Replication-based emulation of the response distribution of stochastic simulators using generalized lambda distributions

X. Zhu, B. Sudret

ETH zürich

Replication-based emulation of the response distribution of stochastic simulators using generalized lambda distributions

X. Zhu 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-2019-008
Arxiv Ref.:	https://arxiv.org/abs/1911.09067 - [stat.CO] [stat.ML]
DOI:	-
Date submitted:	November 20, 2019
Date accepted:	-

Replication-based emulation of the response distribution of stochastic simulators using generalized lambda distributions

Xujia Zhu [*1] and Bruno Sudret[†1]

[1] *Chair of Risk, Safety and Uncertainty Quantification, ETH Zürich, Stefano-Franscini-Platz 5, 8093 Zürich, Switzerland*

November 20, 2019

Abstract

Due to limited computational power, performing uncertainty quantification analyses with complex computational models can be a challenging task. This is exacerbated in the context of stochastic simulators, the response of which to a given set of input parameters, rather than being a deterministic value, is a random variable with unknown probability density function (PDF). Of interest in this paper is the construction of a surrogate that can accurately predict this response PDF for any input parameters. We suggest using a flexible distribution family – the generalized lambda distribution – to approximate the response PDF. The associated distribution parameters are cast as functions of input parameters and represented by sparse polynomial chaos expansions. To build such a surrogate model, we propose an approach based on a local inference of the response PDF at each point of the experimental design based on replicated model evaluations. Two versions of this framework are proposed and compared on analytical examples and case studies.

1 Introduction

Computer models, a.k.a. simulators, are nowadays widely used in the context of design optimization, uncertainty quantification and sensitivity analysis. A simulator is called *deterministic* if repeated runs with the same input parameters produce exactly the same output quantity of interest (QoI); for example, a finite element model of a structure with external load as input and stresses as output is a deterministic simulator. In contrast, a *stochastic simulator* provides different results when run repeatedly with the same input values. In other words, for a given vector of input parameters, the QoI of a stochastic simulator is a *random variable*, whose probability density function (PDF) is of interest. The reason for this intrinsic stochasticity is that

[*]zhu@ibk.baug.ethz.ch

[†]sudret@ethz.ch

some source of randomness inside the model, which can be represented by *latent variables*, is not taken explicitly into account within the input parameters. Therefore, if not all the relevant variables that uniquely determine the output can be fully specified, the model output remains random. Examples of stochastic simulators are encountered when evaluating the performance of a wind turbine under stochastic loads when only some characteristic values of the wind climate are known, or when predicting the price of an option in financial market with only historical data.

Such numerical models can be time-consuming: a single model evaluation may require minutes to hours of simulation, as it is the case for complex fluid dynamic codes. To alleviate the computational burden, surrogate models, a.k.a. emulators, have been successfully developed for deterministic simulators, such as Gaussian processes [1] and polynomial chaos expansions [2, 3]. The construction of surrogate models relies on a set of model evaluations, called the *experimental design* (ED). However, when it comes to stochastic simulators, one single model evaluation for a given vector of input parameter is incapable to fully characterize the associated QoI. As a result, repeated runs with the same input parameters, called *replications*, are necessary to obtain the resulting (unknown) probability distribution of the QoI. Consequently, standard surrogate modeling techniques cannot directly be applied to stochastic simulators, due to the very random nature of the output.

Large efforts have been dedicated to estimate summary scalar quantities of the random output as a function of the input parameters, such as the mean value [4–6], the standard deviation [7–9] and quantiles [10–12]. However, surrogate modeling for the entire response PDF of a stochastic code is a less mature field. Two types of approaches can be found in the literature. The first is known as the *statistical approach*. If the response PDF belongs to the exponential family, generalized linear models (GLM) can be efficiently applied [4, 13]. When the probability distribution is arbitrary and no prior knowledge on its shape is available, nonparametric estimators may be considered, notably kernel density estimators [14, 15] and projection estimators [16]. Nonparametric estimators, however, suffer from the *curse of dimensionality* [17], meaning that the necessary amount of data needed to achieve sufficient accuracy increases drastically with increasing input dimensionality.

A second approach is the *replication-based* method, which capitalizes instead on available replications to represent the response distribution through a suitably general parametric distribution. The parameters of the latter are then treated as outputs of a deterministic simulator. Conventional deterministic surrogate modeling methods may then be used to emulate these parameters as functions of the input. Note that this approach was initially proposed to estimate summary statistics [6, 11]. It has been extended to more general cases given the functional form of the parametric PDF by [18]. So far, nonparametric estimators have been used to estimate the distribution from replications [18, 19]. Thus, many replications are necessary, sometimes as many as 10^4 replications per point of the experimental design, which severely limits the applicability of such an approach.

It is worthwhile to notice that the existing methods either assume a rather restrictive shape of the distribution or require a large number of model evaluations. The present paper aims at designing a replication-based approach which will reduce the necessary amount of replications. To this end, we propose approximating the response PDF of a stochastic simulator by *generalized lambda distributions* (GLD) [20]. The distribution parameters are functions of the input and further represented by polynomial chaos (PC) expansions. To construct such a surrogate model, we present two algorithms in this paper: the first one follows the general idea of the replication-based approach, while the second enriches the former with an additional optimization step.

The paper is organized as follows. Sections 2 and 3 introduce generalized lambda distributions and polynomial chaos expansions, respectively. In Section 4, we present our novel algorithms to infer the response PDF of a stochastic simulator based on limited replicated data. Section 5 validates the proposed methods through two toy examples, and Section 6 illustrates their performance on two applications, namely a stochastic differential equation case study and a wind turbine simulation. Finally, we summarize the main findings of the paper and provide outlooks for future research in Section 7.

2 Generalized lambda distributions

2.1 Formulation

The generalized lambda distribution (GLD) is a highly flexible four-parameter probability distribution function designed to approximate most of the well-known parametric distributions [20]. Figure 1 illustrates how, with the proper choice of parameters, it can accurately approximate, normal, uniform, Student's t, exponential, lognormal, Weibull distributions, among others.

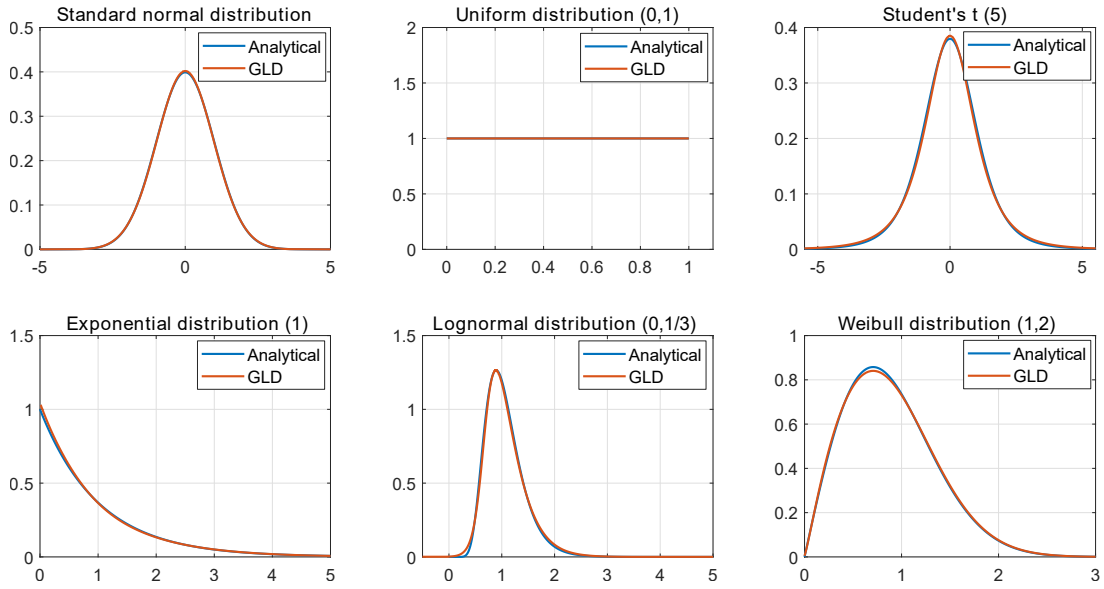


Figure 1: Visual comparison of GLD approximation of several common distributions.

Instead of providing a direct parametrization of the PDF, the GLD parametrizes the *quantile function*, which is the inverse of the cumulative distribution function $Q = F^{-1}(u)$. Therefore, Q is a non-decreasing function defined in $[0, 1]$. In this paper, we consider the GLD of the Freimer-Kollia-Mudholkar-Lin (FKML) family [21], which is defined as:

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

where λ_1 is the location parameter, λ_2 is the scaling parameter, and λ_3 and λ_4 are shape parameters. To ensure valid quantile functions, it is only required that λ_2 be positive.

Parametrizing the quantile function is equivalent to modeling the inverse probability integral transform. More precisely, the random variable Y with Q as quantile function and the random variable $Q(U)$ with $U \sim \mathcal{U}(0, 1)$ follow the same distribution. Therefore, the PDF $f_Y(y)$ of a random variable Y following a GLD can be calculated through a change of variables as follows:

$$f_Y(y) = \frac{f_U(u)}{Q'(u)} = \frac{\lambda_2}{u^{\lambda_3 - 1} + (1-u)^{\lambda_4 - 1}} \mathbb{1}_{[0,1]}(u), \text{ with } u = Q^{-1}(y), \tag{2}$$

where $\mathbb{1}_{[0,1]}$ is the indicator function. A closed form expression of Q^{-1}, and therefore of f_Y, is in general not available.

Figure 2 illustrates some PDF shapes of the FKML generalized lambda distributions in the (λ_3, λ_4) plane. It shows that distributions which belong to this family can cover a wide range of shapes that are determined by λ_3 and λ_4. For example, $\lambda_3 = \lambda_4$ produces symmetric PDFs, and $\lambda_3, \lambda_4 < 1$ yields bell-shaped distributions. Importantly, λ_3 and λ_4 control the support and the tail properties of the resulting PDF. The distribution has lower infinite support for $\lambda_3 \leq 0$ and upper infinite support for $\lambda_4 \leq 0$. In contrast, $\lambda_3 > 0$ implies that the PDF support is left-bounded and $\lambda_4 > 0$ corresponds to right-bounded distributions. More precisely, the support of the PDF, denoted by $\text{supp}\left(f_Y(y)\right) = [B_l, B_u]$, can be derived from Eq. (1) as follows:

$$\begin{aligned} B_l\left(\lambda_1, \lambda_2, \lambda_3\right) &= \begin{cases} -\infty, & \lambda_3 \leq 0 \\ \lambda_1 - \frac{1}{\lambda_2 \lambda_3}, & \lambda_3 > 0 \end{cases}, \\ B_u\left(\lambda_1, \lambda_2, \lambda_4\right) &= \begin{cases} +\infty, & \lambda_4 \leq 0 \\ \lambda_1 + \frac{1}{\lambda_2 \lambda_4}, & \lambda_4 > 0 \end{cases}. \end{aligned} \tag{3}$$

2.2 Estimation of $\boldsymbol{\lambda}$

Many estimation methods have been proposed to fit a generalized lambda distribution to data [22]. [23, 24] compared different methods through exhaustive Monte Carlo simulations with various test cases. All of the estimators show comparable performance, and none of them is shown to always outperform the others. The performance depends on the shape of the true distribution, the sample size and the goodness-of-fit criterion used for comparison. In this paper, we choose to apply the method of moments that relies on matching the four moments: mean, variance, skewness, and kurtosis [25] and the maximum likelihood estimation [26].

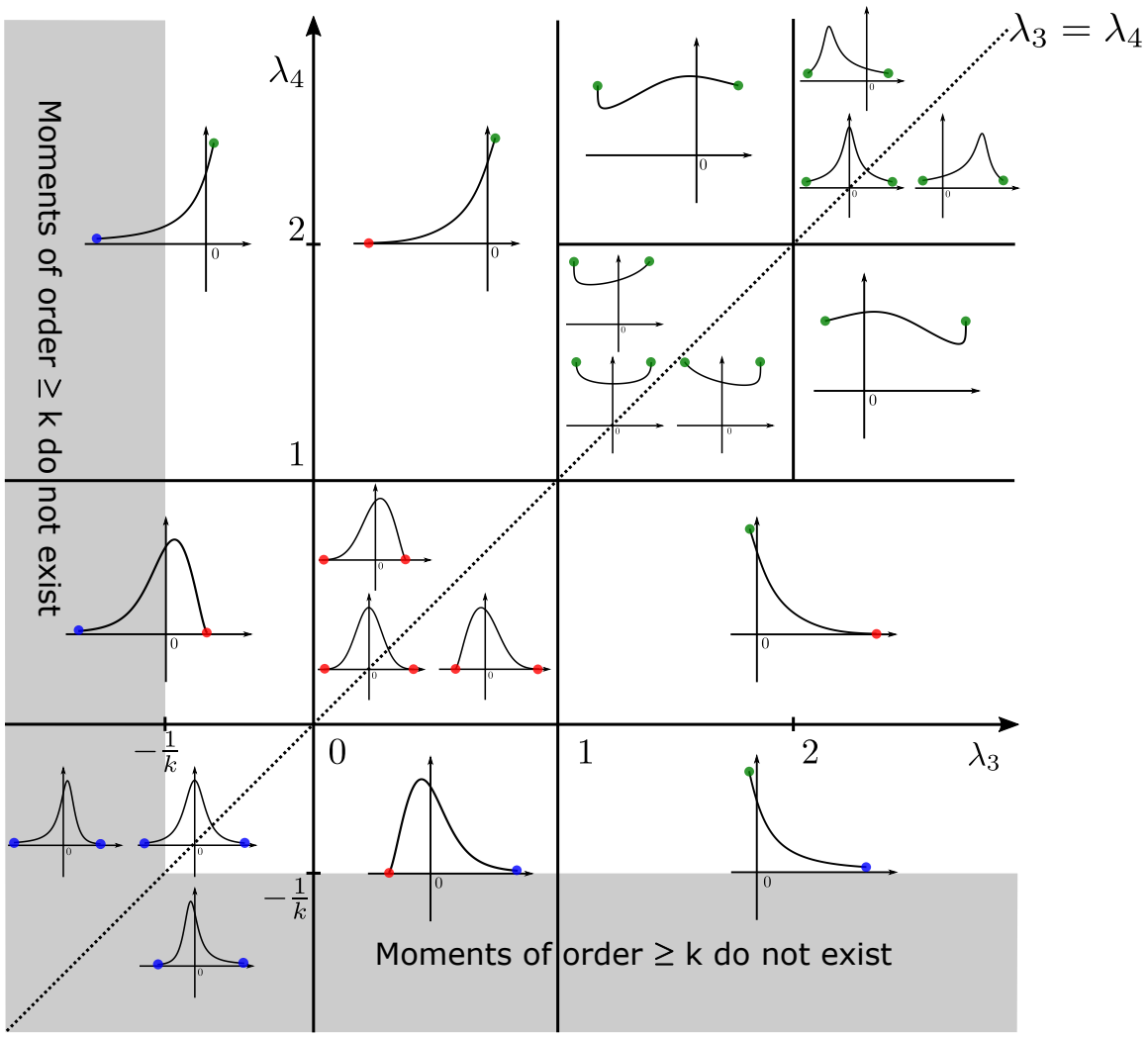


Figure 2: A graphical illustration of the shapes that can be represented by the FKML family of GLD as a function of λ_3 and λ_4. The values of λ_1 and λ_2 are set to 0 and 1, respectively. The dotted line is $\lambda_3 = \lambda_4$, which produces symmetric PDFs. The blue points indicate that the PDF has infinite support in the marked direction. In contrast, both the red and green points denote the boundary points of the PDF support. The PDF $f_Y(y) = 0$ on the red dots, whereas $f_Y(y) = 1$ on the green ones.

