates of P_f can be obtained with relatively small experimental designs, greatly minimizing computational costs. To the author's best knowledge, such an approach is still not available in the context of structural reliability for stochastic emulators.

Furthermore, Table 2 is provided to complement Figure 1, presenting the same data in numerical form. This table allows for precise comparison across sample sizes, showing the median value and coefficient of variation over 50 replications for the estimated probabilities of failure.



(a) GlaM emulator (in green) against direct MCS (in orange)

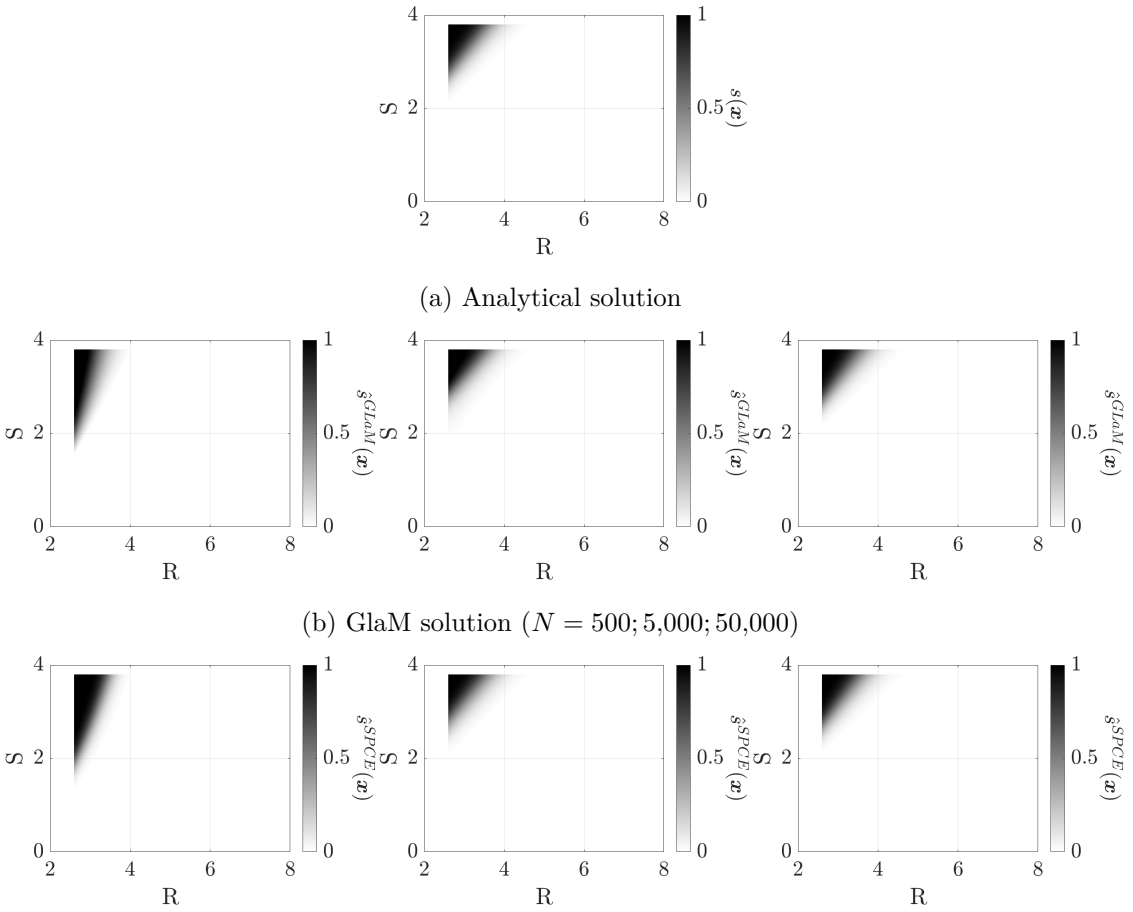
(b) SPCE emulator (in blue) against direct MCS (in orange)

Figure 1: $R-S$ function – Box plots comparing convergence behavior obtained from the emulators and from direct MCS for increasing values of N. The analytical probability of failure is depicted by the dashed black line.

Table 2: $R-S$ function – Median and coefficient of variation of probability of failure estimates for GLaM, MCS, and SPCE across varying sample sizes, obtained from 50 repetitions of the experiments. The analytical probability of failure is 3.154×10^{-3}.

N	GLaM		MCS		SPCE	
	Median $\widehat{P_f}$	Coef. of var.	Median $\overline{P}_f$	Coef. of var.	Median $\widehat{P_f}$	Coef. of var.
500	2.447×10^{-3}	24.7%	2.000×10^{-3}	65.4%	2.389×10^{-3}	25.1%
1,000	2.923×10^{-3}	19.4%	3.000×10^{-3}	55.7%	2.931×10^{-3}	24.3%
5,000	3.000×10^{-3}	5.9%	3.000×10^{-3}	24.8%	3.069×10^{-3}	7.3%
10,000	3.045×10^{-3}	4.8%	3.300×10^{-3}	13.3%	3.114×10^{-3}	4.7%
50,000	3.144×10^{-3}	2.3%	3.180×10^{-3}	6.7%	3.123×10^{-3}	2.3%

Due to the stochastic nature of the problem, depicting a limit-state surface in the x-space is not possible. However, to visually compare the performance of the emulators, we plot heat maps of the function $s(\boldsymbol{x})$ and those estimated by GLaM and SPCE, denoted as $\widehat{s}^{GLaM}(\boldsymbol{x})$ and $\widehat{s}^{SPCE}(\boldsymbol{x})$, respectively. Figure 2 displays the heat maps for three different experimental design sizes: $N = \{500; 5{,}000; 50{,}000\}$. These surfaces correspond to the seeds that led to the median $\widehat{P_f}$ in Figure 1. For $N = 500$, there is a noticeable difference between the reference function and those produced by the emulators. However, as the size of the training set increases, the emulators



(a) Analytical solution

(b) GlaM solution ($N = 500; 5{,}000; 50{,}000$)

(c) SPCE solution ($N = 500; 5{,}000; 50{,}000$)

Figure 2: $R - S$ function: Comparison of the conditional failure probability heat maps for different sizes of the experimental design used for GLaM and SPCE models.

become more accurate, with only minor differences observed for $N = 5{,}000$. For $N = 50{,}000$, no distinguishable differences are evident.

5.2 Stochastic simply supported beam

Let us consider a simply supported beam subjected to uniform load. We are interested in the probability that the mid-span deflection exceeds a given threshold t_{lim}. The associated limit-state function reads:

$$g(\boldsymbol{X}, E) = t_{lim} - \left(\frac{5pL^4}{32Ebh^3} \right). \tag{34}$$

The input variables are $\boldsymbol{X} = \{p, L, b, h\}$, and $Z = E$ is treated as latent variable. By doing this, we aim to simulate a scenario where heterogeneous properties of the materials are modeled using random fields (Grigoriu, 2002), which would naturally require a stochastic treatment. However, since we focus on validating the proposed methodology in an academic context, we do not discuss appropriate ways of modeling these properties. Instead, we simplify the problem by using E as a

latent variable. The distributions of the related random variables, along with their moments and corresponding parameters, are presented in Table 3.

