POLITECNICO DI TORINO
Repository ISTITUZIONALE

Polynomial chaos vs kernel machine learning methods for uncertainty quantification: A comparative study and benchmarking with the hybrid PCE-GPR approach

Original
Polynomial chaos vs kernel machine learning methods for uncertainty quantification: A comparative study and benchmarking with the hybrid PCE-GPR approach / Manfredi, P.. - In: COMPUTER METHODS IN APPLIED MECHANICS AND ENGINEERING. - ISSN 0045-7825. - ELETTRONICO. - 449:(2026), pp. 1-35. [10.1016/j.cma.2025.118523]

Availability:
This version is available at: 11583/3008930 since: 2026-03-19T12:40:49Z

Publisher:
Elsevier

Published
DOI:10.1016/j.cma.2025.118523

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

01 November 2026

Computer Methods in Applied Mechanics and Engineering 449 (2026) 118523

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Polynomial chaos vs kernel machine learning methods for uncertainty quantification: A comparative study and benchmarking with the hybrid PCE-GPR approach

Paolo Manfredi *

Department of Electronics and Telecommunications, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Turin, Italy

ARTICLE INFO

Keywords:
Gaussian process regression
Kernel methods
Machine learning
Polynomial chaos expansion
Surrogate modeling
Uncertainty quantification

ABSTRACT

In the past two decades, the application of surrogate models to expedite uncertainty quantification (UQ) received wide attention in the applied physics and engineering communities. In particular, extensive literature focused on high-dimensional problems and training efficiency with the aim of addressing the so-called "curse of dimensionality" and achieving satisfactory accuracy even with small experimental designs. Besides popular methods based on polynomial chaos expansion (PCE), recent works explored the application of machine learning regression methods to UQ, particularly kernel-based methods such as support vector regression and Gaussian process regression (GPR). Promising performance and superior accuracy were highlighted especially for small dataset sizes. In this context, the present paper provides a twofold contribution. First, it extends the literature survey and benchmarks by comparing PCE and kernel-based machine learning methods. Second, it introduces an open source toolbox that implements "PCE-GPR", a recently proposed hybrid method that combines the advantages of PCE and GPR methods and is suitable for accurate high-dimensional UQ. The analysis shows that kernel techniques tend to outperform PCE in high-dimensional settings and in small-data scenarios, at the price of increased computational cost. Moreover, for most test cases, PCE-GPR is shown to significantly outperform all of the state-of-the-art methods considered. The comparisons are based on popular benchmarks for surrogate modeling and UQ, simulations in various fields of engineering, as well as stochastic boundary value problems with high-dimensional random fields and up to 145 uncertain parameters.

1. Introduction

Surrogate-based uncertainty quantification (UQ) became ubiquitous in science and engineering to accelerate the design exploration of complex systems under the influence of uncertain parameters [1–7]. In this scenario, a computationally cheap surrogate or metamodel is trained from a limited set of observations from the actual underlying physical system. The surrogate is then used as a proxy of the original system to carry out UQ tasks that normally require a large number of simulation samples to converge.

Since the early 2000s, methods based on polynomial chaos expansion (PCE) [8] have gained a great deal of popularity. PCE approximates the input-output system relation using orthogonal polynomials that depend on the distribution of each random input. This guarantees an optimal converge rate as the expansion order is increased. Moreover, thanks to the orthogonality of the basis functions, statistical moments and sensitivity information are easily derived from the PCE coefficients [9].

Several classes of methods were proposed for the calculation of the PCE coefficients. In this work, we focus on data-driven methods, which are the most general and most suitable for engineering systems, since they only require collecting input-output samples, without

* Corresponding author.
E-mail address: paolo.manfredi@polito.it

https://doi.org/10.1016/j.cma.2025.118523
Received 21 May 2025; Received in revised form 22 October 2025; Accepted 24 October 2025
Available online 8 November 2025
0045-7825/© 2025 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

any specific knowledge of the underlying system. Among these, sparse regression techniques were shown to perform well especially in high-dimensional settings [10]. Sparse PCE approaches include least-angle regression (LAR) [11], orthogonal matching pursuit (OMP) [12], subspace pursuit (SP) [13], and Bayesian compressive sensing (BCS) [14]. Applications are found in very diverse fields including, e.g., brittle fracture analysis [15], hydrology [16], aerodynamics [17], eddy-current testing [18,19], FDTD solvers [20], electromagnetic compatibility [21], obesity diagnosis [22], as well as in the analysis of multibody systems [23], beam systems [24], transonic airfoils [25–27], astronautics problems [28], pesticide transfer [29], resonant gating structures [30,31], solar cells [32], multi-energy flow [33], low-pollution burners [34], cracking furnaces [35], water-exit vehicles [36] and underwater vehicles [37], chemical reactions [38], composite laminates [39], slope reliability [40], nonstationary geotechnical systems [41], thrust regulation in turbine aero-engines [42], rotor systems [43], supersonic turbulent jet-in-crossflows [44], fuel centrifugal pumps [45], human body electromagnetic exposure [46], Earth entry vehicles [47], and collision probability with debris in low-Earth orbits [48]. LAR, OMP, SP, and BCS are all implemented in UQLab [49,50], the prime toolbox for high-dimensional UQ. Their performance was extensively assessed in [10] based on several benchmarks.

More recently, the rapid evolution of machine learning and artificial intelligence led researches to explore alternative methods for regression. Although not specifically designed for UQ, machine learning regression techniques can also be adopted to construct surrogates for UQ purposes [51]. While neural networks are less attractive in this regard, as they typically require larger training datasets and incur higher self-training cost, kernel-based machine learning methods like support vector regression (SVR) [52] and Gaussian process regression (GPR) [53] do not assume a predetermined form and exhibit good predictive performance even with small amounts of training data. Applications of SVR and GPR, also known as Kriging, to UQ and sensitivity analysis are found, e.g., in the analysis of compressor cascades [54], mechanical metamaterials [55], dynamic stability analysis [56], dynamical systems [57], solid mechanics [58], lightning strike damage [59], aging corrosion [60], integrated electronic devices [2], high-speed electronic links [61], wireless power transfer efficiency [62], electromagnetic simulations [63], welded steel tubular joints [64], crude distillation units [65], drug effects on the cardiovascular system [66].

A technique combining PCE and GPR, named polynomial-chaos-based Kriging (PCK), was proposed in [67] and shown to provide better prediction performance than PCE and GPR alone. Specifically, PCK uses a sparse PCE in the trend of a Kriging model, while resorting to standard kernels as covariance function. Applications of PCK are found, e.g., in geotechnical engineering [68], civil engineering [69], hydrology [70], aerodynamics [71,72], and structural health monitoring [73].

Another hybrid technique, named “PCE-GPR”, was recently proposed in [74]. PCE-GPR leverages special kernels constructed from an infinite sequence of Hermite or Legendre polynomials, which are the PCE orthogonal bases for Gaussian and uniform variability, respectively. Hence, a fundamental difference with respect to PCK is that the PCE basis functions are embedded in the kernel rather than in the trend. PCE-GPR was shown to outperform both state-of-the-art PCE methods and classical GPR based on standard kernels. Moreover, thanks to the special kernel definition, the model can be fully converted into a PCE with arbitrary precision, enabling the analytical calculation of moments and sensitivity indices with the inclusion of confidence levels.

This paper presents an open-source implementation of PCE-GPR. The PCE-GPR toolbox [75] relies on UQLab to train the model, as it offers advanced features for custom kernel definitions and hyperparameter estimation. Furthermore, the performance of PCE-GPR is assessed based on several benchmarks, including both low- and high-dimensional examples, stochastic boundary value problems, as well as analytical functions and numerical test cases in electronic engineering and structural mechanics. The benchmarking extends the analysis of [10] to classical kernel-based machine learning methods, namely SVR, GPR, PCK and least square support vector machine (LSSVM) regression, and to high-dimensional real-life examples with up to 145 uncertain parameters.

The remainder of the paper is organized as follows. Section 2 provides an overview of the state-of-the-art PCE and kernel methods, as well as of PCE-GPR. Section 3 introduces the test cases that are used in the benchmarking. Section 4 discusses the PCE-GPR toolbox and investigates the performance under various model settings to identify the recommended default configuration. In Section 5, we investigate the performance of PCE-GPR against state-of-the-art methods. Section 6 further explores the application of a subset of high-performing techniques to high-dimensional stochastic boundary value problems and structural mechanics. Finally, we draw conclusions in Section 7.

2. Methods

In this section, we introduce the various methods for UQ that are considered in this work. The goal of forward UQ is to propagate the variability of uncertain parameters $\boldsymbol{x} = (x_1, \dots, x_d) \in \mathcal{X} \subseteq \mathbb{R}^d$, with a known probability distribution $\omega : \mathcal{X} \to \mathbb{R}^+$, to the output y of a system. We generically denote the system as

$$y = \mathcal{M}(\boldsymbol{x}), \tag{1}$$

where $\mathcal{M} : \mathcal{X} \to \mathrm{R}$ is the function or code that needs to be evaluated or run to obtain y for a given configuration of $\boldsymbol{x}$.

We focus on systems with scalar and real-valued outputs and on data-driven methods only. Therefore, we assume that a training dataset $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_l, y_l)\}_{l=1}^{L}$ (the experimental design) is available, with L observations $y_l = \mathcal{M}(\boldsymbol{x}_l)$ computed using (1).

2.1. Polynomial chaos expansion methods

The methods based on PCE approximate the functional relationship (1) as

$$\mathcal{M}_{\text{PCE}}(\boldsymbol{x}) = \sum_{\boldsymbol{\kappa} \in \mathcal{K}} c_{\boldsymbol{\kappa}} \psi_{\boldsymbol{\kappa}}(\boldsymbol{x}), \tag{2}$$

where $\psi_{\boldsymbol{\kappa}}$ are orthogonal polynomials based on the probability distribution of the parameters $\boldsymbol{x}$. In the common case of independent variation of the uncertain parameters $\boldsymbol{x}$, the basis functions are expressed as the product of univariate polynomials, i.e.,

$$\psi_{\boldsymbol{\kappa}}(\boldsymbol{x}) = \prod_{j=1}^{d} \psi_{\kappa_j}^{(j)}(x_j), \tag{3}$$

where $\boldsymbol{\kappa} = (\kappa_1, \dots, \kappa_d) \in \mathbb{N}^d$ are multi-indices defining the degree of the basis function in each input dimension, since $\deg\left(\psi_k\right) = k$. The superscript $^{(j)}$ is used to highlight that the polynomials may differ for each input dimension, based on the corresponding distribution. The univariate polynomials are orthogonal based on the inner product

$$\langle f, g \rangle_j = \int_{\mathcal{X}_j} f(x_j) g(x_j) \omega_j(x_j), \tag{4}$$

where ω_j and $\mathcal{X}_j$ denote the probability density and the domain of the variable x_j, respectively.

For standard distributions, the orthogonal polynomial bases are well-known, e.g., Hermite, Legendre, and Jacobi polynomials for Gaussian, uniform, and beta distributions, respectively [8]. Without loss of generality, we assume that the polynomials are normalized so that

$$\left\| \psi_{\kappa_j}^{(j)} \right\|_2 = \sqrt{\left\langle \psi_{\kappa_j}^{(j)}, \psi_{\kappa_j}^{(j)} \right\rangle_j} = 1 \tag{5}$$

Numerical techniques exist to construct orthogonal polynomials for arbitrary distributions (e.g., [76]).

An attractive feature of PCE is that the first two statistical moments are readily derived from the expansion coefficients as

$$\mathrm{E}(y) \approx c_{\mathbf{0}} \tag{6}$$

and

$$\mathrm{Var}(y) \approx \sum_{\boldsymbol{\kappa} \in \mathcal{K} \setminus \mathbf{0}} c_{\boldsymbol{\kappa}}^2 \tag{7}$$

Moreover, performing the above summation over suitable subsets of the coefficients provides Sobol' sensitivity indices [9]. Specifically, the total Sobol' indices quantify the individual contribution of each random input to the output variance and are obtained from the PCE coefficients as

$$S_{T_i} \approx \frac{\sum_{\boldsymbol{\kappa} \in \mathcal{K}_j} c_{\boldsymbol{\kappa}}^2}{\sum_{\boldsymbol{\kappa} \in \mathcal{K} \setminus \mathbf{0}} c_{\boldsymbol{\kappa}}^2}, \tag{8}$$

where $\mathcal{K}_j \subset \mathcal{K}$ denotes the subset of orthogonal polynomials that are non-constant in the j-th input (i.e., the ones with $\kappa_j > 0$), while the denominator is the PCE estimate of the total variance (7). Selecting the optimal polynomial basis provides an exponential convergence rate for L^2 functions, which correspond to functions with finite variance.

In addition to the selection of the proper orthogonal polynomials, one key definition is the truncation of the basis. Typically, the user first defines a preliminary truncation by identifying a suitable set of multi-indices $\mathcal{K} \subset \mathbb{N}^d$. This is usually achieved by bounding some norm of the multi-indices to a maximum degree. For example, the popular total degree truncation is defined as

$$\mathcal{K} = \left\{ \boldsymbol{\kappa} \,:\, \|\boldsymbol{\kappa}\|_1 = \sum_{j=1}^{d} \kappa_j \leq p \right\} \tag{9}$$

and leads to $|\mathcal{K}| = (p+d)!/(p!d!)$ coefficients in the expansion. The hyperbolic truncation is instead defined as

$$\mathcal{K} = \left\{ \boldsymbol{\kappa} \,:\, \|\boldsymbol{\kappa}\|_q = \left(\sum_{j=1}^{d} \kappa_j^q \right)^{1/q} \leq p \right\}, \tag{10}$$

with $0 < q < 1$, and yields a sparser expansion. The total-degree truncation is equivalent to a hyperbolic truncation with $q = 1$. Once the preliminary truncation is defined, one may compute all the coefficients of the resulting expansion using, e.g., collocation [77] or ordinary least-square (OLS) regression. Conversely, sparse PCE methods [10] are able to further prune the basis and retain important terms only.

In this work, we focus on high-dimensional and fully data-driven methods for the calculation of PCE coefficients. Therefore, we neglect intrusive methods like the stochastic Galerkin method [78], which operates directly on the system of equations by performing Galerkin projections and constructing an augmented system in the expansion coefficients. We also neglect collocation methods, which are usually based on Gauss quadrature rules [77]. These methods require to sample the function at predefined collocation nodes and typically result in a higher number of simulation samples.

Rather, we focus the analysis on regression schemes, which compute the coefficients $c_{\boldsymbol{\kappa}}$ by minimizing the L^2 norm of the approximation residual over the training dataset $\mathcal{D}_{\text{train}}$. The well-known OLS regression solves

$$\hat{\boldsymbol{c}} = \underset{\boldsymbol{c} \in \mathbb{R}^{|\mathcal{K}|}}{\arg\min} \|\boldsymbol{\Psi}\boldsymbol{c} - \boldsymbol{y}\|_2 = \boldsymbol{\Psi}^{+}\boldsymbol{y} \tag{11}$$

where $\boldsymbol{c}$ is the vector of PCE coefficients, $\boldsymbol{y} \in \mathbb{R}^L$ is the vector of observations, $\boldsymbol{\Psi} \in \mathcal{R}^{L\times|\mathcal{K}|}$ is the matrix with the basis functions (regressors) evaluated at the training points, with entries $\Psi_{l\boldsymbol{\kappa}} = \psi_{\boldsymbol{\kappa}}(\boldsymbol{x}_l)$, and $\boldsymbol{\Psi}^+$ is its Moore-Penrose pseudoinverse. However, this approach requires $L > |\mathcal{K}|$ samples to be well-posed, and in general $L \gg |\mathcal{K}|$ to be accurate, which is impractical in high-dimensional settings and/or for highly nonlinear systems, even with hyperbolic truncation schemes.

Sparse regression schemes further prune the basis functions and identify the most relevant coefficients via iterative schemes and/or by introducing a penalty in (11) to promote coefficient sparsity at the price of some (hopefully negligible) accuracy reduction. The method for solving the sparse regression problem and calculate the PCE coefficients becomes therefore the main feature that defines the various methods. An extensive and exhaustive list of state-of-the-art methods is described in [10]. Based on the careful review therein, we consider in this work LAR [11], OMP [12], SP [13], BCS [14,79], and spectral projected gradient descent with ℓ^1-minimization (SPGL1) [80].

LAR and OMP are iterative schemes that create and expand an active set of regressors $\mathcal{A} \subseteq \mathcal{K}$ by progressively picking basis functions from $\mathcal{K}$ based on their correlation with the current residual. In this process, LAR introduces a penalty to (11) and solves the LASSO problem

$$\hat{\boldsymbol{c}} = \underset{\boldsymbol{c}\in\mathbb{R}^{|\mathcal{A}|}}{\arg\min} \|\boldsymbol{\Psi}\boldsymbol{c} - \boldsymbol{y}\|_2 + \lambda\|\boldsymbol{c}\|_1, \tag{12}$$

where λ is a regularization hyperparameter.

OMP uses a greedy algorithm to find, at each iteration, the basis element that is the most correlated with the current approximation residual. Let us denote with $\boldsymbol{\Psi}^{(\mathcal{K})}$ and $\boldsymbol{\Psi}^{(\mathcal{A})}$ the matrix of the candidate and active regressors evaluated in the training samples, respectively. Finding the element that minimizes the resulting residual is equivalent to solving [49]

$$\boldsymbol{\alpha} = \underset{\boldsymbol{\kappa}\in\mathcal{K}}{\arg\max} \left|\boldsymbol{\psi}_{\boldsymbol{\kappa}}^{\mathsf{T}}\boldsymbol{r}\right|, \tag{13}$$

where $\boldsymbol{\psi}_{\boldsymbol{\kappa}}$ denotes the vector of the candidate basis function $\psi_{\boldsymbol{\kappa}}$ evaluated at the training samples and $\boldsymbol{r} = \boldsymbol{y} - \boldsymbol{\Psi}^{(\mathcal{A})}\boldsymbol{c}$ is the current approximation residual, in which the coefficients $\boldsymbol{c}$ are computed via OLS regression over the active set of regressors. The above maximization corresponds to finding the largest element of the vector $\boldsymbol{\Psi}^{(\mathcal{K})\mathsf{T}}\boldsymbol{r}$. The residual is initialized to the vector of training observations $\boldsymbol{y}$. At each iteration, the leave-one-out (LOO) error estimate is stored and the selected basis element with multi-index $\boldsymbol{\alpha}$ is removed from the candidate set $\mathcal{K}$ and added to the active set $\mathcal{A}$. The process is repeated until the number of active regressors has reached the number of observed data L or the cardinality of the original candidate set. The final active set is selected as the one that achieved the lowest LOO error throughout the iterations.

SP uses OLS on subsets of regressors to identify K non-zero coefficients [13]. Specifically, it first adds to the active set the K regressors that are the most correlated to the current residual by taking the largest elements of $\boldsymbol{\Psi}^{(\mathcal{K})\mathsf{T}}\boldsymbol{r}$. Then, it computes the corresponding coefficients $\boldsymbol{c}$ via OLS regression and adds K more regressors that are the most correlated with the updated residual. Next, it calculates the coefficients of the resulting $2K$ regressors, again via OLS regression, and removes from the active set the K regressors with the smallest coefficients. The process is repeated until convergence. Since the optimal number of non-zero coefficients K is not known a priori, it is considered a hyperparameter to be tuned using cross-validation.

BCS recast the regression in a Bayesian framework and uses likelihood functions and priors to promote sparsity in the coefficients via a maximum-a-posteriori estimate. Finally, SPGL1 solves either the LASSO problem

$$\hat{\boldsymbol{c}} = \underset{\boldsymbol{c}\in\mathbb{R}^{|\mathcal{K}|}}{\arg\min} \frac{1}{2}\|\boldsymbol{\Psi}\boldsymbol{c} - \boldsymbol{y}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{c}\|_1 \le \tau \tag{14}$$

or the basis pursuit denoise problem

$$\hat{\boldsymbol{c}} = \underset{\boldsymbol{c}\in\mathbb{R}^{|\mathcal{K}|}}{\arg\min} \|\boldsymbol{c}\|_1 \quad \text{s.t.} \quad \frac{1}{2}\|\boldsymbol{\Psi}\boldsymbol{c} - \boldsymbol{y}\|_2^2 \le \sigma. \tag{15}$$

2.2. Kernel-based machine learning methods

Kernel-based methods follow a different rationale. As opposed to PCE methods, which first construct a candidate set of basis functions, they are nonparametric, meaning that they do not assume a predefined model form. They use combinations of kernel functions centered at the training points, which makes their complexity mainly determined by the number of available data, rather than the number of basis functions. As such, they naturally scale more easily to high-dimensional parameter spaces.

2.2.1. Support-Vector Machine Regression

SVR defines a nonlinear regression by mapping input data to a high- or even infinite-dimensional *feature space* [52] by means of a function set $\boldsymbol{\Phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \dots, \phi_K(\boldsymbol{x}))^{\mathsf{T}}$, with $\boldsymbol{\Phi} : \mathbb{R}^d \to \mathbb{R}^K$, leading to

$$\mathcal{M}_{\text{SVR}}(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x})\rangle + b = \sum_{k=1}^{K} w_k\phi_k(\boldsymbol{x}) + b = \sum_{l=1}^{L}(\alpha_l - \alpha_l^*)\sum_{k=1}^{K}\phi_k(\boldsymbol{x})\phi_k(\boldsymbol{x}_l) + b. \tag{16}$$

The coefficients $\boldsymbol{\alpha}, \boldsymbol{\alpha}^*$ are obtained by minimizing the so-called ε-insensitive loss function

$$\mathcal{L}_{\nu}^{\varepsilon}(\boldsymbol{x}) = \begin{cases} 0 & \text{if} \quad |\mathcal{M}_{\text{SVR}}(\boldsymbol{x}) - y| < \varepsilon \\ \left(|\mathcal{M}_{\text{SVR}}(\boldsymbol{x}) - y| - \varepsilon\right)^{\nu} & \text{otherwise} \end{cases} \tag{17}$$

typically with $\nu = 1$ (linear loss) or $\nu = 2$ (quadratic loss). In (17), ε is a parameter defining the ε-insensitive tube: only points outside it are penalized, whereas predictions within ε of the true value are considered error-free. A peculiar feature of SVR is that only a subset of the training points, lying outside the ε-insensitive tube and called *support vectors*, have non-zero coefficients and contribute to the regression. Therefore, SVR seeks a trade-off between the width ε of the insensitive tube and the model accuracy: a small ε may lead to excessive overfitting, whereas a large ε may lead to poor accuracy.

One of the key advantages is that the evaluation of the inner product

$$\left\langle \boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{x}') \right\rangle = \boldsymbol{\Phi}(\boldsymbol{x})^{\mathsf{T}} \boldsymbol{\Phi}(\boldsymbol{x}') = \sum_{k=1}^{K} \phi_k(\boldsymbol{x}) \phi_k(\boldsymbol{x}') \tag{18}$$

does not need to be carried out explicitly, but it can be achieved by leveraging a kernel function such that $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{x}') \rangle$. Any positive function that satisfies the Mercer's condition [81] corresponds to an inner product in *some* feature space and is therefore a valid kernel. Therefore, the evaluation of (16) becomes a linear combination of L kernel functions, i.e.,

$$\mathcal{M}_{\mathrm{SVR}}(\boldsymbol{x}) = \sum_{l=1}^{L} (\alpha_l - \alpha_l^*) k(\boldsymbol{x}, \boldsymbol{x}_l) + b = \sum_{l \in \mathcal{I}_{\mathrm{SV}}} (\alpha_l - \alpha_l^*) k(\boldsymbol{x}, \boldsymbol{x}_l) + b, \tag{19}$$

regardless of the dimensionality of $\boldsymbol{x}$. In (19), the index set $\mathcal{I}_{\mathrm{SV}}$ defines the subset of training points that are support vectors.

Note that the equivalence between the *dual space* formulation (19) and the *primal space* formulation (16) is trivially demonstrated if the kernel is explicitly defined as in (18), with

$$w_k = \sum_{l=1}^{L} (\alpha_l - \alpha_l^*) \phi_k(\boldsymbol{x}_l), \quad k = 1, \dots, K. \tag{20}$$

However, (19) holds also for implicit kernels, for which the feature space functions are often not available explicitly.