2.2.1 Method of moments

Following Eq. (2), the expectation of any function $g(Y)$ can be calculated as

$$\mathbb{E}\left[g(Y)\right] = \mathbb{E}\left[g(Q(U))\right] = \int_0^1 g(Q(u))du. \tag{4}$$

Accordingly, the k^{th} moment is given by

$$\begin{aligned}\mathbb{E}\left[Y^k\right] &= \int_0^1 \left(\lambda_1 + \frac{1}{\lambda_2}\left(\frac{u^{\lambda_3}-1}{\lambda_3} - \frac{(1-u)^{\lambda_4}-1}{\lambda_4}\right)\right)^k du \\ &= \int_0^1 \left(\lambda_1 - \frac{1}{\lambda_2\lambda_3} + \frac{1}{\lambda_2\lambda_4} + \frac{1}{\lambda_2}\left(\frac{u^{\lambda_3}}{\lambda_3} - \frac{(1-u)^{\lambda_4}}{\lambda_4}\right)\right)^k du,\end{aligned}$$

which is then simplified as

$$\mathbb{E}\left[Y^k\right] = \int_0^1 \left(c + \frac{1}{\lambda_2}s(u)\right)^k du, \tag{5}$$

where

$$\begin{aligned}c &\stackrel{\text{def}}{=} \lambda_1 - \frac{1}{\lambda_2\lambda_3} + \frac{1}{\lambda_2\lambda_4}, \\ s(u) &\stackrel{\text{def}}{=} \frac{u^{\lambda_3}}{\lambda_3} - \frac{(1-u)^{\lambda_4}}{\lambda_4}.\end{aligned}$$

To further elaborate Eq. (5), we calculate

$$v_k = \int_0^1 s(u)^k du = \sum_{j=0}^{k} \frac{(-1)^j}{\lambda_3^{k-j}\lambda_4^j}\binom{k}{j}\mathrm{B}(\lambda_3(k-j)+1, \lambda_4 j + 1), \tag{6}$$

where B denotes the beta function. With the help of Eq. (6), Eq. (5) can be calculated through binomial expansions. As a result, the mean, variance, skewness, and kurtosis are given by (see details in [25])

$$\mu = \mathbb{E}\left[Y\right] = \lambda_1 - \frac{1}{\lambda_2}\left(\frac{1}{\lambda_3+1} - \frac{1}{\lambda_4+1}\right), \tag{7}$$

$$\sigma^2 = \mathbb{E}\left[(Y-\mu)^2\right] = \frac{(v_2 - v_1^2)}{\lambda_2^2}, \tag{8}$$

$$\delta = \mathbb{E}\left[\left(\frac{(Y-\mu)}{\sigma}\right)^3\right] = \frac{v_3 - 3v_1v_2 + 2v_1^3}{(v_2 - v_1^2)^{\frac{3}{2}}}, \tag{9}$$

$$\kappa = \mathbb{E}\left[\left(\frac{(Y-\mu)}{\sigma}\right)^4\right] = \frac{v_4 - 4v_1v_3 + 6v_1^2v_2 - 3v_1^4}{(v_2 - v_1)^2}. \tag{10}$$

The method of moments matches these four quantities to the associated empirical moments $\left(\hat{\mu}, \hat{\sigma}^2, \hat{\delta}, \hat{\kappa}\right)$ computed from the available sample set $\mathcal{Y} = \left\{y^{(1)} \ldots, y^{(N)}\right\}$. Since v_k is only a nonlinear function of λ_3 and λ_4, the skewness and kurtosis are also functions of only λ_3 and λ_4. Therefore, the fitting procedure first estimates λ_3, λ_4 solving Eqs. (9)-(10), which can be

replaced by an optimization problem shown in Eq. (11). The remaining parameters, namely λ_1 and λ_2, are then estimated directly from Eqs. (7)-(8).

$$\left(\hat{\lambda}_3, \hat{\lambda}_4\right) = \arg\min_{\lambda_3,\lambda_4} (\delta(\lambda_3, \lambda_4) - \hat{\delta})^2 + (\kappa(\lambda_3, \lambda_4) - \hat{\kappa})^2. \tag{11}$$

Note that for $\lambda_3 \leq -0.25$ or $\lambda_4 \leq -0.25$, the generalized lambda distribution has infinite fourth order moment as shown in Figure 2 for $k = 4$. Therefore, the method of moments only provides $\lambda_3 > -0.25$ and $\lambda_4 > -0.25$.

2.2.2 Maximum likelihood estimation

Since the PDF of the generalized lambda distribution is not explicitly given, the negative log-likelihood function can only be evaluated numerically according to Eq. (2):

$$l(\boldsymbol{\lambda}) = -\sum_{i=1}^{N} \log\left(\frac{\lambda_2}{u_i^{\lambda_3-1} + (1-u_i)^{\lambda_4-1}}\right), \tag{12}$$

where

$$\text{where } u_i = Q^{-1}(y_i), \;\; y_i = Q(u_i) = \lambda_1 + \frac{1}{\lambda_2}\left(\frac{u_i^{\lambda_3} - 1}{\lambda_3} - \frac{(1-u_i)^{\lambda_4} - 1}{\lambda_4}\right). \tag{13}$$

The maximum likelihood method estimates the distribution parameters by minimizing the negative log-likelihood defined in Eq. (12):

$$\hat{\boldsymbol{\lambda}} = \arg\min_{\boldsymbol{\lambda}} l(\boldsymbol{\lambda}). \tag{14}$$

For a sample set of size N, each likelihood function evaluation requires solving N times the nonlinear equation Eq. (13). Consequently, the maximum likelihood estimation can be time-consuming for large data sets. To alleviate the computational burden, we propose the bisection method [27] to efficiently solve Eq. (13) using the property that $Q(u)$ is monotonic and defined in $[0, 1]$.

3 Polynomial chaos expansions

3.1 Introduction

A deterministic simulator is a function $\mathcal{M}$ that maps a set of input parameters $\boldsymbol{x} = (x_1, x_2, \ldots, x_M)^T \in \mathcal{D}_{\boldsymbol{X}} \subset \mathbb{R}^M$ to the output quantity of interest $y \in \mathbb{R}$. In the context of uncertainty quantification, the input parameters are modeled by a random vector $\boldsymbol{X} = (X_1, X_2, \ldots, X_M)^T$ described by its joint distribution $f_{\boldsymbol{X}}$ with support $\mathcal{D}_{\boldsymbol{X}}$. Therefore, the uncertainty in the input variables propagates through the computational model to the output, which becomes a random variable denoted by $Y = \mathcal{M}(\boldsymbol{X})$.

Under the assumption that Y has finite variance, Y belongs to the Hilbert space $\mathcal{H}$ of square-integrable functions with respect to the following inner product:

$$\langle u, v\rangle_{\mathcal{H}} = \mathbb{E}\left[u(\boldsymbol{X})v(\boldsymbol{X})\right] = \int_{\mathcal{D}_{\boldsymbol{X}}} u(\boldsymbol{x})v(\boldsymbol{x})f_{\boldsymbol{X}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}. \tag{15}$$

If the joint distribution $f_{\boldsymbol{X}}$ satisfies certain conditions [28], the set of multivariate polynomials is dense in $\mathcal{H}$. Hence, $\mathcal{H}$ is a separable Hilbert space admitting a polynomial basis $\left\{\psi_{\boldsymbol{\alpha}}(\cdot), \boldsymbol{\alpha} \in \mathbb{N}^M\right\}$ which satisfies

$$\langle \psi_{\boldsymbol{\alpha}}, \psi_{\boldsymbol{\beta}}\rangle_{\mathcal{H}} = \delta_{\boldsymbol{\alpha}\boldsymbol{\beta}}, \tag{16}$$

with δ being the Kronecker symbol defined by $\delta_{\boldsymbol{\alpha}\boldsymbol{\beta}} = 1$ if $\boldsymbol{\alpha} = \boldsymbol{\beta}$ and $\delta_{\boldsymbol{\alpha}\boldsymbol{\beta}} = 0$ otherwise. Each component α_j of $\boldsymbol{\alpha}$ indicates the polynomial degree of $\psi_{\boldsymbol{\alpha}}$ in the variable x_j. As a results, $\mathcal{M}$ can be represented by

$$\mathcal{M}(\boldsymbol{X}) = \sum_{\boldsymbol{\alpha}\in\mathbb{N}^M} a_{\boldsymbol{\alpha}}\psi_{\boldsymbol{\alpha}}(\boldsymbol{X}), \tag{17}$$

where $a_{\boldsymbol{\alpha}}$ is the coefficient associated to the basis function $\psi_{\boldsymbol{\alpha}}$. The construction of $\left\{\psi_{\boldsymbol{\alpha}}(\cdot), \boldsymbol{\alpha} \in \mathbb{N}^M\right\}$ for arbitrary $f_{\boldsymbol{X}}(\boldsymbol{x})$ is in general difficult. In this study, we consider that $\boldsymbol{X}$ has mutually independent components. Thus, the joint distribution $f_{\boldsymbol{X}}$ is expressed as

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \prod_{j=1}^{M} f_{X_j}(x_j). \tag{18}$$

In this case, each basis function $\psi_{\boldsymbol{\alpha}}$ that fulfils Eq. (16) can be obtained as the tensor product of univariate polynomials:

$$\psi_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \prod_{j=1}^{M} \phi_{\alpha_j}^{(j)}(x_j), \tag{19}$$

where $\left\{\phi_{\alpha_j}^{(j)}, \alpha_j \in \mathbb{N}\right\}$ are orthogonal polynomials with respect to the marginal distribution of f_{X_j}, i.e.,

$$\mathbb{E}\left[\phi_{\alpha_j}^{(j)}(X_j)\cdot\phi_{\beta_j}^{(j)}(X_j)\right] = \delta_{\alpha_j\beta_j}. \tag{20}$$

As a result, the problem of constructing $\psi_{\boldsymbol{\alpha}}$ is reduced to finding univariate orthogonal polynomials. Polynomials that are orthogonal with respect to some classical distributions, e.g., normal, uniform, exponential, are listed in [2]. For arbitrary marginal distributions, orthogonal polynomials can be calculated numerically through the *Stieltjes procedure* [29].

3.2 Sparse PCE

The spectral expansion in Eq. (17) is an infinite series. In practice, truncation schemes must be adopted, which leads to approximating the computational model by a finite series defined by a finite multi-index subset $\mathcal{A} \subset \mathbb{N}^M$.

$$\mathcal{M}(\boldsymbol{x}) \approx \mathcal{M}^{PC}(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}} a_{\boldsymbol{\alpha}}\psi_{\boldsymbol{\alpha}}(\boldsymbol{x}) \tag{21}$$

Once the set of candidates is selected, regression-based algorithms such as ordinary least squares [30] can be applied to the data $(\boldsymbol{\mathcal{X}}, \mathcal{Y}) = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right), i = 1, \ldots, N\right\}$ to build the surrogate model. Here $\boldsymbol{\mathcal{X}}$ denotes the experimental design of the input variables, and $\mathcal{Y}$ are the associated model outputs. One common method to select $\mathcal{A}$ is the full basis of degree p, which contains all the PC basis functions the degree of which is lower than a given value p. However, it is well known that the classical "full" PC approximation suffers from the curse of dimensionality [31], due to the quick increase of the basis size with increasing input dimension or polynomial degree. To overcome this problem, sparse polynomial chaos expansions have been proposed, which select only the most important basis functions among a candidate set [31, 32], before ordinary least squares are used to compute the coefficients. In the present work, we use the hybrid-LAR algorithm [33] implemented in the open source software UQLab [34] for building sparse PCE. The selection procedure of the algorithm is based on *least angle regression* (LAR) [35].

In the sequel, we will combine PCE with the local inference of generalized lambda distributions on each point of the experimental design with replications.

4 Infer-and-Fit algorithm and joint modeling

4.1 Introduction

We assume that the response PDF of the stochastic simulator for a given input realization $\boldsymbol{x}$ follows a generalized lambda distribution, with distribution parameters $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T$ that are functions of $\boldsymbol{x}$:

$$Y(\boldsymbol{x}) \sim \mathrm{GLD}\left(\lambda_1(\boldsymbol{x}), \lambda_2(\boldsymbol{x}), \lambda_3(\boldsymbol{x}), \lambda_4(\boldsymbol{x})\right). \tag{22}$$

Under appropriate assumptions discussed in Section 3, each component of $\boldsymbol{\lambda}(\boldsymbol{x})$ admits a PC representation. For the FKML family, $\lambda_2(\boldsymbol{x})$ is required to be positive (see Section 2), and thus the associated PC approximation is built on the natural logarithm $\log\left(\lambda_2(\boldsymbol{x})\right)$. In a nutshell, $\boldsymbol{\lambda}(\boldsymbol{x})$ are decomposed as

$$\lambda_s\left(\boldsymbol{x}\right) \approx \lambda_s^{\mathrm{PC}}\left(\boldsymbol{x}; \boldsymbol{a}\right) = \sum_{\boldsymbol{\alpha} \in \mathcal{A}_s} a_{s,\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\boldsymbol{x}), \quad s = 1, 3, 4 \tag{23}$$

$$\lambda_2\left(\boldsymbol{x}\right) \approx \lambda_2^{\mathrm{PC}}\left(\boldsymbol{x}; \boldsymbol{a}\right) = \exp\left(\sum_{\boldsymbol{\alpha} \in \mathcal{A}_2} a_{2,\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\boldsymbol{x})\right), \tag{24}$$

where $\boldsymbol{\lambda}^{\mathrm{PC}}\left(\boldsymbol{x}; \boldsymbol{a}\right)$ are the PC approximations of the unknown functions $\boldsymbol{\lambda}(\boldsymbol{x})$. The truncation sets $\{\mathcal{A}_s, s = 1, 2, 3, 4\}$ are to be defined, and the coefficients $a_{s,\boldsymbol{\alpha}}$ are the model parameters to be estimated from the samples. For the purpose of clarification, we explicitly express the model parameters $\boldsymbol{a}$ in the surrogate model $\boldsymbol{\lambda}^{\mathrm{PC}}\left(\boldsymbol{x}; \boldsymbol{a}\right)$ so as to emphasize that $\boldsymbol{a}$ are unknown and need to be estimated from the data.

4.2 Infer-and-Fit algorithm

To account for the intrinsic randomness, the stochastic simulator is repeatedly run R times for each point $\boldsymbol{x}^{(i)}$ of the experimental design $\boldsymbol{\mathcal{X}}$, and the associated output is denoted by $\mathcal{Y}^{(i)} = \left\{y^{(i,1)}, y^{(i,2)}, \ldots, y^{(i,R)}\right\}$, where the upper index (i, r) refers to the output of the r^{th} replication for the i^{th} set of input parameters in the experimental design.

Following [18, 19], one straightforward way to build a surrogate model is the Infer-and-Fit algorithm presented in Algorithm 1.

Algorithm 1 Infer-and-Fit algorithm

1: **for** $i \leftarrow 1, N$ **do**
2: $\quad \hat{\boldsymbol{\lambda}}^{(i)} \leftarrow \hat{\boldsymbol{\lambda}}\left(\mathcal{Y}^{(i)}\right)$
3: **end for**
4: $\hat{\boldsymbol{\Lambda}} \leftarrow \left(\hat{\boldsymbol{\lambda}}^{(1)}, \hat{\boldsymbol{\lambda}}^{(2)}, \ldots, \hat{\boldsymbol{\lambda}}^{(N)}\right)^T$
5: $\boldsymbol{\lambda}^{\mathrm{PC}}(\boldsymbol{x}; \tilde{\boldsymbol{a}}) \leftarrow$ Hybrid-LAR $\left(\boldsymbol{\mathcal{X}}, \hat{\boldsymbol{\Lambda}}\right)$

Function $\hat{\boldsymbol{\lambda}}(\cdot)$ in the second line of Algorithm 1 denotes an estimator of the distribution parameters based on the replications (see Section 2.2), and Hybrid-LAR in the last line is the algorithm [31] used to build sparse PCE for $\boldsymbol{\lambda}(\boldsymbol{x})$.