Table 3: Moments, distributions, and parameters of the considered variables for the simply supported beam example.

Variable	Distribution	Mean	Std. Deviation	λ	ζ
b	Lognormal	0.15 m	7.5×10^{-3} m	-1.8984	0.0500
h	Lognormal	0.3 m	15×10^{-3} m	-1.2052	0.0500
L	Lognormal	5.0 m	0.05 m	1.6094	0.0100
p	Lognormal	10 kN/m	2 kN/m	9.1907	0.1980
E	Lognormal	30,000 MPa	4,500 MPa	24.1133	0.1492

As in the $R-S$ problem, based on the shape of the limit-state function in Eq. (34) and the lognormal input distributions, the probability of failure can be calculated analytically and reads:

$$P_f = \Phi\left(\frac{\ln(t_{lim}) - \ln\left(\frac{5}{32}\right) - \lambda_p - 4\lambda_L + \lambda_E + \lambda_b + 3\lambda_h}{\sqrt{\zeta_p^2 + 16\zeta_L^2 + \zeta_E^2 + \zeta_b^2 + 9\zeta_h^2}}\right). \tag{35}$$

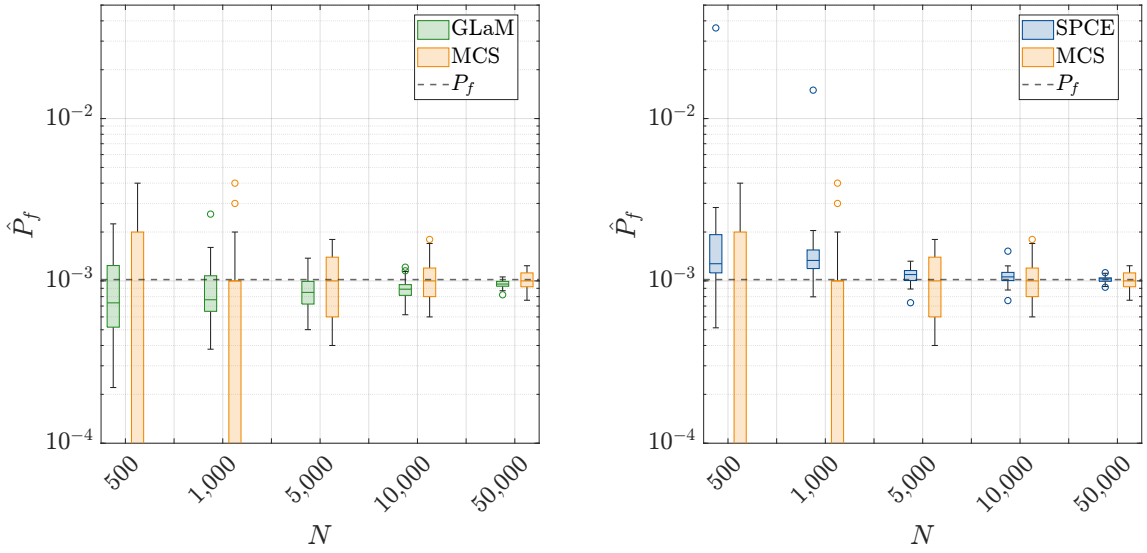
We consider a displacement threshold of $t_{lim} = \text{span}/250 = 0.02$ m, according to European Committee for Standardization (2004). This yields $P_f = 1.019 \times 10^{-3}$.

We evaluate the performance of the emulators using a similar approach as in the previous example. Both emulators were trained with different experimental design sizes $N \in \{500; 5{,}000; 10{,}000; 50{,}000\}$ obtained via LHS (Olsson et al., 2003). The polynomial chaos expansions used to approximate the parameters $\lambda_1(\boldsymbol{x})$ and $\lambda_2(\boldsymbol{x})$ in the GLaM model had adaptive degrees ranging from $p \in [1, 4]$, while the expansions for $\lambda_3(\boldsymbol{x})$ and $\lambda_4(\boldsymbol{x})$ varied from $p \in [0, 2]$. For SPCE, the degree adaptivity ranged from $p \in [1, 7]$. The hyperbolic truncation q-norm for all expansions ranged from $q \in [0.7, 1]$, with increments of 0.1. $\widehat{P_f}$, obtained as the empirical average of the conditional failure probability function, was computed via MCS with $N_{MCS} = 10^6$ samples. To compare with direct MCS, we also estimated $\overline{P_f}$ with the same number of full-scale simulations for training the emulators. All results are displayed in box plots as we carried out 50 experiment repetitions to account for the statistical variability.

Figure 3 compares the various estimates of the failure probability for different sizes of the experimental design. The emulators provide relatively accurate estimates of P_f, even with small EDs ($N \leqslant 1{,}000$), despite a slight bias. In contrast, for many Monte Carlo simulation runs with $N \leqslant 1{,}000$, no failed samples were observed, leading to $\overline{P_f} = 0$, thus the orange box plots shown in Figure 3. Using such small datasets for MCS to estimate P_f on the order of 10^{-3} is generally not recommended. However, we include these results to show that the emulators already provide reasonable estimates with limited data. The bias in the emulator estimates decreases as the experimental design size increases, ultimately converging to the reference probability of failure,

P_f. Additionally, as observed in the previous example, the box plots obtained by the emulators are consistently narrower than those from MCS, indicating that the emulators converge faster to P_f.

Additionally, Table 4 is included to supplement Figure 3. It offers a numerical representation of the same data and enables detailed comparison across sample sizes by providing the median and standard deviation of the estimated failure probabilities.



(a) GlaM emulator (in green) against direct MCS (in orange)

(b) SPCE emulator (in blue) against direct MCS (in orange)

Figure 3: Simply supported beam example – Box plots comparing convergence behavior obtained from the emulators and from direct MCS for increasing values of N. The analytical probability of failure is depicted by the dashed black line.

Table 4: Simply supported beam – Median and standard deviation of probability of failure estimates for GLaM, MCS, and SPCE across varying sample sizes, obtained from 50 repetitions of the experiments. The analytical probability of failure is 1.019×10^{-3}.

N	GLaM		MCS		SPCE	
	Median $\widehat{P_f}$	Coef. of var.	Median $\overline{P}_f$	Coef. of var.	Median $\widehat{P_f}$	Coef. of var.
500	7.324×10^{-4}	55.4%	0	126.0%	1.276×10^{-3}	24.3%
1,000	7.653×10^{-4}	30.3%	1.000×10^{-3}	76.3%	1.337×10^{-3}	15.5%
5,000	8.4953×10^{-4}	22.6%	1.000×10^{-3}	37.6%	1.094×10^{-3}	9.8%
10,000	8.887×10^{-4}	12.4%	1.000×10^{-3}	26.8%	1.059×10^{-3}	8.0%
50,000	9.564×10^{-4}	5.4%	1.000×10^{-3}	12.6%	1.030×10^{-3}	4.4%

5.3 Wind turbine application

Understanding and predicting the performance of wind turbines under realistic wind conditions is paramount when designing them. Consequently, reliability analysis of stochastic models has often focused on wind turbines, as shown in numerous studies Choe et al. (2015, 2016, 2017); Cao and Choe (2019); Pan et al. (2020); Li et al. (2021). The typical design process involves two steps: generating wind excitations and conducting aero-elastic simulations to model the interaction between wind inflow, aerodynamics, and structural dynamics. While the aero-elastic model itself is deterministic, stochasticity arises from the wind excitation. The wind loads are represented as 10-minute spatio-temporal random fields defined by *macroscopic parameters* such as wind speed and turbulence intensity. Although these parameters simplify wind modeling, they do not fully characterize the wind inflow. Consequently, multiple wind fields can be generated from the same parameters, leading to different dynamical responses.

