

# Data-driven polynomial chaos expansion for machine learning regression

E. Torre, S. Marelli, P. Embrechts, B. Sudret

## Abstract

We present a regression technique for data-driven problems based on polynomial chaos expansion (PCE). PCE is a popular technique in the field of uncertainty quantification (UQ), where it is typically used to replace a runnable but expensive computational model subject to random inputs with an inexpensive-to-evaluate polynomial function. The metamodel obtained enables a reliable estimation of the statistics of the output, provided that a suitable probabilistic model of the input is available.

Machine learning (ML) regression is a research field that focuses on providing purely data-driven input-output maps, with the focus on pointwise prediction accuracy. We show that a PCE metamodel purely trained on data can yield pointwise predictions whose accuracy is comparable to that of other ML regression models, such as neural networks and support vector machines. The comparisons are performed on benchmark datasets available from the literature. The methodology also enables the quantification of the output uncertainties, and is robust to noise. Furthermore, it enjoys additional desirable properties, such as good performance for small training sets and simplicity of construction, with only little parameter tuning required.

**Keywords:** polynomial chaos expansions, machine learning, regression, sparse representations, uncertainty quantification, copulas.

## 1 Introduction

Machine learning (ML) is increasingly used today to make predictions of system responses and to aid or guide decision making. Given a  $d$ -dimensional input vector  $\mathbf{X}$  to a system and the corresponding  $q$ -dimensional output vector  $\mathbf{Y}$ , data-driven ML algorithms establish a map  $\mathcal{M} : \mathbf{X} \mapsto \mathbf{Y}$  on the basis of an available sample set  $\mathcal{X} = \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}$  of input observations and of the corresponding output values  $\mathcal{Y} = \mathbf{y}^{(1)}, \dots, \mathbf{y}^{(n)}$ , where  $\mathbf{y}^{(i)} = \mathcal{M}(\mathbf{x}^{(i)}) + \varepsilon$  and  $\varepsilon$  is a noise term. In classification tasks, the output  $\mathbf{Y}$  is discrete, that is, it can take at most a countable number of different values (the class *labels*). In regression tasks, which this paper is concerned with, the output  $\mathbf{Y}$  takes continuous values. Regression techniques

include linear regression, neural networks (NN), kernel methods (such as Gaussian processes, GP), sparse kernel machines (such as support vector machines, SVM), graphical models (such as Bayesian networks and Markov random fields), and others. A comprehensive overview of these methods can be found in Bishop (2009) and in Witten et al. (2016).

Current research on ML algorithms focuses on various open issues. First, there is an increasing interest towards problems where the inputs to the system, and as a consequence the system’s response, are uncertain (see, *e.g.*, Chan and Elsheikh (2018); Kasiviswanathan et al. (2016); Mentch and Hooker (2016)). Properly accounting for both aleatory and epistemic input uncertainties allows one to estimate the response statistics. Second, novel methods are sought to better automatize the tuning of hyperparameters, which several ML methods are highly sensitive to Snoek et al. (2012). Third, the paucity of data in specific problems calls for models that can be effectively trained on few observations only Forman and Cohen (2004). Finally, complex models, while possibly yielding accurate predictions, are often difficult if not impossible to interpret.

This manuscript revisits polynomial chaos expansions (PCE), a well established meta-modelling technique in uncertainty quantification (UQ), as a statistical ML Vapnik (1995) regression algorithm that can deal with these challenges Sudret et al. (2015). UQ classically deals with problems where  $\mathbf{X}$  is uncertain, and is therefore modelled as a random vector. As a consequence,  $\mathbf{Y} = \mathcal{M}(\mathbf{X})$  is also uncertain, but its statistics are unknown and have to be estimated. Differently from ML, in UQ the model  $\mathcal{M}$  is typically available (*e.g.*, as a finite element code), and can be computed pointwise. However,  $\mathcal{M}$  is often a computationally expensive black-box model, so that a Monte Carlo approach to estimate the statistics of  $\mathbf{Y}$  (generate a large input sample set  $\mathcal{X}$  to obtain a large output sample set  $\mathcal{Y} = \{\mathcal{M}(\mathbf{x}), \mathbf{x} \in \mathcal{X}\}$ ) is not feasible. PCE is an UQ spectral technique that is used in these settings to express  $\mathbf{Y}$  as a polynomial function of  $\mathbf{X}$ . PCE thereby allows one to replace the original computational model  $\mathcal{M}$  with an inexpensive-to-run but accurate metamodel. The metamodel can be used to derive, for instance, estimates of various statistics of  $\mathbf{Y}$ , such as its moments or its sensitivity to different components of  $\mathbf{X}$  Saltelli et al. (2000), in a computationally efficient way. PCE was originally developed for independent, identically distributed input data from a Gaussian Wiener (1938) and then a list of named distributions Xiu and Karniadakis (2002), then generalized to marginal and joint distributions with an arbitrary functional form Soize and Ghanem (2004); Wan and Karniadakis (2006). A moment-based approach that does not require a functional form of the marginals to be available (or even to exist) was proposed in Oladyshkin and Nowak (2012), and further extended to arbitrary joint distributions in Paulson et al. (2018). In Torre et al. (2019), we established a general framework to perform UQ (including but not limited to PCE metamodeling) in the presence of complex input dependencies modelled through copulas.

Here, we re-establish PCE in a purely data-driven ML setup, where the goal is to obtain

a model that can predict the response  $\mathbf{Y}$  of a system given its inputs  $\mathbf{X}$ . Compared to classical UQ settings where PCE is typically used, four major challenges arise in this data-driven setup:

- no computational model of the system is available, determining a significant lack of information compared to classical UQ settings;
- as a spectral method, PCE requires the knowledge of the joint PDF  $f_{\mathbf{X}}$  of the input. We address instead scenarios where  $f_{\mathbf{X}}$  is entirely inferred from data, a step that can affect convergence;
- to guarantee generality, inference on the marginal distributions is performed here non-parametrically (by kernel smoothing), the rationale being that too little evidence is available to assume any parametric families;
- the obtained model needs to be robust to noise in the data, a feature that to our knowledge has not been investigated for PCE yet.

For simplicity, we consider the case where  $\mathbf{Y}$  is a scalar random variable  $Y$ . The generalisation to multivariate outputs is straightforward. In the setup of interest here, no computational model  $\mathcal{M}$  is available, but only a set  $(\mathcal{X}, \mathcal{Y})$  of input values and corresponding responses.  $\mathcal{X}$  and  $\mathcal{Y}$  are considered to be (possibly noisy) realisations of  $\mathbf{X}$  and  $Y$ , the true relationship between which is deterministic but unknown.

After recalling the underlying theory (Section 2), we first show by means of simulation that data-driven PCE delivers accurate pointwise predictions (Section 3). In addition, PCE also enables a reliable estimation of the statistics of the response (such as its moments and PDF), thus enabling uncertainty quantification of the predictions being made. Dependencies among the input variables are effectively modelled by copula functions, specifically by vine copulas. Vines copulas have been recently investigated for general UQ applications, including PCE metamodeling, in Torre et al. (2019). Here their applicability to data-driven settings is investigated and demonstrated. The full approach is shown to be robust to noise in the data, a feature that has not been investigated yet for PCE, and that enables denoising. Furthermore, the methodology works well in the presence of small training sets. In Section 4, we further apply PCE to real data previously analyzed by other authors with different NN and/or SVM algorithms, where it achieves a comparable performance. Importantly, the construction of the PCE metamodel does not rely on fine tuning of critical hyper-parameters. This and other desirable features of the PCE modelling framework are discussed in Section 5.

## 2 Methods

### 2.1 Measures of accuracy

Before introducing PCE, which is a popular metamodeling technique in UQ, as an ML technique used to make pointwise predictions, it is important to clarify the distinction between the error types that UQ and ML typically aim at minimizing.