Popular kernel functions are the polynomial kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^{\mathsf{T}} \boldsymbol{x}' + c)^p, \tag{21}$$

the squared-exponential (or Gaussian) kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2} h^2\right), \tag{22}$$

and the family of Matérn functions

$$k(\boldsymbol{x}, \boldsymbol{x}') = C_{p+1/2}(h) = \exp\left(-\sqrt{2p+1}\, h\right) \frac{p!}{(2p)!} \sum_{k=0}^{p} \frac{(p+k)!}{k!(p-k)!} \left(2\sqrt{2p+1}\, h\right)^{p-k}, \tag{23}$$

for which $p = 1$ or $p = 2$ is commonly used. The argument u of the squared-exponential and Matérn kernels is defined as

$$h = \sqrt{\sum_{j=1}^{d} \frac{(x_j - x_j')^2}{\theta_j^2}}, \tag{24}$$

where θ_j are scale parameters. If the scale parameters are allowed to differ for each input, the kernel is said to be *anisotropic*. In contrast, isotropic kernels use the same scale parameter θ for each input, leading to $u = \|\boldsymbol{x} - \boldsymbol{x}'\|_2 / \theta$. The scales $\boldsymbol{\theta}$, as well as the bias term c and the degree p of the polynomial kernel, are *hyperparameters* that are tuned during the training process, typically using CV.

For the case of linear ε-insensitive loss, the model coefficients are computed by solving the quadratic programming problem [82]

$$\begin{aligned} \underset{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*}{\arg\min} \quad & \frac{1}{2} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^* \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \boldsymbol{K} & -\boldsymbol{K} \\ -\boldsymbol{K} & \boldsymbol{K} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^* \end{pmatrix} + \begin{pmatrix} \varepsilon - \boldsymbol{y} \\ \varepsilon + \boldsymbol{y} \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^* \end{pmatrix} \\ \text{s.t.} \quad & \begin{pmatrix} \boldsymbol{1} \\ -\boldsymbol{1} \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^* \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{pmatrix}, \qquad 0 \le \alpha_l, \alpha_l^* \le C \end{aligned} \tag{25}$$

where $\boldsymbol{K} \in \mathbb{R}^{L \times L}$ is the Gram matrix of the kernel evaluated at the training samples, with entries $K_{lm} = k(\boldsymbol{x}_l, \boldsymbol{x}_m)$, for $l, m = 1, \dots, L$. For the quadratic loss, the quadratic programming minimization problem becomes

$$\begin{aligned} \underset{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*}{\arg\min} \quad & \frac{1}{2} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^* \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \tilde{\boldsymbol{K}} & -\tilde{\boldsymbol{K}} \\ -\tilde{\boldsymbol{K}} & \tilde{\boldsymbol{K}} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^* \end{pmatrix} + \begin{pmatrix} \varepsilon - \boldsymbol{y} \\ \varepsilon + \boldsymbol{y} \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^* \end{pmatrix} \\ \text{s.t.} \quad & \begin{pmatrix} \boldsymbol{1} \\ -\boldsymbol{1} \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^* \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{pmatrix}, \qquad \alpha_l, \alpha_l^* \ge 0 \end{aligned}, \tag{26}$$

with the modified Gram matrix $\tilde{\boldsymbol{K}} = \boldsymbol{K} + \frac{1}{C} \boldsymbol{I}$, where $\boldsymbol{I}$ is the $L \times L$ identity matrix. In both cases, C is a regularization parameter that helps avoid overfitting along with ε. The regularizer C and the tube width ε are further hyperparameters to be tuned along with kernel scale(s).

2.2.2. Least-Square Support-Vector Machines

LSSVM is the least-square reformulation of SVR [84]. The dual-space LSSVM model reads

$$\mathcal{M}_{\text{LSSVM}} = \sum_{l=1}^{L} \alpha_l k(\boldsymbol{x}, \boldsymbol{x}_l) + b \tag{27}$$

where now the coefficients $\boldsymbol{\alpha}$ are conveniently computed by solving the linear system

$$\begin{pmatrix} \boldsymbol{K} + \boldsymbol{I}/\gamma & \mathbf{1} \\ \mathbf{1}^{\mathsf{T}} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ b \end{pmatrix} = \begin{pmatrix} \boldsymbol{y} \\ 0 \end{pmatrix}, \tag{28}$$

instead of a quadratic programming problem, where γ is a regularization hyperparameter that plays a similar role as C in (26). In fact, LSSVM is a kernel ridge regression with the addition of the bias term b. Also in this case, the hyperpameters (the regularizer γ and kernel parameters) are tuned using CV. From (26), it is readily shown that the LSSVM is equivalent to an SVR with quadratic loss and $\varepsilon = 0$, in which all training samples contribute as support vectors. Despite the loss of the sparsity of support vectors, the LSSVM has the advantage of requiring a mere solution of a linear system instead of a quadratic programming.

In [85], it was shown that the primal-space LSSVM formulation is equivalent to a PCE if the kernel function is constructed using the PCE basis as feature-space functions, i.e.,

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\boldsymbol{\kappa} \in \mathcal{K}} \psi_{\boldsymbol{\kappa}}(\boldsymbol{x}) \psi_{\boldsymbol{\kappa}}(\boldsymbol{x}') \tag{29}$$

Moreover, special implicit kernels were introduced that correspond to an infinite sequence of Hermite or Legendre polynomials, as will be discussed later on.

2.2.3. Gaussian Process Regression / Kriging

GPR starts from the assumption that the target function (1) is a realization of a Gaussian process (prior) with mean function or trend $\mu(\boldsymbol{x})$ and covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$. The mean and covariance functions are more conveniently expressed as

$$\mu(\boldsymbol{x}) = \sum_{k=0}^{K} \beta_k h_k(\boldsymbol{x}), \tag{30}$$

i.e., a linear combination of basis functions $\{h_k(\boldsymbol{x})\}_{k=0}^{K}$, and

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 r(\boldsymbol{x}, \boldsymbol{x}'), \tag{31}$$

respectively, where $r(\boldsymbol{x}, \boldsymbol{x}')$ is the prior *correlation* function. Although the starting point and interpretation are different, the covariance function plays in fact the same role as a kernel, and the same kernel functions as for SVR and LSSVM are commonly used as prior covariance (e.g., [53,86]).

The GPR model reads

$$\mathcal{M}_{\text{GPR}}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \sum_{l=1}^{L} \alpha_l r(\boldsymbol{x}, \boldsymbol{x}_l), \tag{32}$$

which, in Bayesian settings, corresponds to the mean function of a posterior Gaussian process that is obtained by constraining (conditioning) the prior to the available observations.

If a zero-mean white noise is assumed on the observations, the coefficients $\boldsymbol{\alpha}$ are obtained as

$$\boldsymbol{\alpha} = (1 - \tau) \tilde{\boldsymbol{R}}^{-1} (\boldsymbol{y} - \boldsymbol{\mu}), \tag{33}$$

where $\boldsymbol{\mu} = (\mu(\boldsymbol{x}_1), \dots, \mu(\boldsymbol{x}_L))^{\mathsf{T}}$ is a vector with the trend evaluated at the training points and

$$\tilde{\boldsymbol{R}} = (1 - \tau) \boldsymbol{R} + \tau \boldsymbol{I}. \tag{34}$$

In (34), $\boldsymbol{R}$ is the correlation matrix of the training samples, with elements $R_{lm} = r(\boldsymbol{x}_l, \boldsymbol{x}_m)$, $l, m = 1, \dots, L$, while τ is defined as

$$\tau = \frac{\sigma_n^2}{\sigma^2 + \sigma_n^2}, \tag{35}$$

where σ_n^2 is the noise variance and σ^2 is the kernel variance. If the observations are assumed to be noiseless, then $\sigma_n = 0$ and (32) interpolates them.

One of the attractive features of GPR is that, building upon a Bayesian interpretation, it is possible to associate a posterior covariance function to the model predictions obtained with (32). The posterior covariance reflects the intrinsic uncertainty related to the limited availability of training data and allows assigning confidence intervals to model predictions. In this work, however, we focus the attention only on the predictive capabilities, and hence we disregard the posterior uncertainty and distribution.

The mean function $\mu(\boldsymbol{x})$ embeds prior beliefs on the general trend of the target function. Several options are commonly adopted:

- Simple Kriging: the trend is fully known, i.e., $\beta_k = 1$, $\forall k = 1, \dots, K$;
- Ordinary Kriging: the trend is constant yet unknown, i.e., $\mu(\boldsymbol{x}) = \beta_0$, with $h_0(\boldsymbol{x}) = 1$;
- Universal Kriging: same as (30), with unknown coefficients $\boldsymbol{\beta}$.

A very common case of simple Kriging is $\mu(\boldsymbol{x}) = 0$ (null trend). When the trend coefficients are instead unknown (ordinary and universal Kriging), they are estimated using a generalized least-square regression [86]. In universal Kriging, a sequence of polynomials of increasing order is often assumed. It should be noted that, especially in multi-dimensional settings, this choice partially cancels the advantage of using a kernel formulation, since the number of basis functions in the trend increases exponentially and the coefficients needs to be estimated with a parametric regression. Finally, it is relevant to note that the ordinary Kriging model is equivalent to the LSSVM, where the noise term σ_n^2 plays the same role as the regularizer γ.

Besides the use of classical CV, the hyperparameters of GPR models (i.e., kernel lengthscale, kernel variance, and noise variance or τ) can also be tuned using maximum likelihood (ML) estimation [53,86].

2.2.4. Polynomial-Chaos-Based Kriging

PCK is a special case of universal Kriging in which a PCE is considered for the trend [67], while standard kernel functions are still leveraged for the covariance. It was shown in [67] that PCK provides superior accuracy compared to PCE and GPR alone. PCK is a hybrid method that combines the features of PCE and GPR. However, the model cannot be fully described in terms of PCE coefficients due to the kernel part, and therefore it does not provide statistical and sensitivity information in closed form.

Two implementations of PCK are available: in sequential PCK, the most relevant basis functions for the trend are estimated first using a standard sparse regression method (e.g., LAR) and then plugged into the GPR model. In optimal PCK, the algorithm iterates between PCE estimation and GPR calibration to yield a more accurate model at the price of a reduced training efficiency.

2.3. PCE-GPR

PCE-GPR is a hybrid method recently proposed in [74]. It employs a GPR formulation with null trend and special correlation functions that are built from an infinite sequence of Hermite or Legendre polynomials and originally introduced in [74]. The univariate Hermite correlation reads

$$r(x, x'; \rho) = \frac{1}{\sqrt{1-\rho^2}} \exp\left(\frac{-\rho^2\left(x^2 + x'^2\right) - 2\rho x x'}{2(1-\rho^2)} \right) = \sum_{k=0}^{\infty} \frac{\rho^k}{k!} H_k(x) H_k(x'), \tag{36}$$

where H_k denotes the k-th probabilists' Hermite polynomial. The univeraite Legendre correlation reads

$$r(x, x'; \rho) = \frac{K(u)}{\pi\sqrt{a-b}} = \sum_{k=0}^{\infty} \rho^k P_k(x) P_k(x'), \tag{37}$$

where

$$u = \sqrt{\frac{2b}{b-1}}, \tag{38}$$

$$a = 1 - 2xx'\rho + \rho^2, \tag{39}$$

$$b = -2\rho\sqrt{1-x^2}\sqrt{1-x'^2}, \tag{40}$$

and $K(u)$ is the complete elliptic integral of the first kind, i.e.,

$$K(u) = \int_0^1 \frac{dt}{\sqrt{(1-t^2)(1-u^2t^2)}}. \tag{41}$$

In both cases, $\rho \in (0, 1)$ is a hyperparameter that plays a similar role as the traditional scale. The multivariate correlation is constructed using the separable formulation [81]

$$r(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\rho}) = \prod_{j=1}^{d} r^{(j)}(x_j, x'_j; \rho_j) \tag{42}$$

and embeds all product combinations of the univariate chaos polynomials up to order infinity. In (42), the subscript $^{(j)}$ is used to highlight that different univariate correlations may be used according to the input distribution.

The advantage of using these special kernels, as opposed to the PCK method, is that the GPR model can now be analytically converted from (32) in a PCE by computing the corresponding coefficients up to an arbitrary order as

$$c_{\boldsymbol{\kappa}} = \lambda_{\boldsymbol{\kappa}} \sum_{l=1}^{L} \alpha_l \psi_{\boldsymbol{\kappa}}(\boldsymbol{x}_l), \tag{43}$$

where $\lambda_{\boldsymbol{\kappa}}$ are scale factors related to the correlation function. These are computed as

$$\lambda_{\boldsymbol{\kappa}} = \prod_{j=1}^{d} \lambda_{\kappa_j}^{(j)}, \tag{44}$$

with

$$\lambda_{\kappa_j}^{(j)} = \rho_j^{\kappa} \tag{45}$$

if the j-th input has a Gaussian distribution (Hermite correlation) or

$$\lambda_{\kappa_j}^{(j)} = \frac{\rho_j^{\kappa}}{2\kappa_j + 1} \tag{46}$$

of the j-th input has a uniform distribution (Legendre correlation).

Hence, while it can be used as a classical nonparametric Kriging model, PCE-GPR also allows the analytical calculation of PCE coefficients, from which statistical information is readily obtained. This calculation merely involves the evaluation of the PCE basis functions at the training points. Although the PCE coefficients are obtained individually using (43), they can also be computed compactly for an entire expansion as

$$c = \mathbf{\Lambda}\mathbf{\Psi}^{\mathsf{T}}\boldsymbol{\alpha}, \tag{47}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with entries $\Lambda_{\boldsymbol{\kappa}\boldsymbol{\kappa}} = \lambda_{\boldsymbol{\kappa}}$. Furthermore, it is also possible to calculate the covariance matrix of the estimated PCE coefficients, from which pointwise predictions and statistical information are obtained probabilistically with the inclusion of confidence information. The superior performance of PCE-GPR over standard PCE and GPR was demonstrated for some engineering application examples [74].

At this point, we should highlight an important difference between PCK and PCE-GPR: While the former places the PCE basis functions in the trend, PCE-GPR inherently embeds them in the kernel, thereby fully preserving the nonparametric nature of GPR (a part from the possible post-processing calculation of the PCE coefficients).

2.4. PCE-LSSVM

PCE-LSSVM [85] is another hybrid method and a precursor of PCE-GPR, which uses the same kernel functions but is based on the LSSVM formulation (27). Although conceptually similar, PCE-LSSVM does not rely on a Bayesian framework and therefore does not provide confidence information. Moreover, the hyperparameter tuning is based on CV and is hardly implemented in existing toolboxes. A notable difference is that, in PCE-LSSVM, the zero-order PCE coefficient is explicitly represented by the bias term b.

2.5. Discussion

PCE methods are specifically designed for UQ and readily provide at least the first two statistical moments and sensitivity indices. Even sparse methods require to construct the full candidate set $\mathcal{K}$, which can become computationally demanding in very high-dimensional settings, unless a sparse truncation (e.g., hyperbolic) is used. In any case, the size of $\mathcal{K}$ increases with the number of input dimensions and/or expansion order. The choice of the candidate set was shown in [74] to impact the precision even for sparse methods, since regression typically estimates the coefficients of a smaller basis with greater accuracy for a given number of samples. This may introduce a trade-off on the accuracy: on the one hand, a sparser candidate set increases the accuracy in the calculation of the coefficients; on the other hand, it may exclude a priori relevant expansion terms.

Conversely, kernel machine learning methods provide generic surrogates to replace the expensive computational model (1). The model complexity is determined by the size of the training dataset and is therefore not (directly) related to the input dimensions. The evaluation of the kernel is itself inexpensive both in low and in high-dimensional settings. However, the input dimensionality may indirectly affect the training cost because: 1) a high-dimensional problem may require more training data; 2) the hyperparameter tuning becomes more expensive if anisotropic kernels are used, since in that case the number of scale parameters equals the number of input dimensions and the optimization problem becomes itself high-dimensional. Compared to PCE methods, the training time for a given number of samples is usually higher as they require the inversion of a dense and potentially large kernel matrix.

PCE-GPR is a hybrid method that takes the advantageous features of both classes of methods, i.e., analytical statistical information and nonparametric training with good scalability to high-dimensional settings. It should be noted that the post-processing calculation of the PCE coefficients is per se parametric. However, as opposed to classical PCE methods, in this case the coefficients can be computed individually and up to an arbitrary order.

3. Benchmark functions and test cases

The computational models and test cases that will be used to benchmark the PCE-GPR toolbox against state-of-the-art UQ methods are listed in Table 1 by increasing dimensionality. They are a combination of popular benchmarks for surrogate and UQ methods, most of which are collected from [10], and realistic test cases related to electronic device simulation, which were introduced in [74]. For each test case, all datasets used in the simulations, both for training and validation, are available online [83]. We also provide MATLAB files for the analytical and numerical functions and the SPICE simulation netlists for the electronic designs. The number of training samples is varied as indicated in Table 1. For each training dataset, 50 independent realizations are provided, generated with Latin hypercube sampling (LHS). The test datasets $\mathcal{D}_{\text{test}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ are also generated with LHS. The number of test samples is $N = 10^4$ for both the MATLAB functions and the SPICE test cases. Table 1 also provides relevant parameters such as the number of uncertain parameters, their distribution, the truncation for PCE-based methods, and the size of the available training datasets. The Reader is referred to the indicated references for additional information on the test cases.

The Forrester function is a simple but highly nonlinear one-dimensional function introduced in [87] as

$$\mathcal{M}(x) = (6x - 2)^2 \sin(12x - 4). \tag{48}$$

Table 1
Information on the considered test cases, listed by increasing input dimensionality d. Italic font denotes models that involve numerical simulations, while bold font further indicates test cases that require the use of external software (SPICE); the other models are analytical. For the PCE-based methods, a truncation of order p and hyperbolic truncation factor q is used. For some highly nonlinear analytical functions (Forrester and Friedman) and the state-of-the-art PCE methods available in UQLab, the order is selected adaptively in the range $[5, 20]$. For the SPICE examples, the truncation factor is chosen adaptively between $q = 0.5$ and $q = 0.7$. In SPGL1 and PCE-GPR, the largest value of p and q is used. The available sizes for the training datasets (L) are also indicated, whereas the size of the test datasets is $N = 10^4$ for each case. The last column provides the literature in which the test cases and their input distributions are described in detail.

Test case	d	Distribution	PCE truncation	L	References
Forrester function	1	uniform	$p \in [5, 20]$[a], $q = 1$	10 to 50	[87]
Ishigami function	3	uniform	$p = 14, q = 1$	30 to 200	[10,88]
Friedman function	5	uniform	$p \in [5, 20]$[a], $q = 0.5$	40 to 200	[89]
Duffing-van der Pol oscillator	5	Gaussian/uniform	$p = 5, q = 1$	50 to 250	[23]
Undamped oscillator	6	Gaussian	$p = 5, q = 1$	30 to 150	[10,90]
Borehole function	8	Gaussian/uniform	$p = 5, q = 1$	50 to 300	[10,91]
Wingweight function	10	uniform	$p = 4, q = 1$	50 to 300	[10,87]
Morris function	20	uniform	$p = 8, q = 0.5$	100 to 400	[10,92]
Amplifier gain	25	uniform	$p = 3, q = \{0.5, 0.7\}$[a]	30 to 150	[74,93]
Maximum crosstalk	26	Gaussian	$p = 3, q = \{0.5, 0.7\}$[a]	40 to 120	[74,94]
Interconnect delay	54	Gaussian	$p = 3, q = \{0.5, 0.7\}$[a]	50 to 250	[74,95]

[a] Adaptive

Table 2
Input distributions for the Duffing-van der Pol oscillator, the undamped oscillator, the borehole function, and the wingweight function benchmarks.

Variable	Distribution Duffing-van der Pol oscillator	Undamped oscillator	Borehole function	Wingweight function
x_1	$\mathcal{N}(4, 0.6)$	$\mathcal{N}(1, 0.1)$	$\mathcal{N}(0.1, 0.0161812)$	$\mathcal{U}(150, 200)$
x_2	$\mathcal{N}(-2, 0.4)$	$\mathcal{N}(0.1, 0.01)$	$\mathcal{U}(1120, 1680)$	$\mathcal{U}(220, 300)$
x_3	$\mathcal{N}(2, 0.5)$	$\mathcal{N}(1, 0.05)$	$\mathcal{U}(9855, 12045)$	$\mathcal{U}(6, 10)$
x_4	$\mathcal{U}(0.5, 2.5)$	$\mathcal{N}(0.5, 0.05)$	$\mathcal{U}(63070, 115600)$	$\mathcal{U}(-10, 10)$
x_5	$\mathcal{U}(46, 54)$	$\mathcal{N}(1, 0.2)$	$\mathcal{U}(63.1, 116)$	$\mathcal{U}(16, 45)$
x_6	–	$\mathcal{N}(0.5, 1/12)$	$\mathcal{U}(990, 1110)$	$\mathcal{U}(0.5, 1)$
x_7	–	–	$\mathcal{U}(700, 820)$	$\mathcal{U}(0.08, 0.18)$
x_8	–	–	$\mathcal{N}(7.71, 1.0056)$[a]	$\mathcal{U}(2.5, 6)$
x_9	–	–	–	$\mathcal{U}(1700, 2500)$
x_{10}	–	–	–	$\mathcal{U}(0.025, 0.08)$

[a] Originally lognormal

Since, in its original definition, the function is defined over the support $[0, 1]$, we consider $x \sim \mathcal{U}(0, 1)$ for the UQ analysis.

The Ishigami function [88] is a popular benchmark for UQ methods and is defined as

$$\mathcal{M}(x_1, x_2, x_3) = \sin(x_1) + 7\sin^2(x_2) + 0.1x_3^4 \sin(x_1). \tag{49}$$

The distribution of the input variables is $x_j \sim \mathcal{U}(-\pi, \pi)$, for $j = 1, 2, 3$.

The Friedman function is defined in [89] as

$$\mathcal{M}(x_1, \dots, x_5) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5, \tag{50}$$

with $x_j \sim \mathcal{U}(0, 1)$, for $j = 1, \dots, 5$.

The Duffing-van der Pol oscillator is a popular mathematical model that is used in various scientific fields to describe systems that exhibit self-sustained oscillations with non-linear damping. In our benchmark study, we consider the uncertain description in [23], which involves the numerical solution of the ordinary differential equation

$$x_1\ddot{y} + \sigma(x_2 + by^2)\dot{y} + x_4 y(c + dy^2) = x_3 \cos(x_5 t). \tag{51}$$

In particular, we look at the dynamic response $y(t)$ at time $t = 0.5$ s. The five uncertain parameters $x_1, \dots, x_5$ have the distributions indicated in the second column of Table 2. The remaining parameters are deterministic, with values $\sigma = 1.75$, $b = 1$, $c = 1$, and $d = 1$.

The undamped oscillator was defined in [90] as the solution of a nonlinear mechanical system, given by

$$\mathcal{M}(x_1, \dots, x_6) = 3x_4 - \left| \frac{2x_6}{x_3 w_0^2} \sin\left(\frac{w_0^2 x_5}{2} \right) \right|, \tag{52}$$

with $w_0 = \sqrt{(x_1 + x_2)/x_3}$. The distribution of the six input parameters is indicated in third column of Table 2.

The borehole function simulates the water flow between two aquifers and was defined in [91] as

$$\mathcal{M}(x_1,\dots,x_8) = \frac{2\pi x_4(x_6-x_7)}{\ln\left(e^{x_8}/x_1\right)\left(1+\frac{2x_2x_4}{\left(\ln\left(e^{x_8}/x_1\right)x_1^2x_3\right)}+\frac{x_4}{x_5}\right)}, \tag{53}$$

with the distributions indicated in the fourth column of Table 2. It should be noted that, in the original formulation, a lognormal distribution is ascribed to the parameter $r = e^{x_8}$. However, since the lognormal distribution cannot be handled directly with PCE-GPR, we equivalently consider a Gaussian distribution for $x_8 = \ln(r)$.

The Wingweight function estimates the weight of a light aircraft wing, roughly representative of a Cessna C172 Skyhawk. It was defined in [87] as

$$\mathcal{M}(x_1,\dots,x_{10}) = 0.036\,x_1^{0.758}x_2^{0.0035}\left(\frac{x_3}{\cos(x_4\pi/180)^2}\right)^{0.6}x_5^{0.006}x_6^{0.04}\left(\frac{100x_7}{\cos(x_4\pi/180)}\right)^{-0.3}(x_8x_9)^{0.49}+x_1x_{10}, \tag{54}$$

for which the input distributions are again indicated in the last column of Table 2.