We test our methodology using the dataset from Barone et al. (2012), which includes the equivalent of 96 years of operational data for the onshore 5 MW NREL reference turbine (Jonkman et al., 2009). Stochastic wind inflow is generated using the NREL TurbSim code (Jonkman, 2009), with average wind speed and turbulence intensity as inputs. The wind speed is modeled using a truncated Rayleigh distribution, following Moriarty (2008). The average wind speed U is 10 m/s (untruncated), with cut-in and cut-off speeds of 3 m/s and 25 m/s, respectively. Turbulence intensity is specified deterministically as a function of mean wind speed, following the IEC normal turbulence model (International Electrotechnical Commission, 2005), making wind speed the only random variable in this study. Aero-elastic simulations are performed using the FAST code (Jonkman and Buhl, 2005; Jonkman and Jonkman, 2013).

Of interest is the probability that the maximum flap-wise bending moment at the root of the blade during the 10-minute simulations $M_b(U, \omega)$ exceeds a given threshold τ within the simulated time frame. The associated limit-state function can be defined as follows:

$$g_s(U, \omega) = \tau - M_b(U, \omega), \tag{36}$$

Figure 4a presents the available dataset, displaying $M_b(U, \omega)$ as a function of the average wind speed U (approximately 5 million data points), while Figure 4b shows a normalized histogram representing the PDF of the wind speed U. The dataset comprises approximately 5 million simulator runs, with no replications. We consider a threshold of $\tau = 15{,}000\,\mathrm{kN} \cdot \mathrm{m}$, resulting in a target probability of failure, $P_f = 1.022 \times 10^{-2}$, with a coefficient of variation below 0.5%, computed using all available samples in the dataset. This threshold (and corresponding target P_f) is slightly larger (and smaller, respectively) than those considered in Choe et al. (2015); Cao and Choe (2019).

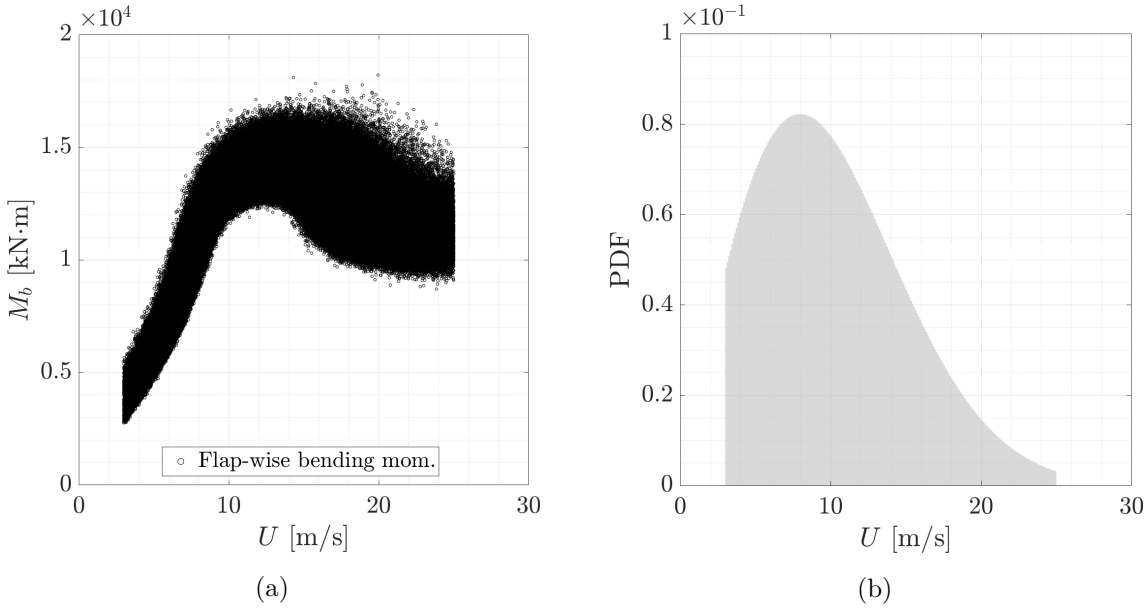


Figure 4: Wind turbine application – (a) Scatter plot depicting the relationship between the average wind speed and the maximum flap-wise bending moment for the given dataset. (b) Normalized histogram representing the PDF of the average wind speed for the given dataset.

The emulators are constructed by subsampling the available data uniformly in the input space. This adjustment, as opposed to using the original Rayleigh distribution, ensures that enough training samples are located in the tails, which is crucial for the emulators accuracy. The emulators are built with three experimental design sizes: $N \in \{500, 1{,}000, 5{,}000\}$. The polynomial chaos expansions used to surrogate the parameters $\lambda_1(\boldsymbol{x})$ and $\lambda_2(\boldsymbol{x})$ in the GLaM model have adaptive degrees ranging from $p \in [1, 10]$, while the expansions for $\lambda_3(\boldsymbol{x})$ and $\lambda_4(\boldsymbol{x})$ vary in the range $p \in [0, 3]$. For the SPCE emulators, the degree adaptivity ranges in $p \in [1, 10]$. Since this example involves only a single random variable, hyperbolic truncation does not apply. A total of 50 different experiments are conducted to account for statistical variability.

Since this is a one-dimensional example, we can visually assess the emulators predictions against those derived from the dataset. Our analysis includes a comparison of the mean function, $\mu(u)$, and the α-quantile function, $Q_\alpha(u)$. Additionally, to gain some insights regarding the heteroskedasticity of the problem, we compute the variance function associated with the simulator. Given the discrete nature of the data, we employ a moving window approach to estimate these functions. For each wind speed u, we define a window as $[u - \Delta, u + \Delta]$, where $\Delta = 0.1$ m/s is a small positive parameter. The mean function at u is then estimated as the empirical mean of the values M_b within this window, *i.e.*,

$$\mu(u) = \frac{1}{N_w} \sum_{i:u^{(i)} \in [u-\Delta, u+\Delta]} M_b^{(i)}, \tag{37}$$

where $M_b^{(i)} = M_b\left(u^{(i)}\right)$ and N_w is the number of data points within the window:

$$N_w = \text{card}\left\{i : u^{(i)} \in [u-\Delta, u+\Delta]\right\}. \tag{38}$$

Similarly, the variance function is estimated empirically as follows:

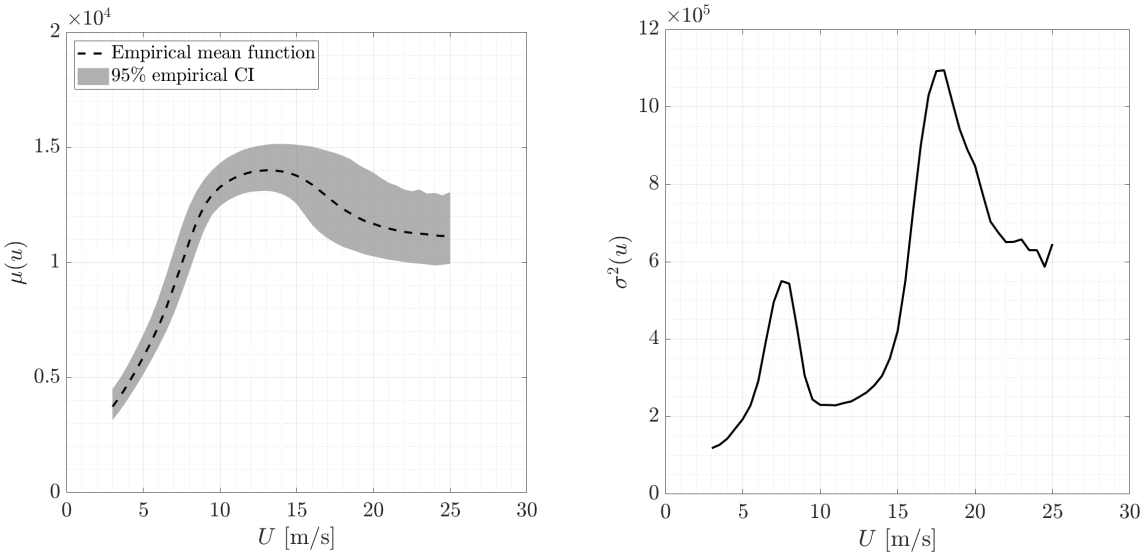
$$\sigma^2(u) = \frac{1}{N_w - 1} \sum_{i:u^{(i)}\in[u-\Delta,u+\Delta]} \left(M_b^{(i)} - \mu(u)\right)^2, \tag{39}$$

Since the conditional responses of the model are not necessarily normally distributed, estimating confidence intervals requires quantile estimation. The empirical α-quantile is estimated by first sorting the values M_b within the window in ascending order. The α-quantile, $Q_\alpha(u)$, is then defined as:

$$Q_\alpha(u) = M_{b_{(\lfloor \alpha N_w \rfloor)}} \tag{40}$$

where $\lfloor \cdot \rfloor$ represents the floor function, which gives the largest integer less than or equal to αN_w and $M_{b_{(\bullet)}}$ are the sorted values of M_b ordered so that $M_{b_{(1)}} \leq M_{b_{(2)}} \leq, \ldots, M_{b_{(N)}}$.

Figure 5 illustrates the results obtained with the moving window approach. Figure 5a displays the mean function, $\mu(u)$, along with the 95% central confidence interval $Q_{2.5\%} - Q_{97.5\%}$ (Eq. (40)). Figure 5b depicts the variance function, $\sigma^2(u)$. While the empirical mean function is relatively smooth, the variance function depicts heteroskedastic and non-monotonic behavior, which generally complicates surrogate modeling.



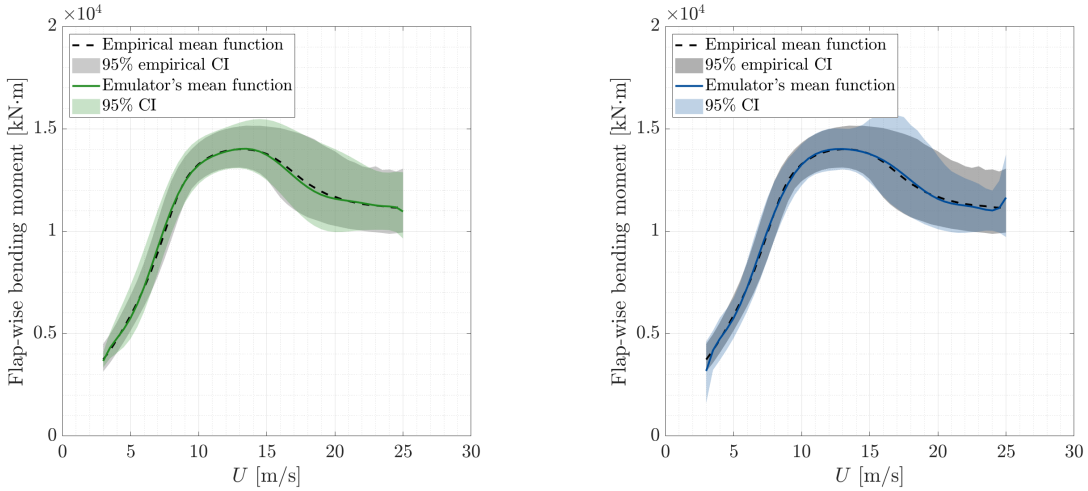
(a) (b)

Figure 5: Wind turbine application – (a) Empirical mean curve (dotted line) and the corresponding 95% empirical confidence interval (gray shaded area) based on the available dataset. (b) Empirical variance function.

Figure 6 shows the fitting of the emulators for different experimental design sizes. For each ED size, the model yielding the median $\hat{P}_f$ is used for illustration. The colored lines represent

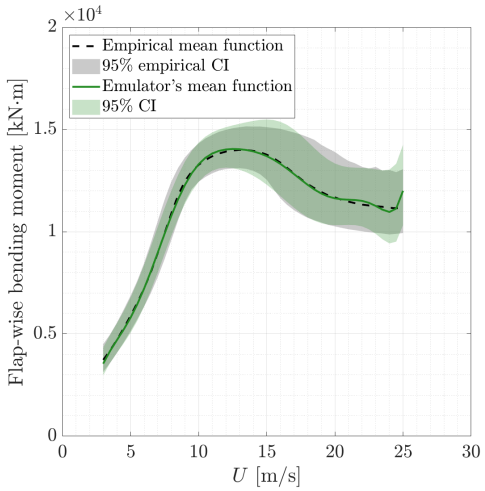
the mean functions obtained by the emulators, while the dashed black lines show the empirical mean computed from the data. The shaded colored areas indicate the 95% confidence interval obtained from the emulators, whereas the ones in gray correspond to the confidence intervals derived directly from the data.

With a relatively small experimental design of 500 points, the surrogate models capture the behavior of the mean function fairly accurately, despite minor discrepancies. As the size of the experimental design increases, the accuracy of the emulators improves, and the mean curve obtained by the emulator gets closer to that obtained from the data. A similar behavior is observed with the confidence intervals. For small experimental designs, discrepancies are more remarkable, but as the experimental design size increases, the emulators converge more closely to the confidence intervals derived from the data, demonstrating a convergence trend.