Algorithm 1 consists of two main steps. The first step is used to capture the intrinsic stochasticity through replications. More precisely, this inference step aims at providing an estimate $\hat{\boldsymbol{\lambda}}^{(i)}$ of the distribution parameters $\boldsymbol{\lambda}\left(\boldsymbol{x}^{(i)}\right)$ for each point of the experimental design $\boldsymbol{\mathcal{X}}$. The second step independently builds four surrogate models for the distribution parameters, based on the estimated parameters at discrete points of the experimental design. For the local inference in the first step, we test both the method of moments and the maximum likelihood estimation (a detailed comparison is presented in Section 5). Besides, in the second step, we choose to use the hybrid-LAR for sparse PCE constructions, but Algorithm 1 is not bounded to this choice: any other regression methods such as ordinary least squares [30], orthogonal matching pursuit [36], etc. can be used equivalently.

In practice, the estimator $\hat{\boldsymbol{\lambda}}^{(i)}$ is calculated from replications of finite size due to finite computational budget, and it is subject to noise. Consequently, the choice of the regression setting for building sparse PCE is advantageous because of its robustness to noise [37]. However, the generalized lambda distribution is rather flexible, so that a few samples cannot guarantee an accurate estimation [24], and none of the existing methods have been proved to produce unbiased estimators. If a consistent bias is present in the estimation, the use of regression algorithms cannot filter it out. Moreover, the four parameters of the GLD, considered as functions of the input variables, are approximated by four PCE built independently. As a result, the Infer-and-Fit algorithm qualitatively requires many replications R to achieve a good estimate (quantitative results are shown in Section 5).

The two separate steps of Algorithm 1 may be seen as two successive, independent optimization problems. First, the four parameters of the GLD are optimally fitted for each point $\boldsymbol{x}^{(i)} \in \boldsymbol{\mathcal{X}}$, leading to $\hat{\boldsymbol{\Lambda}}$. Second, coefficients of the PCE of each parameter $\lambda_s(\boldsymbol{x})$ are optimized based on the data collected in $\hat{\boldsymbol{\Lambda}}$, so as to minimize a mean squared error. Intuitively, it appears that these two successive optimizations are suboptimal. We propose to complement the Infer-and-Fit algorithm with a subsequent joint optimization.

4.3 Joint PCE-GLD fitting

To reduce the computational cost associated with the need for a large number of replications, we propose a similar approach as that of generalized linear models [4]. In this joint modeling method, PC coefficients $\boldsymbol{a}$ of the four λ_s's are calibrated from the original data $(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}})$ through a maximum likelihood estimation, instead of being calibrated from the local estimates $\hat{\boldsymbol{\Lambda}} = \left\{\hat{\boldsymbol{\Lambda}}^{(1)}, \ldots, \hat{\boldsymbol{\Lambda}}^{(N)}\right\}$, as shown in Algorithm 1.

To form such an estimator, a deeper insight into the nature of stochastic simulators is necessary. Running once the stochastic simulator for $\boldsymbol{x}$, the output value is a realization of the random variable $Y(\boldsymbol{x})$, which can also be written as $Y \mid \boldsymbol{X} = \boldsymbol{x}$. As a result, the stochastic simulator can be regarded as a *conditional sampler* with the response PDF $f_{Y|\boldsymbol{X}}(y \mid \boldsymbol{x})$. Therefore, we can write the joint distribution of $(\boldsymbol{X}, Y)$ as $f_{\boldsymbol{X},Y}(\boldsymbol{x}, y) = f_{Y|\boldsymbol{X}}(y \mid \boldsymbol{x})\, f(\boldsymbol{x})$. The GLD surrogate provides an approximation $f_{Y|\boldsymbol{X}}\left(y|\boldsymbol{\lambda}^{\mathrm{PC}}(\boldsymbol{x};\boldsymbol{a})\right)$ to the conditional PDF. Therefore, the joint PDF of the GLD model is $f_{\boldsymbol{X},Y}(\boldsymbol{x}, y; \boldsymbol{a}) = f_{\boldsymbol{X}}(\boldsymbol{x}) f_{Y|\boldsymbol{X}}\left(y|\boldsymbol{\lambda}^{\mathrm{PC}}(\boldsymbol{x};\boldsymbol{a})\right)$.

Minimizing the Kullback-Leibler divergence between $f_{\boldsymbol{X},Y}(\boldsymbol{x}, y)$ and $f_{\boldsymbol{X},Y}(\boldsymbol{x}, y; \boldsymbol{a})$ gives an appropriate approximation of the GLD surrogate to the underlying true model:

$$\boldsymbol{a}_0 = \arg\min_{\boldsymbol{a}} D_{KL}\left(f_{\boldsymbol{X},Y}(\boldsymbol{x}, y) \,\|\, f_{\boldsymbol{X},Y}(\boldsymbol{x}, y; \boldsymbol{a})\right), \tag{25}$$

where:

$$\begin{aligned} D_{KL}\left(f_{\boldsymbol{X},Y}(\boldsymbol{x}, y) \,\|\, f_{\boldsymbol{X},Y}(\boldsymbol{x}, y; \boldsymbol{a})\right) &= \int f_{\boldsymbol{X},Y}(\boldsymbol{x}, y) \log\left(\frac{f_{\boldsymbol{X},Y}(\boldsymbol{x}, y)}{f_{\boldsymbol{X},Y}(\boldsymbol{x}, y; \boldsymbol{a})}\right) \mathrm{d}\boldsymbol{x}\mathrm{d}y \\ &= -\int f_{\boldsymbol{X},Y}(\boldsymbol{x}, y) \log\left(f_{\boldsymbol{X},Y}(\boldsymbol{x}, y; \boldsymbol{a})\right) \mathrm{d}\boldsymbol{x}\mathrm{d}y + \text{const.} \\ &= -\int f_{\boldsymbol{X},Y}(\boldsymbol{x}, y) \log\left(f_{Y|\boldsymbol{X}}(y \mid \boldsymbol{x}; \boldsymbol{a})\, f_{\boldsymbol{X}}(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}\mathrm{d}y + \text{const.} \end{aligned} \tag{26}$$

Since $f_{\boldsymbol{X}}$ does not contain the model parameters $\boldsymbol{a}$, Eq. (25) can be further simplified as

$$\boldsymbol{a}_0 = \arg\min_{\boldsymbol{a}} -\int f_{\boldsymbol{X},Y}(\boldsymbol{x}, y) \log\left(f_{Y|\boldsymbol{X}}(y \mid \boldsymbol{\lambda}(\boldsymbol{x};\boldsymbol{a}))\right) \mathrm{d}\boldsymbol{x}\mathrm{d}y. \tag{27}$$

Thus the model parameters $\boldsymbol{a}_0$ can be obtained by minimizing the following function:

$$l(\boldsymbol{a}) \overset{\text{def}}{=} -\mathbb{E}_{\boldsymbol{X},Y}\left[\log\left(f_{Y|\boldsymbol{X}}\left(Y\middle|\boldsymbol{\lambda}^{\mathrm{PC}}(\boldsymbol{X};\boldsymbol{a})\right)\right)\right]. \tag{28}$$

Note that if the true underlying model can be expressed as the form $f_{Y|\boldsymbol{X}}\left(y|\boldsymbol{\lambda}^{\mathrm{PC}}\left(\boldsymbol{x};\boldsymbol{a}_{\mathrm{true}}\right)\right)$, $\boldsymbol{a}_0$ from Eq. (27) guarantees that $f_{Y|\boldsymbol{X}}\left(y|\boldsymbol{\lambda}^{\mathrm{PC}}(\boldsymbol{x};\boldsymbol{a}_0)\right)$ is the same as that of the true model $\forall \boldsymbol{x} \in \mathcal{D}_{\boldsymbol{x}}$.

To estimate $\boldsymbol{a}_0$, the expectation in Eq. (28) is replaced by some estimator. In most cases, a sample based empirical average $\frac{1}{N}\sum_{i=1}^{N}\log\left(f_{Y|\boldsymbol{X}}\left(y^{(i)}\middle|\boldsymbol{\lambda}^{\mathrm{PC}}\left(\boldsymbol{x}^{(i)};\boldsymbol{a}\right)\right)\right)$ is used, where $\left\{\left(\boldsymbol{x}^{(i)},y^{(i)}\right)\right\}_{i=1}^{N}$ are drawn independently from the joint distribution $f_{\boldsymbol{X},Y}\left(\boldsymbol{x},y\right)$. For a given r, $\left\{\left(\boldsymbol{x}^{(i)},y^{(i,r)}\right)\right\}_{i=1}^{N}$ are independent samples, and thus it is natural to consider the estimator

$$\hat{l}^{(r)}(\boldsymbol{a}) = \frac{1}{N}\sum_{i=1}^{N} -\log\left(f_{Y|\boldsymbol{X}}\left(y^{(i,r)}\middle|\boldsymbol{\lambda}^{\mathrm{PC}}\left(\boldsymbol{x}^{(i)};\boldsymbol{a}\right)\right)\right) \tag{29}$$

to replace the expectation in Eq. (28). Note that $\left\{\hat{l}^{(r)}(\boldsymbol{a})\right\}_{r=1}^{R}$ are unbiased estimators of $l(\boldsymbol{a})$. Therefore, the following estimator $\hat{l}(\boldsymbol{a})$ is also unbiased:

$$\hat{l}(\boldsymbol{a}) = \frac{1}{R}\sum_{r=1}^{R}\hat{l}^{(r)}(\boldsymbol{a}). \tag{30}$$

For a given $\boldsymbol{a}$, $\left\{\hat{l}^{(r)}(\boldsymbol{a})\right\}_{r=1}^{R}$ have the same variance $\sigma^2(\boldsymbol{a})$, because they are the same estimator applied to different samples $\left\{\left(\boldsymbol{x}^{(1)},y^{(1,r)}\right),\ldots,\left(\boldsymbol{x}^{(N)},y^{(N,r)}\right)\right\}$, generated by the same scheme and indexed by r. Nevertheless, these estimators of $l(\boldsymbol{a})$ are not mutually independent due to the presence of replications. Hence, the variance of $\hat{l}(\boldsymbol{a})$ is calculated as follows:

$$\begin{aligned}\mathrm{Var}\left[\hat{l}(\boldsymbol{a})\right] &= \mathrm{Var}\left[\frac{1}{R}\sum_{r=1}^{R}\hat{l}^{(r)}(\boldsymbol{a})\right] \\ &= \frac{1}{R^2}\left(\sum_{r=1}^{R}\mathrm{Var}\left[\hat{l}^{(r)}(\boldsymbol{a})\right] + \sum_{r_1=1}^{R}\sum_{r_2\neq r_1}\mathrm{Cov}\left[\hat{l}^{(r_1)}(\boldsymbol{a}),\hat{l}^{(r_2)}(\boldsymbol{a})\right]\right).\end{aligned} \tag{31}$$

Using the Cauchy-Schwartz inequality, we have

$$\begin{aligned}\mathrm{Var}\left[\hat{l}(\boldsymbol{a})\right] &\leq \frac{1}{R^2}\left(\sum_{r=1}^{R}\mathrm{Var}\left[\hat{l}^{(r)}(\boldsymbol{a})\right] + \sum_{r_1=1}^{R}\sum_{r_2\neq r_1}\sqrt{\mathrm{Var}\left[\hat{l}^{(r_1)}(\boldsymbol{a})\right]}\sqrt{\mathrm{Var}\left[\hat{l}^{(r_2)}(\boldsymbol{a})\right]}\right) \\ &= \frac{1}{R^2}\left(R\cdot\sigma^2(\boldsymbol{a}) + R(R-1)\cdot\sigma^2(\boldsymbol{a})\right) = \sigma^2(\boldsymbol{a}).\end{aligned} \tag{32}$$

The inequality becomes an equality if and only if $\hat{l}^{(r_1)}(\boldsymbol{a})$ is an affine function of $\hat{l}^{(r_2)}(\boldsymbol{a})$. In the context of stochastic simulators, $\hat{l}^{(r_1)}(\boldsymbol{a})$ is not a deterministic function of $\hat{l}^{(r_2)}(\boldsymbol{a})$. Therefore, for a given $\boldsymbol{a}$, $\hat{l}(\boldsymbol{a})$ has less variance than any estimator of the set $\left\{\hat{l}^{(r)}(\boldsymbol{a})\right\}_{r=1}^{R}$, so it is a better estimator in terms of variance. Replacing the expectation in Eq. (28) by $\hat{l}(\boldsymbol{a})$, we end up with a new estimator:

$$\hat{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}}\hat{l}\left(\boldsymbol{a}\right), \tag{33}$$

where

$$\hat{l}\left(\boldsymbol{a}\right) \stackrel{\mathrm{def}}{=} \sum_{i}^{N}\sum_{r}^{R} -\log\left(f_{Y|X}\left(y^{(i,r)}\middle|\boldsymbol{\lambda}^{\mathrm{PC}}\left(\boldsymbol{x}^{(i)};\boldsymbol{a}\right)\right)\right). \tag{34}$$

This estimator by itself does not produce sparsity in the PC representations, meaning that the basis functions for $\boldsymbol{\lambda}^{\mathrm{PC}}(\boldsymbol{x};\boldsymbol{a})$ should be predefined before optimizing Eq. (33). To this end, we first exploit Algorithm 1 to identify a sparse truncation scheme $\{\mathcal{A}_s, s=1,\dots,4\}$ for each component of $\boldsymbol{\lambda}$ in terms of the input vector $\boldsymbol{x}$. Then, we keep this representation and optimize the associated coefficients over the $R \times N$ data points globally (by joint likelihood maximization Eq. (33)), instead of separately. Therefore, this new procedure, which is summarized in Algorithm 2, can be considered as a *refinement* of Algorithm 1, which is expected to improve the surrogate quality with respect to the number of available replications.