ML algorithms are generally used to predict, for any *given* input, the corresponding output of the system. Their performance is assessed in terms of the distance of the prediction from the actual system response, calculated for each input and averaged over a large number of (ideally, all) possible inputs. A common error measure in this sense, also used here, is the mean absolute error (MAE). For a regression model  $\mathcal{M}$  trained on a training data set  $(\mathcal{X}', \mathcal{Y}')$ , the MAE is typically evaluated over a validation data set  $(\mathcal{X}'', \mathcal{Y}'')$  of size  $n''$  by

$$\text{MAE} = \frac{1}{n''} \sum_{(\mathbf{x}, y) \in (\mathcal{X}'', \mathcal{Y}'')} |y - \mathcal{M}(\mathbf{x})|. \quad (1)$$

The associated relative error considers point by point the ratio between the absolute error and the actual response, *i.e.*,

$$\text{rMAE} = \frac{1}{n} \sum_{(\hat{\mathbf{x}}, \hat{y}) \in (\mathcal{X}, \mathcal{Y})} \left| \frac{\hat{y} - \mathcal{M}(\hat{\mathbf{x}})}{\hat{y}} \right| = \frac{1}{n} \sum_{(\hat{\mathbf{x}}, \hat{y}) \in (\mathcal{X}, \mathcal{Y})} \left| 1 - \frac{\mathcal{M}(\hat{\mathbf{x}})}{\hat{y}} \right|, \quad (2)$$

which is well defined if  $\hat{y} \neq 0$  for all  $\hat{y} \in \mathcal{Y}$ .

PCE is instead typically used to draw accurate estimates of various statistics of the output – such as its moments, its PDF, confidence intervals – given a probabilistic model  $f_{\mathbf{X}}$  of an uncertain input. The relative error associated to the estimates  $\hat{\mu}_Y$  of  $\mu_Y$  and  $\hat{\sigma}_Y$  of  $\sigma_Y$  is often quantified by

$$\left| 1 - \frac{\hat{\mu}_Y}{\mu_Y} \right|, \quad \left| 1 - \frac{\hat{\sigma}_Y}{\sigma_Y} \right|, \quad (3)$$

provided that  $\mu_Y \neq 0$  and  $\sigma_Y \neq 0$ . A popular measure of the error made on the full response PDF  $f_Y$  is the Kullback-Leibler divergence

$$\Delta_{\text{KL}}(\hat{f}_Y | f_Y) = \int_y \log \left( \frac{\hat{f}_Y(y)}{f_Y(y)} \right) f_Y(y) dy, \quad (4)$$

which quantifies the average difference between the logarithms of the predicted and of the true PDFs. In the simulated experiments performed in Section 3, we used the error measures in (3) and (4) to assess the quality of the PCE estimates  $\hat{\mu}_Y$ ,  $\hat{\sigma}_Y$ , and  $\hat{f}_Y$ . The reference solutions  $\mu$ ,  $\sigma$ , and  $f_Y$  were obtained by MCS on  $10^7$  samples. The real data analyzed in Section 4, instead, contained too few points to draw sufficiently accurate reference values. Statistical errors for these data could not be quantified.

Provided a suitable model  $f_{\mathbf{X}}$  of the joint PDF of the input, PCE is known to converge (in the mean-square sense) very rapidly as the size of the training set increases, compared for instance to Monte-Carlo sampling Puig et al. (2002); Todor and Schwab (2007); Ernst

et al. (2012). This happens because PCE, which is a spectral method, efficiently propagates the probabilistic information delivered by the input model  $f_{\mathbf{X}}$  to the output (as explained in more details next). Nevertheless, this does not necessarily imply a similarly rapid pointwise convergence of the error, which remains to be demonstrated. Also, the speed of mean-square convergence will generally worsen if  $f_{\mathbf{X}}$  is unknown and needs to be inferred from data. For a discussion of various types of convergence, see Ernst et al. (2012) and references therein.

## 2.2 PCE in data-driven settings

PCE is designed to build an inexpensive-to-evaluate analytical model  $Y_{PC} = \mathcal{M}_{PC}(\mathbf{X})$  mapping an input random vector  $\mathbf{X}$  onto an output random variable  $Y$  Ghanem and Spanos (1990); Xiu and Karniadakis (2002). PCE assumes that an unknown deterministic map  $\mathcal{M} : \mathbb{R}^d \rightarrow \mathbb{R}$  exists, such that  $Y = \mathcal{M}(\mathbf{X})$ .  $Y$  is additionally assumed to have finite variance:  $\mathbb{V}(Y) = \int_{\mathbb{R}^d} \mathcal{M}^2(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} < +\infty$ . Under the further assumption that each  $X_i$  has finite moments of all orders Ernst et al. (2012), the space of square integrable functions with respect to  $f_{\mathbf{X}}(\cdot)$  admits a basis  $\{\Psi_{\alpha}(\cdot), \alpha \in \mathbb{N}^d\}$  of polynomials orthonormal to each other with respect to the probability measure  $f_{\mathbf{X}}$ , *i.e.*, such that

$$\int_{\mathbb{R}^d} \Psi_{\alpha}(\mathbf{x}) \Psi_{\beta}(\mathbf{x}) f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} = \delta_{\alpha\beta}. \quad (5)$$

Here,  $\delta_{\alpha\beta}$  is the Kronecker delta symbol. The element  $\alpha_i$  of the multi-index  $\alpha \in \mathbb{N}^d$  indicates the degree of  $\Psi_{\alpha}$  in the  $i$ -th variable,  $i = 1, \dots, d$ .  $\Psi_{\alpha}$  has a total degree given by  $|\alpha| = \sum_i \alpha_i$ .

Thus,  $Y$  admits the spectral representation

$$Y(\mathbf{X}) = \sum_{\alpha \in \mathbb{N}^d} y_{\alpha} \Psi_{\alpha}(\mathbf{X}). \quad (6)$$

The goal of PCE regression is to determine the coefficients  $y_{\alpha}$  of the expansion, truncated at some polynomial order, given an initial set  $(\mathcal{X}, \mathcal{Y})$  of observations (the *training set* or *experimental design*). The advantage of expressing the functional relationship between the inputs and the output by an orthonormal multivariate basis lies in its spectral decay properties. Indeed, given that finite variance models/phenomenons are considered, the orthonormality of the basis implies that the squared sum of the PCE coefficients (equal to the variance, see Section 2.3.5) is finite. Therefore, the orthonormal basis guarantees a rapidly decaying spectrum of the PCE coefficients. This in turn reduces the number of parameters in the PCE representation, yielding a compressed representation. The orthonormal basis is thus paramount to reduce the number of regression coefficients and thus to avoid overfitting on finite datasets. The superiority of the orthonormal basis with respect to a non-orthogonal one in these settings is demonstrated on simulated data in Section 3.4.

In engineering applications the model  $\mathcal{M}(\cdot)$  is often directly available (*e.g.*, as a finite element model) but computationally expensive to evaluate: in these cases, PCE is used as a surrogate to replace the true model with an inexpensive-to-evaluate metamodel. In the

standard ML settings considered here, conversely,  $\mathcal{M}(\cdot)$  is unknown and has to be modelled solely on the basis of available observations  $(\mathcal{X}, \mathcal{Y})$ . The primary goal of this paper is to show that the accuracy of a PCE metamodel built purely on  $(\mathcal{X}, \mathcal{Y})$  can compete with that of state-of-the-art machine learning regression models, while requiring little parameter tuning and in some cases offering additional advantages.

### 2.3 PCE for independent inputs

In this section, we assume that the input random vector  $\mathbf{X}$  has statistically independent components. The PCE basis is built from the tensor product of univariate orthogonal polynomials. The case of inputs with dependent components is discussed later in Section 2.4.

#### 2.3.1 Orthogonalisation for independent inputs

When  $\mathbf{X}$  has independent components, the  $\Psi_{\alpha}$  can be obtained as the tensor product of  $d$  univariate polynomials,

$$\Psi_{\alpha}(\mathbf{x}) = \prod_{i=1}^d \phi_{\alpha_i}^{(i)}(x_i), \quad (7)$$

where each  $\phi_{\alpha_i}^{(i)}(x_i)$  is the element of degree  $\alpha$  of a basis of univariate polynomials orthonormal with respect to the marginals  $f_i$  of  $\mathbf{X}$ , that is, such that

$$\int_{\mathbb{R}} \phi_{\alpha_i}^{(i)}(\omega) \phi_{\beta_i}^{(i)}(\omega) f_i(\omega) d\omega = \delta_{\alpha_i \beta_i}. \quad (8)$$

The problem of building a basis of mutually orthonormal multivariate polynomials with respect to  $f_{\mathbf{X}}$  hence becomes the problem of building  $d$  bases of mutually orthonormal univariate polynomials, each with respect to a marginal  $f_i$ . The  $d$  bases can be built independently.