Finally, the last analytical benchmark is the Morris function, defined as [92]

$$\mathcal{M}(x_1,\dots,x_{20}) = \sum_{i=1}^{20}\beta_i w_i + \sum_{i<j}^{20}\beta_{ij}w_iw_j + \sum_{i<j<l}^{20}\beta_{ijl}w_iw_jw_l + 5w_1w_2w_3w_4, \tag{55}$$

where

$$w_i = \begin{cases} 2\left(\dfrac{1.1x_i}{x_i+0.1}-0.5\right) & i=3,5,7 \\ 2(x_i-0.5) & \text{otherwise} \end{cases} \tag{56}$$

The input distributions are $x_i \sim \mathcal{U}(0,1)$, for $i = 1,\dots,20$, [96] and the coefficients are defined as

$$\begin{aligned} \beta_i &= \begin{cases} 20 & i \le 10 \\ (-1)^i & i > 10 \end{cases} \\ \beta_{ij} &= \begin{cases} -15 & i,j \le 6 \\ (-1)^{i+j} & i,j > 6 \end{cases} \\ \beta_{ijl} &= \begin{cases} -10 & i,j \le 5 \\ 0 & i,j > 5 \end{cases}. \end{aligned} \tag{57}$$

The electronic design test cases refer to:

1. The gain at 2 GHz of a bipolar junction transistor low-noise amplifier. The design was originally inspired to [93]. The uncertainty is provided by 25 electrical parameters, including element values, physical properties, and geometrical dimensions. For simplicity, in the SPICE netlists the variation of all parameters is standardized so that $x_j \sim \mathcal{U}(-1,1)$, $\forall j = 1,\dots,20$, and the variation is within $\pm 20\%$ around the nominal value.
2. The maximum interference (crosstalk) between adjacent transmission lines occurring in an electrical interconnect with coupled microstrip lines. The design was originally inspired to [94]. The uncertainty is provided by 26 geometrical and material parameters of the microstrip lines. All variations are standardized to $\mathcal{N}(0,1)$ to provide a relative standard deviation of 10% from the nominal value.
3. The propagation delay occurring in the transient simulation of an electrical interconnect with single microstrip lines. The design was originally inspired to [95]. The delay is defined at the time at which the terminal voltage raises above 100 mV and is expressed in nanoseconds. The uncertainty is provided by 54 parameters, including lumped element values and geometrical and material parameters of the microstrip lines. All variations are standardized to $\mathcal{N}(0,1)$ to provide a relative standard deviation of 10% from the nominal value.

4. PCE-GPR toolbox: implementation and assessment

This section briefly introduces the PCE-GPR toolbox [75]. Additional details are provided in the user guide. The toolbox has been implemented in MATLAB and builds upon the UQLab toolbox, release 2.0.0 [50], and specifically on the Kriging module [86]. UQLab provides all necessary features and a robust and well-consolidated implementation, which makes a custom implementation superfluous. UQLab was preferred over the Statistics and Machine Learning Toolbox™ (SMLTB) in MATLAB, and specifically the function `fitrgp` therein available, or the `scikit-learn` package in Python because it allows greater flexibility in the definition of custom kernel functions, as well as in model training and hyperparameter estimation.

Some UQLab definitions are constrained by the peculiar features of PCE-GPR. In particular, the trend function is set to be zero, whereas the kernel function is defined according to the input distribution, as outlined in Section 2.3. The variation of the input parameters is rescaled internally to the standardized distributions typical of PCE, i.e., $\mathcal{N}(0,1)$ and $\mathcal{U}(-1,1)$ for Gaussian and uniform variability, respectively. This step is required to leverage standard Hermite/Legendre polynomials and their respective kernels and is transparent to the user. The user is allowed to choose between the following main options:

Table 3
Configurations considered for the analysis. If noise is activated, the GPR model works in regression mode; otherwise, the model interpolates training data. The last column provides the labels with which the model are identified in the figures. By default, the Kriging model in UQLab uses a noise-free interpolation with an anisotropic kernel and LOO-CV estimation.

Noise	Hyperparameter estimation	Kernel type	Label
True (regression)	ML	Anisotropic	REG-ML-AN
False (interpolation)	ML	Anisotropic	INT-ML-AN
True	LOO-CV	Anisotropic	REG-LOO-CV-AN
False	LOO CV	Anisotropic	INT-LOO-CV-AN[a]
True	10-out CV	Anisotropic	REG-10out-CV-AN
False	10-out CV	Anisotropic	INT-10out-CV-AN
True	ML	Isotropic	REG-ML-IS
False	ML	Isotropic	INT-ML-IS
True	LOO-CV	Isotropic	REG-LOO-CV-IS
False	LOO CV	Isotropic	INT-LOO-CV-IS

[a] UQLab default

- Isotropic or anisotropic kernel;
- ML or CV estimation of the hyperparameters; this is carried out by considering the logarithm of ρ.
- Interpolation mode (noiseless model) or regression mode (noisy model).

Moreover, advanced UQLab settings regarding CV and the hyperparameter optimizer can also be modified.

Before proceeding with the benchmarking against state-of-the-art methods, we provide a comparative analysis of some implementational features to assess the performance of the toolbox and identify the recommended default settings. The analysis is carried out on the basis of the electronic design test cases, using datasets of three different sizes and, for each dataset, all the 50 independent realizations available. We compare the prediction performance based on the relative mean square error on the available test samples, defined as [10]

$$\text{RelMSE} = \frac{\sum_{i=1}^{N} \left(y_i - \hat{\mathcal{M}}(\boldsymbol{x}_i)\right)^2}{\sum_{i=1}^{N} \left(y_i - \bar{y}\right)^2} \tag{58}$$

where

$$\bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i \tag{59}$$

is the dataset mean and $\hat{\mathcal{M}}$ denotes the model prediction.

4.1. Standard options: kernel type, hyperparameter estimation, and noise

In this section, we assess the impact of the standard settings the user is allowed to select, namely kernel type (isotropic or anisotropic), hyperparameter estimation method (ML or CV), and noise (true or false). For anisotropic kernels, we consider both a LOO and a K-fold CV scheme. For the latter, we leave $k = 10$ samples out, which corresponds to using $K = \lceil L/10 \rceil$ folds, where L is the total number of training samples in the dataset [86]. The hybrid covariance matrix adaptation-evolution strategy (HCMAES) optimizer is used. Table 3 summarized the various configurations considered for the analysis. It should be noted that training data are noiseless, but the noise term in GPR acts as a regularizer and may help avoid overfitting.

The boxplots in Fig. 1 show the resulting RelMSE. The configurations yielding the smallest and largest median across the 50 datasets are highlighted using green and red colors, respectively. Notably, the configuration providing the best result is always the one without noise and using an anisotropic kernel with a ML estimation of the hyperparameters. Conversely, in most cases the worst model is provided by a regression model with either 10-out CV or isotropic kernel. Based on this analysis, we set anisotropic kernels with noise-free ML estimation as default options.

4.2. Optimizer

Next, we assess the performance of the different optimizers available in UQLab for the hyperparameter estimation. The optimizer is invoked to solve the minimization problem associated to the estimation strategy: the minimization of the CV error or the minimization of the negative log-likelihood function (which is equivalent to maximizing the likelihood function). The available options are [86]:

- Broyden-Fletcher-Goldfarb-Shanno (BFGS) method;
- Genetic algorithm (GA);
- Hybrid genetic algorithm (HGA; UQLab default);

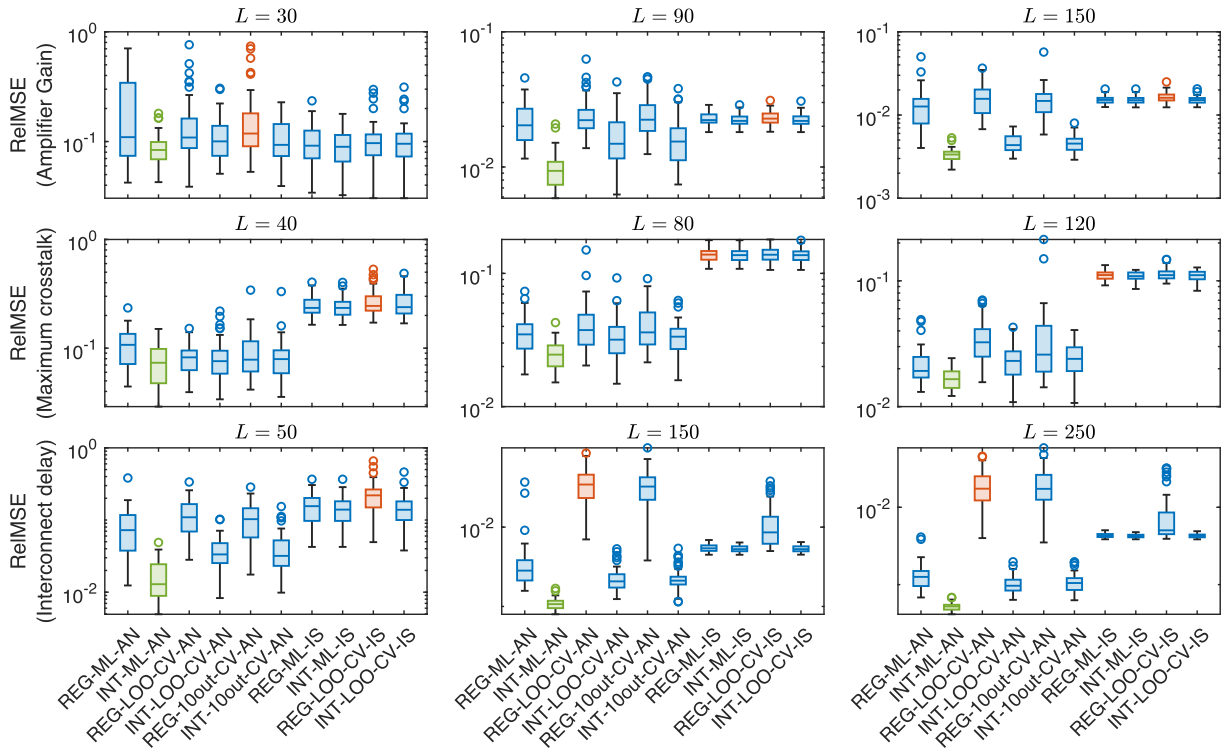


Fig. 1. Prediction error obtained with various model configurations for the three SPICE test cases and three training dataset sizes. Green and red colors highlight the models with the lowest and largest median error, respectively.

Table 4
Performance ranking of the five optimizers for the hyperparameter estimation, based on the Borda score. The Friedman tests suggests that the performance is significantly different across optimization methods with $p = 0.0002$.

Method	Rank	Borda score
HCMAES	1	17
HGA	2	21
CMAES	3	26
BFGS	4	27
GA	5	45

- Covariance matrix adaptation-evolution strategy (CMAES);
- Hybrid covariance matrix adaptation-evolution strategy (HCMAES);

For the standard options, we use the default values identified with the previous analysis, i.e., a noise-free model with an anisotropic kernel and ML estimation.

The results are illustrated in the boxplots of Fig. 2. In this case, the unanimous conclusion is that GA has the worst performance in all cases, with both higher median and larger dispersion across the dataset realizations. Conversely, HGA, CMAES, and HCMAES exhibit a better and rather similar performance. In Table 4, the five methods are ranked based on the Borda score, which assigns points based on the rank in each test case. The method with the lowest total score is ranked best. Overall, the HCMAES ranks first followed by HGA, despite the latter exhibits the best performance in five out of nine cases. Indeed, even when HCMAES is not the best-performing method, its error remains comparable. Although BFGS ranks third in terms of median error, Fig. 2 shows that it exhibits a rather large error dispersion across different training datasets. A Friedman test suggests that the difference in performance is significant with a p-value of 0.0002. Based on this analysis, we select the HCMAES as the default optimization method.

4.3. Hyperparameter bounds

In standard GPR, the main hyperparameter is usually the kernel scale, which is constrained to be strictly positive but is not upper-bounded. In the PCE-GPR kernels, the hyperparameter ρ is limited to the interval $(0, 1)$. The extrema are excluded since the kernels

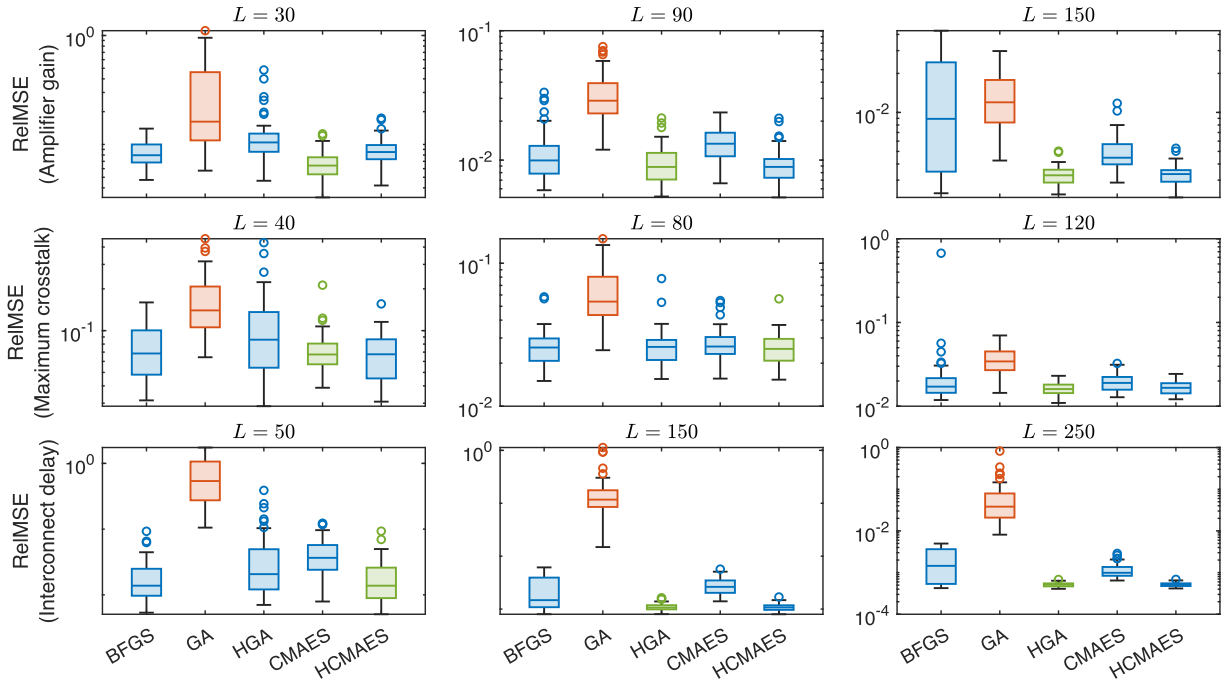


Fig. 2. Prediction error obtained with various optimizers.

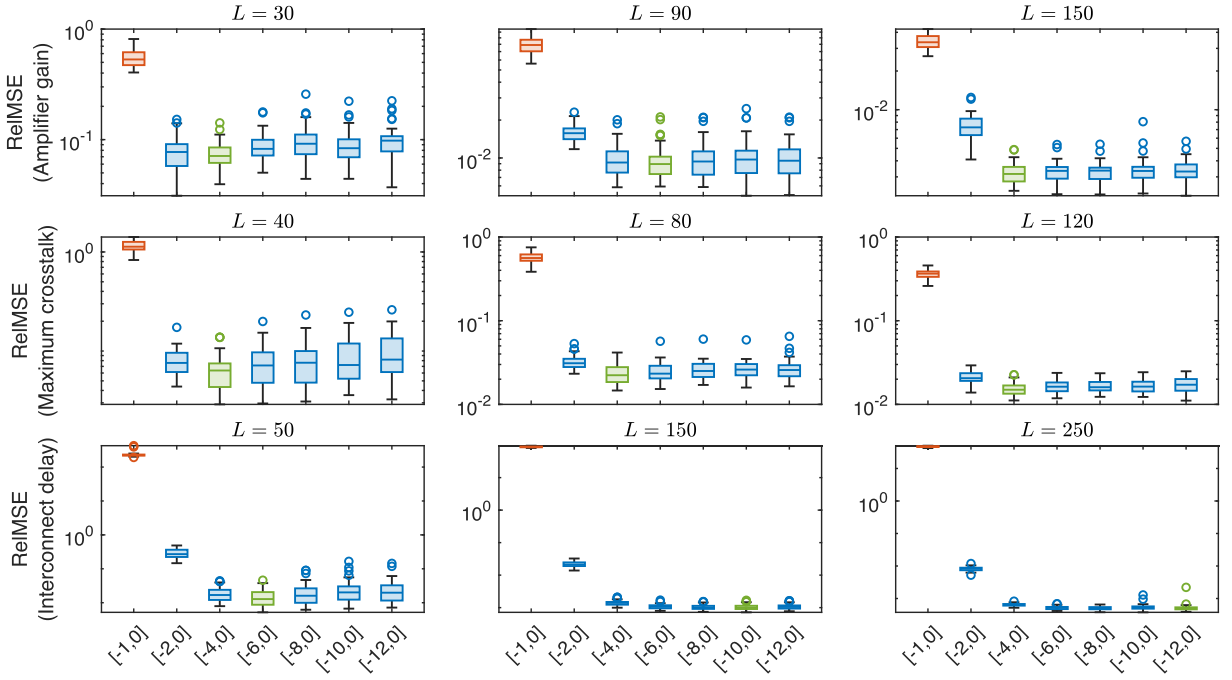


Fig. 3. Prediction error obtained with various hyperparameter optimization bounds.

are singular for $\rho = 1$, while they would yield a vanishing PCE for $\rho = 0$. As ρ plays a crucial role on the accuracy, it is fundamental to understand how close its value may get to these extrema and set correct lower and upper bounds for the optimization. Since we are using a logarithmic transform in the optimization problem, the definition of the lower bound is particularly critical, as clearly this cannot be $-\infty$. Therefore, we assess the prediction accuracy for different hyperparameter bounds. In particular, we consider $\log \rho \in \{[-1,0],[-2,0],[-4,0],[-6,0],[-8,0],[-10,0],[-12,0]\}$.

Table 5
Methods considered for the comparative analysis. The toolbox used for the implementation is indicated along with the specific settings considered.

Method	Type	Toolbox	Version	Specific settings
LAR	PCE	UQLab [97]	v2.0.0	Default
OMP	PCE	UQLab [97]	v2.0.0	Default
SP	PCE	UQLab [97]	v2.0.0	Hyperparameter tuning: 4-fold CV
SP-LOO	PCE	UQLab [97]	v2.0.0	Hyperparameter tuning: LOO-CV (default)
BCS	PCE	UQLab [97]	v2.0.0	Default
SPGL1	PCE	`spgl1` [98]	v2.1	Hyperparameter tuning: 4-fold CV
SVR-L1	kernel	UQLab [82]	v2.0.0	Loss function: L1 (default) Kernel: anisotropic Gaussian Hyperparameter tuning: span LOO estimate (default) QP solver: interior point method Optimizer: HCMAES
SVR-L2	kernel	UQLab [82]	v2.0.0	Loss function: L2 Other settings: same as above
SVR (SMLTB)	kernel	SMLTB [99] (`fitrsvm`)	v24.2	Kernel: isotropic Gaussian Hyperparameter tuning: 5-fold CV QP solver: sequential minimal optimization (default)
LSSVM	kernel	LS-SVMlab [100]	v1.8	Kernel: isotropic Gaussian Hyperparameter tuning: LOO-CV Optimizer: simplex method
GPR (UQLab)	kernel	UQLab [86]	v2.0.0	Trend: constant Kernel: anisotropic Gaussian Noise: false Hyperparameter tuning: ML Optimizer: HCMAES
GPR (SMLTB)	kernel	SMLTB [99] (`fitrgp`)	v24.2	Trend: constant Kernel: anisotropic Gaussian Noise: true Hyperparameter tuning: ML (default) Optimizer: quasi-Newton (default)
PCK (sequential)	kernel	UQLab [101]	v2.0.0	Mode: sequential Kernel: anisotropic Gaussian Noise: false Hyperparameter tuning: ML Optimizer: HCMAES
PCK (optimal)	kernel	UQLab [101]	v2.0.0	Mode: optimal Other settings: same as above
PCE-GPR (ML)	hybrid	PCE-GPR [75]	v1.0	Kernel: anisotropic (default) Hyperparameter tuning: ML (default) Noise: false (default)
PCE-GPR (CV)	hybrid	PCE-GPR [75]	v1.0	Hyperparameter tuning: LOO-CV Other settings: same as above

The results, summarized in Fig. 3, indicate that the lowest median error is obtained by setting the lower bound for ρ to either 10^{-4} or 10^{-6}, while setting it to 10^{-2} or higher significantly reduces accuracy. The prediction error generally stabilizes for lower bounds of 10^{-4} and below, with smaller values having little impact on accuracy. Therefore, we set $[-6, 0]$ as the default optimization bounds for $\log(\rho)$; this can be adjusted through UQLab-specific options.

5. Numerical results and benchmarking

In this section, we benchmark the PCE-GPR toolbox against the state-of-the-art PCE and kernel methods reviewed in Section 2, based on the test cases listed in Table 1. The analysis is carried out in terms of predictive accuracy, assessed using the RelMSE, and computational time. We implement existing methods using available toolboxes.

Table 5 lists all the methods considered, along with the indication of the toolbox used and pertinent information regarding the method's settings. LAR, OMP, SP, and BCS are all available in the PCE module of UQLab [97]. For LAR, OMP, and BCS we use standard default settings. Since SP requires the definition of a CV scheme to estimate the number K of non-zero coefficients, we consider the same two implementations as in [10]: one using a 4-fold CV, which we simply denote as "SP", and one using a LOO-CV scheme, denoted as "SP-LOO". We use instead the `spgl1-2.1` toolbox [98] to implement the SPGL1 method. The toolbox generically allows solving either the LASSO (14) or the basis pursuit denoise (15) problem. We align with the suggestion in [10]: for a given σ, we solve (14) for different values of τ to find the minimum that meets the constraint in (15). In the process, σ is considered as a hyperparameter that is tuned via a 4-fold CV leveraging dedicated features available in the SMLTB. It should be noted that, for the analytical benchmarks, the analysis of the PCE methods replicates the one in [10] and is repeated here only for the sake of completeness and to facilitate a direct and transparent comparison between the two classes of PCE and kernel methods. Yet, the same analysis is extended to numerical and higher-dimensional test cases that were not considered in the previous literature.

As to the kernel methods, the SVR is implemented leveraging both UQLab, via the dedicated module [82], and the SMLTB using the function `fitrsvm`. The UQLab implementation allows choosing between a linear (L1) and a quadratic (L2) ϵ-insensitive loss and supports anisotropic kernels. By default, it uses a span estimate of the LOO-CV error for tuning the hyperparameters. Conversely, `fitrsvm` only supports a linear loss and isotropic kernels. By default, it calculates hyperparameters empirically based on the training data, i.e., it does not tune their value, which results in poor predictive performance. Therefore, we rather choose to tune them using a 5-fold CV scheme.

The LSSVM is implemented via the LS-SVMlab toolbox [100]. We consider an isotropic Gaussian kernel (therein termed "radial basis function" kernel). Anisotropic kernels are not supported. We also investigated other kernel types (e.g., polynomial), which resulted in poorer predictive performance and whose results are therefore omitted. The hyperparameters are tuned via a LOO-CV estimation based on the simplex method.

For the implementation of the standard GPR, we again consider both UQLab and the `fitrgp` function available in the SMLTB. In both cases, we consider a constant trend function, an anisotropic Gaussian kernel, and a ML estimation for tuning the hyperparameters (the SMLTB does not support CV for anisotropic kernels). In UQLab, we use a noise-free model and the HCMAES optimizer for consistency with PCE-GPR; in the SMLTB instead, we consider a noisy model since a noise-free interpolation is not supported (the noise standard deviation must be set to a minimum non-zero value) and the default quasi-Newton optimizer. Matérn kernels were also tested resulting in similar or worse performance and therefore are not considered in our analysis.

The PCK is implemented using the corresponding module in UQLab [101]. We consider both the "sequential" and "optimal" strategy for estimating the trend coefficients: the former first identifies the relevant PCE basis for the trend and then trains the GPR model; the latter implements an iterative algorithm in which the trend coefficients and the GPR model are calibrated concurrently. The remaining settings are chosen consistently with the standard GPR method: anisotropic Gaussian kernel, noise-free model, and ML estimation of the hyperparameters based on the HCMAES optimizer.

All the above methods are compared against PCE-GPR with default settings (cfr. Section 4) except for the hyperparameter estimation method, for which we consider both ML (default) and LOO-CV estimation. It should be noted that, in Table 5, the PCE-GPR method is defined as "hybrid" because it is able to provide a full PCE model. In PCK instead, the PCE is available only for the trend part, while the kernel contribution cannot be analytically represented as such.

5.1. Predictive accuracy

We first compare the methods in terms of predictive performance, starting from the low-dimensional synthetic benchmarks, i.e., the Forrester, Ishigami, and Friedman functions. These functions are highly nonlinear and specifically designed to test surrogate models and stress their predictive performance. For the PCE methods implemented in UQLab, we let the toolbox adaptively select the expansion order p between 5 and 20, with the exception of the Ishigami function, for which we consider $p = 14$ for the sake of consistency with [10]. In the adaptive case, the algorithm tends to pick the highest order ($p = 20$), especially for the largest datasets.