Algorithm 2 Joint PCE-GLD fitting

1: Apply Algorithm 1 to get the sparse PCE truncation schemes $\{\mathcal{A}_s, s=1,\dots,4\}$, and the associated coefficients $\tilde{\boldsymbol{a}}$

2: $\hat{\boldsymbol{a}} \leftarrow \arg\min_{\boldsymbol{a}} \hat{l}\,(\boldsymbol{a})$, where $\hat{l}\,(\boldsymbol{a})$ is defined in Eq. (34) and

$$\lambda_s^{\mathrm{PC}}(\boldsymbol{x};\boldsymbol{a}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}_s} a_{s,\boldsymbol{\alpha}}\psi_{\boldsymbol{\alpha}}(\boldsymbol{x}) \quad s=1,3,4 \tag{35}$$

$$\lambda_2^{\mathrm{PC}}(\boldsymbol{x};\boldsymbol{a}) = \exp\left(\sum_{\boldsymbol{\alpha}\in\mathcal{A}_2} a_{2,\boldsymbol{\alpha}}\psi_{\boldsymbol{\alpha}}(\boldsymbol{x})\right) \tag{36}$$

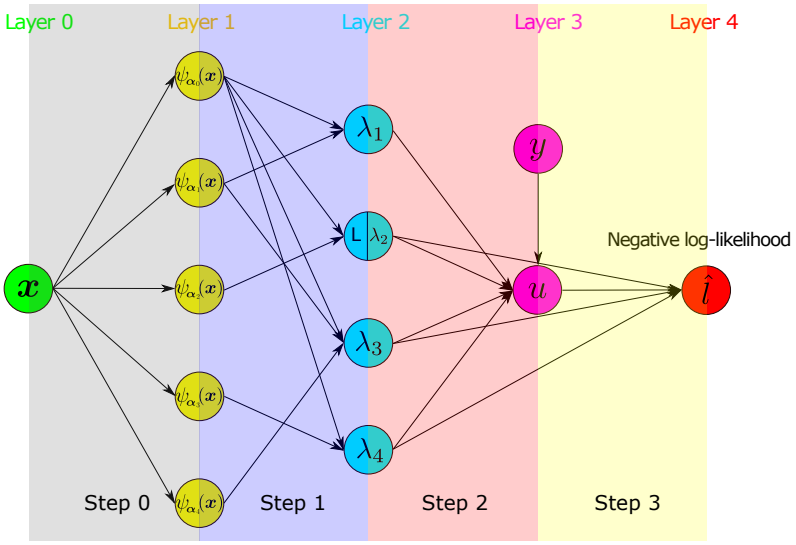


Figure 3: Flow chart of the negative log-likelihood calculation

In the second step of Algorithm 2, the log-likelihood $\hat{l}(\boldsymbol{a})$ needs to be evaluated with given PC coefficients $\boldsymbol{a}$ for each data point $\left(\boldsymbol{x}^{(i)}, y^{(i,r)}\right)$. The computation details are illustrated in Figure 3 and described here. The preliminary step (referred to as Step 0 in Figure 3) evaluates the basis functions $\{\psi_{\boldsymbol{\alpha}}, \boldsymbol{\alpha}\in\mathcal{A}_s\}$ at all $\boldsymbol{x}^{(i)} \in \mathcal{X}$. Step 1 calculates the distribution parameters $\boldsymbol{\lambda}^{(i)} = \boldsymbol{\lambda}^{\mathrm{PC}}\left(\boldsymbol{x}^{(i)};\boldsymbol{a}\right)$ according to Eqs. (35)-(36), in which the model parameters $\boldsymbol{a}$ are used. The two layers involved in this step (Layer 1 and Layer 2 in Figure 3) are not fully connected because

the sparse basis sets are independently selected for each components of $\boldsymbol{\lambda}^{\mathrm{PC}}(\boldsymbol{x};\boldsymbol{a})$ in Algorithm 1. Step 2 solves the nonlinear equation $u_{i,r} = Q^{-1}\left(y^{(i,r)}\right)$, where the current values of $\boldsymbol{\lambda}^{(i)}$'s are used, see Eq. (1). The nonlinear equation is explicitly written as

$$y^{(i,r)} = \lambda_1^{(i)} + \frac{1}{\lambda_2^{(i)}}\left(\frac{u_{i,r}^{\lambda_3^{(i)}} - 1}{\lambda_3^{(i)}} - \frac{(1-u_{i,r})^{\lambda_4^{(i)}} - 1}{\lambda_4^{(i)}}\right). \tag{37}$$

Eventually, Step 3 computes the negative log-likelihood using Eq. (2). More precisely, we have

$$-\log\left(f_{Y|\boldsymbol{X}}\left(y^{(i,r)}\middle|\boldsymbol{\lambda}^{(i)}\right)\right) = \log\left(\frac{u_{i,r}^{\lambda_3^{(i)}-1} + (1-u_{i,r})^{\lambda_4^{(i)}-1}}{\lambda_2^{(i)}}\right). \tag{38}$$

The optimization problem in the second step of Algorithm 2 is not only highly nonlinear but also subject to complex constraints. As discussed in Section 2, the FKML family can produce PDFs with bounded support (see Eq. (3)), which implies that the negative log-likelihood function will take value $+\infty$ if the data are outside the support. To avoid numerical issues, constraints need to be introduced, and the complete optimization problem becomes

$$\hat{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}} \hat{l}(\boldsymbol{a}) \tag{39}$$

$$\text{such that } \forall i \quad \begin{cases} B_l\left(\lambda_1^{\mathrm{PC}}\left(\boldsymbol{x}^{(i)};\boldsymbol{a}\right), \lambda_2^{\mathrm{PC}}\left(\boldsymbol{x}^{(i)};\boldsymbol{a}\right), \lambda_3^{\mathrm{PC}}\left(\boldsymbol{x}^{(i)};\boldsymbol{a}\right)\right) \leq \min_r y^{(i,r)} \\ B_u\left(\lambda_1^{\mathrm{PC}}\left(\boldsymbol{x}^{(i)};\boldsymbol{a}\right), \lambda_2^{\mathrm{PC}}\left(\boldsymbol{x}^{(i)};\boldsymbol{a}\right), \lambda_4^{\mathrm{PC}}\left(\boldsymbol{x}^{(i)};\boldsymbol{a}\right)\right) \geq \max_r y^{(i,r)} \end{cases}, \tag{40}$$

where B_l and B_u are computed from Eq. (3).

In general, the fact that the negative log-likelihood function can reach $+\infty$ is not a problem because Eq. (39) is a minimization problem. Therefore, we can always treat the optimization problem as unconstrained. However, numerical issues can occur when applying unconstrained derivative-based algorithms. For this reason, we choose to use the derivative-based algorithm *trust region without constraints* [38] in the first place. If it does not converge, which implies that some constraints are activated, the constrained (1+1)-CMA-ES algorithm [39] available in UQLab [40] is used instead.

For derivative-based algorithms, the choice of a relevant starting point is important to ensure convergence. In the proposed method, we use the coefficients resulting from the Infer-and-Fit algorithm as the starting point, namely $\tilde{\boldsymbol{a}}$. However, $\tilde{\boldsymbol{a}}$ is generally not guaranteed to be feasible. If it violates the inequality conditions in Eq. (40), additional operations are necessary to have a feasible starting point, see details in Appendix A.1.

When applying derivative-based optimizers to solve Eq. (39), using finite difference to calculate gradients would be time-consuming and inaccurate. This is because the likelihood (Eq. (34)) is expensive to evaluate, especially considering that NR nonlinear equations (Eq. (37)) that need to be solved. To alleviate the computational burden, we derived analytical expressions of the derivatives through implicit differentiations of Eqs. (1)-(2) and the chain rule (see Appendix A.2

for details). Besides, the Hessian matrix (second order derivatives of $\hat{l}$ with respect to $\boldsymbol{a}$) has also been derived. As a result, each iteration of the trust region algorithm only evaluates once the likelihood function $\hat{l}(\boldsymbol{a})$.

5 Analytical examples

In this section, we investigate the performance of the Infer-and-Fit and of the joint modeling algorithm using two analytical examples. The examples are built such that the PDF of $Y(\boldsymbol{x})$ is known but does not follow the generalized lambda distribution, so as to test the flexibility of the proposed approaches. As an inference tool for the first algorithm, we apply both the method of moments and the maximum likelihood estimation to get the values $\hat{\boldsymbol{\lambda}}^{(i)}$ for each $\boldsymbol{x}^{(i)} \in \boldsymbol{\mathcal{X}}$. The associated surrogate models built from the Infer-and-Fit algorithm are respectively denoted by *GLD MM* and *GLD MLE*. Similarly, the joint PCE-GLD algorithm provides another two models denoted by *GLD joint_MM* and *GLD joint_MLE*. Note that when building these two joint models following Algorithm 2, both of them rely on solving the optimization problem in Eq. (39). However, results are not identical because the sparse truncation sets $\{\mathcal{A}_s, s = 1, \ldots, 4\}$ for $\boldsymbol{\lambda}^{\mathrm{PC}}(\boldsymbol{x}; \boldsymbol{a})$ as well as the starting points for the optimization generally differ.

The error measure between the emulated PDF and the true one is computed using the Hellinger distance, which is then averaged over all possible $\boldsymbol{x}$. More precisely, we define

$$\epsilon = \mathbb{E}_{\boldsymbol{X}}\left[d_{\mathrm{HD}}\left(f_{Y|\boldsymbol{X}}(y \mid \boldsymbol{X}), f_{Y|\boldsymbol{X}}\left(y \mid \boldsymbol{\lambda}^{\mathrm{PC}}\left(\boldsymbol{X}; \hat{\boldsymbol{a}}\right)\right)\right)\right]. \tag{41}$$

It is reminded that the Hellinger distance between two continuous PDFs p and q reads

$$\begin{aligned} d_{\mathrm{HD}}\left(p(y), q(y)\right) &= \frac{1}{\sqrt{2}}\|\sqrt{p(y)} - \sqrt{q(y)}\|_2 \\ &= \sqrt{\frac{1}{2}\int_{-\infty}^{+\infty}\left(\sqrt{p(y)} - \sqrt{q(y)}\right)^2 \mathrm{d}y} = \sqrt{1 - \int_{-\infty}^{+\infty}\sqrt{p(y)q(y)}\mathrm{d}y}. \end{aligned} \tag{42}$$

Another natural choice for measuring the distance between two PDFs would have been the KL divergence. However, $D_{KL}(p\|q)$ tends to $+\infty$ if $\mathrm{supp}(p) \setminus \mathrm{supp}(q)$ has non zero probability with respect to p, which is not suitable for the comparison.

In practice, the integral in Eq. (42) is computed using numerical integration. In this paper, we restrict the integral interval from $(-\infty, +\infty)$ to $[Q_p^{0.001}, Q_p^{0.999}] \cup [Q_q^{0.001}, Q_q^{0.999}]$, where $Q_p^{0.001}$ and $Q_p^{0.999}$ denote the 0.1% and 99.9% quantiles of a random variable having p as PDF (similar notations are used for q). Note that this is feasible here because the two densities in Eq. (41) we want to compare have analytical expressions for the specific examples handled.

To calculate the expectation in Eq. (41), quasi Monte Carlo simulation is used with $N_{\mathrm{test}} = 1,000$ samples generated by the Sobol' sequence [41] in the input space. The Sobol' sequence sampler is also used to draw the experimental design (ED). To study the performance of the proposed methods, data are generated for various combinations of the experimental design size N and

the amount of replications R per ED point. Each scenario is run 100 times with independent experimental designs to account for statistical uncertainty. Error estimates for each scenario (N, R) are thus represented by box plots.

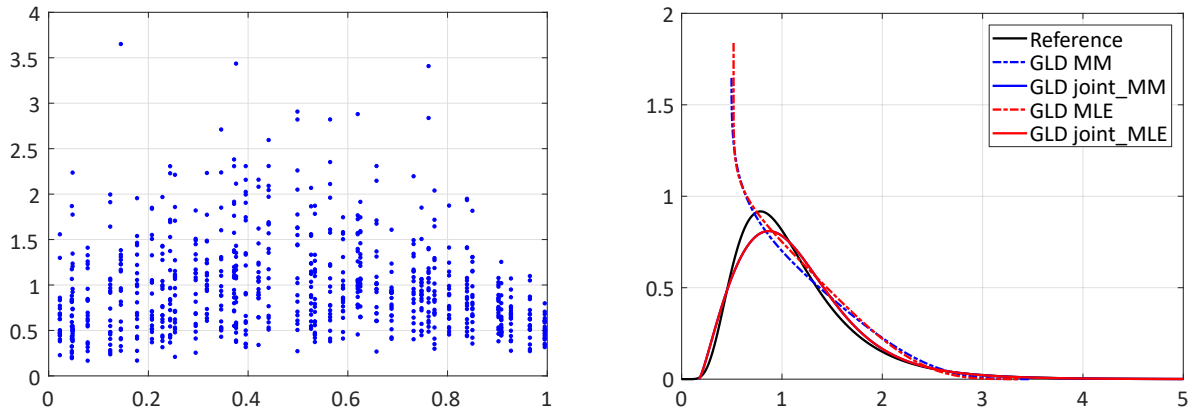
5.1 Example 1: a one-dimensional simulator

The first example is defined as follows:

$$Y(X,\omega) = \sin\left(\frac{2\pi}{3}X + \frac{\pi}{6}\right) \cdot (Z_1(\omega) \cdot Z_2(\omega))^{\cos(X)}, \tag{43}$$

where $X \sim \mathcal{U}(0,1)$ is the input parameter, and $Z_1(\omega) \sim \mathcal{LN}(0, 0.25)$ and $Z_2(\omega) \sim \mathcal{LN}(0, 0.5)$ are latent variables following lognormal distributions. Under this definition, $Y(x,\omega)$ follows a lognormal distribution $\mathcal{LN}(\ell(x), \zeta(x))$ with $\ell(x) = \log\left(\sin\left(\frac{2\pi}{3}x + \frac{\pi}{6}\right)\right)$ and $\zeta(x) = \sqrt{\frac{3}{8}}\cos(x)$, $x \in [0,1]$. As mentioned in Section 2, the lognormal distribution, which is widely used in engineering, can be approximated by the generalized lambda distribution. The nonlinearity in its parameters leads to nonlinear functions of $\boldsymbol{\lambda}(\boldsymbol{x})$ in the GLD approximation, and thus the PC representations $\boldsymbol{\lambda}^{\mathrm{PC}}(\boldsymbol{x})$ are also nonlinear.

Figure 4 shows one realization of an experimental design of $N = 40$ and $R = 20$ for each point and the predicted PDF of the four surrogate models for $X = 0.5$. We observe that the two models *GLD MM* and *GLD MLE* built using the Infer-and-Fit algorithm cannot capture the shape of the true distribution. In contrast, the two joint models *GLD joint_MM* and *GLD joint_MLE* produce a PDF that not only has the correct shape but also is an accurate approximation of the underlying distribution. We remark that with the data illustrated in Figure 4, *GLD joint_MM* and *GLD joint_MLE* are identical, implying that even though *GLD MM* and *GLD MLE* are different, their associated joint models can still be identical if they select the same sparse truncation sets $\{\mathcal{A}_s, s = 1, \ldots, 4\}$. Therefore, the selected algorithm is appears to be not too sensitive to the starting point.



(a) Generated data

(b) PDF prediction for $X = 0.5$

Figure 4: Example 1 – 40 ED points and 20 replications

Figures 5 to 6 show quantitative comparisons of the convergence behavior of the four models. It turns out that in general all the GLD models converge when increasing the size of experimental design N and the number of replications R. Moreover, the two joint models outperform the models built of the Infer-and-Fit approach, especially when only a few replications are available.

In the case with only 20 replications (Figure 5), the convergence behavior of *GLD MM* and *GLD MLE* shows a weak dependence on N. This is because in the first step of Algorithm 1, estimators $\hat{\boldsymbol{\lambda}}^{(i)}$ from both the method of moments and the maximum likelihood estimation might be biased. Then regression used in the second step is not able to filter the bias. Moreover, a few replications lead to high variance of the estimators, which together with the bias explains the non convergent behavior of *GLD MM* and *GLD MLE*. When increasing the number of replications, the bias becomes less significant and the variance of the error decreases. Therefore, ϵ decreases with increasing N for *GLD MM* and *GLD MLE* in Figure 6.

In contrast, *GLD joint_MLE* exhibits a fast error decay even with a small number of replications. This is because all the available data are used at once to estimate the model parameters, which reduces both the bias and the variance. *GLD joint_MM* appears to provide less accurate PDF estimation than *GLD joint_MLE*, which is due to the less appropriate truncation scheme selected by *GLD MM*. Nevertheless, *GLD joint_MM* still improves the results of *GLD MM* and outperforms *GLD MLE*.