#### 2.3.2 Specification of the marginals and PCE construction

The construction of the polynomial basis requires a model of the marginal input distributions. Families of univariate orthonormal polynomials associated to classical continuous distributions are described in Xiu and Karniadakis (2002). An input variable  $X_i$  with a different distribution  $F_i$  (continuous, strictly increasing) may be transformed into a random variable  $\tilde{X}_i$  with distribution  $\Phi$  belonging to one of the classical families by the transformation

$$\tilde{X}_i = \Phi^{-1}(F_i(X_i)). \quad (9)$$

This relation offers a first possible (but usually not recommended) approach to build the PCE of  $Y$  for inputs  $\mathbf{X}$  with generic continuous marginals  $F_i$ : transform  $\mathbf{X}$  into  $\tilde{\mathbf{X}}$  through (9), and then build a PCE of  $Y$  in terms of  $\tilde{\mathbf{X}}$ . The PCE has to be trained on  $(\tilde{\mathcal{X}}, \mathcal{Y})$ , where  $\tilde{\mathcal{X}}$  is obtained from  $\mathcal{X}$  using (9).

A second approach, undertaken in this study, consists in directly constructing a basis of orthonormal polynomials with respect to  $F_i$ , by Stiltjes or Gram-Schmidt orthogonalisation Soize and Ghanem (2004); Wan and Karniadakis (2006). In practice, performing PCE on the original input  $\mathbf{X}$  is often preferable to building the PCE on inputs transformed via (9). Indeed, the latter is usually a highly non-linear transformation, making the map from  $\tilde{\mathbf{X}}$  to  $Y$  more difficult to approximate by polynomials Oladyshkin and Nowak (2012). In the applications presented in this paper, we opt for non-parametric inference of the input marginals  $f_i$  by kernel density estimation (KDE) Rosenblatt (1956); Parzen (1962). Given a set  $\mathcal{X}_i = \{x_i^{(1)}, \dots, x_i^{(n)}\}$  of observations of  $X_i$ , the kernel density estimate  $\hat{f}_i$  of  $f_i$  reads

$$\hat{f}_i(x) = \frac{1}{nh} \sum_{j=1}^n k\left(\frac{x - x_i^{(j)}}{h}\right), \quad (10)$$

where  $k(\cdot)$  is the kernel function and  $h$  is the appropriate kernel bandwidth that is learned from the data. Different kernels are used in practice, such as the Epanechnikov or Gaussian kernels. Here we use the latter, that is,  $k(\cdot)$  is selected as the standard normal PDF.

A third approach for PCE construction, first introduced in Oladyshkin and Nowak (2012) under the name of arbitrary PCE (aPCE), consists in constructing a basis of polynomials orthonormal to the input moments (not to the input PDF directly). The method was first introduced for independent input variables and later extended to correlated variables in Paulson et al. (2018). aPCE offers the advantage to be more general, as it requires only the input moments and does not need (or even assume the existence of) a functional form of the input PDF. An accurate estimation of moments of higher order, however, requires a large number of input samples, thus effectively limiting its applicability in the settings considered here Oladyshkin and Nowak (2018). For this reason, we opt instead for the second approach outlined above.

After estimating the input marginals by KDE, we resort to PCE to build a basis of orthonormal polynomials. The PCE metamodel is then trained on  $(\mathcal{X}, \mathcal{Y})$ .

### 2.3.3 Truncation schemes

The sum in (6) involves an infinite number of terms. For practical purposes, it is truncated to a finite sum. Truncation is a critical step: a wrong truncation will lead to underfitting (if too many terms are removed) or to overfitting (if too many terms are retained compared to the number of available training points). The model complexity can be controlled by various truncation schemes.

The standard basis truncation Xiu and Karniadakis (2002) retains the subset of terms defined by

$$\mathcal{A}^{d,p} = \{\boldsymbol{\alpha} \in \mathbb{N}^d : |\boldsymbol{\alpha}| \leq p\},$$

where  $p \in \mathbb{N}^+$  is the chosen maximum polynomial degree and  $|\boldsymbol{\alpha}| = \sum_{i=1}^d \alpha_i$  is the total degree of  $\Psi_{\boldsymbol{\alpha}}$ . Thus, only polynomials of degree up to  $p$  are considered.  $\mathcal{A}^{d,p}$  contains  $\binom{d+p}{p}$

elements. To further reduce the basis size, several additional truncation schemes have been proposed. Hyperbolic truncation Blatman and Sudret (2011) retains the polynomials with indices in

$$\mathcal{A}^{d,p,q} = \{\boldsymbol{\alpha} \in \mathcal{A}^{d,p} : \|\boldsymbol{\alpha}\|_q \leq p\},$$

where  $q \in (0, 1]$  and  $\|\cdot\|_q$  is the  $q$ -norm defined by  $\|\boldsymbol{\alpha}\|_q = \left(\sum_{i=1}^d \alpha_i^q\right)^{1/q}$ .

A complementary strategy is to set a limit to the number of non-zero elements in  $\boldsymbol{\alpha}$ , that is, to the number of interactions among the components  $X_i$  of  $\mathbf{X}$  in the expansion. This maximum interaction truncation scheme retains the polynomials with indices in

$$\mathcal{A}^{d,p,r} = \{\boldsymbol{\alpha} \in \mathcal{A}^{d,p} : \|\boldsymbol{\alpha}\|_0 \leq r\},$$

where  $\|\boldsymbol{\alpha}\|_0 = \sum_{i=1}^d \mathbf{1}_{\{\alpha_i > 0\}}$ .

In our case studies below we combined hyperbolic and maximum truncation, thus retaining the expansion's polynomials with coefficients in

$$\mathcal{A}^{d,p,q,r} = \mathcal{A}^{d,p,q} \cap \mathcal{A}^{d,p,r}. \quad (11)$$

This choice is motivated by the sparsity of effect principle, which assumes that only few meaningful interactions influence system responses and which holds in most engineering applications. The three hyperparameters  $(p, q, r)$  can all be automatically tuned by cross-validation within pre-assigned ranges, as illustrated in Blatman and Sudret (2011). To reduce the already high computational cost due to the large number of simulations performed in the current study, we fixed  $q = 0.75$  and tuned  $p$  and  $r$  only, within pre-assigned ranges  $[1, p_{\max}]$  and  $[1, r_{\max}]$ .  $p_{\max}$  and  $r_{\max}$  were selected depending on the amount of training data available. Starting from simpler models (lower  $p$  and  $r$ ), more complex models were favored if yielding a lower cross-validation error. The calibration of each parameter stopped either if no increase in performance was attained in two consecutive steps, or if the maximum allowed value was reached.

### 2.3.4 Calculation of the coefficients

Selected a truncation scheme and the corresponding set  $\mathcal{A} = \{\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_{|\mathcal{A}|}\}$  of multi-indices, the coefficients  $y_k$  in

$$Y_{PC}(\mathbf{X}) = \sum_{k=1}^{|\mathcal{A}|} y_k \Psi_{\boldsymbol{\alpha}_k}(\mathbf{X}) \quad (12)$$

need to be determined on the basis of a set  $(\mathcal{X}, \mathcal{Y}) = \{(\mathbf{x}^{(j)}, y^{(j)}), j = 1, \dots, n\}$  of observed data. In these settings, the vector  $\mathbf{y} = (y_{\boldsymbol{\alpha}_1}, \dots, y_{\boldsymbol{\alpha}_{|\mathcal{A}|}})$  of expansion coefficients can be determined by regression.

When the number  $|\mathcal{A}|$  of regressors is smaller than the number  $n$  of observations,  $\mathbf{y}$  can be determined by solving the ordinary least squares (OLS) problem

$$\mathbf{y} = \arg \min_{\mathbf{y}} \sum_{j=1}^n \left( y^{(j)} - Y_{PC}(\mathbf{x}^{(j)}) \right)^2 = \arg \min_{\mathbf{y}} \sum_{j=1}^n \left( y^{(j)} - \sum_{k=1}^{|\mathcal{A}|} y_k \Psi_{\boldsymbol{\alpha}_k}(\mathbf{x}^{(j)}) \right)^2. \quad (13)$$

The solution reads

$$\mathbf{y} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{pmatrix}, \quad (14)$$

where  $A_{jk} = \Psi_{\alpha_k}(x^{(j)})$ ,  $j = 1, \dots, n$ ,  $k = 1, \dots, |\mathcal{A}|$ .