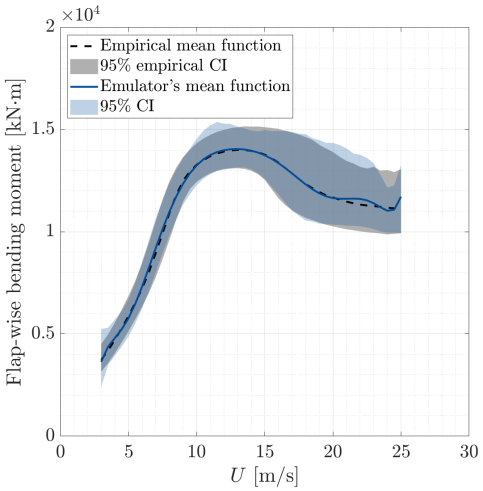


(a) GLaM, $N = 500$

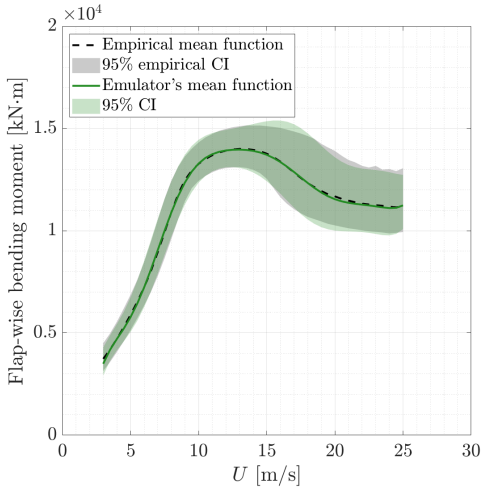
(b) SPCE, $N = 500$



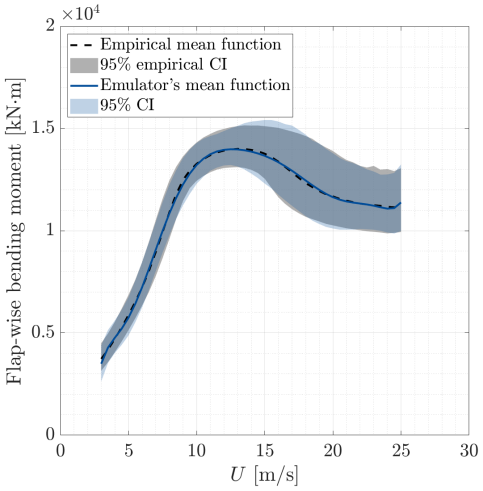
(c) GLaM, $N = 1,000$



(d) SPCE, $N = 1,000$



(e) GLaM, $N = 5,000$



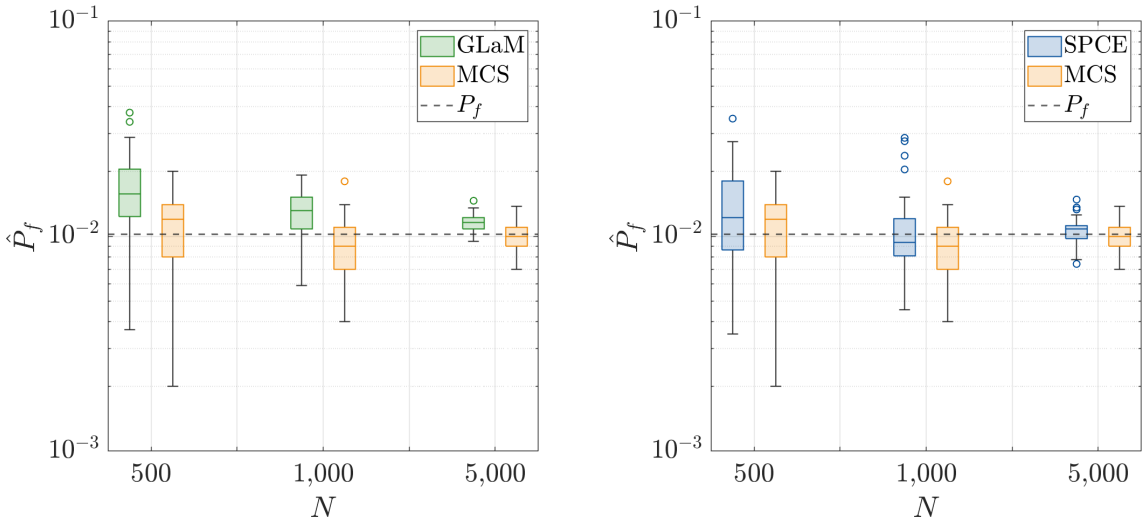
(f) SPCE, $N = 5,000$

Figure 6: Wind turbine application – Empirical mean curve (dotted line) and the corresponding 95 % empirical confidence interval (gray shaded area) based on the available dataset. The full, colored line represents the mean function obtained by the emulator, while the colored shaded area shows the 95 % centered confidence interval generated by the emulators when trained with an ED of size N.

Figure 7 shows the probabilities of failure obtained by the emulators and via MCS, obtained by drawing subsamples of the original dataset. The green boxplots represent results from GLaM, while the blue ones correspond to those obtained via SPCE. The orange boxplots display the results from crude MCS. Similar to previous examples, Table 5 provides a numerical summary of the data shown in Figure 7. Despite the strong heteroskedasticity in the simulator, we observe that the emulators yield reasonable estimates of P_f, even for small ED, such as $N = 500$. Nevertheless, a small bias is observed in the results. We attribute these results to an inaccurate fitting in the tails of the conditional distributions, especially in the input regions with high probability density, which significantly impacts the estimated probability of failure. As shown in Figure 6a and 6b, for $N = 500$ points, the emulators overestimate the $Q_{97.5\%}$. While this estimation improves as the experimental design size increases, the bias does not fully vanish and persists at $N = 5{,}000$ in the range $14 \leqslant u \leqslant 18$.

Even though convergence is observed as the ED size increases, the emulators do not consistently outperform Monte Carlo simulation. There are two main reasons for this. First, the relatively large failure probability requires only a limited number of samples for accurate estimation by MCS. Second, the conditional distributions generated by the emulators are not sufficiently accurate in areas contributing the most to the failure probability. While accuracy improves with larger ED, convergence remains relatively slow, hence favoring MCS. However, the convergence rate of the emulators could be accelerated by selectively targeting training points in these critical areas, rather than the the current global training approach. As in surrogate-based reliability analysis with deterministic simulators, active learning can be leveraged to achieve this targeted refinement of the ED.

In addition to reliability estimates, the emulators provide valuable insights into the behavior of the simulators. Once trained, they enable the estimation of the conditional failure probability function, providing valuable information on input configurations that are most likely to lead to system failure. Furthermore, the emulators allow some degree of extrapolation, enabling reliability estimations that would be infeasible with crude MCS.



(a) GlaM emulator (in green) against direct MCS (in orange)

(b) SPCE emulator (in blue) against direct MCS (in orange)

Figure 7: Wind turbine application – Box plots comparing convergence behavior obtained from the emulators and from direct MCS for increasing values of N . The reference probability of failure is depicted by the dashed black line.

Table 5: Wind turbine application – Median and standard deviation of probability of failure estimates for GLaM, MCS, and SPCE across varying sample sizes, obtained from 50 repetitions of the experiments. The reference probability of failure is 1.022×10^{-2}.