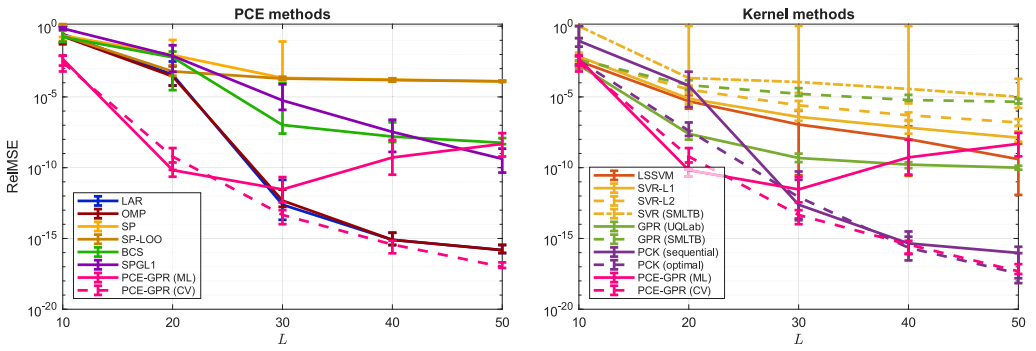
The results are shown in Fig. 4, which provides a compact representation of boxplots. Specifically, the lines indicate the median RelMSE across the 50 training datasets, while the upper and lower bars denote the 75th and 25th percentile, respectively. To avoid cluttering the figure, we collect PCE methods in the left panel and kernel methods in the right panel. In this regard, PCK is considered a kernel method because it does not provide PCE coefficients for the whole model. For PCE-GPR, we report in the right panel the prediction of the kernel model and in the left panel the prediction of the PCE model, which is obtained in post-processing using the highest expansion degree (cfr. Table 1). The difference between the two models is negligible, since the expansion order turns out to be high enough in all cases. It is worth mentioning that, for the scope of our analysis, it is not relevant to convert the model to a PCE, but this is in general useful for analytically obtaining statistical information like Sobol' sensitivity indices, as will be shown later in Section 5.3.

For the Forrester function, PCE-GPR significantly outperforms all the state-of-the-art PCE methods, achieving a median relative accuracy of almost 10^{-3} with 10 training samples only and of about 10^{-10} with 20 training samples. The latter is about six orders of magnitude smaller compared to the other PCE methods. LAR and OMP achieve similar accuracy beyond 30 training samples. PCE-GPR also tends to be more accurate than other kernel methods, exhibiting performance similar to that of PCK for larger training datasets. The ML and CV estimations of the PCE-GPR hyperparameters are found to provide similar accuracy for the smaller datasets, whereas the accuracy of the former degrades when the number of training samples increases.

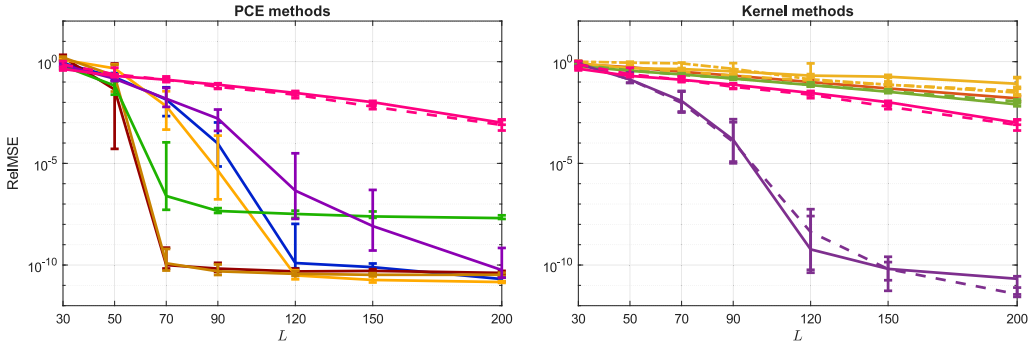
For the Ishigami function instead, all kernel methods, including PCE-GPR, perform much worse than PCE except for PCK, for which the PCE trend likely contributes to most of the accuracy. The optimal strategy overall achieves slightly better accuracy compared to the sequential one. PCE-GPR achieves better performance only for the smallest dataset with $L = 30$ training samples. OMP and SP-LOO provide very similar accuracy, which is slightly worse than SP for the largest training datasets. However, SP exhibits a much larger error for smaller datasets. Among the other kernel methods, PCE-GPR exhibits the best accuracy, with a median relative error of about 10^{-3} for the largest datasets.

For the Friedman function, all methods provide comparable accuracy, with SPGL1 and SP-LOO performing the worst and the best among PCE methods, respectively. PCE-GPR and PCK achieve similar performance as SP-LOO, especially for the smaller datasets. All in all, PCE-GPR is found to perform similarly or better than the state-of-the-art PCE and kernel methods except for the Ishigami function.

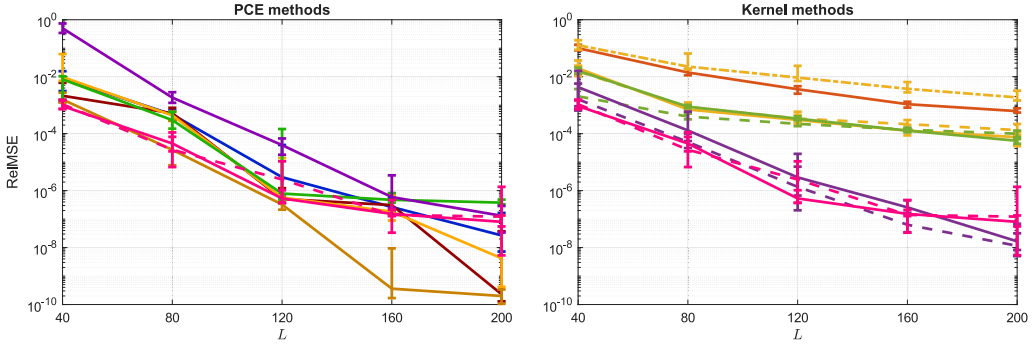
Next, we consider the remaining low-dimensional examples, i.e., the ones with $d \leq 10$, which are characterized by a moderate nonlinearity. The results are shown in Fig. 5. In all cases, PCE-GPR achieves an error that is one to two orders of magnitude smaller than the best performing state-of-the-art technique. Specifically, the comparison with standard GPR, which virtually considers the same



(a) Forrester function ($d = 1$)



(b) Ishigami function ($d = 3$)



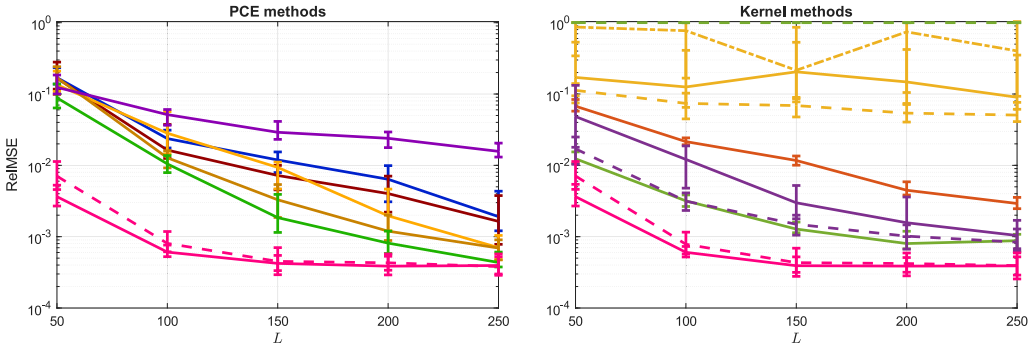
(c) Friedman function ($d = 5$)

Fig. 4. Predictive performance of PCE methods (left panels) and kernel methods (right panels) for the Forrester function (a), Ishigami function (b), and Friedman function (c). For PCE-GPR, the left panels report the prediction of the PCE model obtained using the same truncation as for the other PCE methods.

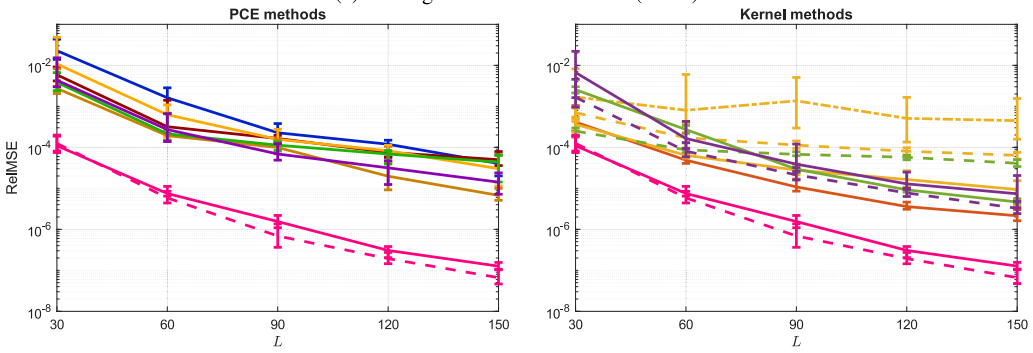
settings apart from the kernel function, suggests that the special kernels leveraged by PCE-GPR play a fundamental role in improving the accuracy. Regarding the PCE-GPR hyperparameter estimation, the performance of ML and CV is found to be comparable.

Among the state-of-the-art kernel methods based on support vector machines, the LSSVM achieves the best performance for the Duffing-van der Pol oscillator and the undamped oscillator. For the Borehole and Wingweight functions, the SVR-L1 in UQLab provides comparable performance, although the accuracy tends to degrade for larger training datasets. Conversely, the SVR model trained with `fitrsvm` provides a much larger error for all test cases. As to PCK, the optimal training strategy provides, as expected, a better performance. However, as we will discuss later on, this comes at the price of a much larger training cost. Compared to PCK, the GPR model trained with `fitrgp` achieves even better performance for the Borehole and Wingweight functions and the smaller training datasets, while it fails to provide an accurate model for the Duffing-van der Pol oscillator.

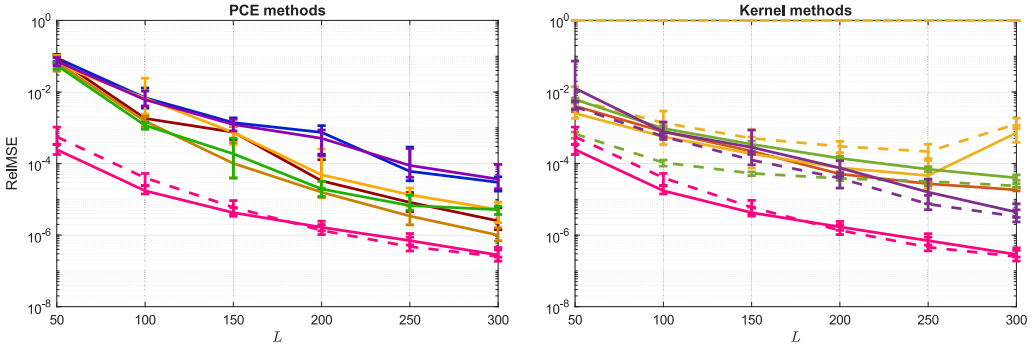
In order to better benchmark the PCE-GPR performance against state-of-the-art methods, Fig. 6 restricts the comparison to the best performing PCE and kernel techniques for each test case. For the Duffing-van der Pol oscillator and the Wingweight function, BCS provides the best accuracy among the standard PCE methods. SP-LOO is instead the best PCE method for the undamped oscillator and Borehole function. As far as kernel methods are concerned, the LSSVM provides the best performance for the undamped oscillator. For the other three benchmarks, GPR performs the best but – for the Borehole and Wingweight functions – is outperformed by the optimal



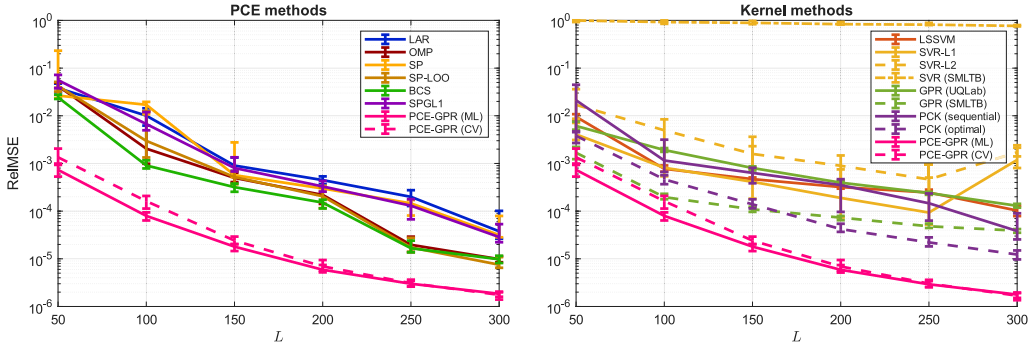
(a) Duffing-van der Pol oscillator ($d = 5$)



(b) Undamped oscillator ($d = 6$)



(c) Borehole function ($d = 8$)



(d) Wingweight function ($d = 10$)

Fig. 5. Predictive performance of PCE methods (left panels) and kernel methods (right panels) for the Duffing-van der Pol oscillator (a), the undamped oscillator (b), the Borehole function (c), and the Wingweight function (d).

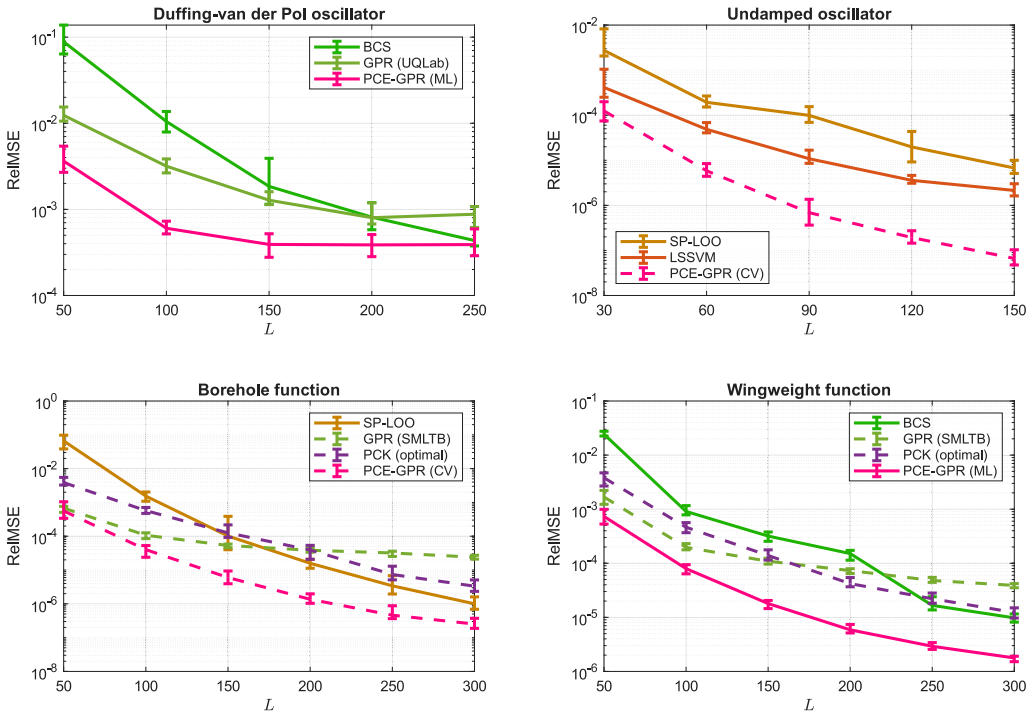


Fig. 6. Comparison between PCE-GPR and the best performing PCE and kernel methods for the low-dimensional test cases: Duffing-van der Pol oscillator ($d = 5$), undamped oscillator ($d = 6$), Borehole function ($d = 8$), and Wingweight function ($d = 10$). Line style and color are the same as in Fig. 5.

Table 6
Summary of best performing techniques for the considered test cases.

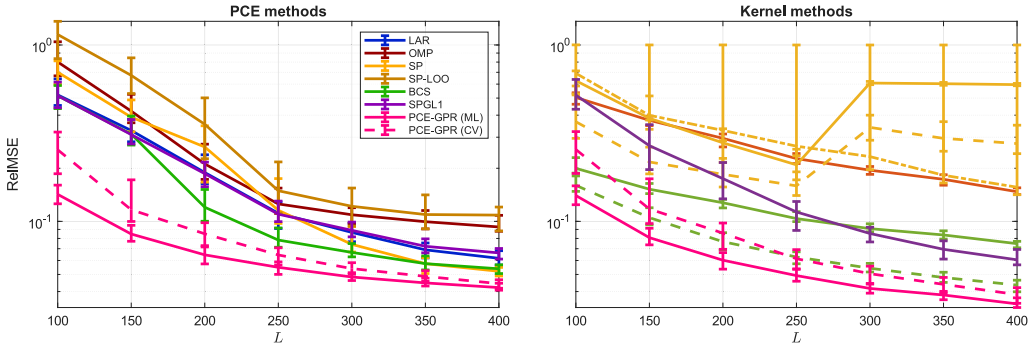
Test case	Best state-of-the-art PCE	Best state-of-the-art kernel	Best overall	PCE-GPR rank
Forrester function	LAR / OMP	PCK	PCE-GPR	1st
Ishigami function	OMP / SP-LOO	PCK	OMP / SP-LOO	7th
Friedman	SP-LOO	PCK	SP-LOO	1st to 5th
Duffing-van der Pol oscillator	BCS	GPR	PCE-GPR	1st
Undamped oscillator	SP-LOO	LSSVM	PCE-GPR	1st
Borehole function	SP-LOO	GPR / PCK	PCE-GPR	1st
Wingweight function	BCS	GPR / PCK	PCE-GPR	1st
Morris function	BCS	GPR	PCE-GPR	1st
Amplifier gain	BCS	GPR	PCE-GPR	1st
Maximum crosstalk	BCS	GPR	PCE-GPR	1st
Interconnect delay	BCS	PCK	PCE-GPR	1st

PCK when using larger training datasets. The analysis of Fig. 6 suggests that the kernel methods tend to perform better than PCE techniques, especially when the number of training samples is small. Moreover, PCE-GPR outperforms all state-of-the-art methods.

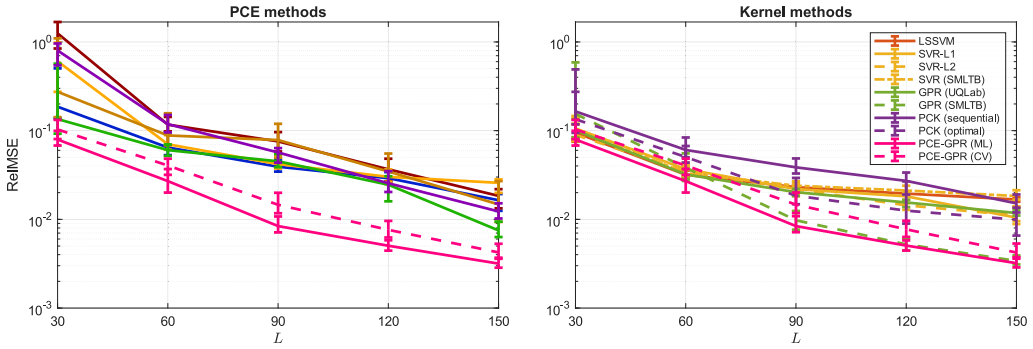
Similar results are provided for the high-dimensional examples in Fig. 7. The performance of the various methods is found to be more consistent between the different test cases. In particular, BCS always performs the best among PCE methods. As to kernel methods, the GPR models trained with `fitrgp` perform the best except for the interconnect delay test case, for which the model returned a nearly constant prediction. For this test case, UQLab failed to train the SVR models and the best accuracy is obtained with the optimal PCK. The training of the optimal PCK model run out of memory for the Morris function, due to the combination of high dimensionality and large number of training data. For PCE-GPR, ML estimation consistently yields a more accurate model than CV. Therefore, ML appears to be more suitable to estimate hyperparameters in high-dimensional settings.

As before, we restrict the comparison of PCE-GPR against the best performing PCE and kernel methods in Fig. 8. It is observed that kernel methods outperform PCE methods except for the interconnect delay test case, for which BCS and optimal PCK provide similar accuracy. The comparisons also show that PCE-GPR performs similarly to or better than state-of-the-art methods.

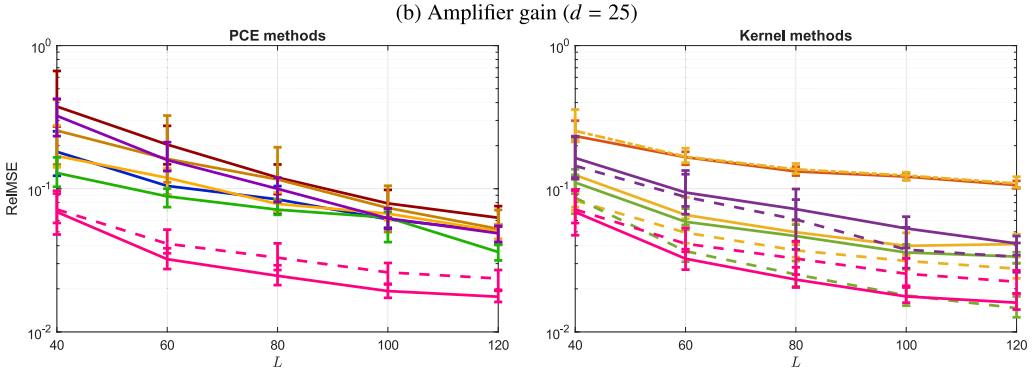
Table 6 summarizes, for each test case, the best-performing methods among the state-of-the-art PCE and kernel techniques. It also highlights the overall best method, considering in this case also PCE-GPR, and reports the rank of PCE-GPR among the ten different techniques evaluated (excluding their variants). Overall, SP-LOO and BCS exhibit the best performance among PCE methods, while GPR and PCK lead among kernel methods. Although treated here as a distinct method, PCK is in fact a specific configuration of GPR.



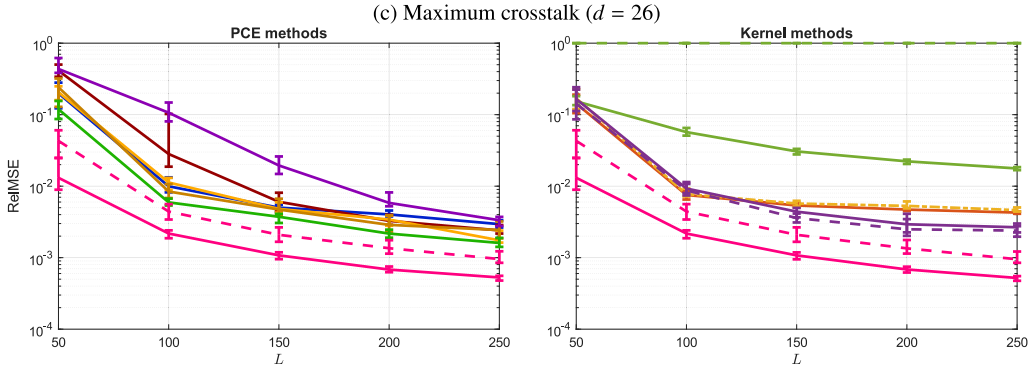
(a) Morris function ($d = 20$)



(b) Amplifier gain ($d = 25$)



(c) Maximum crosstalk ($d = 26$)



(d) Interconnect delay ($d = 54$)

Fig. 7. Predictive performance of PCE methods (left panels) and kernel methods (right panels) for the Morris function (a), amplifier gain (b), maximum crosstalk (c), and interconnect delay (d). For PCE-GPR, the left panels report the prediction of the PCE model obtained using the same truncation as for the other PCE methods.