The OLS problem cannot be solved when  $n < |\mathcal{A}|$ , because in this case the matrix  $\mathbf{A}^T \mathbf{A}$  in (14) is not invertible. Also, OLS tends to overfit the data when  $|\mathcal{A}|$  is large. Simpler models with fewer coefficients can be constructed by sparse regression.

Least angle regression (LAR), proposed in Efron et al. (2004), is an algorithm that achieves sparse regression by solving the regularised regression problem

$$\mathbf{y} = \arg \min_{\tilde{\mathbf{y}}} \left\{ \sum_{j=1}^n \left( y^{(j)} - \sum_{k=1}^{|\mathcal{A}|} y_k \Psi_{\alpha_k}(\mathbf{x}^{(j)}) \right)^2 + \lambda \|\tilde{\mathbf{y}}\|_1 \right\}. \quad (15)$$

The last addendum in the expression is the regularisation term, which forces the minimisation to favour sparse solutions. One advantage of using LAR is that it does not require explicit optimization with respect to the  $\lambda$  parameter, which in turn is never explicitly calculated. The use of LAR in the context of PCE was initially proposed in Blatman and Sudret (2011), which the reader is referred to for more details. In the applications illustrated in Sections 3 and 4, we adopt LAR to determine the PCE coefficients.

### 2.3.5 Estimation of the output statistics

Given the statistical model  $F_{\mathbf{Z}}$  of the input and the PCE metamodel (12) of the input-output map, the model response  $Y_{\text{PC}}$  is not only known pointwise, but can also be characterised statistically. For instance, the orthonormality of the polynomial basis ensures that the mean and the variance of  $Y_{\text{PC}}$  read, respectively,

$$\mathbb{E}[Y_{\text{PC}}] = y_{\mathbf{0}}, \quad \mathbb{V}[Y_{\text{PC}}] = \sum_{\alpha \in \mathcal{A} \setminus \{\mathbf{0}\}} y_{\alpha}^2. \quad (16)$$

This property provides a useful interpretation of the expansion coefficients in terms of the first two moments of the output. Other output statistics, such as the Sobol partial sensitivity indices Sobol' (1993), can be obtained from the expansion coefficients analytically Sudret (2008).

Higher-order moments of the output, as well as other statistics (such as the full PDF  $F_Y$ ), can be efficiently estimated through Monte-Carlo simulation, by sampling sufficiently many realisations of  $\mathbf{X}$  and evaluating the corresponding PCE responses. Polynomial evaluation is computationally cheap and can be trivially vectorised, making estimation by resampling extremely efficient.

## 2.4 PCE for mutually dependent inputs

If the input vector  $\mathbf{X}$  has statistically dependent components, its joint PDF  $f_{\mathbf{X}}$  is not the product of the marginals and the orthogonality property (5) does not hold. As a consequence,

one cannot construct multivariate orthogonal polynomials by tensor product of univariate ones, as done in (7). Constructing a basis of orthogonal polynomials in this general case is still possible, for instance by Gram-Schmidt orthonormalisation NAVARRO et al. (2014). However, the procedure becomes more and more computationally demanding as the number of inputs or the expansion order grow, as it involves the numerical evaluation of higher-dimensional integrals. For this reason, we investigate two alternative strategies.

The first strategy consists in ignoring the input dependencies and in building a basis of polynomials orthonormal with respect to  $\prod_i f_i(x_i)$  by arbitrary PCE. This approach is labelled from here on as *aPCEonX*. While not forming a basis of the space of square integrable functions with respect to  $f_{\mathbf{X}}$ , the considered set of polynomials may still yield a similarly good pointwise approximation.

Accurate estimates of the output statistics may be obtained *a posteriori* by modelling the input dependencies through copula functions, as detailed in A. The joint CDF of a random vector  $\mathbf{X}$  with copula distribution  $C_{\mathbf{X}}$  and marginals  $F_i$  (here, obtained by KDE) is given by (1). Resampling from such a probabilistic input model yields more accurate estimates of the output statistics than resampling from the distribution  $\prod_i F_i(x_i)$ , that is, than ignoring dependencies.

The second possible approach, presented in more details in B, consists in modelling directly the input dependencies by copulas, and then in transforming the input vector  $\mathbf{X}$  into a set of independent random variables  $\mathbf{Z}$  with prescribed marginals (*e.g.*, standard uniform) through the associated Rosenblatt transform  $\mathcal{T}^{(\text{II})}$  Rosenblatt (1952), defined in (18). The PCE metamodel is then built between the transformed input vector  $\mathbf{Z}$  and the output  $Y$ . When the input marginals are standard uniform distributions, the corresponding class of orthonormal polynomials is the Legendre family, and the resulting PCE is here indicated by *lPCEonZ*. The asymptotic properties of an orthonormal PCE, such as the relations (16), hold. The expression of  $Y$  in terms of  $\mathbf{Z}$  is given by the relation (1).

At first, the second approach seems advantageous over the first one. It involves, in a different order, the same steps: modelling the input marginals and copula, and building the PCE metamodel. By capturing dependencies earlier on, it enables the construction of a basis of mutually orthonormal polynomials with respect to the inferred joint PDF of the (transformed) inputs. It thereby provides a model with spectral properties (except for unavoidable errors due to inference, truncation of the expansion, and estimation of the expansion parameters). The experiments run in Section 3, however, show that the first approach yields typically more accurate pointwise predictions (although not always better statistical estimates). This is due to the fact that it avoids mapping the input  $\mathbf{X}$  into independent variables  $\mathbf{Z}$  via the (typically strongly non-linear) Rosenblatt transform. The shortcomings of mapping dependent inputs into independent variables were noted, in classical UQ settings, by Eldred et al. (2008), and are thus extended here to data-driven settings. For

a more detailed discussion, see Section 3.5.

### 3 Validation on synthetic data

#### 3.1 Validation scheme

We first investigate data-driven regression by PCE on two different simulated data sets. The first data set is obtained through an analytical, highly non-linear function of three variables, subject to three uncertain inputs. The second data set is a finite element model of a horizontal truss structure, subject to 10 uncertain inputs. In both cases, the inputs are statistically dependent, and their dependence is modelled through a canonical vine (C-vine) copula (see A).

We study the performance of the two alternative approaches described in Section 2.4: *aPCEonX* and *lPCEonZ*. In both cases we model the input copula as a C-vine, fully inferred from the data. For the *aPCEonX* the choice of the copula only plays a role in the estimation of the output statistics, while for the *lPCEonZ* it affects also the pointwise predictions. We additionally tested the performance obtained by using the true input copula (known here because it was used to generate the synthetic data) and a Gaussian copula inferred from data. We also investigated the performance obtained by using the true marginals instead of the ones fitted by KDE. Using the true copula and marginals yielded better statistical estimates, but is of little practical interest, as it would not be known in real applications. The Gaussian copula yielded generally similar or slightly worse accuracy. For brevity, we do not show these results.

To assess the pointwise accuracy, we generate a training set  $(\mathcal{X}', \mathcal{Y}')$  of increasing size  $n'$ , build the PCE metamodels both by *aPCEonX* and by *lPCEonZ*, and evaluate their rMAE on a validation set  $(\mathcal{X}'', \mathcal{Y}'')$  of fixed size  $n'' = 10,000$  points. Both  $\mathcal{X}'$  and  $\mathcal{X}''$  are sampled from the probabilistic input model with true marginals and vine copula. The statistical accuracy is quantified instead in terms of error on the mean, standard deviation, and full PDF (by the Kullback-Leibler divergence) of the true models, as defined in (2)-(4). Reference solutions are obtained by MC sampling from the true input model, while the PCE estimates are obtained by resampling from the inferred marginals and copula. The sample size is  $10^7$  in both cases. The statistical estimates obtained by the two PCE approaches are furthermore compared to the corresponding sample estimates obtained from the same training data (sample mean, sample standard deviation, and KDE of the PDF).

For each value of  $n'$  and each error type, error bands are obtained as the minimum to maximum error obtained across 10 different realisations of  $(\mathcal{X}', \mathcal{Y}')$  and  $(\mathcal{X}'', \mathcal{Y}'')$ . The minimum error is often taken in machine learning studies as an indication of the best performance that can be delivered by a given algorithm in a given task. The maximum error represents analogously the worst-case scenario.