N	GLaM		MCS		SPCE	
	Median $\widehat{P_f}$	Coef. of var.	Median $\overline{P}_f$	Coef. of var.	Median $\widehat{P_f}$	Coef. of var.
500	1.570×10^{-2}	33.1%	1.200×10^{-2}	39.0%	1.219×10^{-2}	43.9%
1,000	1.315×10^{-2}	21.6%	9.000×10^{-3}	25.9%	9.333×10^{-3}	22.0%
5,000	1.116×10^{-4}	8.7%	1.000×10^{-2}	14.7%	1.080×10^{-3}	11.1%

6 Conclusion

Structural reliability methods are well established for systems in which the limit state function is deterministic. However, modern simulators found in, *e.g.*, wind turbine design or earthquake engineering are stochastic in nature due to the presence of latent variables $\boldsymbol{Z}$ on top of the basic input variables $\boldsymbol{X}$ in their formulation. In this paper, we present the established definition of the probability of failure P_f in the context of stochastic simulators and outline its associated estimators. Although crude Monte Carlo simulation based on sampling in the $(\boldsymbol{X}, \boldsymbol{Z})$ space is feasible, it reveals intractable for realistic problems.

To address this issue, we leverage the estimator that expresses the probability of failure as the expected value (over the $\boldsymbol{X}$-space) of the conditional failure probability function $s(\boldsymbol{x})$. Moreover,

we propose using two types of recently developed stochastic emulators as surrogate models for the (stochastic) limit state function, which yield *semi-analytical* expressions of $s(\boldsymbol{x})$, thus leading to a significant cost reduction. These are the generalized lambda models (GlaM) and stochastic polynomial chaos expansions (SPCE). These two emulators do not require replicated runs of the simulator to construct the training set (experimental design), thus offering a versatile data-driven approach. We also prove that the proposed estimator of P_f has a smaller variance compared to the crude MCS one.

We benchmark our emulator-based approach on two analytical examples for which both P_f and the conditional failure probability function are derived analytically. This allow us to show the convergence of our method with the increasing size of the experimental design. Both GlaM and SPCE provide similar performances, with fairly accurate results already obtained with 5,000 samples, and a smaller coefficient of variation (computed from 50 independent replicated runs of the full analysis) three times smaller than that of the crude Monte Carlo approach. The third example is related to the reliability analysis of a 5 MW wind turbine w.r.t. the maximum flap-wise bending moment at the blades toe. For this case study, we use a publicly available set of simulations of the NREL 5-MW reference wind turbine, performed by Sandia National Laboratories. We show that the two emulators reproduce the mean function and $[2.5\% - 97.5\%]$ quantiles of the bending moment with a remarkable accuracy compared to those obtained by nonparametric methods. Regaridng failure probability estimation, the emulator-based approach performs comparably to MCS and even outperforms it when employing SPCE with 5,000 training points.

In contrast to recent reliability methods developed for deterministic limit state functions, the proposed approach does not (yet) leverage an iterative construction of the emulators using an active learning method. This work in progress should allow us to drastically reduce the overall computational cost by increasing the accuracy of the surrogates only in the regions of interest.

A Variance of estimator for unknown $s(\boldsymbol{x})$

This section demonstrates that when the conditional failure probability function $s(\boldsymbol{x})$ is unknown, estimating P_f using the single-loop estimator $\overline{P_f}$ is more efficient than using the double-loop estimator, denoted as $\overline{\overline{P_f}}$. The single-loop estimator is defined as:

$$\overline{P_f} = \frac{1}{N_{\mathrm{MCS}}} \sum_{i=1}^{N_{\mathrm{MCS}}} \mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(i)}, \boldsymbol{Z}^{(i)}\right), \tag{41}$$

where $\left\{\left(\boldsymbol{X}^{(i)}, \boldsymbol{Z}^{(i)}\right), i = 1, \ldots, N_{\mathrm{MCS}}\right\}$ are i.i.d. copies of $(\boldsymbol{X}, \boldsymbol{Z})$. Its variance is given by:

$$\mathrm{Var}\left[\overline{P_f}\right] = \frac{1}{N_{\mathrm{MCS}}} \mathrm{Var}\left[\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X}, \boldsymbol{Z})\right] = \frac{\overline{P_f}\left(1 - \overline{P_f}\right)}{N_{\mathrm{MCS}}}. \tag{42}$$

In contrast, the double-loop estimator $\overline{\overline{P_f}}$ requires multiple replications to estimate $s(\boldsymbol{x})$ for each $\boldsymbol{x}$, significantly increasing computational demand. The double-loop estimator is defined as:

$$\overline{\overline{P_f}} = \frac{1}{N}\sum_{i=1}^{N} \overline{s}\left(\boldsymbol{X}^{(i)}\right), \tag{43}$$

where $\overline{s}\left(\boldsymbol{X}^{(i)}\right)$ is an estimator of the conditional probability failure function based on R replications:

$$\overline{s}\left(\boldsymbol{X}^{(i)}\right) = \frac{1}{R}\sum_{j=1}^{R} \mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(i)}, \boldsymbol{Z}^{(i,j)}\right). \tag{44}$$

That is,

$$\overline{\overline{P_f}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{R}\sum_{j=1}^{R} \mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(i)}, \boldsymbol{Z}^{(i,j)}\right), \tag{45}$$

where N is the number of samples of $s(\boldsymbol{x})$ evaluated.

To show that $\overline{P_f}$ converges faster than $\overline{\overline{P_f}}$, we compare their variances for a given budget. Using a notation consistent with Sec. 4, we define the total budget as $N_{\text{MCS}} = NR$. The variance of the double-loop estimator $\overline{\overline{P_f}}$ is given by:

$$\operatorname{Var}\left[\overline{\overline{P_f}}\right] = \operatorname{Var}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{R}\sum_{j=1}^{R} \mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(i)}, \boldsymbol{Z}^{(i,j)}\right)\right]. \tag{46}$$

Applying Bienaymé's identity (Klenke, 2013) to Eq. (46) gives:

$$\begin{aligned}\operatorname{Var}\left[\overline{\overline{P_f}}\right] = \frac{1}{N^2R^2}\Bigg(&\sum_{i=1}^{N}\sum_{j=1}^{R}\underbrace{\operatorname{Cov}\left[\mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(i)}, \boldsymbol{Z}^{(i,j)}\right), \mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(i)}, \boldsymbol{Z}^{(i,j)}\right)\right]}_{\text{(I)}} \\ &+ \sum_{i=1}^{N}\sum_{\substack{j,l=1\\ j\neq l}}^{R}\underbrace{\operatorname{Cov}\left[\mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(i)}, \boldsymbol{Z}^{(i,j)}\right), \mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(i)}, \boldsymbol{Z}^{(i,l)}\right)\right]}_{\text{(II)}} \\ &+ \sum_{\substack{i,k=1\\ i\neq k}}^{N}\sum_{j,l=1}^{R}\underbrace{\operatorname{Cov}\left[\mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(i)}, \boldsymbol{Z}^{(i,j)}\right), \mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(k)}, \boldsymbol{Z}^{(k,l)}\right)\right]}_{\text{(III)}}\Bigg).\end{aligned} \tag{47}$$

The first component in the variance, (I), simplifies to $\operatorname{Var}\left[\mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}, \boldsymbol{Z}\right)\right]$, which represents the variance of the direct MCS estimator $\overline{P_f}$, as shown in Eq. (23).