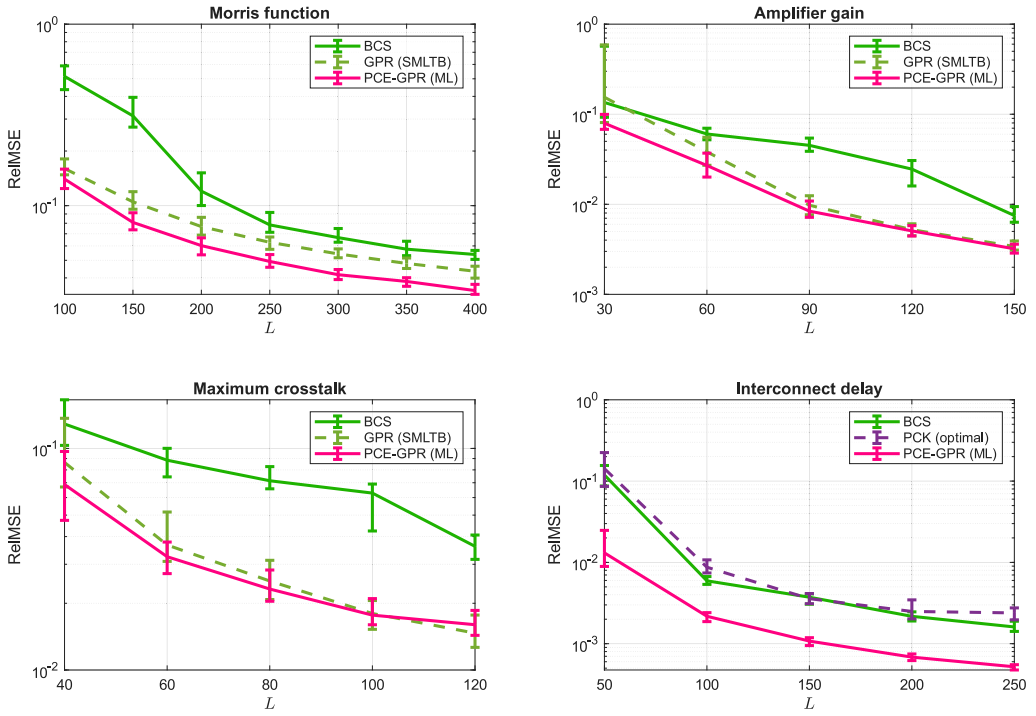


Fig. 8. Comparison between PCE-GPR and the best performing PCE and kernel methods for the high-dimensional test cases: Morris function ($d = 20$), Amplifier gain ($d = 25$), Maximum crosstalk ($d = 26$), and Interconnect delay ($d = 54$). Line style and color are the same as in Fig. 5.

As previously noted, kernel methods generally outperform PCE techniques, particularly in high-dimensional examples and with small experimental designs. In most cases, PCE-GPR achieves the highest rank, with the exception of the Ishigami and Friedman functions, which are two of the most nonlinear benchmarks. For the latter, the ranking is highly sensitive to the dataset size and ranges from first to fifth place. This highlights a possible limitation, or a reduced efficacy, of PCE-GPR in modeling highly nonlinear functions, although its accuracy remains within acceptable levels. It is worth noting, however, that these are highly artificial examples, whereas for the test cases that are representative of physical systems – both the analytical and numerical ones – PCE-GPR consistently and significantly outperforms all other methods.

5.2. Comparison against the PCE-LSSVM method

In this section, we further compare the PCE-GPR method against its LSSVM-based predecessor [85], which is however not available in a public toolbox. We base the analysis on the three SPICE examples, i.e., amplifier gain, maximum crosstalk, and interconnect delay, for which reference information and results are available in [85]. As in [85], we consider isotropic kernels, since anisotropic kernels are found to provide similar or worse performance. The PCE-LSSVM hyperparameters (ρ and γ) are tuned based on the LOO-CV error and Bayesian optimization. For PCE-GPR instead, we tune the hyperparameters based on ML estimation, since it was shown in Section 5.1 to provide better results for these benchmarks.

The PCE-LSSVM and PCE-GPR methods are compared in terms of prediction accuracy in Fig. 9. PCE-GPR is shown to significantly outperform PCE-LSSVM in all three test cases. Despite using a similar kernel formulation, PCE-GPR takes advantage of the ML estimation of the hyperparameters and the robust training algorithms available in UQLab.

5.3. Sobol' indices

As highlighted before, PCE-GPR allows the calculation of statistical information in closed form thanks to the analytical conversion to a PCE. In this section, we compare the performance of PCE-GPR against standard PCE methods in the calculation of Sobol' sensitivity indices (8) for four selected examples, namely the Ishigami function, the undamped oscillator, the Borehole function, and the Wingweight function. The comparison is provided against the best performing PCE method according to Table 6, i.e., the SP-LOO method for the first three test cases and the BCS method for the Wingweight function.

Reference values are obtained with a quasi-Monte Carlo analysis based on Sobol' sampling sequences [102]. This approach involves the calculation of $N(d+2)$ samples, where N is the baseline number of Monte Carlo evaluations. For an accurate reference, we use $N = 10^5$. Therefore, the total number of samples is $5 \cdot 10^5$ for the Ishigami function, $8 \cdot 10^5$ for the undamped oscillator, 10^6 for the

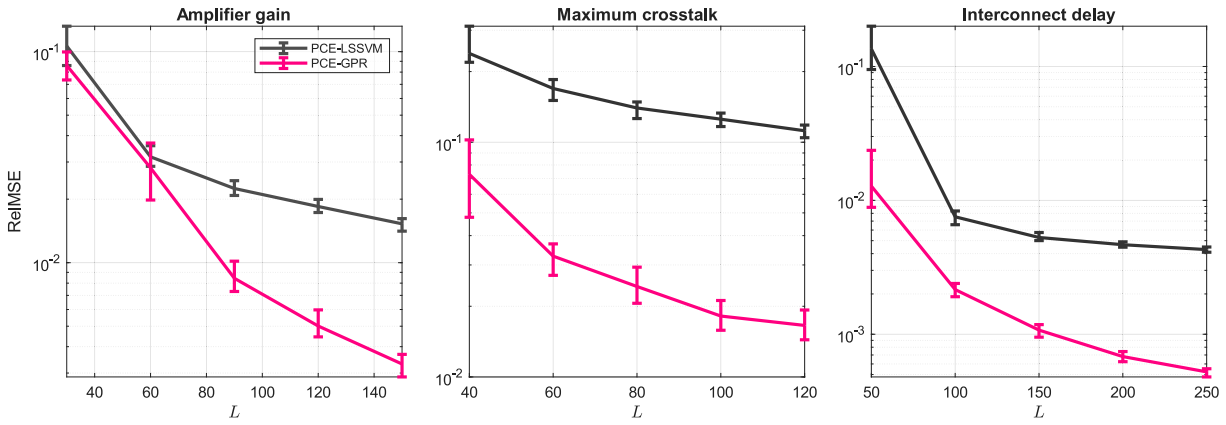


Fig. 9. Comparison between PCE-GPR and PCE-LSSVM for the three SPICE examples.

Table 7
Comaparison between PCE-GPR and the standard PCE methods in the calculation of Sobol' sensitivity indices for the Ishigami function, the undamped oscillator, the Borehole function, and the Wingweight function. For each Sobol' index, the minimum and maximum prediction across the 50 independent realizations of the smallest and largest datasets are reported.

			Smallest dataset		Largest dataset	
Function	Sobol' index	Monte Carlo	Standard PCE	PCE-GPR	Standard PCE	PCE-GPR
Ishigami	S_{T_1}	0.5650	[0.2619, 1.0000]	[0.0259, 0.8573]	[0.5576, 0.5576]	[0.5544, 0.5720]
	S_{T_2}	0.4484	[0.3391, 1.0000]	[0.0895, 0.8713]	[0.4424, 0.4424]	[0.4376, 0.4539]
	S_{T_3}	0.2469	[0.2547, 0.9607]	[0.0001, 0.9569]	[0.2437, 0.2437]	[0.2416, 0.2685]
Undamped oscillator	S_{T_1}	0.0010	[0, 0.0510]	[0.0006, 0.0013]	[0.0009, 0.0010]	[0.0009, 0.0010]
	S_{T_2}	0.0000	[0, 0.0087]	[0, 0.0004]	[0, 0]	[0, 0]
	S_{T_3}	0.0128	[0.0092, 0.0578]	[0.0118, 0.0141]	[0.0125, 0.0127]	[0.0126, 0.0126]
	S_{T_4}	0.6251	[0.2921, 0.6472]	[0.6088, 0.6180]	[0.6140, 0.6154]	[0.6148, 0.6149]
	S_{T_5}	0.2056	[0.1847, 0.6075]	[0.1965, 0.2117]	[0.2018, 0.2031]	[0.2022, 0.2023]
	S_{T_6}	0.1786	[0.1382, 0.2063]	[0.1706, 0.1783]	[0.1754, 0.1759]	[0.1756, 0.1757]
Borehole	S_{T_1}	0.6982	[0.6286, 0.7747]	[0.6722, 0.7126]	[0.6940, 0.6945]	[0.6941, 0.6943]
	S_{T_2}	0.1034	[0.0662, 0.2542]	[0.0940, 0.1090]	[0.1027, 0.1029]	[0.1027, 0.1028]
	S_{T_3}	0.0253	[0, 0.1649]	[0.0199, 0.0263]	[0.0251, 0.0252]	[0.0251, 0.0251]
	S_{T_4}	0.0000	[0, 0.0576]	[0, 0.0024]	[0, 0]	[0, 0]
	S_{T_5}	0.0000	[0, 0.0429]	[0, 0.0035]	[0, 0]	[0, 0]
	S_{T_6}	0.1068	[0.0688, 0.1948]	[0.0970, 0.1145]	[0.1060, 0.1062]	[0.1060, 0.1062]
	S_{T_7}	0.1067	[0.0712, 0.2324]	[0.1003, 0.1130]	[0.1058, 0.1062]	[0.1061, 0.1062]
	S_{T_8}	0.0000	[0, 0.0973]	[0, 0.0032]	[0, 0]	[0, 0]
Wingweight	S_{T_1}	0.1286	[0.1044, 0.1592]	[0.1222, 0.1394]	[0.1276, 0.1282]	[0.1277, 0.1280]
	S_{T_2}	0.0000	[0, 0.0068]	[0, 0.0008]	[0, 0]	[0, 0]
	S_{T_3}	0.2272	[0.1987, 0.2606]	[0.2201, 0.2379]	[0.2256, 0.2263]	[0.2259, 0.2261]
	S_{T_4}	0.0005	[0, 0.0063]	[0, 0.0008]	[0.0005, 0.0005]	[0.0005, 0.0005]
	S_{T_5}	0.0001	[0, 0.0180]	[0, 0.0006]	[0.0001, 0.0001]	[0.0001, 0.0001]
	S_{T_6}	0.0019	[0, 0.0255]	[0.0013, 0.0027]	[0.0018, 0.0019]	[0.0019, 0.0019]
	S_{T_7}	0.1459	[0.1159, 0.1658]	[0.1344, 0.1535]	[0.1447, 0.1454]	[0.1449, 0.1453]
	S_{T_8}	0.4219	[0.3604, 0.4868]	[0.4121, 0.4330]	[0.4191, 0.4202]	[0.4194, 0.4199]
	S_{T_9}	0.0881	[0.0557, 0.1161]	[0.0759, 0.0929]	[0.0874, 0.0878]	[0.0875, 0.0877]
	$S_{T_{10}}$	0.0034	[0, 0.0193]	[0.0021, 0.0045]	[0.0033, 0.0034]	[0.0033, 0.0034]

Borehole function, and $1.2 \cdot 10^6$ for the Wingweight function. Hence, the calculation of an accurate reference becomes prohibitive for the numerical benchmarks.

The results are collected in Table 7 for two scenarios, considering the smallest and largest available datasets. The minimum and maximum prediction of each Sobol' index, obtained across the 50 realizations available in the public dataset [83], are reported. The performance is consistent with the one observed in Section 5.1. Both the standard PCE and PCE-GPR struggle to predict the Sobol' indices of the Ishigami function with the small datasets, exhibiting a large dispersion between the minimum and maximum value. The results are instead accurate and comparable for the large datasets. For the other three benchmarks, the PCE-GPR result is much closer to the reference and exhibits low dispersion already for the small datasets. The accuracy is again similar for the large ones. Yet, the PCE-GPR prediction is more consistent across the different realizations. This analysis corroborates the conclusion that PCE-GPR tends to be more accurate than standard PCE methods, especially in small data regimes.

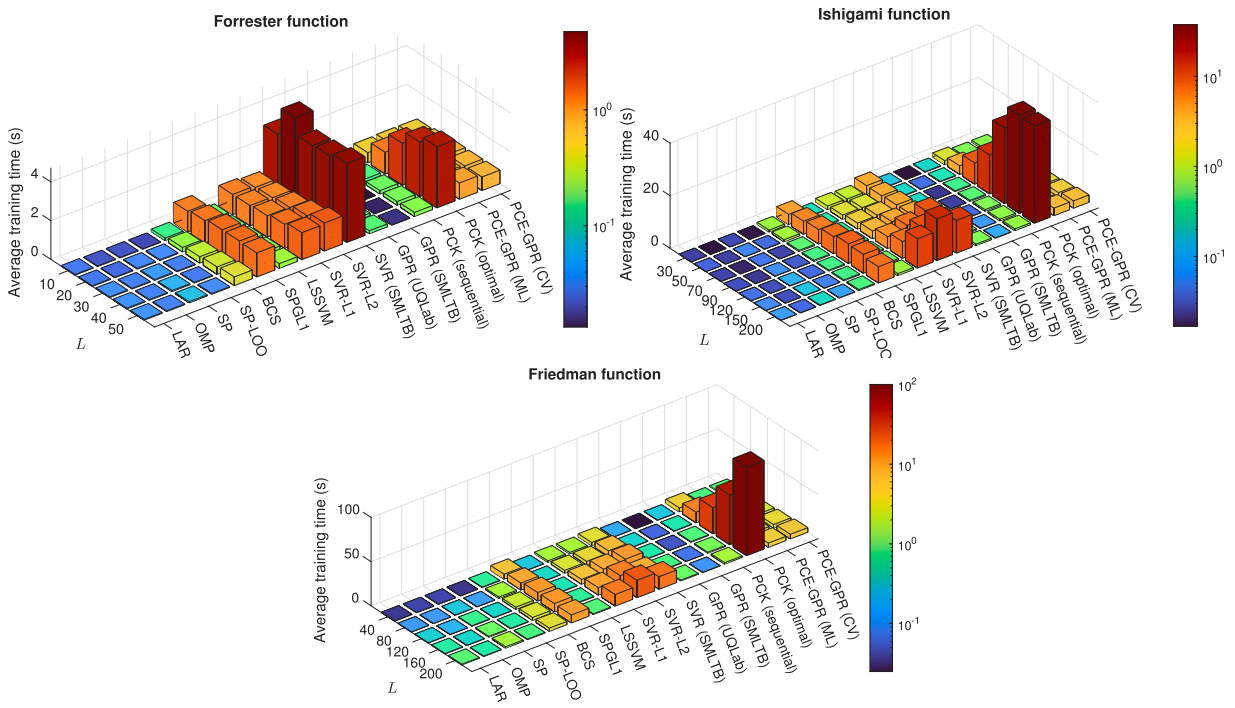


Fig. 10. Training cost for the highly nonlinear low-dimensional test cases. For each dataset size, the cost is computed by averaging the training time over the 50 independent realizations. The color of the bars reflects their relative height on a logarithmic scale.

5.4. Computational time

We now compare the performance in terms of training cost. To this end, we record the time taken by training each model and we take the average across the 50 independent runs. The results are shown by means of colored bars in Fig. 10 for the highly nonlinear examples, in Fig. 11 for the moderately nonlinear and low-dimensional benchmarks, and in Fig. 12 for the high-dimensional test cases. The color reflects the height of the bar on a logarithmic scale. The training cost is then summarized in Table 8, which reports the largest training time taken by each method for the various test cases, usually corresponding to the largest training dataset.

The analysis shows that the training of PCE techniques is generally faster compared to kernel methods, especially for larger training datasets. This is a known limitation of kernel methods, which is due to the inversion of a potentially large kernel/covariance matrix. Among the state-of-the-art PCE methods, SPGL1 is the slowest, followed by BCS. In absolute terms, the slowest training time for SPGL1 is 35.1 s for the interconnect delay, while it is 22.8 s for BCS with the Morris function. The remaining methods (i.e., LAR, OMP, SP, and SP-LOO) are significantly faster, with SP taking the largest training time, i.e., 1.6 s, for the Friedman function with $L = 200$.

As to kernel methods, SVR is usually slower because it requires to solve a quadratic programming minimization problem rather than a linear system, as it is instead the case for LSSVM and GPR. In this regard, the training of LSSVM is below 3 s, while it remains below 6 s for standard GPR. This difference is also due to the fact that the LS-SVMlab only supports isotropic kernels, whereas the training of anisotropic GPR models is more expensive. In contrast, SVR takes up to 99 s for the Morris function and quadratic loss function. As to PCK, the optimal strategy is by far more expensive than the sequential strategy, taking up to 349 s for the interconnect delay problem. The sequential strategy is always below 5 s instead. The difference between optimal and sequential strategies is as large as 146× for the Wingweight function (281 s vs 2 s). Compared to PCE techniques, standard GPR and sequential PCK exhibit similar training cost, while often performing better in terms of accuracy.

Concerning PCE-GPR, the training cost remains within feasible limits, usually tens of seconds, except for the Morris function, for which it reaches 188 s. In this regard, it is observed that the training is more time-consuming for problems with uniform distribution due to the evaluation of the elliptic integral within the Legendre kernel. No significant difference is found instead by using a ML or CV estimation of the hyperparameters. Hence, the superior accuracy that PCE-GPR was shown to provide comes at the price of a slightly higher training cost. It should be noted, however, that in realistic applications the collection of training samples is often expensive, which makes a training cost within tens or hundreds of seconds still acceptable.

6. Additional examples

In this section, we report seven additional examples to further enhance the comparison and demonstrate the performance of the hybrid PCE-GPR method. All examples are taken from relevant UQ-related literature and include applications to partial differential

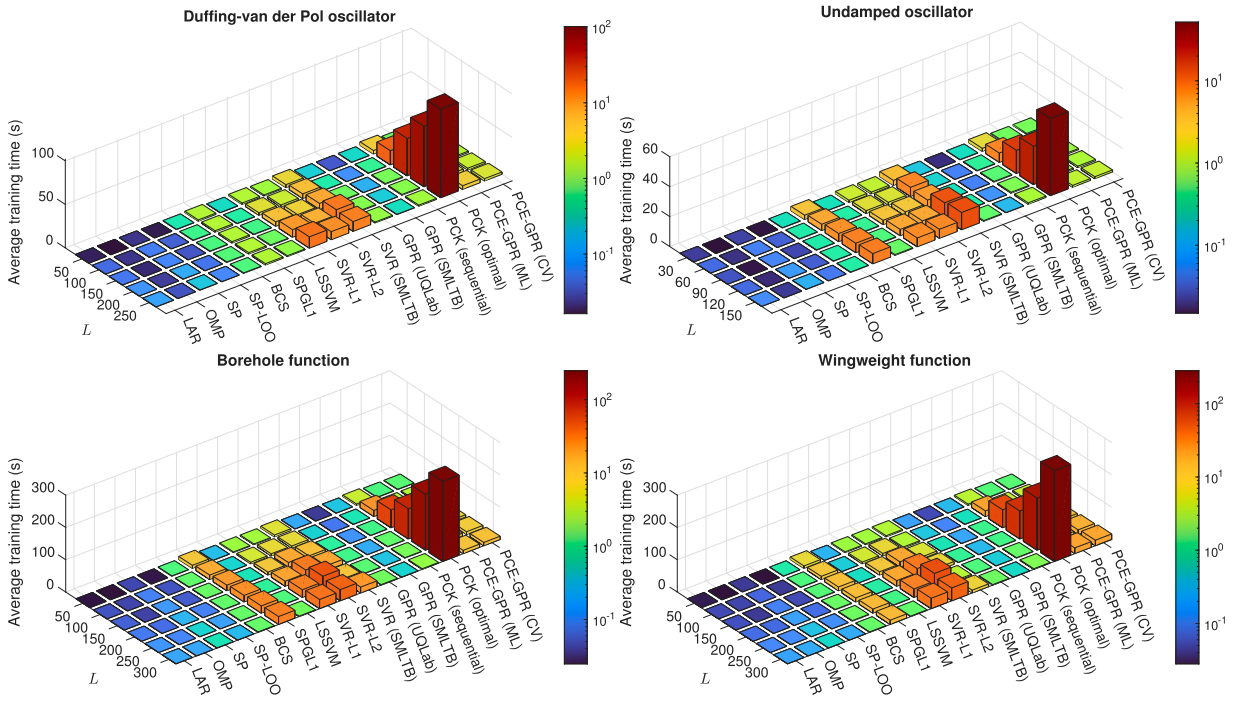


Fig. 11. Training cost for the low-dimensional test cases with moderate nonlinearity.

Table 8
Maximum average training cost (time in seconds) for the considered methods and the various test cases.

Method	Forrester	Ishigami	Friedman	Duffing-van der Pol	Undamped oscillator	Borehole	Wingweight	Morris	Gain	Crosstalk	Delay
LAR	0.04	0.08	0.82	0.11	0.06	0.18	0.20	0.14	0.08	0.06	0.21
OMP	0.04	0.04	0.47	0.04	0.02	0.18	0.22	0.44	0.06	0.04	0.17
SP	0.06	0.19	1.59	0.18	0.10	0.54	0.47	0.56	0.12	0.10	0.48
SP-LOO	0.04	0.09	0.72	0.05	0.06	0.25	0.22	0.23	0.06	0.05	0.21
BCS	0.45	0.79	3.67	0.59	0.28	1.37	1.59	22.83	1.64	1.52	6.03
SPGL1	1.28	7.50	9.59	1.62	6.88	25.45	13.18	21.13	6.28	6.61	35.12
LSSVM	0.24	0.75	0.77	1.06	0.52	1.46	1.47	2.56	0.54	0.43	1.09
SVR-L1	1.40	11.15	13.09	13.05	5.93	28.54	32.38	83.03	12.99	9.24	n/a
SVR-L2	1.45	15.82	19.75	9.38	6.05	41.43	50.22	99.46	17.10	7.36	n/a
SVR (SMLTB)	4.71	12.43	14.18	12.33	11.58	18.21	8.70	10.69	7.96	5.42	6.06
GPR (UQLab)	0.16	0.57	0.93	1.10	0.49	1.46	1.63	5.08	0.96	0.70	2.99
GPR (SMLTB)	0.02	0.10	0.12	0.27	0.12	0.37	0.49	4.80	0.80	0.61	1.07
PCK (sequential)	0.21	0.69	1.68	1.27	0.97	2.03	1.93	4.84	1.35	0.90	3.78
PCK (optimal)	3.15	37.41	100.26	101.87	51.67	248.12	281.33	n/a	75.02	27.42	348.78
PCE-GPR (ML)	0.84	2.88	4.76	4.20	1.24	10.07	17.67	188.06	20.10	2.93	24.19
PCE-GPR (CV)	0.70	2.79	5.22	2.67	1.35	10.25	17.19	183.91	16.84	1.91	9.73

equations with stochastic coefficients described by high-dimensional random fields, as well as thermal and structural mechanics problems. This analysis aims to validate the performance of the hybrid PCE-GPR method on realistic, high-dimensional problems that are relevant to various engineering fields and require numerical solutions. Based on the results of the previous section, we restrict the analysis to LAR, OMP, SP-LOO, and BCS among the PCE methods, and to GPR, PCK, and PCE-GPR among the kernel methods. For GPR, we consider only the UQLab implementation for a direct comparison with the PCE-GPR toolbox. The PCK simulations are limited to the sequential strategy. For GPR, PCK, PCE-GPR, we consider a ML estimation of the hyperparameters. Reference results are generated with a Monte Carlo analysis based on $N = 10^4$ samples.