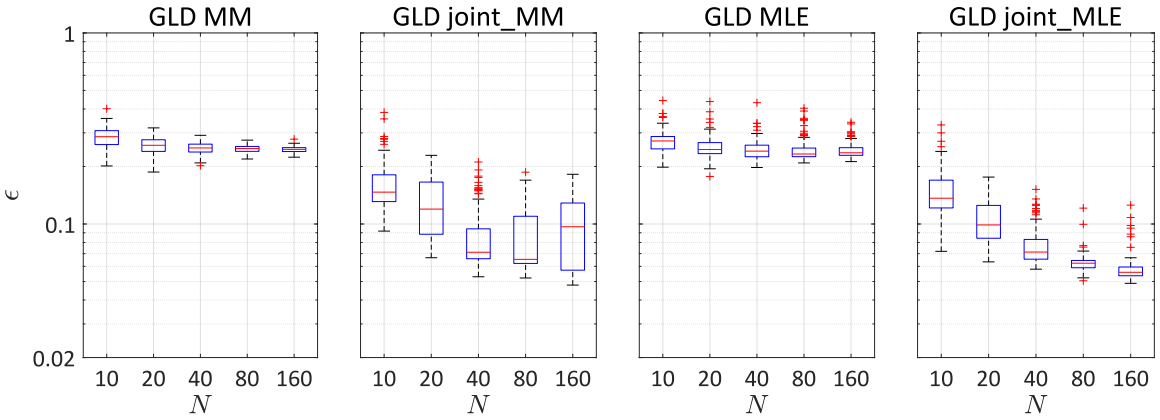


Figure 5: Example 1 – Hellinger distance between the surrogate model built with $R = 20$ and the true response PDF, averaged over $\mathcal{X}_{\text{test}}$ (log-scale).

We have run the simulation for $N = \{10, 20, 40, 80, 160\}$ and $R = \{10, 20, 40, 80, 160\}$. Figure 7 summarizes the total number of model runs NR (only up to 1,600) against the error measure. The results are consistent with what we have observed in the case of fixed number of replications. More precisely, the two models built with the joint modeling algorithm outperform those based on the Infer-and-Fit algorithm: with 400 models runs, *GLD joint_MM* and *GLD joint_MLE* provide more accurate PDF estimations than *GLD MM* and *GLD MLE* with 1,600 model evaluations.

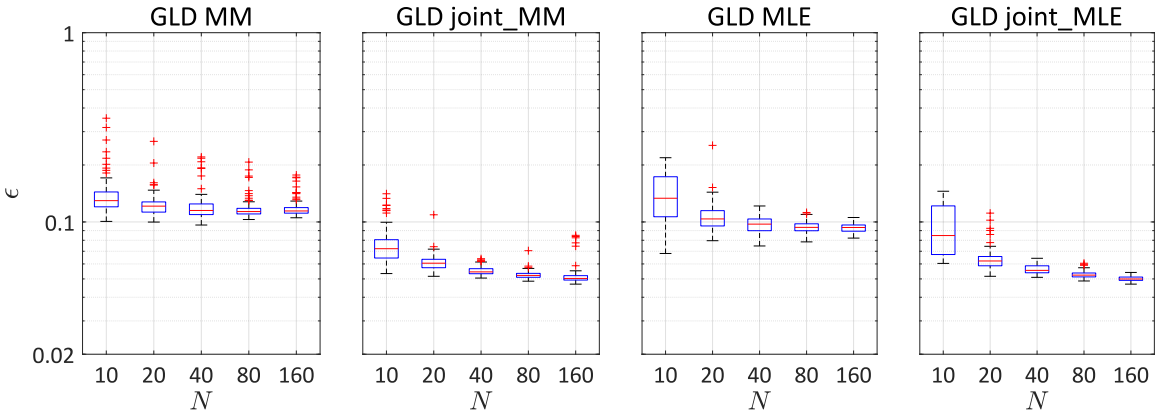


Figure 6: Example 1 – Hellinger distance between the surrogate model built with $R = 80$ and the true response PDF, averaged over $\mathcal{X}_{\text{test}}$ (log-scale).

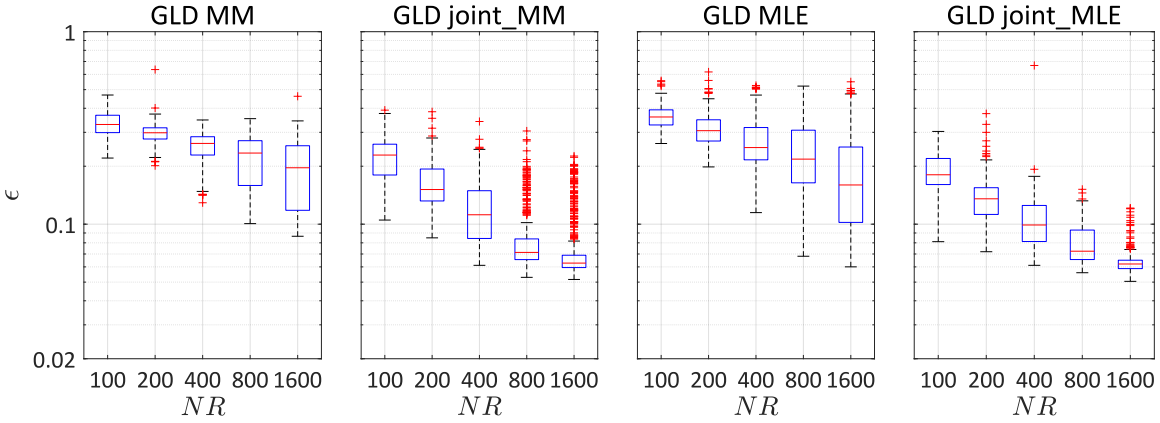


Figure 7: Example 1 – Hellinger distance between surrogate models built with different total number of model runs and the true response PDF, averaged over $\mathcal{X}_{\text{test}}$ (log-scale).

5.2 Example 2: a five-dimensional simulator

The second analytical example is defined as follows:

$$Y(\boldsymbol{X}, \omega) = \mu(\boldsymbol{X}) + \sigma(\boldsymbol{X}) \cdot Z(\omega), \tag{44}$$

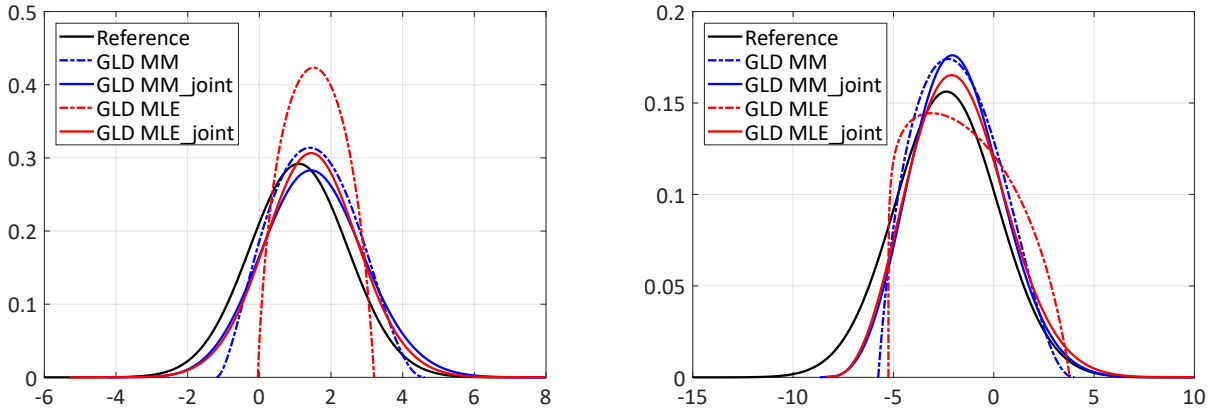
where $Z(\omega) \sim \mathcal{N}(0,1)$ is the latent variable that represents the source of randomness, and $\boldsymbol{X}$ is a five-dimensional random vector, with independent components having uniform distribution $\mathcal{U}(0,1)$. $Y(\boldsymbol{x})$ is a Gaussian random variable with mean $\mu(\boldsymbol{x})$ and standard deviation $\sigma(\boldsymbol{x})$. In this example, the mean function $\mu(\boldsymbol{x})$ reads

$$\mu(\boldsymbol{x}) = 3 - \sum_{j=1}^{5} j x_j + \frac{1}{5}\sum_{j=1}^{5} j x_j^3 + \frac{1}{15}\log\left(1 + \sum_{j=1}^{5} j(x_j^2 + x_j^4)\right) + x_1\, x_2^2 - x_5\, x_3 + x_2\, x_4, \tag{45}$$

and the standard deviation $\sigma(\boldsymbol{x})$ is given by

$$\sigma(\boldsymbol{x}) = \exp\left(\frac{1}{4}\sum_{j=1}^{5} x_j\right), \tag{46}$$

which implies a strong heteroskedastic effect with a highly nonlinear mean function. This example is used to show the performance of the proposed methods in moderate dimensional problems.



(a) PDF at $\boldsymbol{x} = (0.25, 0.25, 0.25, 0.25, 0.25)^T$ (b) PDF at $\boldsymbol{x} = (0.75, 0.75, 0.75, 0.75, 0.75)^T$

Figure 8: Example 2 – PDF predictions with an experimental design of size 50 and 25 replications.

Similar to the previous example, the GLD models demonstrate a convergent behavior, as illustrated in Figures 9 to 11. The two joint models yield more accurate estimates than those built with the Infer-and-Fit algorithm. In the case of a few replications, both *GLD MM* and *GLD MLE* fail to capture the shape of the PDF (Figure 8), and thus converge rather slowly with respect to N (see Figure 9). In contrast, the two joint models are less sensitive to the number of replications, and their performance mainly depends on the ED size. Unlike the first example, using the method of moment turns out to produce slightly more accurate estimates when applying the Infer-and-Fit algorithm. However, the parametric estimation methods employed to get $\hat{\boldsymbol{\lambda}}^{(i)}$ do not have a significant influence on the accuracy of the joint algorithm: *GLD joint_MM* and *GLD joint_MLE* show very similar convergence.

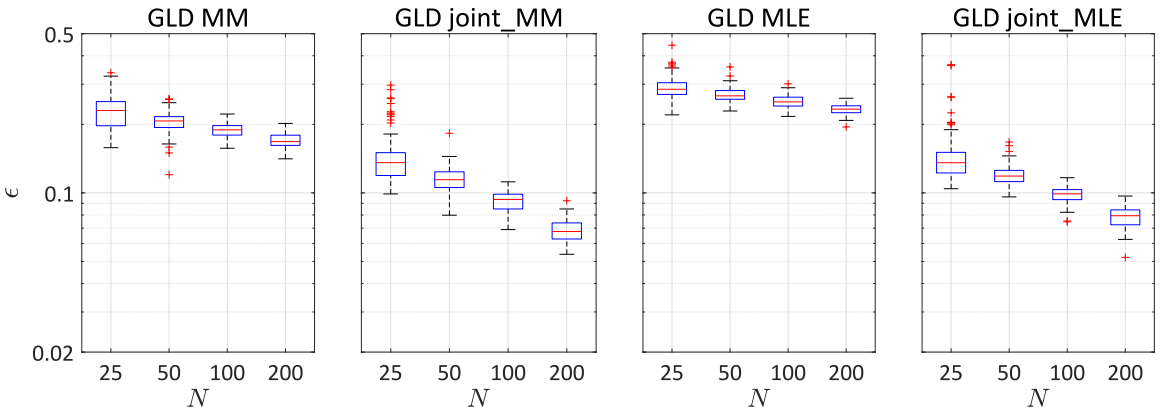


Figure 9: Example 2 – Hellinger distance between the surrogate model built with $R = 25$ and the true response PDF, averaged over $\mathcal{X}_{\text{test}}$ (log-scale)

In this section, only the error measure based on the Hellinger distance is reported for convergence studies. Nevertheless, quantitative comparisons using other metrics such as the Kolmogorov-

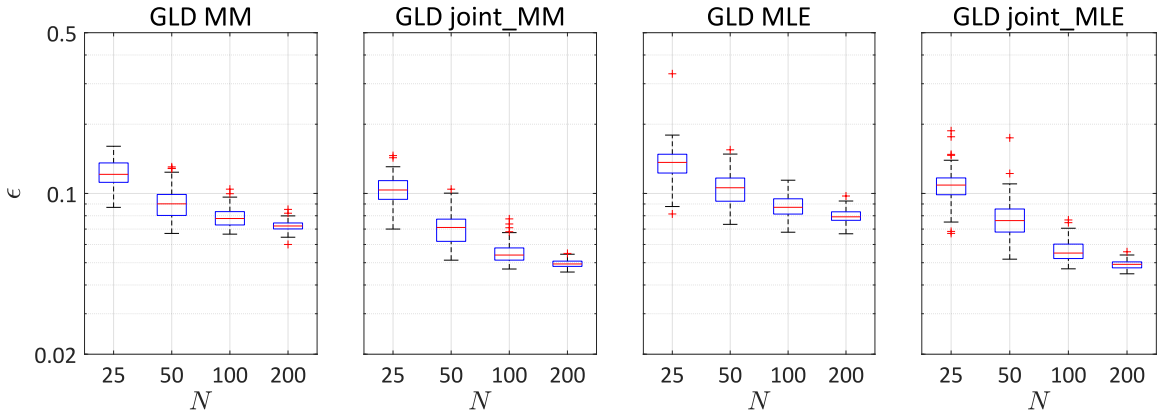


Figure 10: Example 2 – Hellinger distance between the surrogate model built with $R = 100$ and the true response PDF, averaged over $\mathcal{X}_{\text{test}}$ (log-scale)

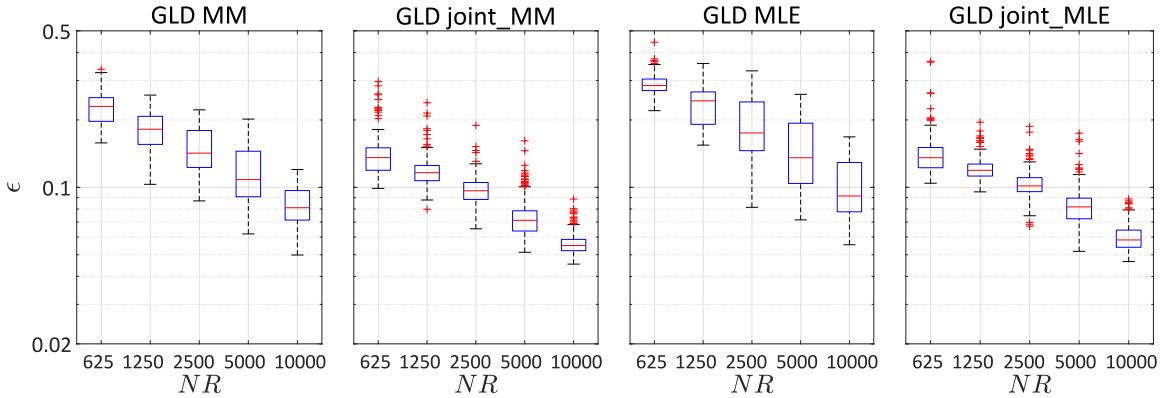


Figure 11: Example 2 – Hellinger distance between surrogate models built with different total number of model runs and the true response PDF, averaged over $\mathcal{X}_{\text{test}}$ (log-scale).

Smirnov distance, the mean value and the 95% quantile of the predicted distributions show similar trends.