Finally, we assess the robustness to noise by adding a random perturbation  $\varepsilon$  to each output observations in  $\mathcal{Y}$  used to train the PCE model. The noise is drawn independently for each observation from a univariate Gaussian distribution with mean  $\mu_\varepsilon = 0$  and prescribed standard deviation  $\sigma_\varepsilon$ .

### 3.2 Ishigami function

The first model we consider is

$$Y = 1 + \frac{\text{ish}(X_1, X_2, X_3) + 1 + \pi^4/10}{9 + \pi^4/5}, \quad (17)$$

where

$$\text{ish}(x_1, x_2, x_3) = \sin(x_1) + 7 \sin^2(x_2) + 0.1x_3^4 \sin(x_1) \quad (18)$$

is the Ishigami function Ishigami and Homma (1990), defined for inputs  $x_i \in [-\pi, \pi]$  and taking values in  $[-1 - \frac{\pi^4}{10}, 8 + \frac{\pi^4}{10}]$ . The Ishigami function is often used as a test case in global sensitivity analysis due to its strong non-linearity and non-monotonicity. Model (17) is rescaled here to take values in the interval  $[1, 2]$ . Rescaling does not affect the approximation accuracy of the PCE, but enables a meaningful evaluation of the rMAE (2) by avoiding values of  $Y$  close to 0.

We model the input  $\mathbf{X}$  as a random vector with marginals  $X_i \sim \mathcal{U}([-\pi, \pi])$ ,  $i = 1, 2, 3$ , and C-vine copula with density

$$c_{\mathbf{X}}(u_1, u_2, u_3) = c_{12}^{(g)}(u_1, u_2; 2) \cdot c_{13}^{(t)}(u_1, u_3; 0.5, 3).$$

Here,  $c_{12}^{(g)}(\cdot, \cdot; \theta)$  and  $c_{13}^{(t)}(\cdot, \cdot; \theta)$  are the densities of the pairwise Gumbel and t- copula families defined in rows 11 and 19 of Table A.4. Thus,  $X_1$  correlates positively with both  $X_2$  and  $X_3$ .  $X_2$  and  $X_3$  are also positively correlated, but conditionally independent given  $X_1$ . The joint CDF of  $\mathbf{X}$  can be obtained from its marginals and copula through (1). The PDF of  $Y$  in response to input  $\mathbf{X}$ , its mean  $\mu_Y$  and its standard deviation  $\sigma_Y$ , obtained on  $10^7$  sample points, are shown in the left panel of Figure 1.

Next, we build the *aPCEonX* and the *lPCEonZ* on training data  $(\mathcal{X}', \mathcal{Y}')$ , and assess their performance as described in Section 3.1. The errors are shown in Figure 2 (red: *aPCEonX*; green: *lPCEonZ*), as a function of the training set size  $n'$ . The dotted line indicates the average error over the 10 repetitions, while the shaded area around it spans the range from the minimum to the maximum error across 10 repetitions. The *aPCEonX* yields a considerably lower rMAE. This is due to the strong non-linearity of the Rosenblatt transform used by the *lPCEonZ* to de-couple the components of the input data. Importantly, the methodology works well already in the presence of relatively few data points: the pointwise error and the Kullback-Leibler divergence both drop below 1% when using as few as  $n' = 100$  data points. The central panel of Figure 1 shows the histogram obtained from  $n' = 1,000$  output observations of one training set, the true PDF (black), and the PDFs obtained by

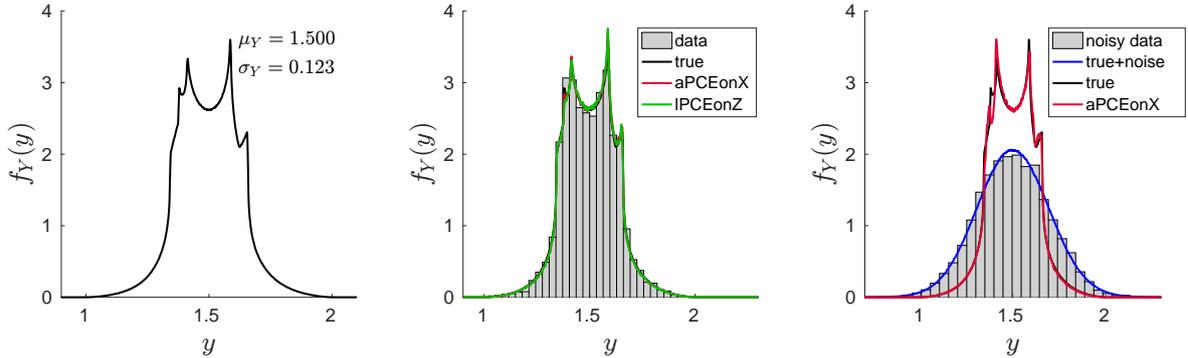


Figure 1: **Response PDFs of the Ishigami function.** *Left panel:* true PDF  $f_Y$  of the Ishigami model's response, obtained on  $10^7$  sample points by KDE. *Central panel:* histogram obtained from  $n' = 1,000$  output observations used for training (gray bars), true response PDF as in the left panel (black), PDFs obtained from the *aPCEonX* (red) and the *lPCEonZ* (green) by resampling. *Right panel:* as in the central panel, but for training data perturbed with Gaussian noise ( $\sigma_\varepsilon = 0.15 = 1.22 \sigma_Y$ ). The blue line indicates the true PDF of the perturbed model.

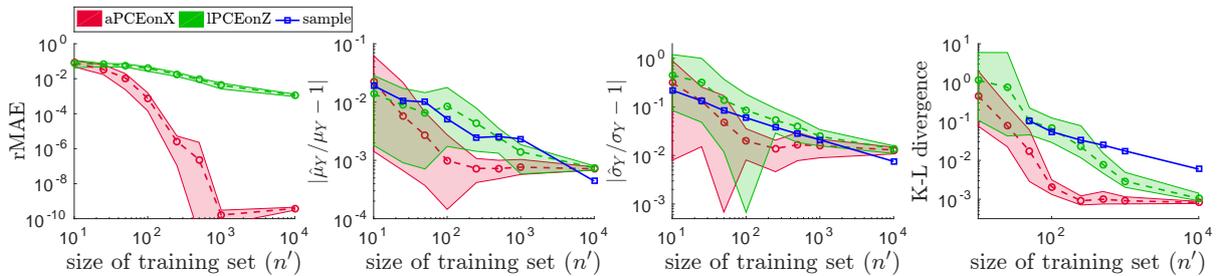


Figure 2: **PCE of Ishigami model: performance.** From left to right: rMAE, error on the mean, error on the standard deviation, and Kullback-Leibler divergence of *aPCEonX* (red) and *lPCEonZ* (green), for a size  $n'$  of the training set increasing from 10 to 10,000. The dash-dotted lines and the bands indicate, respectively, the average and the minimum to maximum errors over 10 simulations. In the second to fourth panels, the blue lines correspond to the empirical estimates obtained from the training data (error bands not shown).

resampling from the *aPCEonX* and the *lPCEonZ* built on that training set. The statistics of the true response are better approximated by the *aPCEonX* than by the *lPCEonZ* or by sample estimation (blue solid lines in Figure 2).