Next, we compute the second component, (II). To do this, we introduce $\boldsymbol{Z}'$ as a copy of $\boldsymbol{Z}$. In this case, the covariance term can be rewritten as:

$$\operatorname{Cov}\left[\mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(i)}, \boldsymbol{Z}^{(i,j)}\right), \mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(i)}, \boldsymbol{Z}^{(i,l)}\right)\right] = \operatorname{Cov}\left[\mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}, \boldsymbol{Z}\right), \mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}, \boldsymbol{Z}'\right)\right]. \tag{48}$$

Expanding the covariance leads to:

$$\begin{aligned}\operatorname{Cov}\left[\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z}),\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z'})\right] &= \mathbb{E}_{\boldsymbol{X},\boldsymbol{Z},\boldsymbol{Z'}}\left[\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z}),\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z'})\right] \\ &\quad - \mathbb{E}_{\boldsymbol{X},\boldsymbol{Z}}\left[\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z})\right]\mathbb{E}_{\boldsymbol{X},\boldsymbol{Z'}}\left[\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z'})\right].\end{aligned} \tag{49}$$

$\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z})$ and $\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z'})$ are independent realizations of the Bernoulli random variable $\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z})$. Moreover, $\mathbb{E}_{\boldsymbol{X},\boldsymbol{Z}}\left[\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z})\right]$ and $\mathbb{E}_{\boldsymbol{X},\boldsymbol{Z'}}\left[\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z'})\right]$ correspond to the replication-free estimator of the probability of failure $\overline{P_f}$. Thus, we have:

$$\operatorname{Cov}\left[\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z}),\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z'})\right] = \mathbb{E}_{\boldsymbol{X}}\left[\mathbb{E}_{\boldsymbol{Z}}\left[\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z})\right]\mathbb{E}_{\boldsymbol{Z'}}\left[\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z'})\right]\right] - \overline{P_f}^2. \tag{50}$$

Now, using the definition of the conditional probability of failure function $s(\boldsymbol{x})$ from Eq. (8), we can rewrite the covariance as:

$$\begin{aligned}\operatorname{Cov}\left[\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z}),\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z'})\right] &= \mathbb{E}_{\boldsymbol{X}}\left[s^2(\boldsymbol{X})\right] - \overline{P_f}^2 \\ &= \operatorname{Var}\left[s(\boldsymbol{X})\right].\end{aligned} \tag{51}$$

For the third component, (III), where $i \neq k$, the covariance terms vanish due to the independence between $\mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(i)},\boldsymbol{Z}^{(i,j)}\right)$ and $\mathbb{1}_{\mathcal{D}_f}\left(\boldsymbol{X}^{(k)},\boldsymbol{Z}^{(k,l)}\right)$.

Substituting these components into Eq. (47), we obtain the final expression for the variance of the double-loop estimator:

$$\begin{aligned}\operatorname{Var}\left[\overline{\overline{P_f}}\right] &= \frac{1}{N^2R^2}\left(NR\operatorname{Var}\left(\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z})\right) + NR(R-1)\operatorname{Var}\left[s(\boldsymbol{X})\right]\right) \\ &= \frac{1}{N_{\text{MCS}}}\left(\operatorname{Var}\left(\mathbb{1}_{\mathcal{D}_f}(\boldsymbol{X},\boldsymbol{Z})\right) + (R-1)\operatorname{Var}\left[s(\boldsymbol{X})\right]\right) \\ &= \frac{1}{N_{\text{MCS}}}\left(\overline{P_f}\left(1-\overline{P_f}\right) + (R-1)\operatorname{Var}\left[s(\boldsymbol{X})\right]\right).\end{aligned} \tag{52}$$

Thus, for a fixed computational budget $N_{\text{MCS}} = NR$, the variance of $\overline{\overline{P_f}}$ is larger than the variance of $\overline{P_f}$ (Eq. (23)) due to the additional term $(R-1)\operatorname{Var}\left[s(\boldsymbol{X})\right]$ in the expression. In fact, this variance can be reduced by setting $R = 1$, which corresponds to the case without replications. This demonstrates that the replication-free estimator $\overline{P_f}$ is more efficient than $\overline{\overline{P_f}}$ when $s(\boldsymbol{x})$ is unknown.

Acknowledgments

The authors express their sincere appreciation to Dr. Nora Lüthen for her contributions to the derivations presented in Appendix A.

The support of European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 955393 is greatly acknowledged.

References

Ankenman, B., B. Nelson, and J. Staum (2010). Stochastic Kriging for simulation metamodeling. *Operations Research 58*(2), 371–382.

Au, S. K. and J. L. Beck (2001). Estimation of small failure probabilities in high dimensions by subset simulation. *Probabilistic Engineering Mechanics 16*(4), 263–277.

Barone, M., J. Paquette, B. Resor, and L. Manuel (2012). Decades of wind turbine load simulation. In *50th American Institute of Aeronautics and Astronautics Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, Nashville, United States of America*, pp. 1–11.

Binois, M., R. B. Gramacy, and M. Ludkovski (2018). Practical heteroscedastic Gaussian process modeling for large simulation experiments. *Journal of Computational and Graphical Statistics 27*(4), 808–821.

Bourinet, J.-M., F. Deheeger, and M. Lemaire (2011). Assessing small failure probabilities by combined subset simulation and support vector machines. *Struct. Saf. 33*(6), 343–353.

Britton, T. (2010). Stochastic epidemic models: A survey. *Mathematical Biosciences 225*(1), 24–35.

Cao, Q. D. and Y. Choe (2019). Cross-entropy based importance sampling for stochastic simulation models. *Reliability Engineering & System Safety 191*, 106526.

Choe, Y., E. Byon, and N. Chen (2015). Importance sampling for reliability evaluation with stochastic simulation models. *Technometrics 57*(3), 351–361.

Choe, Y., H. Lam, and E. Byon (2017). Uncertainty quantification of stochastic simulation for black-box computer experiments. *Methodology and Computing in Applied Probability 20*(4), 1155–1172.

Choe, Y., Q. Pan, and E. Byon (2016). Computationally efficient uncertainty minimization in wind turbine extreme load assessments. *Journal of Solar Energy Engineering 138*(4), 041012.

De Angelis, M., E. Patelli, and M. Beer (2014). Line Sampling for assessing structural reliability with imprecise failure probabilities. In M. Beer, S.-K. Au, and J. W. Hall (Eds.), *Vulnerability, Uncertain. Risk*, Liverpool, UK, pp. 915–924. ASCE.

Ditlevsen, O. and H. O. Madsen (1996). *Structural reliability methods*. J. Wiley and Sons, Chichester.

European Committee for Standardization (2004). European Standard EN 1992-1-1:2004. Standard, European Committee for Standardization.

Freimer, M., G. Kollia, G. S. Mudholkar, and C. T. Lin (1988). A study of the generalized Tukey lambda family. *Communications in Statistics – Theory and Methods 17*(10), 3547–3567.