6 Applications

6.1 Stochastic differential equation

Stochastic differential equations (SDE) are widely used to model the evolution of complex systems in many fields, e.g., finance [42], epidemics [43], and meteorology [44]. Due to the stochastic process (e.g., Wiener processes) involved in a SDE, the associated solution is also a stochastic process. As a results, when fixing the parameters of a SDE, any scalar-valued deterministic function of the solution process produces a random variable, which can be regarded as a stochastic simulator. In this case study, we consider the example proposed by [45], the governing equation

of which reads

$$\mathrm{d}Y_t = (X_1 - Y_t)\mathrm{d}t + (\nu Y_t + 1)X_2\,\mathrm{d}W_t, \tag{47}$$

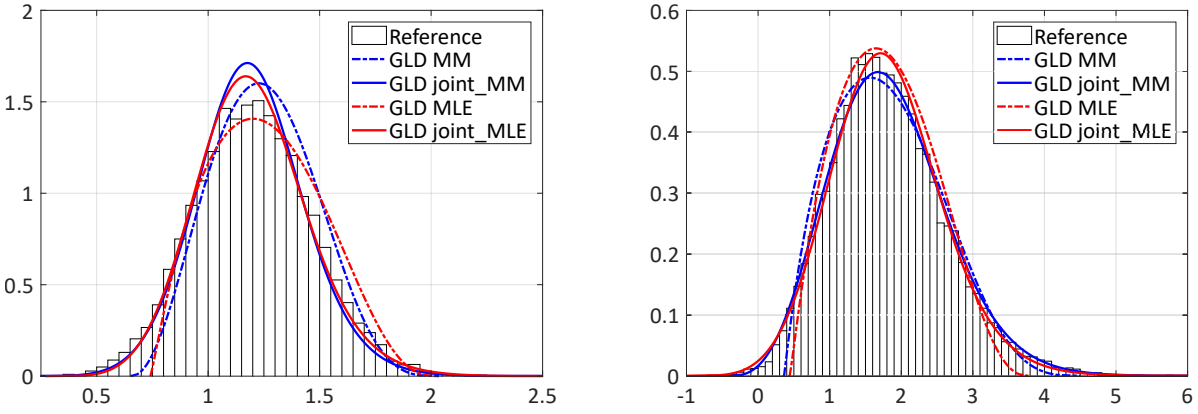
with the initial condition defined by

$$Y_0 = 0 \text{ almost surely.}$$

In this equation, $\boldsymbol{X} = (X_1, X_2)^T$ are the SDE parameters, and W_t is a standard Wiener process that represents the source of randomness. We denote $Y_t(\boldsymbol{x})$ the solution of Eq. (47) for $\boldsymbol{X} = \boldsymbol{x}$, and we focus on the value of $Y_t(\boldsymbol{x})$ at $t = 10$, i.e., $Y_{10}(\boldsymbol{x})$ is the scalar QoI.

Note that the value of ν controls how the Wiener process affects the QoI: for $\nu = 0$, $\mathrm{d}W_t$ is multiplied with a constant, and thus the solution $Y_t(\boldsymbol{x})$ is a Gaussian process; whereas for $\nu \neq 0$, W_t interacts with the unknown process $Y_t(\boldsymbol{x})$, and the marginal distribution of $Y_t(\boldsymbol{x})$ does not have an analytical closed-form. We set $\nu = 0.2$ in this study. To numerically solve Eq. (47), we apply the classical Euler-Maruyana method [46] with time step $\Delta t = 0.01$. Therefore, the discretized version of Eq. (47) has a large number of latent random variables $\boldsymbol{Z}$ equal to $\frac{10}{\Delta t}$ =1,000. This problem is representative of cases with low dimensionality in $\boldsymbol{X}$ and very large size of $\boldsymbol{Z}$.

The original definition of $\boldsymbol{X}$ proposed by [45] follows $X_1 \sim \mathcal{U}(0.95, 1.15)$ and $X_2 \sim \mathcal{U}(0.02, 0.22)$. According to some preliminary tests, we found that under this setting, the response PDF is close to a normal distribution and does not vary significantly with respect to $\boldsymbol{x}$ because the range of definition of the input parameters is rather narrow. In order to have richer shapes for the output PDF of $Y_{10}(\boldsymbol{x})$ and challenge our algorithm, we choose $X_1 \sim \mathcal{U}(0.9, 2)$, $X_2 \sim \mathcal{U}(0.1, 1)$ in this paper. Thus, the response PDF can have normal-like shape and can also be right-skewed depending on $\boldsymbol{x}$.



(a) PDF at $X = (1.2, 0.3)^T$ (b) PDF at $X = (1.8, 0.8)^T$

Figure 12: Stochastic differential equation – PDF predictions with an experimental design of size 80 using 40 replications. The reference histogram is calculated based on 10,000 replications.

Figure 12 shows the results when applying the developed methods to an experimental design of $N = 80$ and $R = 40$. We observe that all the four models can generally well approximate the underlying distributions. Detailed comparison shows that the Infer-and-Fit algorithm is not able to correctly emulate the shape variation of the response PDF: when the underlying PDF is close to a normal distribution, both *GLD MM* and *GLD MLE* predict a slightly right-skewed PDF; whereas for positively skewed PDF, neither of them is able to accurately approximate the tail. In contrast, *GLD joint_MM* and *GLD joint_MLE* not only capture the shape variation but also better represent the underlying PDF.

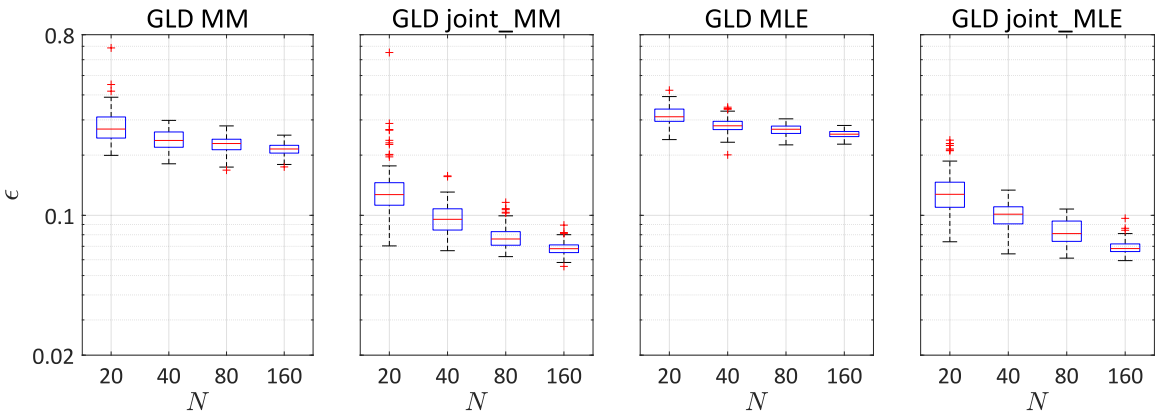


Figure 13: Stochastic differential equation – Hellinger distance between the surrogate model built with $R = 20$ and the reference response PDF, averaged over $\mathcal{X}_{\text{test}}$ (log-scale).

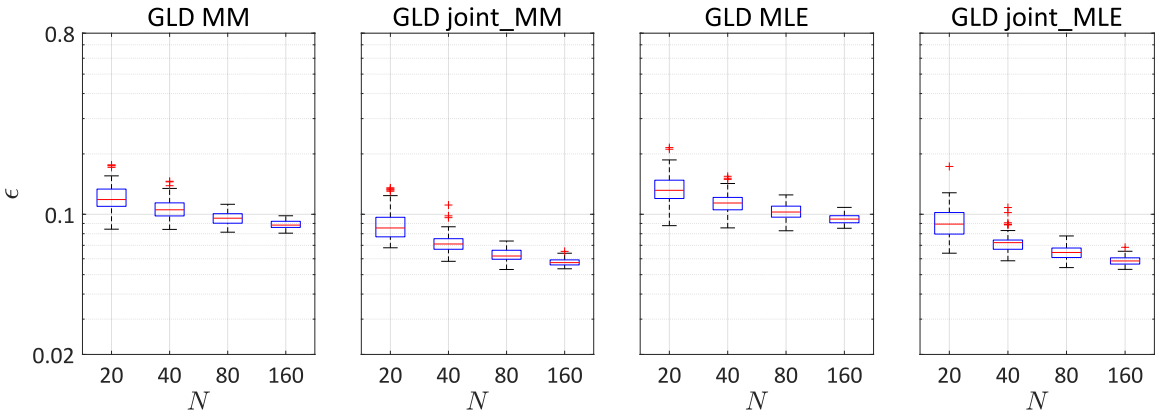


Figure 14: Stochastic differential equation – Hellinger distance between the surrogate model built with $R = 80$ and the reference response PDF, averaged over $\mathcal{X}_{\text{test}}$ (log-scale).

Similar to the analytical examples in Section 5, we investigate the convergence behavior of the developed methods. Since the analytical PDF is not available, we use kernel density estimation [47] using 10,000 replications as the reference distribution for each point in the test set. The Hellinger distance between the predicted PDF and the reference is averaged over a test set $\mathcal{X}_{\text{test}}$ containing 100 points generated with a Sobol' sequence.

The convergence study of the four models is reported in Figures 13 to 14. As expected from the

analytical examples, the joint modeling algorithm appears much more efficient. In particular, both *GLD joint_MM* and *GLD joint_MLE* yield an error around 0.07 in the case of 20 replications and 80 ED points, i.e., 1,600 model evaluations, whereas *GLD MM* and *GLD MLE* can barely achieve this accuracy even when 80 replications and 160 ED points, i.e., a total of 12,800 model evaluations, are available. More generally, the joint algorithm produce more accurate results than the Infer-and-Fit algorithm when a large number of replications are available.

6.2 Wind turbine design

In the wind turbine design process, structural components need to be analyzed under diverse environmental loads to assess their performance and reliability. Typical simulations consist of two parts, namely the generation of the external excitations (i.e., wind inflow) and the aero-servo-elastic simulation as illustrated in Figure 15. The latter refers to the complex multi-physics scenario including mutual interactions of wind inflow, aerodynamics, structural dynamics (elastic deflections) and control systems.

The wind field generator used in this study is TurbSim [48], which is a stochastic inflow turbulence simulator. It takes five macroscopic parameters as input: (1) the mean wind velocity U at reference altitude z_{ref}; (2) the turbulence intensity I, denoting the coefficient of variation of the wind time series, i.e., $I = \sigma/U$; (3) the wind shear exponent α, describing the variation of the mean wind velocity with the altitude according to the following equation:

$$U(z) = U \cdot \left(\frac{z}{z_{\text{ref}}}\right)^{\alpha}; \tag{48}$$

(4) the air density ρ and (5) the inclination angle β (see [49] for details). Since these five parameters cannot fully determine a wind field, random seeds are used on top of these macroscopic parameters in TurbSim to generate a coherent turbulent three-dimensional velocity time series [48].

The wind turbine structure studied here is the reference 5 MW upwind turbine described in [50]. The aero-servo-elastic simulator is FAST [50], a deterministic computational model that takes inflow wind fields as input and calculates the structural response as output. However, due to the

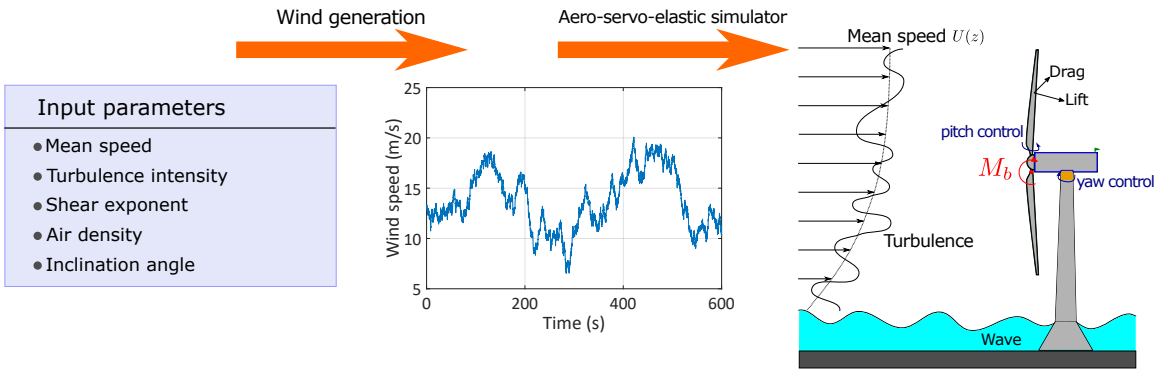


Figure 15: Wind turbine simulation scheme

Table 1: Wind turbine case study–description of the input variables

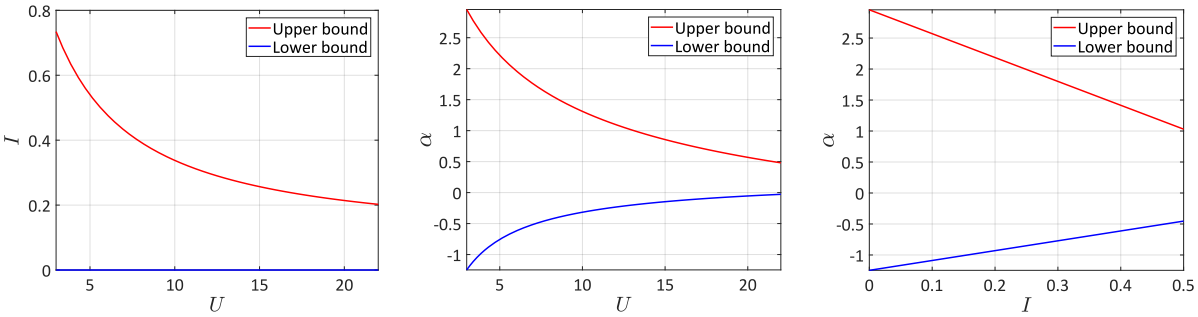
Name	Description	Distribution	Parameters
U	Mean speed (m)	Uniform	$[3, 22]$
I	Turbulence intensity	Uniform	$[0, 0.5]$
α	Shear exponent	Uniform	$[-2, 5]$
ρ	Air density (kg/m^3)	Uniform	$[0.8, 1.4]$
β	Declination angle (deg)	Uniform	$[-10, 10]$

use of random seeds in the turbulent wind generation, simulations of wind turbines are stochastic with respect to the five input macroscopic parameters. In other words, fixing the five quantities described above, any number of three-dimensional wind fields can be simulated, each of which leads to a different response and predicted performance of the wind turbine. Note that this is also what happens in reality for wind turbines.

Of interest is the maximum flap-wise bending moment at the blade root M_b within the simulated time (10 minutes) for a given wind climate defined by the 5 macroscopic parameters, as illustrated in Figure 15. To build a stochastic surrogate, the Latin hypercube sampling (LHS) method [51] with rejection is used to create an experimental design of 485 points in dimension 5. More precisely, input samples are firstly generated by the LHS following Table 1, and then the samples that are outside the bounds that are calibrated from real wind climate are removed. The bounds are respectively defined in the (I, U) plane, (α, U) plane, and (α, I) plane, as illustrated in Figure 16 [52]. Importantly, when the turbulence standard deviation $\sigma = I \cdot U$ is close to zero, the wind speed barely varies in time, and thus the response PDF is close to being degenerate, which can cause numerical problems. In this case, the simulator can be considered as deterministic and does not fit to the GLD framework. Hence, we introduce an additional bound in σ, and only samples with $\sigma > 0.05$ are simulated. Finally, the simulator is run 50 times for each set of input parameters as replications.

Considering the physical process, we chose to use the turbulence standard deviation σ rather than the turbulence intensity I to build the surrogate models. Hence, the input parameters are pre-processed as $\boldsymbol{X} = (U, \sigma, \alpha, \beta, \rho)^T$ for training.

Because of the sampling scheme, the input parameters U, I and α are not independent, which violates the independent assumption when building PC basis. One possibility to tackle this problem would be to use the Rosenblatt transform [53] to map the dependent inputs into a set of independent random variables and then build PC basis of the latter. However, [37] shows that this approach, while yielding improved estimates of the output statistics, is typically detrimental to the accuracy of pointwise predictions. This is because the Rosenblatt transform is highly nonlinear, resulting in a transformed model whose PCE spectrum decays typically more slowly than the original one. Therefore, we ignore the dependence when building PC basis functions, and only the marginal distribution of each input variables is needed. Since the marginal distributions



(a) Bounds in (I, U) plane (b) Bounds in (α, U) plane (c) Bounds in (α, I) plane

Figure 16: Bounds on the physical parameters (U, I, α) calibrated from real wind data

are difficult to be derived analytically due to the bounds, we apply the kernel density estimators to 10,000 samples generated according to the rejection sampling scheme described before. Note that the air density ρ and the inclination angle β are uniform variables, and they are independent from U, I and α. Therefore, we use Legendre polynomials as the associated univariate PC basis functions for these two variables.