Finally, we examine the robustness of *aPCEonX* to noise. We perturb each observation in  $\mathcal{Y}'$  by adding noise drawn from a Gaussian distribution with mean 0 and standard deviation  $\sigma_\varepsilon$ .  $\sigma_\varepsilon$  is assigned as a fixed proportion of the model's true mean  $\mu_Y$ : 1%, 10%, and 30% of  $\mu_Y$  (corresponding to 12%, 122%, and 367% of  $\sigma_Y$ , respectively). The results, shown in Figures 3-4, indicate that the methodology is robust to noise. Indeed, the errors of all types are significantly smaller than the magnitude of the added noise, and decrease with increasing sample size (see Figure 3). For instance, the rMAE for  $\sigma_\varepsilon = 0.15 = 1.22 \sigma_Y$  drops to  $10^{-2}$  if 100 or more training points are used. The error on  $\mu_Y$  is minorly affected, which is expected since the noise is unbiased. More importantly,  $\sigma_Y$  and  $f_Y$  can be recovered with

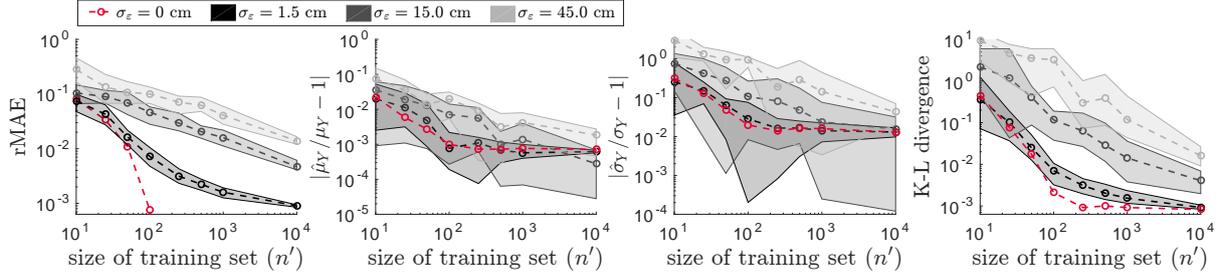


Figure 3: *aPCEonX* of Ishigami model: **robustness to noise (for multiple noise levels)**. From left to right: rMAE, error on the mean, error on the standard deviation, and Kullback-Leibler divergence of the *aPCEonX* for an increasing amount of noise:  $\sigma_\varepsilon = 0.015$  (dark gray),  $\sigma_\varepsilon = 0.15$  (mid-light gray), and  $\sigma_\varepsilon = 0.45$  (light gray). The dash-dotted lines and the bands indicate, respectively, the average and the minimum to maximum error over 10 simulations. The red lines, reported from Figure 2 for reference, indicate the mean error obtained for the noise-free data.

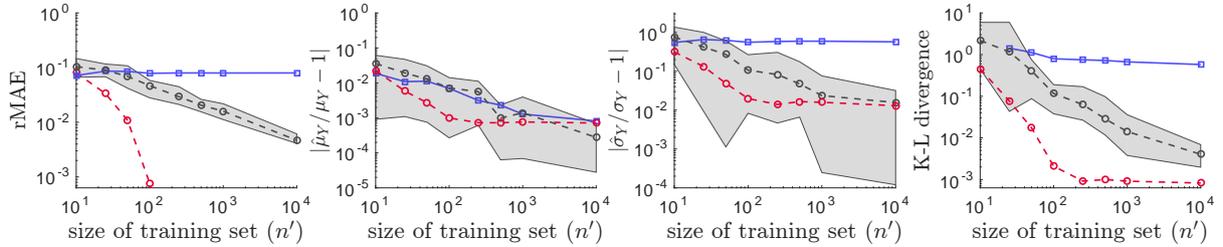


Figure 4: *aPCEonX* of Ishigami model: **robustness to noise (versus sample estimation)**. From left to right: rMAE, error on the mean, error on the standard deviation, and Kullback-Leibler divergence obtained by *aPCEonX* (gray) and by direct sample estimation (blue), for noise  $\sigma_\varepsilon = 0.15 = 1.22 \sigma_Y$ . The dash-dotted lines and the bands indicate, respectively, the average and the minimum to maximum error over 10 simulations. The red lines, reported from Figure 2 for reference, indicate the mean error obtained for the noise-free data.

high precision even in the presence of strong noise (see also Figure 1, fourth panel). In this case, the PCE predictor for the standard deviation works significantly better than the sample estimates, as illustrated in Figure 4.

### 3.3 23-bar horizontal truss

We further replicate the analysis carried out in the previous section on a more complex finite element model of a horizontal truss Blatman and Sudret (2011). The structure consists of 23 bars connected at 6 upper nodes, and is 24 meters long and 2 meters high (see Figure 5). The bars belong to two different groups (horizontal and diagonal bars), both having uncertain

Young modulus  $E_i$  and uncertain cross-sectional area  $A_i$ ,  $i = 1, 2$ :

$$\begin{aligned} E_1, E_2 &\sim \mathcal{LN}(2.1 \cdot 10^{11}, 2.1 \cdot 10^{10}) \text{ Pa}, \\ A_1 &\sim \mathcal{LN}(2.0 \cdot 10^{-3}, 2.0 \cdot 10^{-4}) \text{ m}^2, \\ A_2 &\sim \mathcal{LN}(1.0 \cdot 10^{-3}, 1.0 \cdot 10^{-4}) \text{ m}^2, \end{aligned}$$

where  $\mathcal{LN}(\mu, \sigma)$  is the univariate lognormal distribution with mean  $\mu$  and standard deviation  $\sigma$ .

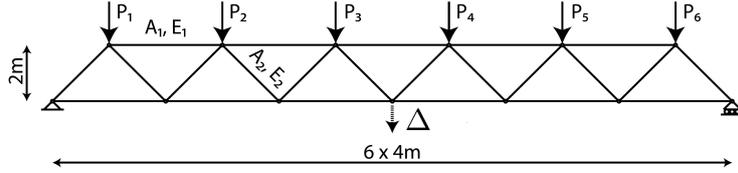


Figure 5: **Scheme of the horizontal truss model.** 23-bar horizontal truss with bar cross-section  $A_i$  and Young modulus  $E_i$  ( $i = 1, 2$ : horizontal and vertical bars, respectively), subject to loads  $P_1, \dots, P_6$ . Modified from Blatman and Sudret (2011).

The four variables can be considered statistically independent, and their values influence the structural response to loading. An additional source of uncertainty comes from six random loads  $P_1, P_2, \dots, P_6$  the truss is subject to, one on each upper node. The loads have Gumbel marginal distribution with mean  $\mu = 5 \times 10^4$  N and standard deviation  $\sigma = 0.15\mu = 7.5 \times 10^3$  N:

$$F_i(x; \alpha, \beta) = e^{-e^{-(x-\alpha)/\beta}}, \quad x \in \mathbb{R}, i = 1, 2, \dots, 6, \quad (19)$$

where  $\beta = \sqrt{6}\sigma/\pi$ ,  $\alpha = \mu - \gamma\beta$ , and  $\gamma \approx 0.5772$  is the Euler–Mascheroni constant. In addition, the loads are made mutually dependent through the C-vine copula with density

$$c_{\mathbf{X}}^{(\mathcal{G})}(u_1, \dots, u_6) = \prod_{j=2}^6 c_{1j; \theta=1.1}^{(\mathcal{GH})}(u_1, u_j), \quad (20)$$

where each  $c_{1j; \theta}^{(\mathcal{GH})}$  is the density of the pair-copula between  $P_1$  and  $P_j$ ,  $j = 2, \dots, 6$ , and belongs to the Gumbel–Hougaard family defined in Table A.4, row 11.

The presence of the loads causes a downward vertical displacement  $\Delta$  at the mid span of the structure.  $\Delta$  is taken to be the system’s uncertain response to the 10-dimensional random input  $\mathbf{X} = (E_1, E_2, A_1, A_2, P_1, \dots, P_6)$  consisting of the 4 structural variables and the 6 loads. The true statistics (mean, standard deviation, PDF) of  $\Delta$  are obtained by MCS over  $10^7$  sample points, and are shown in the left panel of Figure 6.

We analyse the system with the same procedure undertaken for the Ishigami model: we build  $aPCEonX$  and  $lPCEonZ$  on each of 10 training sets  $(\mathcal{X}', \mathcal{Y}')$  of increasing size  $n'$ , and validate their performance. The pointwise error is evaluated on 10 validation sets  $(\mathcal{X}'', \mathcal{Y}'')$  of fixed size  $n'' = 10,000$ , while the statistical errors are determined by large resampling.