Geyer, S., I. Papaioannou, and D. Straub (2019). Cross entropy-based importance sampling

using Gaussian densities revisited. *Structural Safety 76*, 15–27.

Gramstad, O., C. Agrell, E. Bitner-Gregersen, B. Guo, E. Ruth, and E. Vanem (2020). Sequential sampling method using Gaussian process regression for estimating extreme structural response. *Marine Structures 72*, 102780.

Grigoriu, M. (2002). *Stochastic Calculus: Applications in Science and Engineering*. Springer Science+Business Media.

Hao, P., S. Feng, H. Liu, Y. Wang, B. Wang, and B. Wang (2021, October). A novel nested stochastic Kriging model for response noise quantification and reliability analysis. *Computer Methods in Applied Mechanics and Engineering 384*, 113941.

Hastie, T. and R. Tibshirani (1990). *Generalized additive models*, Volume 43 of *Monographs on Statistics and Applied Probability*. Chapman and Hall.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies 6*(2), 327–343.

International Electrotechnical Commission (2005). 61400-1: Wind turbines part 1: Design requirements. Standard, International Electrotechnical Commission.

Iooss, B. and M. Ribatet (2009). Global sensitivity analysis of computer models with functional inputs. *Reliability Engineering & System Safety 94*, 1194–1204.

Jonkman, B. J. (2009). Turbsim user's guide: Version 1.50. Technical Report NREL/TP-500-46198, National Renewable Energy Laboratory: Golden, Colorado.

Jonkman, B. J. and J. M. Jonkman (2013). Addendum to the user's guides for FAST, A2AD, and AeroDyn released March 2010-February 2013. Technical report, National Renewable Energy Laboratory: Golden, Colorado.

Jonkman, J., S. Butterfield, W. Musial, and G. Scott (2009). Definition of a 5-MW reference wind turbine for offshore system development. Technical Report NREL/TP-500-38060, National Renewable Energy Laboratory: Golden, Colorado.

Jonkman, J. M. and M. L. J. Buhl (2005). FAST user's guide - updated August 2005. Technical Report NREL/TP-500-38230, National Renewable Energy Laboratory: Golden, Colorado.

Klenke, A. (2013). *Wahrscheinlichkeitstheorie*. Springer Berlin Heidelberg.

Koutsourelakis, P. S., H. J. Pradlwarter, and G. I. Schuëller (2004). Reliability of structures in high dimensions, part I: algorithms and applications. *Probabilistic Engineering Mechanics 19*(4), 409–417.

Kurtz, N. and J. Song (2013). Cross-entropy-based adaptive importance sampling using Gaussian mixture. *Structural Safety 42*, 35–44.

Lemaire, M. (2009). *Structural reliability*. Wiley.

Li, S., Y. M. Ko, and E. Byon (2021). Nonparametric importance sampling for wind turbine

reliability analysis with stochastic computer models. *The Annals of Applied Statistics 15*(4), 1850 – 1871.

Lüthen, N., S. Marelli, and B. Sudret (2023). A spectral surrogate model for stochastic simulators computed from trajectory samples. *Computer Methods in Applied Mechanics and Engineering 406*(115875), 1–29.

Lüthen, N., X. Zhu, S. Marelli, and B. Sudret (2024a). UQLab user manual – Generalized Lambda Models. Technical report, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland. Report UQLab-V2.1-120.

Lüthen, N., X. Zhu, S. Marelli, and B. Sudret (2024b). UQLab user manual – Stochastic polynomial chaos expansions. Technical report, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland. Report UQLab-V2.1-121.

Lüthen, N., S. Marelli, and B. Sudret (2021). Sparse polynomial chaos expansions: Literature survey and benchmark. *SIAM/ASA Journal on Uncertainty Quantification 9*(2), 593–649.

Marelli, S. and B. Sudret (2014). UQLab: A framework for uncertainty quantification in Matlab. In *Vulnerability, Uncertainty, and Risk (Proc. 2nd Int. Conf. on Vulnerability, Risk Analysis and Management (ICVRAM2014), Liverpool, United Kingdom)*, pp. 2554–2563.

Marelli, S. and B. Sudret (2018). An active-learning algorithm that combines sparse polynomial chaos expansions and bootstrap for structural reliability analysis. *Structural Safety 75*, 67–74.

Marrel, A., B. Iooss, S. Da Veiga, and M. Ribatet (2012). Global sensitivity analysis of stochastic computer models with joint metamodels. *Statistics and Computing 22*, 833–847.

McCullagh, P. and J. Nelder (1989). *Generalized linear models*, Volume 37 of *Monographs on Statistics and Applied Probability*. Chapman and Hall.

McNeil, A. J., R. Frey, and P. Embrechts (2015). *Quantitative risk management: Concepts, techniques and tools*. Princeton, NJ: Princeton University Press.

Melchers, R. and A. Beck (2018). *Structural reliability analysis and prediction*. John Wiley & Sons.

Melchers, R. E. (1989). Importance sampling in structural systems. *Structural Safety 6*, 3–10.

Moriarty, P. (2008). Database for validation of design load extrapolation techniques. *Wind Energy 11*(6), 559–576.

Moustapha, M., S. Marelli, and B. Sudret (2022). Active learning for structural reliability: Survey, general framework and benchmark. *Structural Safety 96*, 102714.

Moutoussamy, V., S. Nanty, and B. Pauwels (2015). Emulators for stochastic simulation codes. *ESAIM: Proceedings and Surveys 48*, 116–155.

Olsson, A., G. Sandberg, and O. Dahlblom (2003). On Latin Hypercube sampling for structural reliability analysis. *Structural Safety 25*, 47–68.

Pan, Q., E. Byon, Y. M. Ko, and H. Lam (2020). Adaptive importance sampling for extreme quantile estimation with stochastic black-box computer models. *Naval Research Logistics 67*(7), 524–547.

Papaioannou, I., C. Papadimitriou, and D. Straub (2016). Sequential importance sampling for structural reliability analysis. *Struct. Saf. 62*, 66–75.

Teixeira, R., M. Nogal, and A. O'Connor (2021). Adaptive approaches in metamodel-based reliability analysis: A review. *Structural Safety 89*, 102019.

Wilensky, U. (2015). *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with Netlogo*. The MIT Press.

Zheng, X., J. Zhang, G. Tang, T. Jiang, W. Peng, and W. Yao (2022). A reliability analysis method based on quantile regression and feedforward neural network. In *12th International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE 2022)*, pp. 291–297. Emeishan, China.

Zhu, X. and B. Sudret (2020). Replication-based emulation of the response distribution of stochastic simulators using generalized lambda distributions. *International Journal for Uncertainty Quantification 10*(3), 249–275.

Zhu, X. and B. Sudret (2021). Emulation of stochastic simulators using generalized lambda models. *SIAM/ASA Journal on Uncertainty Quantification 9*(4), 1345–1380.

Zhu, X. and B. Sudret (2023). Stochastic polynomial chaos expansions to emulate stochastic simulators. *International Journal for Uncertainty Quantification 13*(2), 31–52.