Unlike the previous example in Section 6.1, the wind turbine simulation is costly, and thus we cannot run as many times the simulator as needed to have a reliable estimate of the error defined in Equation (41). To assess the performance of the proposed methods, 120 samples of input parameters are generated by the same scheme as the training set. The simulator is repeatedly run 500 times for each test point. We then compare some sample statistics with those predicted by the stochastic emulators built by the developed methods. The former are considered as references. The metrics used for comparison are the mean, the standard deviation (std) and the 5%, 10%, 50%, 90% and 95% quantiles.

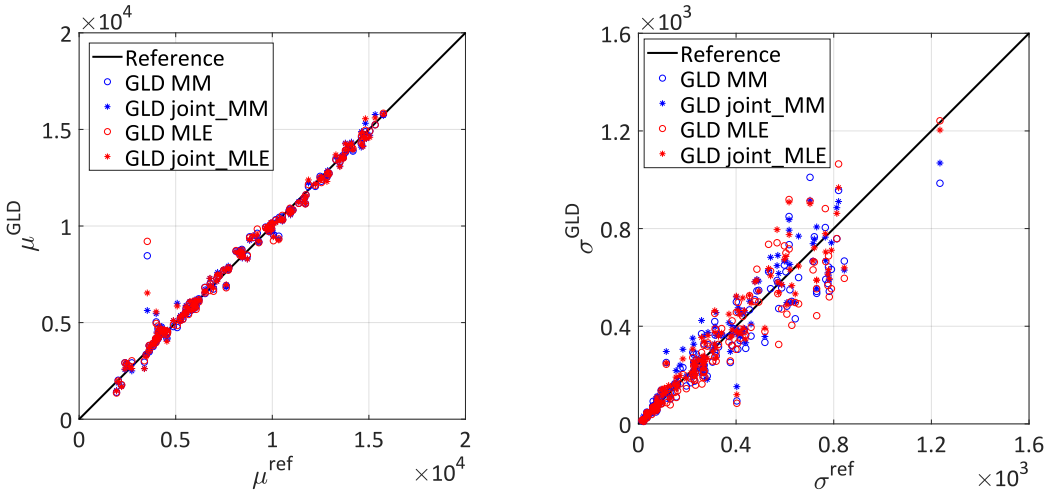
The results of the four GLD models are shown in Figure 17 and Figure 18. Comparisons of the scalar quantities show that all the four GLD models demonstrate a good fit to the simulated scenario. Among the scalar quantities, the mean and the quantiles are estimated with high accuracy, whereas the standard deviation estimation is relatively poor.

To have more quantitative comparison among the four GLD models, we define the normalized mean squared error:

$$\epsilon = \frac{\sum_{i=1}^{N_{\text{test}}} \left(q_{GLD}^{(i)} - \hat{q}^{(i)} \right)^2}{\sum_{i=1}^{N_{\text{test}}} \left(\hat{q}^{(i)} - \bar{\hat{q}} \right)^2}, \text{ with } \bar{\hat{q}} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \hat{q}^{(i)}, \tag{49}$$

where q is the statistical quantity of interest (mentioned above), $q_{GLD}^{(i)}$ is the value predicted by the GLD model, and $\hat{q}^{(i)}$ denotes the estimated quantity (empirical mean, standard deviation and quantiles) based on the replications for $\boldsymbol{x}^{(i)}$.

The errors associated with the scalar quantities are reported in Table 2. We can observe that the method of moments outperforms the maximum likelihood estimation for both the Infer-and-Fit



(a) Mean estimation

(b) Std estimation

Figure 17: Wind turbine case study – Comparison of the mean and the standard deviation estimation of the maximum flapwise bending moment ($kN \cdot m$). The x-axis (reference) is the empirical quantity calculated from the 500 replications.

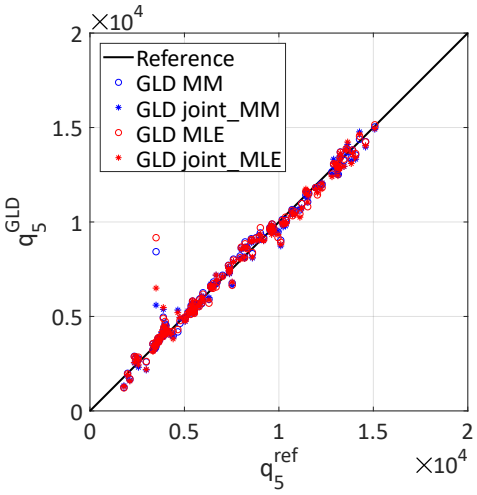
Table 2: Normalized mean squared error of various quantities in the test set. The best results among the four GLD models are highlighted in bold.

GLD models	mean	std	Q_{05}	Q_{10}	Q_{50}	Q_{90}	Q_{95}
GLD MM	0.0185	**0.1125**	0.0231	0.0221	0.0188	0.0166	0.0165
GLD joint_MM	**0.0099**	0.124	**0.0133**	**0.0154**	**0.0103**	**0.009**	**0.0091**
GLD MLE	0.0235	0.1488	0.0292	0.0280	0.0237	0.0214	0.0213
GLD joint_MLE	0.0128	0.1642	0.0167	0.0155	0.0131	0.0121	0.0125

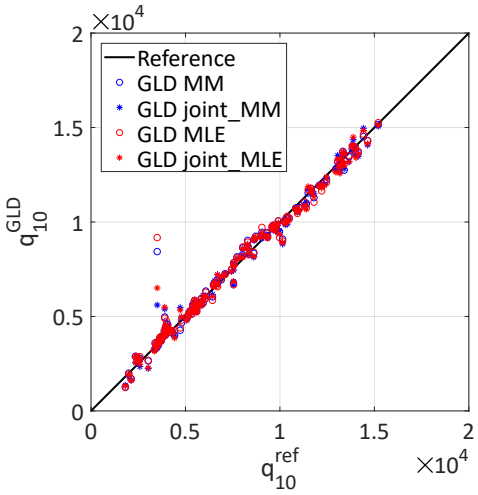
and the joint model in terms of all the error measures used here. The joint models generally improve their associated Infer-and-Fit models, and *GLD joint_MM* provides the best estimates.

Apart from the detailed quantitative comparison of the scalar quantities, we visualize also the PDF prediction in Figure 19 for two specific values of $\boldsymbol{x}$. The reference histograms are obtained based on the 500 replications. Since only 500 samples are available, the histograms are less smooth than those of the previous example in Section 6.1. We observe that all the four surrogate models can well capture the location of the underlying distribution. In addition, the two joint models demonstrate better performance on the shape approximation. For example, in Figure 19, the two Infer-and-Fit models produce narrower support than the range of the samples, whereas the support is accurately approximated by the two joint models.

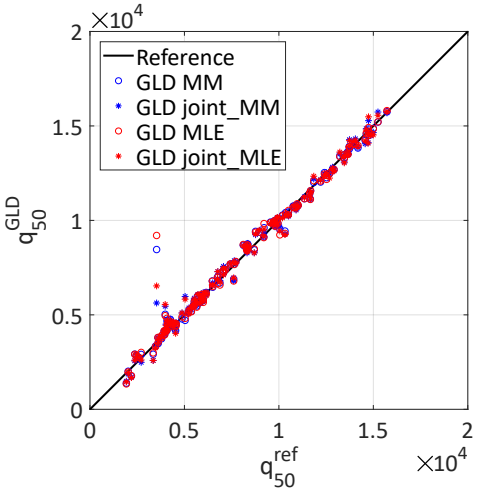
As a conclusion, the GLD joint models allows for accurate prediction of the PDF of the maximum flapwise bending moment at the blade root at a total cost of about 24,000 runs of Tubsim+FAST. The calculation have been carried out on the ETH Euler cluster using 96 cores for a physical time of about 20.5 hours. Interestingly, the 95% quantile, which is of interest for design assessment, is



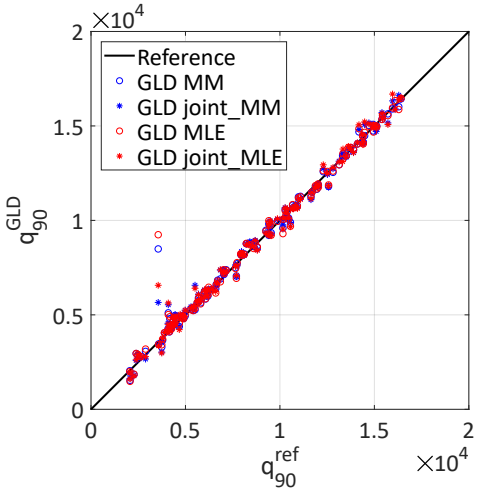
(a) 5% quantile estimation



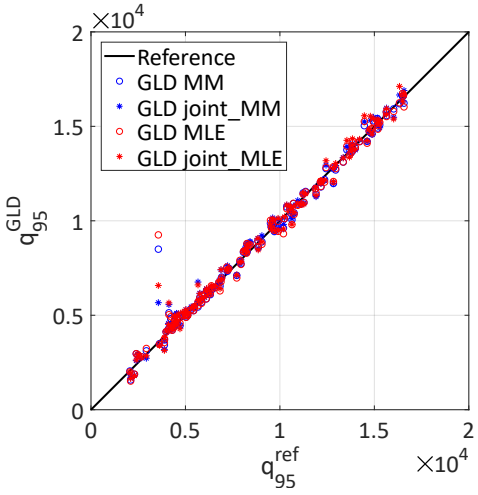
(b) 10% quantile estimation



(c) 50% quantile estimation

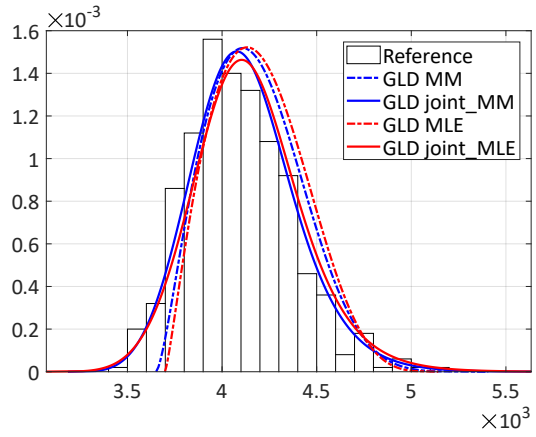


(d) 90% quantile estimation

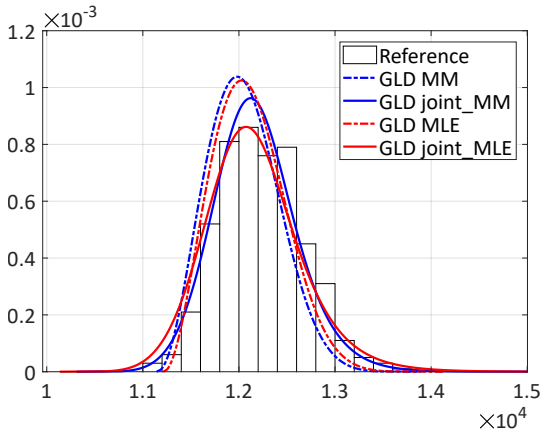


(e) 95% quantile estimation

Figure 18: Wind turbine case study – Comparison of the quantiles estimation of the maximum flapwise bending moment ($kN \cdot m$). The x-axis (reference) is the empirical quantity calculated from the 500 replications.



(a) PDF at $\boldsymbol{x} = (4.9, 1.2, 0.6, 0.9, 7.2)^T$



(b) PDF at $\boldsymbol{x} = (11.8, 2.9, 0.3, 1.0, 0.2)^T$

Figure 19: Wind turbine case study – PDF predictions with the experimental design of size 485 using 50 replications. The reference histogram is calculated based on 500 replications.

remarkably predicted all over the space of input parameters.

7 CONCLUSIONS

The aim of this paper is to build efficient and accurate surrogate models for stochastic simulators within the replication-based framework. Generalized lambda distributions are used to flexibly approximate the output PDF, while the relationship of their parameters with the inputs is approximated through polynomial chaos expansions. To construct surrogate models in a non-intrusive manner, we first proposed the Infer-and-Fit algorithm which consists of solving two consecutive problems. In the first step, the distribution parameters are inferred based on repeated model evaluations for each point of the experimental design. Then the estimated values are used to build a PCE surrogate model for each distribution parameters. The Infer-and-Fit algorithm allows us to use conventional regression techniques to construct PCE. However, this method is sensitive to the number of replications due to the two-step strategy, whereby the model responses are only used in the first step. In order to build accurate surrogate models even when a few replications are available, we proposed in a second part the joint modeling method described in Algorithm 2. This approach carries out one more optimization step after getting a first estimate from the Infer-and-Fit approach, which is used to provide sparse truncation sets for $\boldsymbol{\lambda}^{\mathrm{PC}}(\boldsymbol{x}; \boldsymbol{a})$ and a starting point for the optimization. This enrichment allows us to use all the available data at once and provides a maximum likelihood estimator of the model parameters, namely the coefficients of the polynomial chaos expansions of the $\boldsymbol{\lambda}$'s. Due to the complexity of the likelihood function, this additional optimization problem can be expensive to solve. To alleviate the computational burden, we vectorized the implementation in the Matlab environment and derived analytically the gradient and the Hessian matrix of the objective function. As a result,

we can efficiently apply derivative-based optimizers.

For the analytical examples in Section 5 and the stochastic differential equation case study in Section 6.1, the proposed two algorithms are both able to approximate the reference distributions with high accuracy, even though the data generation scheme does not follow the generalized lambda distribution. As expected, the joint models show better performance when only a few replications are available. For the wind turbine application in Section 6.2, due to the cost of the simulator, only some important statistical scalar quantities are compared to a reference solution obtained from a large Monte Carlo simulation, whereas PDFs at selected input points are only compared visually. Both developed methods demonstrate high accuracy for the mean and quantiles estimation.

In all the examples and applications, joint models are observed to consistently improve the result of the associated models built with the Infer-and-Fit algorithm. Besides, for the parametric estimation in the first step of the Infer-and-Fit algorithm, the method of moments and maximum likelihood show comparable performance. This observation matches the conclusion in [24].

In the joint modeling method, the main role played by the replications is to obtain a truncation scheme for each component of $\boldsymbol{\lambda}^{\mathrm{PC}}(\boldsymbol{x};\boldsymbol{a})$ as well as to find an initial starting point for the following optimization step. Therefore, replications are not necessary if the basis functions of each distribution parameter are known or preselected. Work is in progress to improve the proposed method by using advanced statistical techniques that completely avoid the need for replications and thus drastically reduce the computational cost.

Acknowledgments

This paper is a part of the project "Surrogate Modeling for Stochastic Simulators (SAMOS)" funded by the Swiss National Science Foundation (Grant #200021_175524). The authors gratefully thank Dr. Imad Abdallah (ETH Zurich) and René Slot (Aalborg University) for extensive fruitful discussions on applications, and Nora Lüthen (ETH Zurich) for the computation of the wind turbine data.

References

[1] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, Cambridge, Massachusetts, Internet edition, 2006.

[2] D. Xiu and G. E. Karniadakis. The Wiener-Askey polynomial chaos for stochastic differential equations. *SIAM J. Sci. Comput.*, 24(2):619–644, 2002.

[3] R. Ghanem and P. Spanos. *Stochastic Finite Elements: A Spectral Approach.* Courier Dover Publications, Mineola, 2nd edition, 2003.

[4] P. McCullagh and J. Nelder. *Generalized Linear Models*, volume 37 of *Monographs on Statistics and Applied Probability*. Chapman and Hall/CRC, 1989.

[5] B. Iooss and M. Ribatet. Global sensitivity analysis of computer models with functional inputs. *Reliab. Eng. Sys. Safety*, 94:1194–1204, 2009.

[6] B. Ankenman, B. Nelson, and J. Staum. Stochastic Kriging for simulation metamodeling. *Oper. Res.*, 58:371–382, 2009.