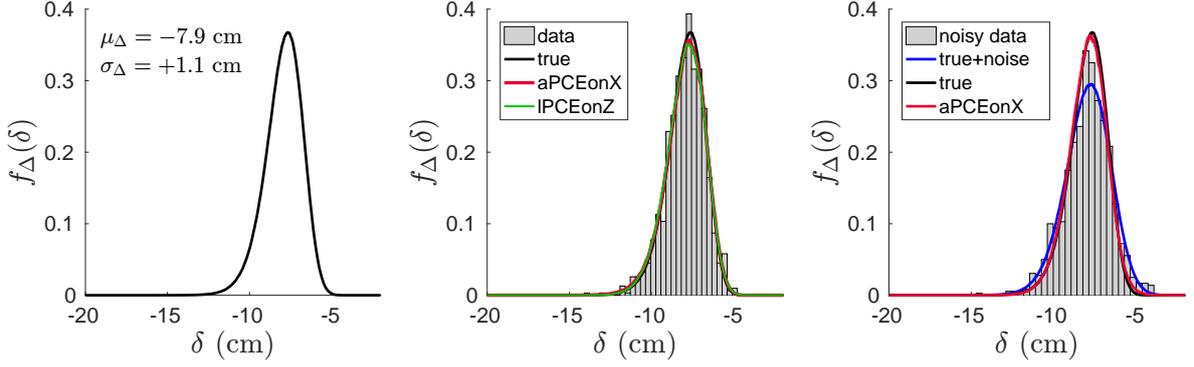


Figure 6: **Response PDFs of the horizontal truss.** *Left panel:* true PDF  $f_Y$  of the truss response, obtained on  $10^7$  sample points by KDE. *Central panel:* probability histogram obtained from  $n' = 1,000$  output observations used for training (gray bars), true response PDF as in the left panel (black), PDFs obtained from the *aPCEonX* (red) and the *lPCEonZ* (green) by resampling. *Right panel:* as in the central panel, but for training data perturbed with Gaussian noise ( $\sigma_\varepsilon = 0.79$  cm =  $0.70\sigma_\Delta$ ). The blue line indicates the true PDF of the perturbed model.

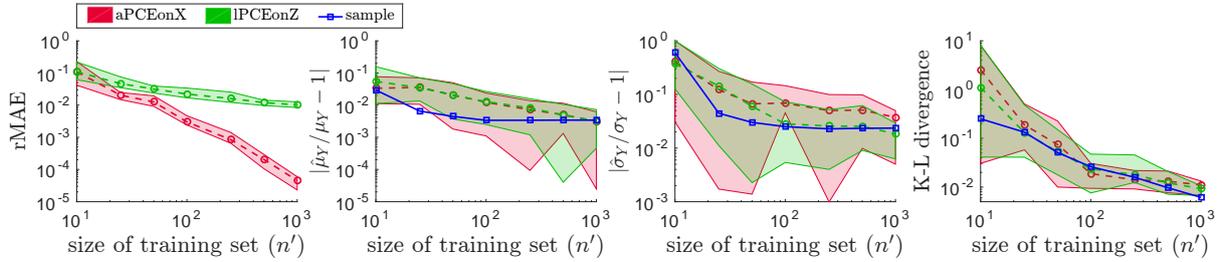


Figure 7: **PCE performance for the horizontal truss.** From left to right: rMAE, error on the mean, error on the standard deviation, and Kullback-Leibler divergence of *aPCEonX* (red) and *lPCEonZ* (green), for a size  $n'$  of the training set increasing from 10 to 1,000. The dash-dotted lines and the bands indicate, respectively, the average and the minimum to maximum errors over 10 simulations. In the second to fourth panels, the blue lines correspond to the empirical estimates obtained from the training data (error bands not shown).

The results are shown in Figure 7. Both PCEs exhibit high performance, yet the *aPCEonX* yields a significantly smaller pointwise error (first panel). The *lPCEonZ* yields a better estimate of the standard deviation, yet the empirical estimates obtained from the training data are the most accurate ones in this case.

Having selected the *aPCEonX* as the better of the two metamodels, we further assess its performance in the presence of noise. We perturb the response values used to train the model by adding Gaussian noise with increasing standard deviation  $\sigma_\varepsilon$ , set to 1%, 10%, and 30% of  $|\mu_\Delta|$  (equivalent to 7%, 70%, and 210% of  $\sigma_\Delta$ , respectively). The results are shown in Figures 8-9. The errors of all types are significantly smaller than the magnitude of the added noise, and decrease with increasing sample size for all noise levels (Figure 8). Also, the PCE estimates are significantly better than the sample estimates (Figure 9; see also Figure 6, fourth panel).

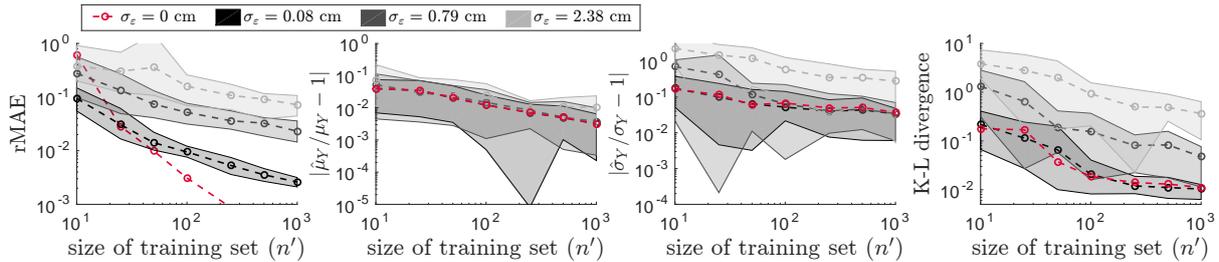


Figure 8: ***aPCEonX* of horizontal truss: robustness to noise (for multiple noise levels)**. From left to right: rMAE, error on the mean, error on the standard deviation, and Kullback-Leibler divergence of *aPCEonX* for an increasing amount of noise:  $\sigma_\varepsilon = 0.079$  cm (dark gray),  $\sigma_\varepsilon = 0.79$  cm (mid-light gray), and  $\sigma_\varepsilon = 2.38$  cm (light gray). The dash-dotted lines and the bands indicate, respectively, the average and the minimum to maximum error over 10 simulations. The red lines, reported from Figure 7 for reference, indicate the error for the noise-free data.

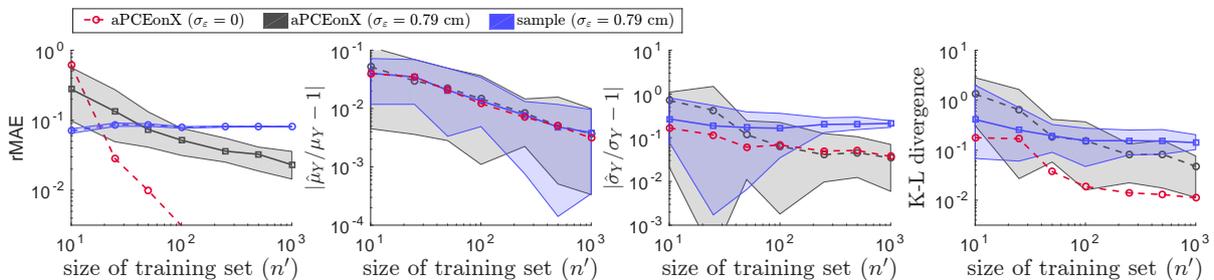


Figure 9: ***aPCEonX* of horizontal truss: robustness to noise (w.r.t. sample estimation)**. From left to right: rMAE, error on the mean, error on the standard deviation, and Kullback-Leibler divergence obtained by *aPCEonX* (gray) and by direct sample estimation (blue), for noise  $\sigma_\varepsilon = 0.79$  cm  $= 0.7 \sigma_Y$ . The dash-dotted lines and the bands indicate, respectively, the average and the minimum to maximum error over 10 simulations. The red lines, reported from Figure 2 for reference, indicate the mean error obtained for the noise-free data.

### 3.4 Regularization without orthonormalization

As discussed in the Section 2.3, PCE first determines a basis of polynomials which are mutually orthonormal with respect to the input PDF, then expresses the system response as a sum of such polynomials. The expansion coefficients are determined, in large dimension  $M$  and in the examples above, by sparse regression. Here in particular, LAR was selected as a method to solve sparse regression.

In the presence of correlations among the inputs, we further explored two strategies. The first one, *lPCEonZ*, transforms the inputs into statistically independent random variables, and then builds an orthonormal basis with respect to the new variables. The second strategy, *aPCEonX*, ignores the input dependencies (copula) and builds a basis orthonormal with respect to the product of their marginals. *aPCEonX* ultimately outperforms *lPCEonZ* in terms of pointwise accuracy, because it avoids highly non-linear transformations that make the expansion spectrum decay slower and thus the polynomial representation less efficient.