6.1. Stochastic boundary value problems

The first example is taken from [103] and concerns the solution of the stochastic partial differential equation

$$-\frac{d}{dx}\left(a(x;\xi)\frac{d}{dx}u(x;\xi)\right) = 1, \tag{60}$$

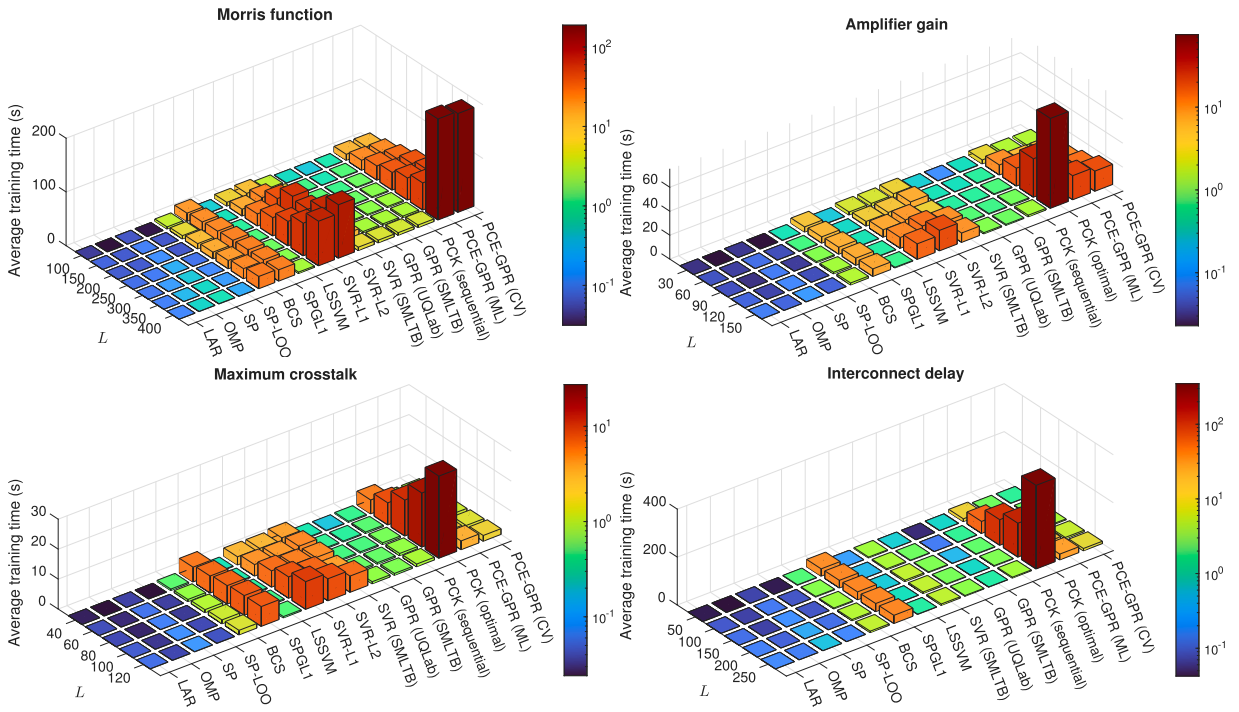


Fig. 12. Training cost for the high-dimensional test cases.

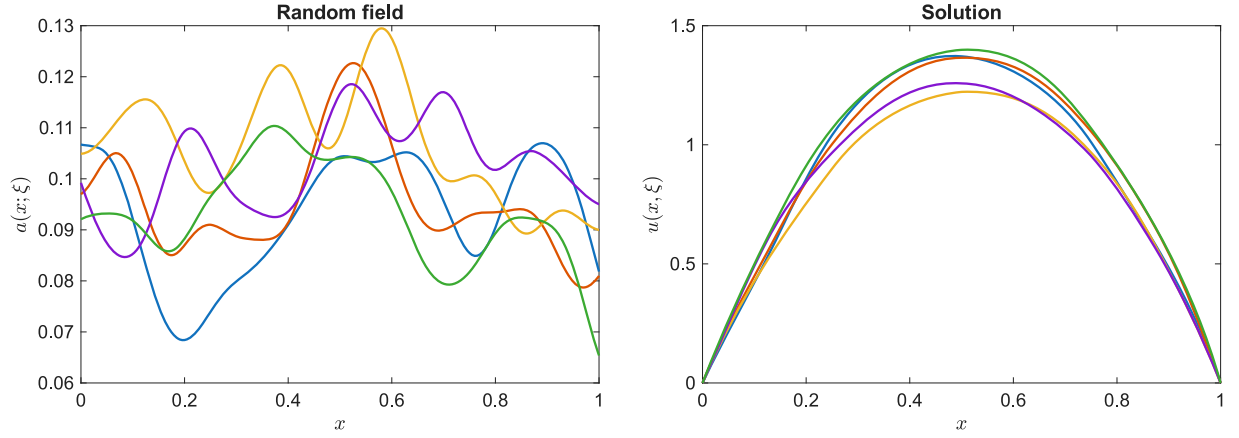


Fig. 13. Five realizations of the Gaussian random field (left) and corresponding solutions of (60) (right).

with $x \in [0,1]$ and Dirichlet boundary conditions $u(0) = u(1) = 0$. The random diffusion coefficient is expressed as the truncated Karhunen-Loève expansion (KLE) [104]

$$a(x;\boldsymbol{\xi}) = \bar{a}(x) + \sigma_a \sum_{k=1}^{d} \sqrt{\lambda_k}\phi_k(x)\xi_k, \tag{61}$$

with $\bar{a} = 0.1$, $\sigma_a = 0.021$, and $d = 40$ terms, where $\{\lambda_k\}_{k=1}^{d}$ and $\{\phi_k(x)\}_{k=1}^{d}$ are the eigvenvalues and eigenfunctions of the Gaussian covariance kernel

$$c(x,x') = \exp\left(-\frac{(x-x')^2}{l_x^2}\right) \tag{62}$$

with autocorrelation length $l_x = 1/14$. It should be noted that, for consistency with the literature, x now denotes the spatial variable, while the uncertain variables are denoted with ξ. The random variables $\{\xi_j\}_{j=1}^{40}$ are uniformly distributed in $[-1,1]$, i.e., $\xi_j \sim \mathcal{U}(-1,1)$ for $j = 1,\ldots,40$. The Eq. (60) is solved numerically in MATLAB using the Partial Differential Equation Toolbox [105]. The eigenvalues

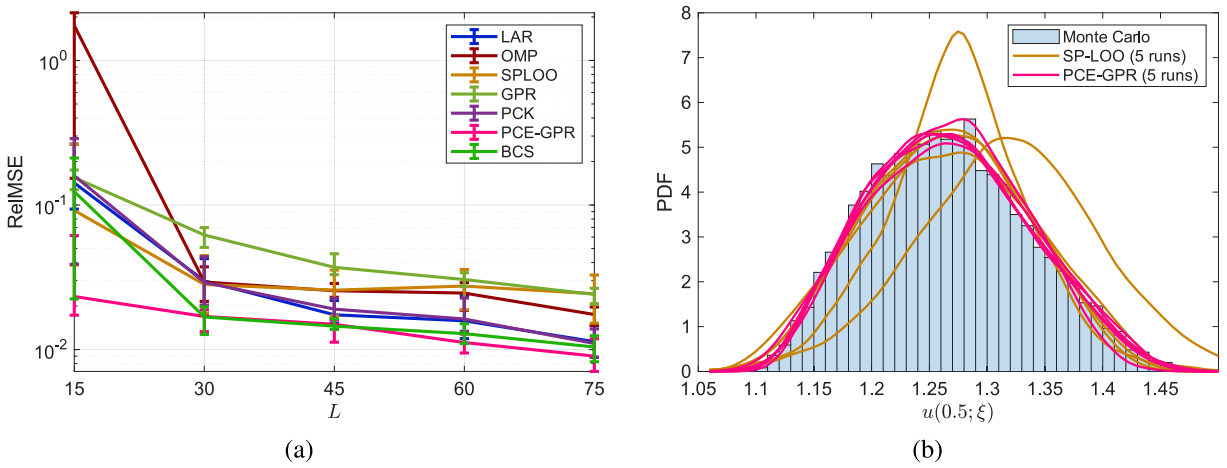


Fig. 14. Results for the solution $u(x;\xi)$ of (60) at $x = 0.5$: (a) prediction accuracy for the various methods and different sizes of the training dataset; (b) comparison between SP-LOO and PCE-GPR in the prediction of the PDF with $L = 15$ training samples. The PDF predictions are provided for five independent runs with randomized training data. The histogram represents the reference result from the Monte Carlo analysis.

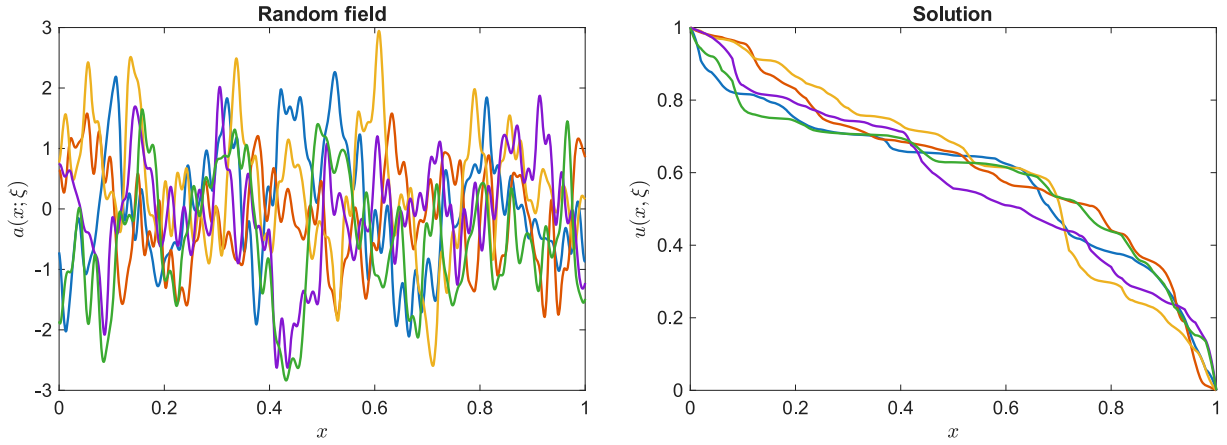


Fig. 15. Five realizations of the lognormal random field (left) and corresponding solutions of (63) (right).

and eigenfunctions are obtained by evaluating the autocorrelation matrix at 1001 spatial points. Fig. 13 illustrates five realizations of the random field (left panel) and the corresponding solutions (right panel).

To compare the performance of the various methods, we focus on the solution of at $x = 0.5$, in analogy with the results reported in [103]. For the state-of-the-art PCE methods, we consider a hyperbolic truncation with $q = 0.5$ and an adaptive selection of the expansion order between $p = 2$ and $p = 4$. Fig. 14(a) shows the results for the prediction accuracy based on 20 independent realizations of training datasets of various sizes. PCE-GPR achieves the best performance and is comparable to BCS for $L \geq 30$. However, for the smallest dataset size ($L = 15$), the median RelMSE is about an order of magnitude lower compared to the state-of-the-art method that performs best, SP-LOO in that case.

To further highlight the difference in predictive performance between these two methods, we consider the probability density function (PDF) of the solution. Fig. 14(b) compares the distribution of the reference samples from the Monte Carlo simulation (histogram) and the predictions obtained with SP-LOO and PCE-GPR for five realizations of the training dataset with $L = 15$. The results show that, for this very small sample size, the PCE-GPR prediction is consistently more accurate and exhibits a much lower dispersion between the different runs, in agreement with the performance observed in Fig. 14(a).

The second example is taken from [106]. Despite being similar to the previous test case, it is more challenging since the random field exhibits a larger variation. The stochastic boundary value problem now reads

$$-\frac{d}{dx}\left(a(x;\xi)\frac{d}{dx}u(x;\xi)\right) + Cu(x,\xi) = F, \tag{63}$$

with $x \in [0,1]$, $C = 15$, $F = 10$, and Dirichlet boundary conditions $u(0) = 1$ and $u(1) = 0$. The random field is lognormal and is constructed by letting $\ln(a(x;\xi))$ be a Gaussian field with mean $\bar{a} = 0$ and exponential covariance function

$$c(x,x') = \sigma_a^2 \exp\left(-\frac{|x-x'|}{l_x}\right), \tag{64}$$

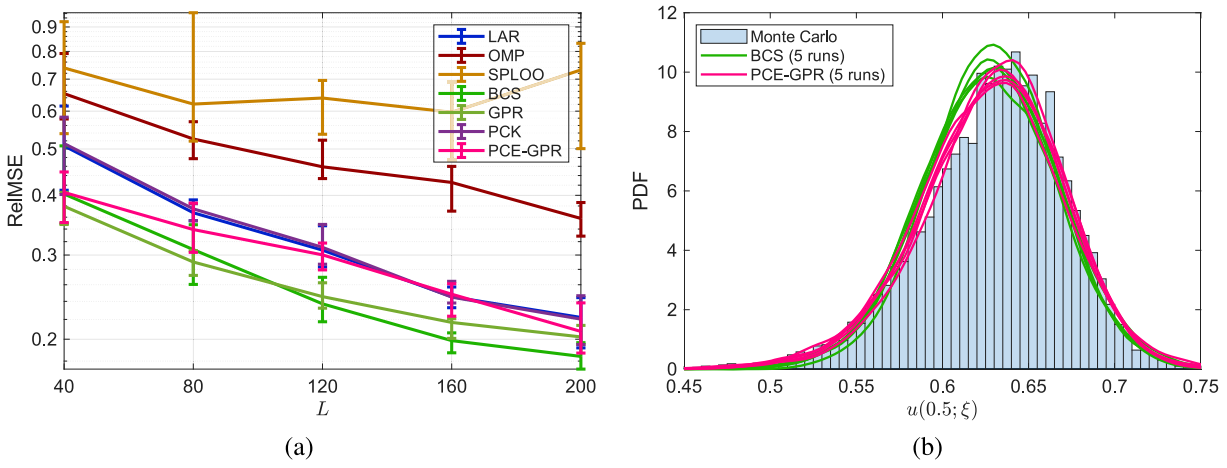


Fig. 16. Results for the solution $u(x;\boldsymbol{\xi})$ of (63) at $x = 0.505$: (a) performance of the various methods for different sizes of the training dataset; (b) comparison between BCS and PCE-GPR in the prediction of the PDF for five realizations of the training dataset with $L = 200$.

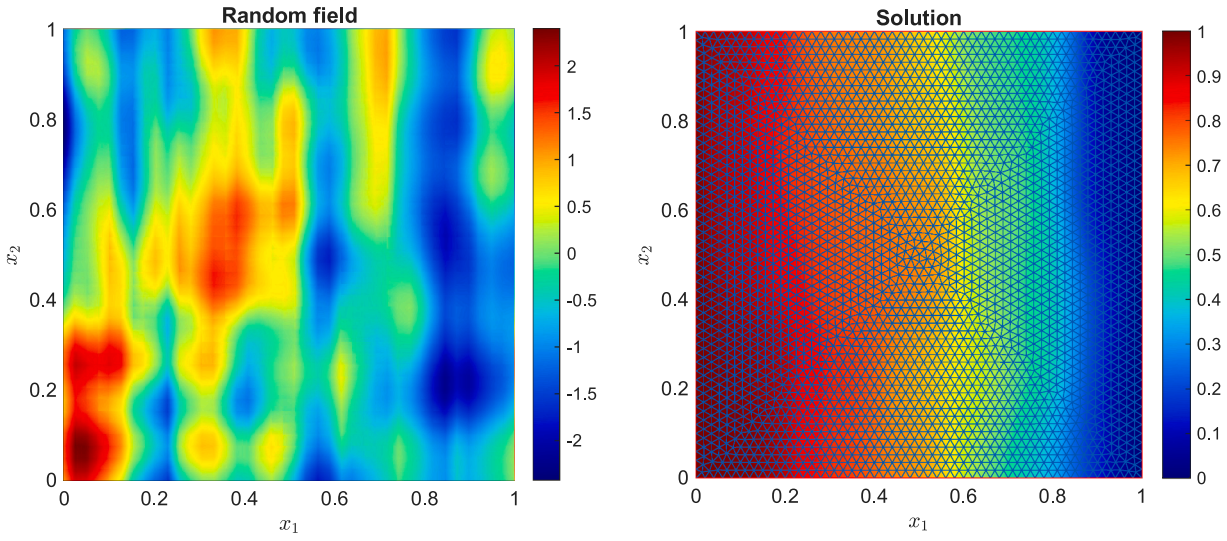


Fig. 17. Lognormal random field (left) and corresponding solution of (65) (right). The mesh is also shown in the right panel.

where $\sigma_a^2 = 1$ and $l_x = 0.03$. The random field is again approximated as (61). The KLE is calculated by decomposing the covariance matrix evaluated at 1001 spatial points and is truncated to retain all eigenvalues such that $\lambda_j/\lambda_1 \geq 0.01$, $\forall j \leq d$, where λ_1 is the largest eigenvalue. This leads to $d = 108$ terms in the expansion. Fig. 15 shows five realizations of the random field (left panel) and the corresponding solutions (right panel).

For the PCE methods, we consider an adaptive order selection between $p = 2$ and $p = 4$ and a hyperbolic truncation with $q = 0.5$. Fig. 16(a) compares the predictive performance by considering the solution at $x = 0.505$ and 20 independent realizations of training dataset of various sizes. In this case, all methods exhibit slow convergence, with BCS and GPR performing slightly better than PCE-GPR, LAR, and PCK. Nevertheless, the predictive performance of PCE-GPR is also rather good, as highlighted in Fig. 16(b). The figure compares the PDF of the solution obtained with the best-performing method (BCS) and with PCE-GPR for five independent realizations of $L = 200$ training data. Both results agree well with the reference distribution (histogram).

The next example is taken again from [106] and considers the two-dimensional stochastic boundary value problem

$$-\nabla \cdot (a(\boldsymbol{x};\boldsymbol{\xi})\nabla u(\boldsymbol{x};\boldsymbol{\xi})) = 0, \tag{65}$$

with $\boldsymbol{x} = (x_1, x_2) \in [0,1]^2$ and boundary conditions

$$\begin{aligned} &u = 1, \quad \forall x_1 = 0 \\ &u = 0, \quad \forall x_1 = 1 \\ &\boldsymbol{n}^{\mathsf{T}}(a(\boldsymbol{x};\boldsymbol{\xi})\nabla u(\boldsymbol{x};\boldsymbol{\xi})) = 0, \quad \forall x_2 = 0, x_2 = 1. \end{aligned} \tag{66}$$

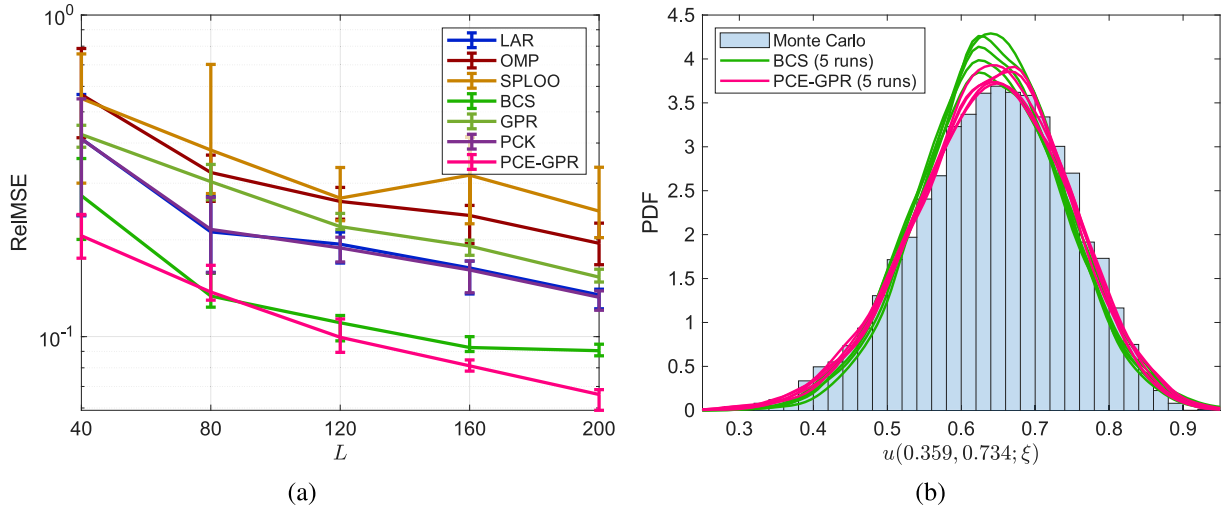


Fig. 18. Results for the solution $u(\boldsymbol{x};\boldsymbol{\xi})$ of (65) at $\boldsymbol{x} = (0.359, 0.734)$: (a) predictive performance of the various methods for different sizes of the training dataset; (b) comparison between BCS and PCE-GPR in the prediction of the PDF for five realizations of the training dataset with $L = 200$.

The random field $a(\boldsymbol{x};\boldsymbol{\xi})$ is lognormal and the underlying Gaussian field has exponential covariance function

$$c(\boldsymbol{x},\boldsymbol{x}') = \sigma_a^2 \exp\left(-\sum_{i=1}^{2} \frac{|x_i - x_i'|}{l_{x_i}}\right), \tag{67}$$

where $\sigma_a^2 = 0.75$, $l_{x_1} = 0.1$, and $l_{x_2} = 0.3$. The field is approximated using a truncated KLE. The eigenvalues and eigenfunctions of the Gaussian field are obtained through the eigenvalue decomposition of the autocorrelation matrix computed over a uniform grid of 32×32 points. We retain the terms such that $\lambda_j/\lambda_1 \geq 0.01$, $\forall j \leq d$, which leads to $d = 145$ terms in the KLE. Fig. 17 shows one realization of the random field (left panel) and the corresponding solution along with the mesh used for its computation (right panel).

We align with the analysis in [106] and focus on the solution at $\boldsymbol{x} = (0.359, 0.734)$. The same settings as for the previous example are considered for the PCE methods, i.e., adaptive order selection between $p = 2$ and $p = 4$ and a hyperbolic truncation with $q = 0.5$. The results comparing the predictive performance of the various methods are collected in Fig. 18(a) and highlight that PCE-GPR outperforms all state-of-the-art methods, while BCS performs slightly worse in terms of median error. Fig. 18(b) compares the accuracy of PCE-GPR and BCS in predicting the PDF of the solution based on $L = 200$ samples and five independent runs. Both methods agree well with the Monte Carlo distribution (histogram), yet PCE-GPR better matches the reference PDF.

6.2. Heat diffusion problem

We now consider a two-dimensional heat diffusion problem [107], described by the partial differential equation

$$-\nabla \cdot (\kappa(\boldsymbol{x})\nabla T(\boldsymbol{x})) = Q\mathbb{I}_A(\boldsymbol{x}) \tag{68}$$

over the squared domain $D = [-0.5, 0.5]^2$ m, with a Dirichlet boundary condition $T = 0$ on the top boundary and Neumann boundary conditions $\nabla T \cdot \boldsymbol{n} = 0$ on the remaining boundaries. A heat source with strength $Q = 500\ \mathrm{W/m^2}$ is located in the region $A = [0.2, 0.3]^2$ m, as illustrated in Fig. 19(a). The term $\mathbb{I}_A$ is an indicator function such that

$$\mathbb{I}_A(\boldsymbol{x}) = \begin{cases} 1 & \boldsymbol{x} \in A \\ 0 & \boldsymbol{x} \notin A \end{cases}. \tag{69}$$

The quantity of interest y is the average temperature in the region $B = [-0.3, -0.2]^2$ m. The computational domain is illustrated in Fig. 19(a).

The diffusion coefficient $\kappa(\boldsymbol{x})$ is modeled as a lognormal random field with mean $\mu_\kappa = 1\ \mathrm{W/°C \cdot m}$ and standard deviation $\sigma_\kappa = 0.3\ \mathrm{W/°C \cdot m}$. The underlying Gaussian field has an isotropic squared-exponential autocorrelation function with correlation length $l = 0.2$ m. The random field is discretized using the Expansion Optimal Linear Estimation (EOLE) method [108] with $d = 53$ normally distributed terms. Fig. 19(b) shows an example of the solution for one realization of the random diffusion coefficient. A public dataset containing both training and validation data is available [109] and is used in our simulations. The training samples are available for 20 independent realizations.

The expansion order for PCE methods is set to $p = 4$ and a hyperbolic truncation with $q = 0.5$ is used. Fig. 20(a) compares the predictive performance of the various methods. PCE-GPR is over four times more accurate than state-of-the-art methods. Among these, the standard GPR performs the best with the smallest datasets, while BCS becomes (slightly) more accurate with larger experimental designs. This confirms once again that the kernel methods tend to be more accurate when the number of training data is scarce.

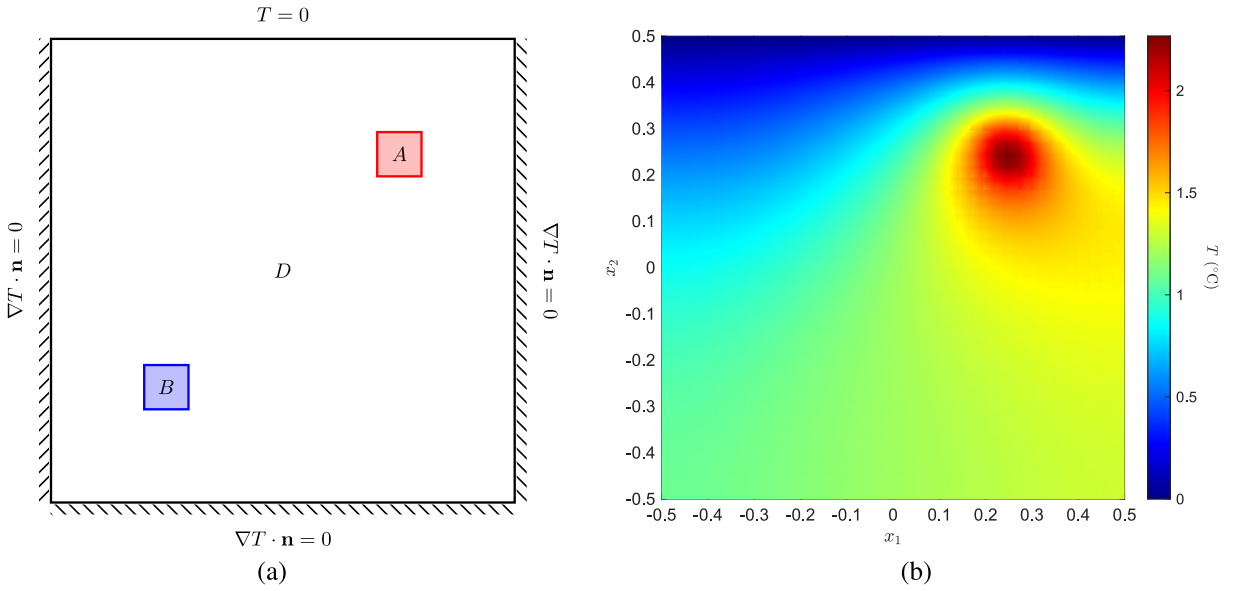


Fig. 19. Two-dimensional heat diffusion problem: (a) domain and boundary conditions; (b) solution example for one realization of the random diffusion coefficient.