[7] M. Dacidian and R. J. Carroll. Variance function estimation. *J. Amer. Stat. Assoc.*, 400:1079–1091, 1987.

[8] J. Fan and Q. W. Yao. Efficient estimation of conditional variance functions in stochastic regression. *Biometrika*, 85:645–660, 1998.

[9] A. Marrel, B. Iooss, S. Da Veiga, and M. Ribatet. Global sensitivity analysis of stochastic computer models with joint metamodels. *Stat. Comput.*, 22:833–847, 2012.

[10] P. K. Bhattacharya and A. K. Gangopadhyay. Kernel and nearest-neighbor estimation of a conditional quantile. *Ann. Statist.*, 18(3):1400–1415, 1990.

[11] M. Plumlee and R. Tuo. Building accurate emulators for stochastic simulations via quantile Kriging. *Technometrics*, 56:466–473, 2014.

[12] R. Koenker. Quantile regression: 40 years on. *Annu. Rev. Econ.*, 9:155–176, 2017.

[13] T. Hastie and R. Tibshirani. *Generalized Additive Models*, volume 43 of *Monographs on Statistics and Applied Probability*. Chapman and Hall/CRC, 1990.

[14] J. Fan and I. Gijbels. *Local Polynomial Modelling and Its Applications*, volume 66 of *Monographs on Statistics and Applied Probability*. Chapman and Hall/CRC, 1996.

[15] P. Hall, J. Racine, and Q. Li. Cross-validation and the estimation of conditional probability densities. *J. Amer. Stat. Assoc.*, 99:1015–1026, 2004.

[16] S. Efromovich. Dimension reduction and adaptation in conditional density estimation. *J. Amer. Stat. Assoc.*, 105:761–774, 2010.

[17] A. B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer Series in Statistics. Springer, Cambridge, Massachusetts, 2009.

[18] V. Moutoussamy, S. Nanty, and B. Pauwels. Emulators for stochastic simulation codes. *ESAIM: Math. Model. Num. Anal.*, 48:116–155, 2015.

[19] T. Browne, B. Iooss, L. Le Gratiet, J. Lonchampt, and E. Rémy. Stochastic simulators based optimization by Gaussian process metamodels – application to maintenance investments planning issues. *Quality Reliab. Eng. Int.*, 32(6):2067–2080, 2016.

[20] Z. A. Karian and E. J. Dudewicz. *Fitting Statistical Distributions: The Generalized Lambda Distribution and Generalized Bootstrap Methods*. Chapman and Hall/CRC, 2000.

[21] M. Freimer, G. Kollia, G. S. Mudholkar, and C. T. Lin. A study of the generalized Tukey

lambda family. *Comm. Stat. Theor. Meth.*, 17:3547–3567, 1988.

[22] Y. Chalabi, D. J. Scott, and D. Würtz. The generalized lambda distribution as an alternative to model financial returns. *Eidgenössische Technische Hochschule and University of Auckland, Zurich and Auckland. Retrieved from Rmetrics*, 2011.

[23] Z. A. Karian and E. J. Dudewicz. *Handbook of Fitting Statistical Distributions with R*. Taylor & Francis, 2010.

[24] C. G. Corlu and M. Meterelliyoz. Estimating the parameters of the generalized lambda distribution: Which method performs best? *Comm. Stat. Simul. Comput.*, 45(7):2276–2296, 2016.

[25] A. Lakhany and H. Massuer. Estimating the parameters of the generalised lambda distribution. *Algo Research Quarterly*, 3(3):47–58, 2000.

[26] S. Su. Numerical maximum log likelihood estimation for generalized lambda distributions. *Comput. Stat. Data Anal.*, 51(8):3983–3998, 2007.

[27] R. L. Burden, J. D. Faires, and A. M. Burden. *Numerical analysis*. Cengage Learning, 2015.

[28] O. G. Ernst, A. Mugler, H. J. Starkloff, and E. Ullmann. On the convergence of generalized polynomial chaos expansions. *ESAIM: Math. Model. and Num. Anal.*, 46:317–339, 2012.

[29] W. Gautschi. *Orthogonal polynomials: computation and approximation*. Oxford University Press, 2004.

[30] M. Berveiller, B. Sudret, and M. Lemaire. Stochastic finite elements: a non intrusive approach by regression. *Eur. J. Comput. Mech.*, 15(1-3):81–92, 2006.

[31] G. Blatman and B. Sudret. Adaptive sparse polynomial chaos expansion based on Least Angle Regression. *J. Comput. Phys.*, 230:2345–2367, 2011.

[32] G. Blatman and B. Sudret. An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis. *Prob. Eng. Mech.*, 25:183–197, 2010.

[33] S. Marelli and B. Sudret. UQLab user manual – Polynomial chaos expansions. Technical report, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2019. Report # UQLab-V1.3-104.

[34] S. Marelli and B. Sudret. 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)*, pages 2554–2563, 2014.

[35] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Ann. Statist.*, 32:407–499, 2004.

[36] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. Inf. Theory*, 53(12):4655–4666, 2007.

[37] E. Torre, S. Marelli, P. Embrechts, and B. Sudret. Data-driven polynomial chaos expansion

for machine learning regression. *J. Comput. Phys.*, 388:601–623, 2019.

[38] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM J. Num. Anal.*, 20(3):626–637, 1983.

[39] D. V. Arnold and N. Hansen. A (1+1)-CMA-ES for constrained optimisation. In Terence Soule and Jason H. Moore, editors, *Proc. of the Genetic and Evolutionary Computation Conference 2012 (GECCO 2012)*, pages 297–304, 2012.

[40] M. Moustapha, C. Lataniotis, P. Wiederkehr, , D. Wagner, P.-R.and Wicaksono, S. Marelli, and B. Sudret. UQLib user manual. Technical report, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2019. Report # UQLab-V1.3-201.

[41] I. M. Sobol'. Distribution of points in a cube and approximate evaluation of integrals. *U.S.S.R Comput. Maths. Math. Phys.*, 7:86–112, 1967.

[42] A. J. McNeil, R. Frey, and P. Embrechts. *Quantitative Risk Management: Concepts, Techniques, and Tools*. Princeton Series in Finance. Princeton University Press, Princeton, New Jersey, 2005.

[43] A. Gray, D. Greenhalgh, L. Hu, X. Mao, and J. Pan. A stochastic differential equation SIS epidemic model. *SIAM J. Appl. Math.*, 71(3):876–902, 2011.

[44] E. B. Iversen, J. M. Morales, J. K. Moller, X. Mao, and H. Madsen. Short-term probabilistic forecasting of wind speed using stochastic differential equations. *Int. J. Forecast.*, 32:981–990, 2015.

[45] M. N. Jimenez, O. P. Le Maître, and O. M. Knio. Nonintrusive polynomial chaos expansions for sensitivity analysis in stochastic differential equations. *SIAM J. Uncer. Quant.*, 5:212–239, 2017.

[46] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Springer, 1992.

[47] M. Wand and M. C. Jones. *Kernel smoothing*. Chapman and Hall, Boca Raton, 1995.

[48] B. J. Jonkman. TurbSim user's guide: version 1.50. Technical report, National Renewable Energy Laboratory, U.S. Department of Energy, 2009.

[49] I. Abdallah, C. Lataniotis, and B. Sudret. Parametric hierarchical Kriging for multi-fidelity aero-servo-elastic simulators – application to extreme loads on wind turbines. *Prob. Eng. Mech.*, 55:67–77, 2019.

[50] J. Jonkman, S. Butterfield, W. Musial, and G. Scott. Definition of a 5-MW reference wind turbine for offshore system development. Technical report, National Renewable Energy Laboratory, U.S. Department of Energy, 2009.

[51] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code.

Technometrics, 21(2):239–245, 1979.

[52] R.M.M Slot, J. D. Sorensen, B. Sudret, L. Svenningsen, and M.L. Thogersen. Surrogate model uncertainty in wind turbine reliability assessment. *Renewable Energy*, 2020. Accepted.

[53] M. Rosenblatt. Remarks on a multivariate transformation. *Ann. Math. Stat.*, 23:470–472, 1952.

A Appendix

A.1 Feasible starting point

For the additional optimization problem introduced in the joint algorithm, the coefficients $\tilde{\boldsymbol{a}}$ fitted from the Infer-and-Fit algorithm are chosen as an appropriate starting point for the optimization. However, as discussed in Section 4, the objective function $l(\boldsymbol{a})$ can take the value $+\infty$, and thus complex constraints are present, which is summarized in Eq. (40). Therefore, additional operations are necessary to have a feasible starting point if $\tilde{\boldsymbol{a}}$ does not satisfy the constraints.

It is observed from Eq. (3) that the lower (upper) bound of the support is a monotonic function of λ_3 (λ_4) for fixed λ_1 and λ_2. Therefore, reducing the coefficients $a_{3,\mathbf{0}}$ and $a_{4,\mathbf{0}}$ that are associated with the constant functions in Eq. (35) broadens the support of the response PDF for all $\boldsymbol{x} \in \mathcal{D}_{\boldsymbol{X}}$. Based on this property, the following procedure is proposed to adjust $\tilde{\boldsymbol{a}}$ to be feasible:

1. Evaluate $\boldsymbol{\lambda}^{(i)} = \boldsymbol{\lambda}^{\mathrm{PC}}\left(\boldsymbol{x}^{(i)}; \tilde{\boldsymbol{a}}\right)$ for all $\boldsymbol{x}^{(i)} \in \boldsymbol{\mathcal{X}}$
2. Collect the index i into the set I, whose associated $\lambda_3^{(i)}$ is positive
3. For all $i \in I$, calculate $\check{\lambda}_3^{(i)}$ such that the minimum value of the associated replication results located exactly on the lower bound. More precisely, according to Eq. (3), we have

$$\check{\lambda}_3^{(i)} = \frac{1}{\lambda_2^{(i)}\left(\lambda_1^{(i)} - \min\limits_r y^{(i,r)}\right)} \tag{50}$$

4. Decrease the value of $\tilde{a}_{3,\mathbf{0}}$ so that $\lambda_3^{(i)} < \check{\lambda}_3^{(i)}$ for all $i \in I$.

This algorithm only deals with the constraints related to the lower bounds. The same method can be used for those from upper bounds, which consists in modifying the constant $\tilde{a}_{4,\mathbf{0}}$ based on $\tilde{\lambda}_4^{(i)}$.

A.2 Analytical derivations for Algorithm 2

In this section, we compute the analytical derivatives of the negative log-likelihood function l with respect to the model parameters $\boldsymbol{a}$. Since the objective function is a composition of several

functions as shown in Figure 3, the derivatives can be calculated through the chain rule, which flows from Step 3 to Step 1. Starting from

$$l = \log\left(\frac{u^{\lambda_3-1} + (1-u)^{\lambda_4-1}}{\lambda_2}\right), \tag{51}$$

we get the following partial derivatives:

$$\frac{\partial l}{\partial u} = \frac{(\lambda_3-1)u^{\lambda_3-2} - (\lambda_4-1)(1-u)^{\lambda_4-2}}{u^{\lambda_3-1} + (1-u)^{\lambda_4-1}}, \tag{52}$$

$$\frac{\partial l}{\partial \lambda_2} = -\frac{1}{\lambda_2}, \tag{53}$$

$$\frac{\partial l}{\partial \lambda_3} = \frac{u^{\lambda_3-1}\log(u)}{u^{\lambda_3-1} + (1-u)^{\lambda_4-1}}, \tag{54}$$

$$\frac{\partial l}{\partial \lambda_4} = \frac{(1-u)^{\lambda_4-1}\log(1-u)}{u^{\lambda_3-1} + (1-u)^{\lambda_4-1}}. \tag{55}$$

The differentiation at Step 2 is more complex because u is not an explicit function of $\boldsymbol{\lambda}$, and thus it involves derivatives of a highly nonlinear implicit function Eq. (1). Based on

$$y = Q(u) = \lambda_1 + \frac{1}{\lambda_2}\left(\frac{u^{\lambda_3}-1}{\lambda_3} - \frac{(1-u)^{\lambda_4}-1}{\lambda_4}\right),$$

and because y is given, differentiating both side gives

$$0 = \mathrm{d}\left(\lambda_1 + \frac{1}{\lambda_2}\left(\frac{u^{\lambda_3}-1}{\lambda_3} - \frac{(1-u)^{\lambda_4}-1}{\lambda_4}\right)\right),$$

where d stands for the total differentiation.

Expanding and rearranging the equations above, we have

$$\frac{\partial u}{\partial \lambda_1} = -\frac{\lambda_2}{u^{\lambda_3-1} + (1-u)^{\lambda_4-1}}, \tag{56}$$

$$\frac{\partial u}{\partial \lambda_2} = \frac{1}{\lambda_2\left(u^{\lambda_3-1} + (1-u)^{\lambda_4-1}\right)}\left(\frac{u^{\lambda_3}-1}{\lambda_3} - \frac{(1-u)^{\lambda_4}-1}{\lambda_4}\right), \tag{57}$$

$$\frac{\partial u}{\partial \lambda_3} = \frac{u^{\lambda_3} - \lambda_3 u^{\lambda_3}\log(u) - 1}{\lambda_3^2\left(u^{\lambda_3-1} + (1-u)^{\lambda_4-1}\right)}, \tag{58}$$

$$\frac{\partial u}{\partial \lambda_4} = \frac{-(1-u)^{\lambda_4} + \lambda_4(1-u)^{\lambda_4}\log(1-u) + 1}{\lambda_4^2\left(u^{\lambda_3-1} + (1-u)^{\lambda_4-1}\right)}. \tag{59}$$

As illustrated in Figure 3, the derivatives of the negative log-likelihood function with respect to $\boldsymbol{\lambda}$ come from two parts: one is from the direct derivative (Eqs. (53)-(55)) in Step 3, the other part is contributed by the implicit differentiation (Eqs. (56)-(59)) in Step 2. As a result, we have

$$\frac{\mathrm{d}l}{\mathrm{d}\lambda_s} = \frac{\partial l}{\partial \lambda_s} + \frac{\partial l}{\partial u}\frac{\partial u}{\partial \lambda_s}, \quad s = 1, 2, 3, 4. \tag{60}$$

Finally, the derivatives flow back to the model parameters $\boldsymbol{a}$ at Step 1 as follows:

$$\frac{\mathrm{d}l}{\mathrm{d}a_{s,\boldsymbol{\alpha}}} = \frac{\mathrm{d}l}{\mathrm{d}\lambda_s}\frac{\partial \lambda_s}{\partial a_{s,\boldsymbol{\alpha}}} = \frac{\mathrm{d}l}{\mathrm{d}\lambda_s}\psi_{\boldsymbol{\alpha}}(\boldsymbol{x}), \quad s = 1, 3, 4, \tag{61}$$

$$\frac{\mathrm{d}l}{\mathrm{d}a_{2,\boldsymbol{\alpha}}} = \frac{\mathrm{d}l}{\mathrm{d}\lambda_2}\frac{\partial \lambda_2}{\partial a_{2,\boldsymbol{\alpha}}} = \frac{\mathrm{d}l}{\mathrm{d}\lambda_2}\frac{1}{L'(\lambda_2)}\psi_{\boldsymbol{\alpha}}(\boldsymbol{x}), \tag{62}$$

where L denotes the transform that is used to guarantee the positiveness of $\lambda_2(\boldsymbol{x})$. Recall that we chose to use $L(\lambda_2) = \log(\lambda_2)$ in this paper. Similar techniques can be used to derive the Hessian matrix of the negative log-likelihood function, which is necessary for the trust-region algorithm. Due to the lengthy derivation, we omit the result here. Eq. (60) calculates the derivatives of the log-likelihood function with respect to the distribution parameters $\boldsymbol{\lambda}$. Hence, it can be used in the maximum likelihood estimation of the distribution parameters of a random variable following a generalized lambda distribution.