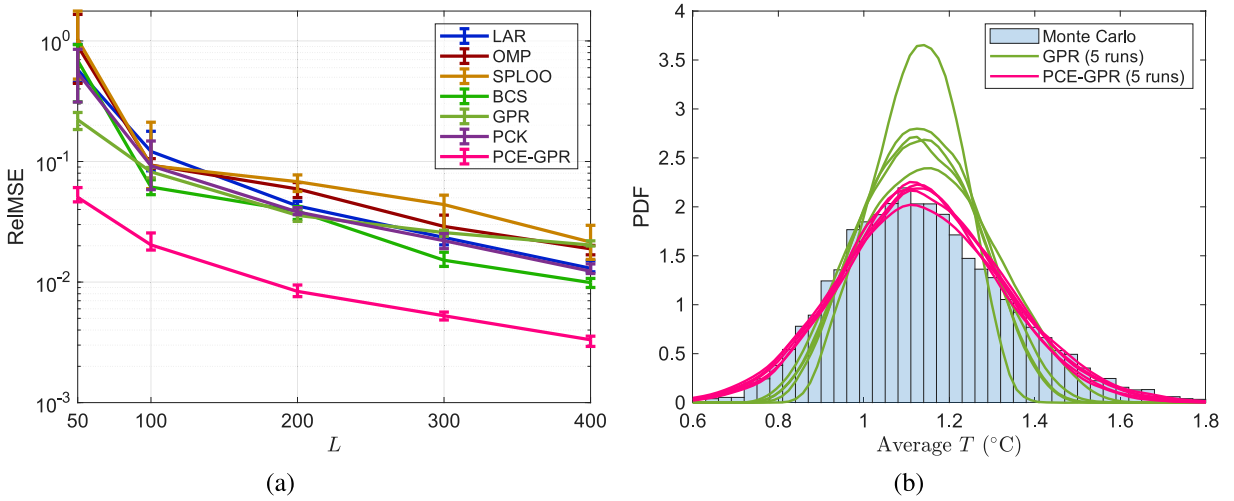


Fig. 20. Results for the heat diffusion problem: (a) predictive accuracy of the various methods for different sizes of the training dataset; (b) PDF of the average temperature obtained by GPR and PCE-GPR for five independent runs using $L = 50$ samples (lines), compared against the reference distribution from the Monte Carlo analysis (histogram).

To further highlight the PCE-GPR performance with small experimental designs, Fig. 20(b) compares PDF of the average temperature obtained for five realizations of the training dataset with $L = 50$ samples. The performance is compared against the standard GPR, which is the best-performing state-of-the-art method for this samples size, and with the reference distribution of the Monte Carlo samples (histogram). The predictions achieved by PCE-GPR are remarkably better and exhibit a much lower dispersion across the different runs.

6.3. Problems in structural mechanics

Finally, we consider three examples from structural mechanics. The first example investigates the midspan deflection V of the simply supported beam illustrated in Fig. 21(a). There are $d = 5$ uncertain parameters, namely the beam width b, height h, and length ℓ, as well as the Young's modulus E and the uniform load p, all with a lognormal distribution [107]. The code and parameters for this example are provided in the UQLab toolbox [49]. We equivalently consider a normal distribution of the logarithm of these parameters to facilitate the application of PCE-GPR, which only supports normal and uniform distributions. The original distributions

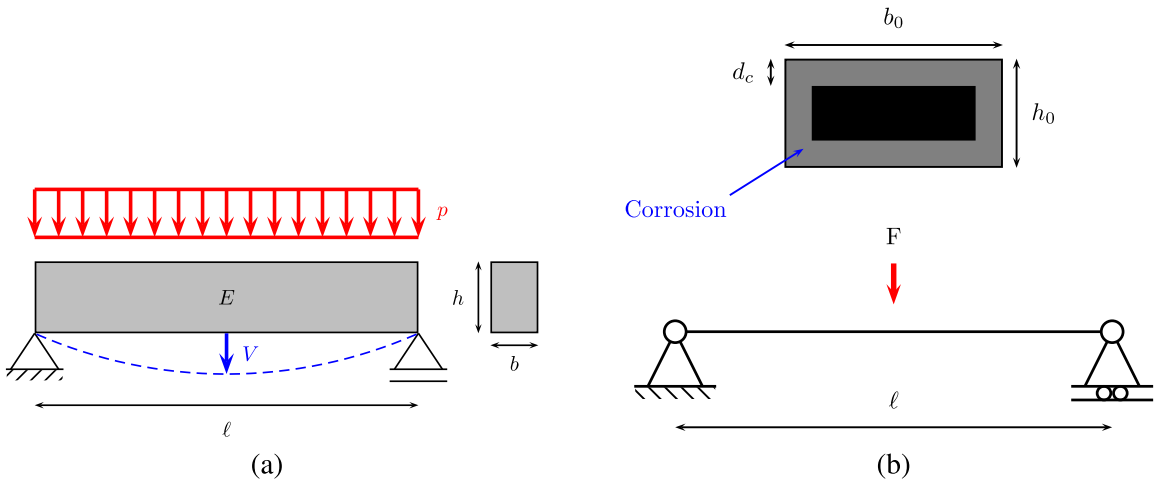


Fig. 21. Illustration of the first two structural mechanics examples: (a) simply supported beam with midspan deflection; (b) steel beam with corrosion and midspan plastic hinge.

Table 9
Input distributions for the simply supported beam example. The actual distribution and the approximate distribution considered for the analysis indicated.

Variable	Actual distribution	Mean	Standard deviation	Variable and distribution used for the analysis
b (m)	Lognormal	0.15	0.0075	$\ln(b) \sim \mathcal{N}(-1.8984, 0.05)$
h (m)	Lognormal	0.3	0.015	$\ln(h) \sim \mathcal{N}(-1.2052, 0.05)$
ℓ (m)	Lognormal	5	0.05	$\ln(\ell) \sim \mathcal{N}(1.6094, 0.01)$
E (Pa)	Lognormal	3×10^{10}	4.5×10^{9}	$\ln(E) \sim \mathcal{N}(24.1133, 0.1492)$
p (N/m)	Lognormal	1×10^{4}	2×10^{3}	$\ln(p) \sim \mathcal{N}(9.1907, 0.1980)$

and those used in the analysis are indicated in Table 9. For the state-of-the-art PCE methods, we use an adaptive selection of the expansion order in the interval $[2, 10]$ and a hyperbolic truncation with $q = 0.5$.

The second example assesses the resistance of a steel beam subject to corrosion over a 10-year span. The beam is illustrated in Fig. 21(b) and has uncertain breadth b_0, height h_0, and is subject to a random yield stress f_y, all with lognormal distribution. We equivalently consider their logarithm to be Gaussian, with $\ln(b_0) \sim \mathcal{N}(12.3834, 0.1)$, $\ln(h_0) \sim \mathcal{N}(-1.61, 0.03)$, and $\ln(f_y) \sim \mathcal{N}(-3, 0.03)$. Furthermore, the beam is subject to a stochastic load that is modeled as a random process with Gaussian autocorrelation and a correlation length of 3 months. For modeling purposes, the stochastic process is approximated using an expansion of 20 standard normal random variables. Therefore, the total number of uncertain parameters for this example is $d = 23$ (three beam parameters plus the twenty parameters describing the stochastic load).

The remaining parameters are deterministic. The corrosion depth is modeled as $d_c = \kappa \cdot t$, i.e., a linear increase over time, with a kinematic coefficient of $\kappa = 1$ mm/year. The beam length is $\ell = 5$ m and the steel mass density is $\rho = 78.5$ kN/mm. We consider as output the formation of a plastic hinge midspan after a period of 10 years, which is computed using the simulation code available within the UQLab package [49]. For the state-of-the-art PCE methods, we consider an adaptive selection of the order between $p = 3$ and $p = 10$ and a hyperbolic truncation with $q = 0.5$.

Fig. 22 compares the predictive accuracy of the various methods over 50 independent realizations of training datasets of various sizes. For the simply supported beam example (left panel), the standard GPR and PCE-GPR perform similarly when using the smallest training dataset ($L = 10$). However, the latter outperforms all the methods for larger datasets. In particular, the median error is about an order of magnitude lower than the state-of-the-art PCE methods for $L = 20$ and than the kernel methods for $L = 50$.

The results for the steel beam with corrosion are provided in the right panel of Fig. 22. PCE-GPR is found to be more accurate than all the alternative kernel methods and also outperforms the mainstream PCE methods when the number of training samples is lower than $L = 90$. In particular, with $L = 30$ and $L = 60$ training samples, it achieves an accuracy that is approximately two orders of magnitude better than the best performing PCE method.

We further compare the performance in the prediction of the output PDF with small experimental designs. Fig. 23 compares the results obtained with the standard GPR and PCE-GPR against the reference Monte Carlo distribution (histograms). The results are obtained with $L = 20$ training samples for the simply supported beam (left panel) and with $L = 30$ samples for the steel beam with corrosion (right panel). We consider five independent runs with random realizations of the training dataset. In both cases, PCE-GPR achieves a more accurate and consistent result, especially for the steel beam with corrosion.

The third and last example refers to the truss structure depicted in Fig. 24(a) [11,107]. The output of interest is the midspan deflection V. The uncertain variables are:

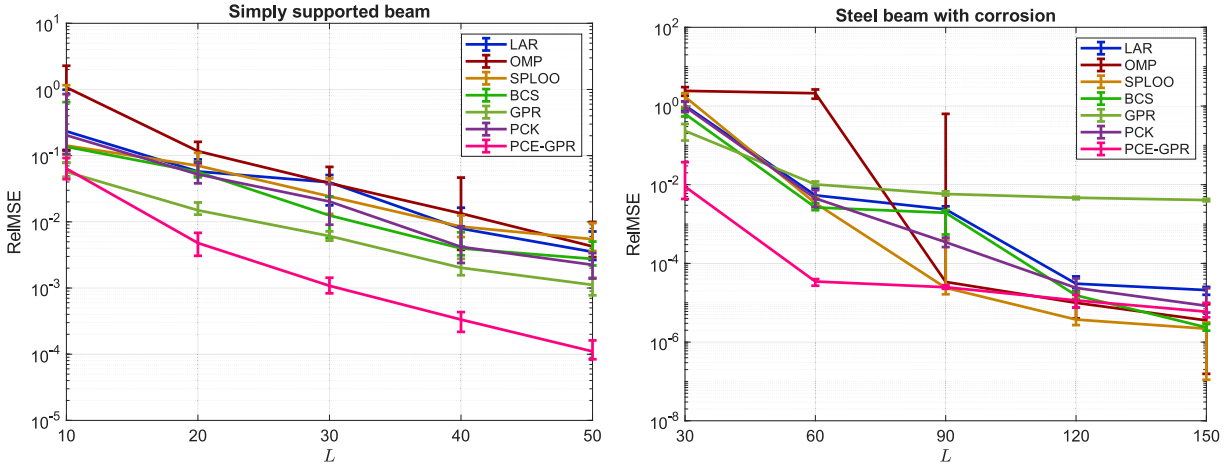


Fig. 22. Prediction accuracy for the structural mechanics examples of Fig. 21: simply supported beam (left) and steel beam with corrosion (right).

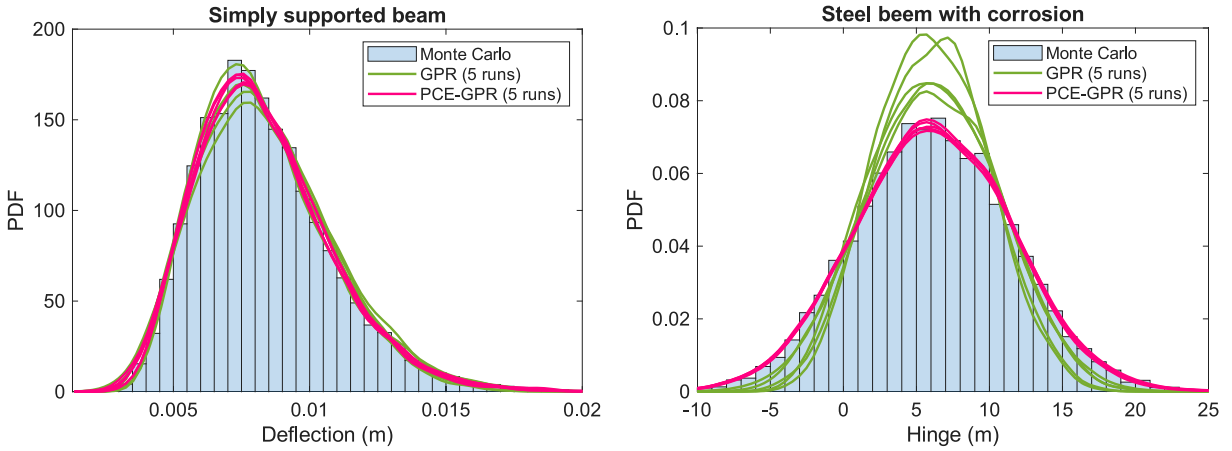


Fig. 23. Comparison between standard GPR and PCE-GPR in the prediction of the output PDF for the simply supported beam (left) and the steel beam with corrosion (right). Five independent runs with $L = 20$ and $L = 30$ training samples are considered, respectively.

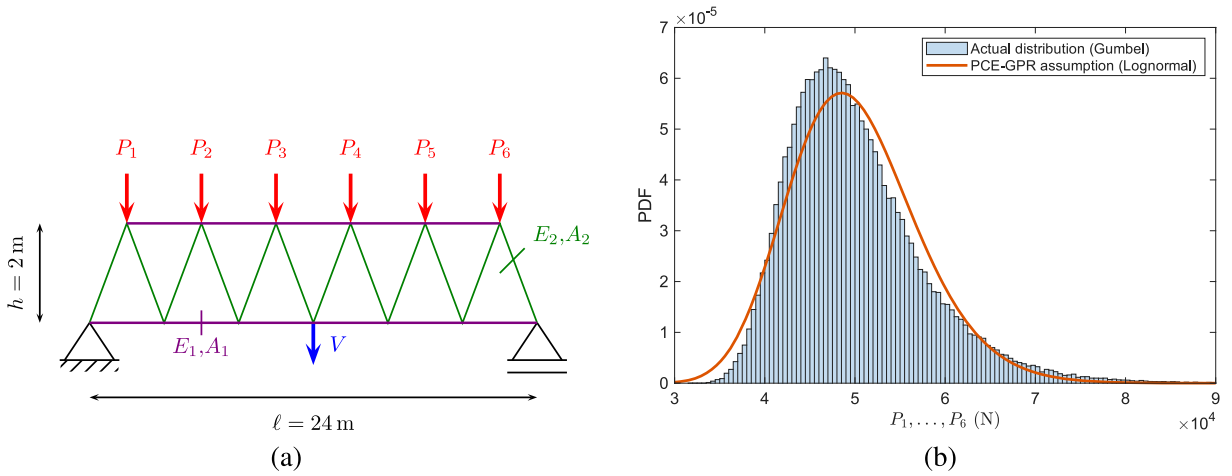


Fig. 24. Truss model example: (a) Illustration of the structure; (b) actual distribution (histogram) and assumption made for PCE-GPR (line).

Table 10
Input distributions for the truss structure. The actual distribution and the approximate distribution considered for PCE-GPR are indicated.

Variable	Actual distribution	Mean	Standard deviation	Variable and distribution used for PCE-GPR
E_1, E_2 (Pa)	Lognormal	2.1×10^{11}	2.1×10^{10}	$\ln(E_1), \ln(E_2) \sim \mathcal{N}(26.06545, 0.09955)$
A_1 (mm^2)	Lognormal	2×10^{-3}	2×10^{-4}	$\ln(A_1) \sim \mathcal{N}(-6.2198, 0.0996)$
A_2 (mm^2)	Lognormal	1×10^{-3}	1×10^{-4}	$\ln(A_2) \sim \mathcal{N}(-6.9129, 0.0999)$
$P_1, \dots, P_6$ (N)	Gumbel	5×10^{4}	7.5×10^{3}	$\ln(P_1), \dots, \ln(P_6) \sim \mathcal{N}(10.81, 0.1425)$

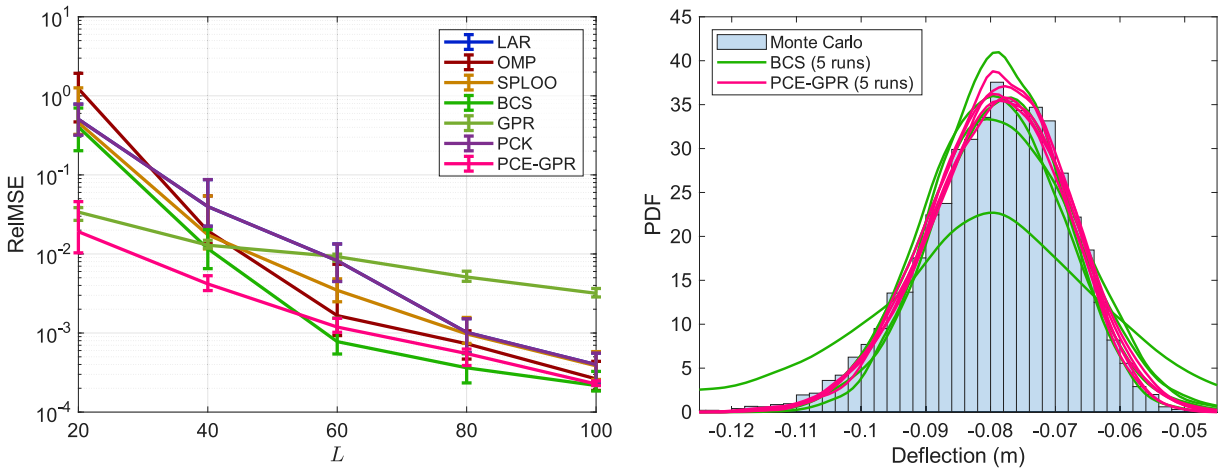


Fig. 25. Results for the truss structure: (a) predictive accuracy of the various methods for different sizes of the training dataset; (b) PDF of the midspan deflection obtained with BCS and PCE-GPR for five runs using $L = 20$ samples (lines), compared against the reference distribution from the Monte Carlo analysis (histogram).

- The cross-sectional area and the Young's modulus of the horizontal bars, denoted with A_1 and E_1, respectively;
- The cross-sectional area and the Young's modulus of the vertical bars, denoted with A_2 and E_2, respectively;
- The six vertical loads, denoted with $P_1, \dots, P_6$.

The total number of random variables is therefore $d = 10$. The cross-sectional areas and the Young's moduli have a lognormal distribution, while the loads follow a Gumbel distribution. Both distributions are supported in UQLab. For the PCE-GPR toolbox instead, we consider a normal distribution for the natural logarithm of the uncertain variables. Although this transformation is exact for the lognormal distribution, it is an approximation for the Gumbel distribution, as illustrated in Fig. 24(b). Therefore, the use of Hermite kernels in PCE-GPR is nonoptimal for these variables. However, it should be noted that this assumption influences only the selection of kernel functions, while input data remain unchanged. These considerations do not affect standard kernel methods, since no specific assumption is made on the input distribution. The actual distributions, considered for the state-of-the-art methods, and those assumed for PCE-GPR are summarized in Table 10. A public dataset with training and validation data for this example is available [110].

Fig. 25(a) shows the predictive accuracy achieved by the various methods with different samples sizes over 20 independent runs. For the PCE methods, a hyperbolic truncation with order $p = 4$ and $q = 0.5$ is considered. Remarkably, PCE-GPR provides again the most accurate prediction for the smallest datasets with $L = 20$ and $L = 40$ training samples, while it is (slightly) outperformed by BCS for $L = 60$ and $L = 80$. The standard GPR ranks second for $L = 20$ but exhibits slow convergence, becoming the least accurate method for $L > 60$. With $L = 20$ samples, PCE-GPR achieves a median prediction error that is over one order of magnitude lower compared to state-of-the-art PCE techniques. This can be further appreciated in Fig. 25(b), which compares BCS and PCE-GPR in predicting the PDF of the deflection for five realizations of the training dataset. Once again, the PCE-GPR result is more accurate and more consistent across the different runs. These results show that PCE-GPR maintains its accuracy even when input parameters are drawn from distributions that are not directly supported by the current implementation.

The results in this section confirm that PCE-GPR scales effectively to high-dimensional problems and maintains high predictive accuracy even in small-sample regimes. This dual performance arises from the combination of two key factors: 1) the use of a kernel method, which is inherently nonparametric and whose model complexity grows with the amount of training data rather than with the input dimensionality; and 2) the construction of the kernel from Hermite and Legendre polynomials, which — according to the Wiener-Askey scheme [8] — form the optimal L^2 basis for functions with Gaussian or uniform inputs.

7. Conclusions

This paper reviewed state-of-the-art PCE and kernel-based machine learning methods for UQ and introduced a novel open-source toolbox, named "PCE-GPR", for accurate high-dimensional UQ. The toolbox is based on UQLab and implements a recently proposed hybrid method that leverages a GPR formulation with special kernels constructed from Hermite or Legendre polynomials. This formulation enables higher prediction accuracy and the analytical calculation of PCE coefficients, from which statistical information is readily obtained. A systematic analysis was conducted to identify the recommended default settings, which resulted in an anisotropic kernel formulation with a ML estimation of the hyperparameters using the HCMAES optimizer.

Moreover, the paper provided an extensive benchmarking of PCE-GPR against the state-of-the-art techniques in terms of predictive accuracy and training cost. The analysis, based on both popular analytical benchmarks and numerical simulations of physical and electronic systems, extends previous literature in comparing the performance of kernel machine learning methods against PCE techniques in high-dimensional UQ. All datasets and test cases have been made publicly available.

The results show that SP-LOO and BCS usually perform the best among the PCE techniques. However, kernel methods tend to outperform PCE, especially for high-dimensional problems and with small training datasets. Furthermore, the proposed PCE-GPR toolbox is shown to substantially outperform all state-of-the-art methods in most test cases, with the exception of the Ishigami and Friedman functions, for which lower or comparable accuracy was observed. This highlighted a potential limitation of PCE-GPR in accurately modeling strongly nonlinear functions. The comparison with PCK, which uses a GPR formulation with a polynomial trend and provided an accurate model for these benchmarks, suggests that the limited accuracy may be related to using a constant or zero trend when surrogating highly nonlinear functions with kernel methods. Also, an ML estimation of PCE-GPR hyperparameters was found to yield more accurate models in high-dimensional benchmarks. PCE-GPR was also compared with standard PCE techniques in the calculation of Sobol' indices, for which it turned out to provide accurate and more consistent results across different training datasets and with a small number of samples already, and against PCE-LSSVM, highlighting a significant improvement in predictive performance.

In terms of computational time, PCE methods are shown to be generally faster than kernel methods, although BCS and SPGL1 are significantly slower than LAR, OMP, and SP. In this regard, PCE-GPR is found to be faster than SVR and optimal PCK, and slightly slower than LSSVM and standard GPR or sequential PCK, with a training time that is comparable to BCS and SPGL1.

Further comparisons were performed for three stochastic boundary value problems with high-dimensional random fields, a heat diffusion problem, and three test cases in structural mechanics. This analysis confirmed that BCS stands out among PCE methods but tends to be outperformed by PCE-GPR, especially for small experimental designs. The results also demonstrated that PCE-GPR remains accurate even when the input parameters follow distributions beyond the supported normal and uniform cases.

All in all, PCE-GPR showed the potential to become a cutting-edge technique for data-driven and high-dimensional UQ. Currently, the implementation is limited to Gaussian and uniform distributions. Future work will focus on extending the approach to other input distributions, exploring alternative kernel formulations, and incorporating non-constant trend definitions.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the author used ChatGPT and Writefull (the Overleaf's AI-based language feedback) in a limited capacity to improve the clarity and style of the manuscript. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the content of the published article.

CRediT authorship contribution statement

Paolo Manfredi: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Data availability

The data used for the benchmark study of Section 5 are available in [83].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M.B. Yelten, T. Zhu, S. Koziel, P.D. Franzon, M.B. Steer, Demystifying surrogate modeling for circuits and systems, IEEE Circuits Syst. Mag. 12 (1) (2012) 45–63.
[2] R. Trinchero, M. Larbi, H.M. Torun, F.G. Canavero, M. Swaminathan, Machine learning and uncertainty quantification for surrogate models of integrated devices with a large number of parameters, IEEE Access 7 (2018) 4056–4066.
[3] T. Nguyen, B. Shi, H. Ma, E.-P. Li, X. Chen, A.C. Cangellaris, J. Schutt-Aine, Comparative study of surrogate modeling methods for signal integrity and microwave circuit applications, IEEE Trans. Components, Packaging Manufact. Technol. 11 (9) (2021) 1369–1379.
[4] J. Kudela, R. Matousek, Recent advances and applications of surrogate models for finite element method computations: A review, Soft Computing 26 (24) (2022) 13709–13733.

[5] Y. Feng, D. Wu, M.G. Stewart, W. Gao, Past, current and future trends and challenges in non-deterministic fracture mechanics: A review, Computer Methods in Applied Mechanics and Engineering 412 (2023) 116102.
[6] H. Zhao, C. Fu, Y. Zhang, W. Zhu, K. Lu, E.M. Francis, Dimensional decomposition-aided metamodels for uncertainty quantification and optimization in engineering: A review, Computer Methods Appl. Mech. Eng. 428 (2024) 117098.
[7] Z. Azarhoosh, M.I. Ghazaan, A review of recent advances in surrogate models for uncertainty quantification of high-dimensional engineering applications, Computer Methods Appl. Mech. Eng. 433 (2025) 117508.
[8] D. Xiu, G.E. Karniadakis, The Wiener–Askey polynomial chaos for stochastic differential equations, SIAM Journal on Scientific Computing 24 (2) (2002) 619–644. https://doi.org/10.1137/S1064827501387826
[9] B. Sudret, Global sensitivity analysis using polynomial chaos expansions, Reliability Engineering & System Safety 93 (7) (2008) 964–979. https://doi.org/10.1016/j.ress.2007.04.002
[10] N. Lüthen, S. Marelli, B. Sudret, Sparse polynomial chaos expansions: Literature survey and benchmark, SIAM/ASA J. Uncertainty Quantif. 9 (2) (2021) 593–649.
[11] G. Blatman, B. Sudret, Adaptive sparse polynomial chaos expansion based on least angle regression, J. Comput. Phys. 230 (6) (2011) 2345–2367. https://doi.org/10.1016/j.jcp.2010.12.021
[12] Y.C. Pati, R. Rezaiifar, P.S. Krishnaprasad, Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition, in: Proceedings of 27th Asilomar Conference on Signals, Systems and Computers, IEEE, 1993, pp. 40–44.
[13] P. Diaz, A. Doostan, J. Hampton, Sparse polynomial chaos expansions via compressed sensing and D-optimal design, Computer Methods in Applied Mechanics and Engineering 336 (2018) 640–666.
[14] K. Sargsyan, C. Safta, H.N. Najm, B.J. Debusschere, D. Ricciuto, P. Thornton, Dimensionality reduction for complex models via Bayesian compressive sensing, Int. J. Uncertainty Quantif. 4 (1) (2014).
[15] A. Modak, U.M. Krishnan, A. Gupta, T. Gangwar, R. Chowdhury, Sparse polynomial chaos expansion and adaptive mesh refinement for enhanced fracture prediction using phase-field method, Theoretical and Applied Fracture Mechanics 133 (2024) 104639. https://www.sciencedirect.com/science/article/pii/S0167844224003896. https://doi.org/10.1016/j.tafmec.2024.104639
[16] M.C. Dwelle, J. Kim, K. Sargsyan, V.Y. Ivanov, Streamflow, stomata, and soil pits: Sources of inference for complex models with fast, robust uncertainty quantification, Adv. Water Resour. 125 (2019) 13–31. https://www.sciencedirect.com/science/article/pii/S0309170818306997. https://doi.org/10.1016/j.advwatres.2019.01.002
[17] Z. Guo, W. Chu, H. Zhang, C. Liang, D. Meng, Aerodynamic evaluation of cascade flow with actual geometric uncertainties using an adaptive sparse arbitrary polynomial chaos expansion, Phys. Fluids 35 (3) (2023) 036122. https://doi.org/10.1063/5.0144937
[18] T. Zygiridis, A. Kyrgiazoglou, S. Amanatiadis, N. Kantartzis, T. Theodoulidis, Uncertainty quantification and sensitivity analysis in subsurface defect detection with sparse models, J. Nondestructive Eval. 43 (4) (2024) 104.
[19] Y. Bao, J. Qiu, P. Gurrala, J. Song, Efficient MAPoD via least angle regression based polynomial chaos expansion metamodel for eddy current NDT, Applied Computational Electromagnetics Society Journal (ACES) (2024) 461–469.
[20] R. Hu, V. Monebhurrun, R. Himeno, H. Yokota, F. Costen, An adaptive least angle regression method for uncertainty quantification in FDTD computation, IEEE Trans. Antennas Prop. 66 (12) (2018) 7188–7197.
[21] H. Jiang, F. Ferranti, G. Antonini, Enhanced sparse polynomial chaos expansion for electromagnetic compatibility uncertainty quantification problems, IEEE Transactions on Electromagnetic Compatibility (2025).
[22] E.M.D. Moya, R.A. Rodriguez, A. Pietrus, S. Bernard, A study of fractional optimal control of overweight and obesity in a community and its impact on the diagnosis of diabetes, Mathematical Modelling and Numerical Simulation with Applications 4 (4) (2024) 514–543.
[23] X. Guo, Y. Jin, Novel algorithm for flexible multibody systems with hybrid uncertainties, Appl. Math. Modell. 113 (2023) 573–595. https://www.sciencedirect.com/science/article/pii/S0307904X22004528. https://doi.org/10.1016/j.apm.2022.09.029
[24] B. Rong, X. Rui, L. Tao, G. Wang, J. Zhang, Z. Xiao, Efficient dynamics modeling and analysis for probabilistic uncertain beam systems with geometric nonlinearity and thermal coupling effect, Nonlinear Dynam. 111 (1) (2023) 39–66.
[25] R. Mura, T. Ghisu, S. Shahpar, Least squares approximation-based polynomial chaos expansion for uncertainty quantification and robust optimization in aeronautics, in: AIAA Aviation 2020 Forum, 2020, p. 3163.
[26] H. Handuo, S. Yanping, Y. Jianyang, L. Yao, G. Wenxiu, C. Fu, Investigation on uncertainty quantification of transonic airfoil using compressive sensing greedy reconstruction algorithms, Aerospace Science and Technology 147 (2024) 109000. https://www.sciencedirect.com/science/article/pii/S1270963824001330. https://doi.org/10.1016/j.ast.2024.109000
[27] M.S. Karimi, R. Mohammadi, M. Raisee, Bi-fidelity adaptive sparse reconstruction of polynomial chaos using Bayesian compressive sensing, Engineering with Computers 41 (2025) 2461–2481.
[28] B.A. Jones, T.N. Wolf, Incorporating directional uncertainties into polynomial chaos expansions for astronautics problems, The J. Astronaut. Sci. 70 (4) (2023) 19.
[29] E. Rouzies, C. Lauvernet, B. Sudret, A. Vidard, How is a global sensitivity analysis of a catchment-scale, distributed pesticide transfer model performed? Application to the PESHMELBA model, Geosci. Model Development 16 (11) (2023) 3137–3163. https://gmd.copernicus.org/articles/16/3137/2023/. https://doi.org/10.5194/gmd-16-3137-2023
[30] A.D. Papadopoulos, T.T. Zygiridis, E.N. Glytsis, N.V. Kantartzis, C.S. Antonopoulos, Uncertainty study of periodic-grating wideband filters with sparse polynomial-chaos expansions, IEEE Photonics Technology Letters 31 (18) (2019) 1499–1502.
[31] A.D. Papadopoulos, T. Zygiridis, Y. Kopsinis, E.N. Glytsis, An anisotropic sparse adaptive polynomial chaos method for the uncertainty quantification of resonant gratings-performance, Journal of Lightwave Technology 40 (12) (2022) 3640–3646.
[32] A. Samadhiya, K. Namrata, Probabilistic screening and behavior of solar cells under Gaussian parametric uncertainty using polynomial chaos representation model, Complex Intell. Syst. 8 (2) (2022) 989–1004.
[33] B. Dong, P. Li, H. Yu, H. Ji, J. Li, J. Wu, C. Wang, A non-intrusive probabilistic multi-energy flow calculation method and its application in operation risk analysis of integrated energy systems, Sustain. Energy Technol. Assessments 54 (2022) 102834.
[34] G. Hu, Q. Tao, R. Ying, J. Long, Multi-objective robust optimization design framework for low-pollution emission burners, Chemical Eng. Res. Design 210 (2024) 180–189.
[35] G. Hu, L. Xu, L. Zhao, W. Du, F. Qian, Uncertainty analysis of NOx and CO emissions in industrial ethylene cracking furnace using high-precision sparse polynomial chaos expansion, Combustion Sci. Technol. 196 (2) (2024) 195–222.
[36] G. Ma, C. Lu, M. Fang, B. Zhao, S. Fan, Q. Zhou, J. Lu, X. Yao, Uncertainty quantification in the motion of a water-exit vehicle and analysis of the robustness enhancement characteristics of pressure-equalizing exhaust, Physics of Fluids 36 (12) (2024).
[37] W. He, G. Li, Z. Nie, An adaptive sparse polynomial dimensional decomposition based on Bayesian compressive sensing and cross-entropy, Structural and Multidisciplinary Optimization 65 (1) (2022) 26.
[38] Z. Lian, J. Zhang, F. Zhao, W. Yu, Optimization of methane simplified chemical kinetic mechanism based on uncertainty quantitation analysis by sparse polynomial chaos expansions, Fuel 339 (2023) 127393.
[39] S. Kushari, T. Mukhopadhyay, A. Chakraborty, S.R. Maity, S. Dey, Probability-based unified sensitivity analysis for multi-objective performances of composite laminates: a surrogate-assisted approach, Composite Structures 294 (2022) 115559.
[40] P. Li, Y. Wang, Development of an efficient response surface method for highly nonlinear systems from sparse sampling data using Bayesian compressive sensing, ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering 7 (4) (2021) 04021050.
[41] P. Li, Active learning of small failure probabilities of highly nonstationary geotechnical systems by adaptive Bayesian compressive sensing and subset simulation, ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering 11 (1) (2025) 04024083.
[42] S. Li, Z. Wei, S. Zhang, Z. Cen, E. Tsoutsanis, Polynomial chaos expansion-based uncertainty model for fast assessment of gas turbine aero-engines thrust regulation: A sparse regression approach, J. Eng. Gas Turbines Power 147 (1) (2025) 011031.

[43] B.-S. Xu, X.-M. Yang, A.-C. Zou, C.-P. Zang, Efficient metamodeling and uncertainty propagation for rotor systems by sparse polynomial chaos expansion, Prob. Eng. Mech. 79 (2025) 103723.
[44] X. Huan, C. Safta, K. Sargsyan, Z.P. Vane, G. Lacaze, J.C. Oefelein, H.N. Najm, Compressive sensing with cross-validation and stop-sampling for sparse polynomial chaos expansions, SIAM/ASA Journal on Uncertainty Quantification 6 (2) (2018) 907–936.
[45] X. Liu, S. Zhong, X. Zheng, J. Fu, Research on uncertainties in fuel centrifugal pump based on prediction and reconstruction of internal flow field, Physics of Fluids 36 (6) (2024).
[46] T. Wang, B. Li, Q. Yu, Y. Wu, L. Xu, Y. Chi, B. Li, Uncertainty quantification of electromagnetic exposure of human body with medical aortic valve stent implants under an EV-WPT device, Progress in Electromagnetics Research C 136 (2023).
[47] T.K. West, C.O. Johnston, Efficient parametric uncertainty analysis of an Earth entry vehicle concept using least angle regression, in: AIAA Scitech 2023 Forum, 2023, p. 2430.
[48] B.A. Jones, N. Parrish, A. Doostan, Postmaneuver collision probability estimation using sparse polynomial chaos expansions, Journal of Guidance, Control, and Dynamics 38 (8) (2015) 1425–1437.
[49] S. Marelli, B. Sudret, UQLab: A framework for uncertainty quantification in Matlab, in: Vulnerability, uncertainty, and risk: quantification, mitigation, and management, 2014, pp. 2554–2563. https://doi.org/10.1061/9780784413609.257
[50] S. Marelli, B. Sudret, UQLab: The framework for uncertainty quantification, release 2.0.0, 2022. https://www.uqlab.com/.
[51] Y. Shi, P. Wei, K. Feng, D.-C. Feng, M. Beer, A survey on machine learning approaches for uncertainty quantification of engineering systems, Machine Learn. Comput. Sci. Eng. 1 (1) (2025) 1–39.
[52] H. Drucker, C.J. Burges, L. Kaufman, A. Smola, V. Vapnik, Support vector regression machines, Adv. Neural Inf. Process. Syst. 9 (1996) 155–161.
[53] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, MIT Press, 2006.
[54] Y.P. Ju, A dimension-reduction metamodeling approach to simulation-based uncertainty quantification problems with high dimensionalities, Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 236 (1) (2022) 43–62.
[55] Y. Tian, Y. Feng, W. Gao, Virtual modelling framework-based inverse study for the mechanical metamaterials with material nonlinearity, Modelling 6 (1) (2025) 24.
[56] L. Bo, J. Zhang, H. Wang, Advanced virtual modelling aided stochastic nonlinear dynamic stability analysis of the GPLR-FGP plate in thermal environments, Thin-Walled Structures 199 (2024) 111833.
[57] B. Bhattacharyya, Uncertainty quantification of dynamical systems by a POD–Kriging surrogate model, J. Comput. Sci. 60 (2022) 101602.
[58] L. Cicci, S. Fresca, M. Guo, A. Manzoni, P. Zunino, Uncertainty quantification for nonlinear solid mechanics using reduced order models with Gaussian process regression, Comput. Math. Appl. 149 (2023) 1–23.
[59] R.S. Chahar, J. Lee, T. Mukhopadhyay, On quantifying uncertainty in lightning strike damage of composite laminates: A hybrid stochastic framework of coupled transient thermal-electrical simulations, Aerosp. Sci. Technol. 142 (2023) 108597.
[60] Y. Yu, B. Dong, W. Gao, A. Sofi, Physics-based stochastic aging corrosion analysis assisted by machine learning, Prob. Eng. Mech. 69 (2022) 103270.
[61] H. Ma, E.-P. Li, A.C. Cangellaris, X. Chen, Support vector regression-based active subspace (SVR-AS) modeling of high-speed links for fast and accurate sensitivity analysis, IEEE Access 8 (2020) 74339–74348.
[62] X. Shang, L. Xu, Q. Yu, B. Li, G. Lv, Y. Chi, T. Wang, Uncertainty quantification and optimal design of EV-WPT system efficiency based on adaptive Gaussian process regression, Applied Computational Electromagnetics Society Journal (2023) 929–940.
[63] S. Kasdorf, J.J. Harmon, B.M. Notaroš, Kriging methodology for uncertainty quantification in computational electromagnetics, IEEE Open Journal of Antennas and Propagation 5 (2) (2024) 474–486.
[64] B. Chen, L. Chen, R. Mo, Z. Wang, L. Zheng, C. Zhang, Y. Chen, Strength prediction and uncertainty quantification of welded CHS tubular joints via Gaussian process regression, Eng. Struct. 332 (2025) 120030.
[65] L.Q. Minh, P.L.T. Duong, M. Lee, Global sensitivity analysis and uncertainty quantification of crude distillation unit using surrogate model based on Gaussian process regression, Ind. Eng. Chem. Res. 57 (14) (2018) 5035–5044.
[66] F.S. Costabal, K. Matsuno, J. Yao, P. Perdikaris, E. Kuhl, Machine learning in drug development: Characterizing the effect of 30 drugs on the QT interval using Gaussian process regression, sensitivity analysis, and uncertainty quantification, Computer Methods Appl. Mech. Eng. 348 (2019) 313–333.
[67] R. Schobi, B. Sudret, J. Wiart, Polynomial-chaos-based Kriging, International Journal for Uncertainty Quantification 5 (2) (2015).
[68] W. Zhang, M.H. El Naggar, P. Ni, M. Zhao, X. Du, Bayesian updating of geotechnical parameters with polynomial chaos Kriging model and Gibbs sampling, Computers and Geotechnics 180 (2025) 107087.
[69] E. García-Macías, F. Ubertini, Real-time Bayesian damage identification enabled by sparse PCE-Kriging meta-modelling for continuous SHM of large-scale civil engineering structures, J. Building Eng. 59 (2022) 105004.
[70] V.N. Tran, J. Kim, Robust and efficient uncertainty quantification for extreme events that deviate significantly from the training dataset using polynomial chaos-kriging, Journal of Hydrology 609 (2022) 127716.
[71] H. Zhao, Z.-H. Gao, L. Xia, Efficient aerodynamic analysis and optimization under uncertainty using multi-fidelity polynomial chaos-Kriging surrogate model, Comput. Fluids 246 (2022) 105643.
[72] J. Weinmeister, X. Gao, S. Roy, Analysis of a polynomial chaos-kriging metamodel for uncertainty quantification in aerodynamics, AIAA J. 57 (6) (2019) 2280–2296.
[73] A. Amini, A. Abdollahi, M.A. Hariri-Ardebili, U. Lall, Copula-based reliability and sensitivity analysis of aging dams: Adaptive Kriging and polynomial chaos Kriging methods, Appl. Soft Comput. 109 (2021) 107524.
[74] P. Manfredi, A hybrid polynomial chaos expansion–Gaussian process regression method for Bayesian uncertainty quantification and sensitivity analysis, Computer Methods in Applied Mechanics and Engineering 436 (2025) 117693.
[75] P. Manfredi, PCE-GPR: A toolbox for high-dimensional uncertainty quantification, 2025. https://doi.org/10.5281/zenodo.15348860.
[76] C. Cui, Z. Zhang, Stochastic collocation with non-Gaussian correlated process variations: Theory, algorithms, and applications, IEEE Trans. Components, Packaging Manufact. Technol. 9 (7) (2018) 1362–1375.
[77] D. Xiu, J.S. Hesthaven, High-order collocation methods for differential equations with random inputs, SIAM Journal on Scientific Computing 27 (3) (2005) 1118–1139.
[78] R.G. Ghanem, P.D. Spanos, Stochastic Finite Elements: A Spectral Approach, Courier Corporation, 2003.
[79] S.D. Babacan, R. Molina, A.K. Katsaggelos, Bayesian compressive sensing using Laplace priors, IEEE Trans. Image Process. 19 (1) (2009) 53–63.
[80] E. van den Berg, M.P. Friedlander, Probing the Pareto frontier for basis pursuit solutions, Siam Journal on Scientific Computing 31 (2) (2009) 890–912.
[81] V. Vapnik, The Nature of Statistical Learning Theory, Springer science & business media, 2013.
[82] M. Moustapha, C. Lataniotis, S. Marelli, B. Sudret, UQLab user manual – Support vector machines for regression, Technical Report, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2022. Report UQLab-V2.0-111.
[83] P. Manfredi, Benchmark datasets for the PCE-GPR toolbox, version v1.1.0, 2025. https://doi.org/10.5281/zenodo.16410102.
[84] T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, J. Suykens, T. Van Gestel, Least Squares Support Vector Machines World Scientific, 2002.
[85] P. Manfredi, R. Trinchero, Nonparametric formulation of polynomial chaos expansion based on least-square support-vector machines, Eng. Appl. Artif. Intell. 133 (2024) 108182.
[86] C. Lataniotis, D. Wicaksono, S. Marelli, B. Sudret, UQLab user manual – Kriging (Gaussian process modeling), Technical Report, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2022. Report UQLab-V2.0-105.
[87] A. Forrester, A. Sobester, A. Keane, Engineering Design via Surrogate Modelling: A Practical Guide, John Wiley & Sons, 2008.
[88] T. Ishigami, T. Homma, An importance quantification technique in uncertainty analysis for computer models, in: [1990] Proceedings. First international symposium on uncertainty modeling and analysis, IEEE, 1990, pp. 398–403.
[89] J.H. Friedman, E. Grosse, W. Stuetzle, Multidimensional additive spline approximation, SIAM J. Scient. Stat. Comput. 4 (2) (1983) 291–301.

[90] B. Echard, N. Gayton, M. Lemaire, N. Relun, A combined importance sampling and kriging reliability method for small failure probabilities with time-demanding numerical models, Reliab. Eng. Syst. Saf. 111 (2013) 232–240.
[91] W.V. Harper, S.K. Gupta, Sensitivity/uncertainty analysis of a borehole scenario comparing Latin hypercube sampling and deterministic sensitivity approaches, Office of Nuclear Waste Isolation, Battelle Memorial Institute Columbus, Ohio, 1983.
[92] M.D. Morris, Factorial sampling plans for preliminary computational experiments, Technometrics 33 (2) (1991) 161–174. https://doi.org/10.1080/00401706.1991.10484804
[93] T. Buss, 2 GHz low noise amplifier with the BFG425W, Technical Report, Philips Semiconductors B.V., Nijmegen, The Netherlands, 1996. Appl. Note RNR-T45-96-B-773.
[94] M. Ahadi, S. Roy, Sparse linear regression (SPLINER) approach for efficient multidimensional uncertainty quantification of high-speed circuits, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 35 (10) (2016) 1640–1652.
[95] T.K. Tang, M.S. Nakhla, Analysis of high-speed VLSI interconnects using the asymptotic waveform evaluation technique, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 11 (3) (1992) 341–352.
[96] G. Blatman, Chaos polynomial creux et adaptatif pour la propagation d'incertitudes et l'analyse de sensibilité, Ph.D. thesis, Université Blaise Pascal-Clermont-Ferrand II, 2009.
[97] S. Marelli, N. Lüthen, B. Sudret, UQLab user manual – Polynomial chaos expansions, Technical Report, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2022. Report UQLab-V2.0-104.
[98] M.P. Berg, E. van den and Friedlander, A Solver for Sparse Least Squares, Version 2.1, 2025, (https://friedlander.io/spgl1/). Accessed: 2025-02-13.
[99] Statistics and Machine Learning Toolbox, Version 25.1, The MathWorks Inc., Natick, MA, USA, Natick, MA, USA, 2025.
[100] K. De Brabanter, P. Karsmakers, F. Ojeda, C. Alzate, J. De Brabanter, K. Pelckmans, B. De Moor, J. Vandewalle, J.A.K. Suykens, LS-SVMlab toolbox user's guide: version 1.8, Technical Report, Katholieke Universiteit Leuven, 2010.
[101] R. Schöbi, S. Marelli, B. Sudret, UQLab user manual – Polynomial chaos Kriging, Technical Report, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2022. Report UQLab-V2.0-109.
[102] I.M. Sobol, Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates, Mathematics and Computers in Simulation 55 (1-3) (2001) 271–280. https://doi.org/10.1016/S0378-4754(00)00270-6
[103] X. Yang, G.E. Karniadakis, Reweighted ℓ1 minimization method for stochastic elliptic differential equations, J. Comput. Phys. 248 (2013) 87–108.
[104] M. Loève, Probability Theory, Dover Publications, Inc., Mineola, NY, 2017.
[105] Partial Differential Equation Toolbox, Version 25.1, The MathWorks Inc., Natick, MA, USA, 2025.
[106] S. Karumuri, R. Tripathy, I. Bilionis, J. Panchal, Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks, J. Comput. Phys. 404 (2020) 109120.
[107] K. Konakli, B. Sudret, Global sensitivity analysis using low-rank tensor approximations, Reliab. Eng. Syst. Saf. 156 (2016) 64–83.
[108] C.-C. Li, A. Der Kiureghian, Optimal discretization of random fields, J. Eng. Mech. 119 (6) (1993) 1136–1154.
[109] A. Hlobilová, S. Marelli, B. Sudret, Benchmark case datasets – Two-dimensional heat diffusion model, 2024. https://doi.org/10.5281/zenodo.12701147.
[110] A. Hlobilová, S. Marelli, B. Sudret, Benchmark case datasets – Truss model, 2024. https://doi.org/10.5281/zenodo.12699396.