An even simpler alternative to *aPCEonX* would be to ignore not only the input copula, but its marginal distributions as well, and to use any non-orthonormal polynomial basis. Various parametric and non-parametric schemes exist to this end. We consider three different ones, which we apply to the Ishigami and horizontal truss models. This investigation allows us to quantify the benefit of basis orthonormality.

The first of such methods is LASSO Tibshirani (1996), which directly solves the problem (15). In standard LASSO the complete basis comprises all and only monomials of the type  $\prod_{i=1}^M x_i^{k_i}$ ,  $k_i \in \mathbb{N}^+$ . In practice though, the number of different monomials is reduced by neglecting interactions, that is, by considering only monomials of the type  $x_i^{k_i}$ . Further simplification is typically achieved by considering only the linear terms. It is clear that, in such settings, LASSO would miss the complex interactions that characterize both the Ishigami and the horizontal truss systems. Indeed, using the standard Matlab implementation of linear LASSO (with 10-fold cross-validation), we could not achieve a better accuracy than  $\text{rMAE} = 0.05$  for either case.

A recent improvement of LASSO is SALSA Kandasamy and Yu (2016). The method considers interactions among the input variables, and automatically selects the relevant terms in the expansion by an approach analogous to kernel ridge regression. We analyse the Ishigami and horizontal truss data with SALSA, using the Matlab implementation provided by the authors at [github.com/kirthevasank/salsa](https://github.com/kirthevasank/salsa). The results are illustrated in Figure 10. While performing overall better than LASSO, SALSA (cyan curve) fails to capture the highly non-linear behavior of the Ishigami function, and performs overall worse than *aPCEonX* (red) both on the Ishigami and the horizontal truss models.

The third attempt to solve sparse regression using a non-orthonormal basis is by performing PCE on the data  $\mathcal{X}$  using Legendre polynomials. We refer to this approach as *lPCEonX*. Legendre polynomials are the family of orthonormal polynomials with respect to uniform distributions in an interval  $[a, b]$ . We apply *lPCEonX* to the horizontal truss model only, as for the Ishigami model - whose inputs are truly uniform - it would correspond to the non-interesting case of assuming the correct marginals. We obtain the interval  $[a, b]$  for each input variable from the training data, as the observation range enlarged by 1% both to the left and to the right. The results are shown in Figure 10, right, orange curve. *lPCEonX* performs similarly to SALSA, and consistently worse than *aPCEonX*.

These comparisons demonstrate numerically the benefit of performing regression using an orthonormal basis, which ensures, due to finite variance of the response, better compression and thus more rapid convergence.

### 3.5 Preliminary conclusion

The results obtained in the previous section allow us to draw some important preliminary conclusions on data-driven PCE. The methodology:

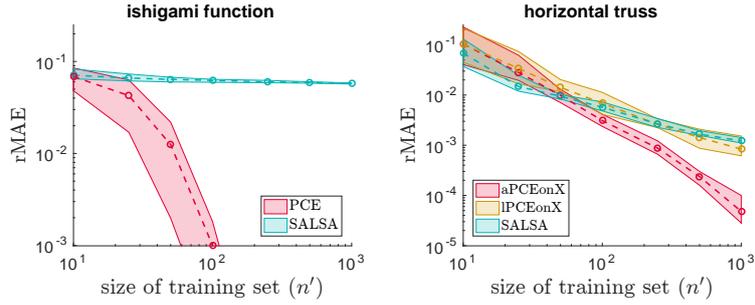


Figure 10: **PCE vs SALSA analysis of Ishigami and horizontal truss data.** rMAE of *aPCEonX* (red) vs SALSA (cyan) on the Ishigami (left) and horizontal truss (right) data, for increasing training set size.

- delivers reliable predictions of the system response to multivariate inputs;
- produces reliable estimates of the response statistics if the input dependencies are properly modelled, as done here through copulas (for *aPCEonX*: a-posteriori);
- works well already when trained on few observations;
- deals effectively with noise, thus providing a tool for denoising;
- involves only few hyperparameters, which have a clear interpretation and are easily tuned by cross-validation;
- outperforms sparse regression schemes based on non-orthonormal bases, such as LASSO Tibshirani (1996), SALSA Kandasamy and Yu (2016), or *lPCEonX* (see previous section).

In order to build the expansion when the inputs are mutually dependent, we investigated two alternative approaches, labelled as *lPCEonZ* and *aPCEonX*. Of the two strategies, *aPCEonX* appears to be the most effective one in purely data-driven problems. It is worth mentioning, though, that *lPCEonZ* may provide superior statistical estimates if the joint distribution of the input is known with greater accuracy than in the examples shown here. This was the case, for instance, when we replaced the inferred marginals and copula used to build the *lPCEonZ* with the true ones (not shown here): in both examples above, we obtained more accurate estimates of  $\mu_Y$ ,  $\sigma_Y$ , and  $F_Y$  (but not better pointwise predictions) than using *aPCEonX*.

The simulation results show a mismatch between the fast speed of MAE convergence and the slower convergence speed of the other metrics. Such mismatch is explained by the data-driven setup. To see this, let us start from an extreme example. Consider the ideal case where the true model is an exact polynomial. Inferring the input joint PDF  $f_{\mathbf{X}}$  from data (first step of the analysis) will generally lead to an inferred pdf  $\hat{f}_{\mathbf{X}} \neq f_{\mathbf{X}}$ . The bases associated to  $f_{\mathbf{X}}$  and  $\hat{f}_{\mathbf{X}}$  will thus also differ. Nevertheless, performing regression on the basis associated to  $\hat{f}_{\mathbf{X}}$  may still lead to an exact expansion, that is, to a PCE model that is

identical to the true polynomial. In this ideal scenario, the MAE would be 0. Yet, estimating the output statistics would involve resampling input points from the wrong pdf  $\hat{f}_{\mathbf{X}}$ . The error would be propagated by the correct PCE, and yield positive errors in the statistical estimates despite the PCE model being exact. The numerical examples shown here are not so extreme: the original model  $\mathcal{M}$  is not a polynomial and the obtained PCE only approximates  $\mathcal{M}$ . Still, it is a good approximation, in the sense that the resulting MAE is very small even if the wrong input PDF (and therefore a basis which is not exactly orthonormal) is used. The output statistics instead have a larger estimation error, because their accuracy is more strongly affected by errors in the inferred input distribution.

## 4 Results on real data sets

We now demonstrate the use of *aPCEonX* on three different real data sets. The selected data sets were previously analysed by other authors with different machine learning algorithms, which establish here the performance benchmark.

### 4.1 Analysis workflow

#### 4.1.1 Statistical input model

The considered data sets comprise samples made of multiple input quantities and one scalar output. Adopting the methodology outlined in Section 2, we characterize the multivariate input  $\mathbf{X}$  statistically by modelling its marginals  $\hat{f}_i$  through KDE, and we then resort to arbitrary PCE to express the output  $Y$  as a polynomial of  $\mathbf{X}$ . The basis of the expansion thus consists of mutually orthonormal polynomials with respect to  $\prod_i \hat{f}_i(x_i)$ , where  $\hat{f}_i$  is the marginal PDF inferred for  $X_i$ .

#### 4.1.2 Estimation of pointwise accuracy

Following the pointwise error assessment procedure carried out in the original publications, for the case studies considered here we assess the method's performance by cross-validation. Standard  $k$ -fold cross-validation partitions the data  $(\mathcal{X}, \mathcal{Y})$  into  $k$  subsets, trains the model on  $k - 1$  of those (the training set  $(\mathcal{X}', \mathcal{Y}')$ ), and assesses the pointwise error between the model's predictions and the observations on the  $k$ -th one (the validation set  $(\mathcal{X}'', \mathcal{Y}'')$ ). The procedure is then iterated over all  $k$  possible combinations of training and validation sets. The final error is computed as the average error over all validation sets. The number  $k$  of data subsets is chosen as in the reference studies. Differently from the synthetic models considered in the previous section, the true statistics of the system response are not known here, and the error on their estimates cannot be assessed.

A variation on standard cross-validation consists in performing a  $k$ -fold cross validation on each of multiple random shuffles of the data. The error is then typically reported as the

average error obtained across all randomisations, ensuring that the final results are robust to the specific partitioning of the data in its  $k$  subsets. In the following, we refer to a  $k$ -fold cross validation performed on  $r$  random permutations of the data (i.e.  $r$  random  $k$ -fold partitions) as an  $r \times k$ -fold randomised cross-validation.

### 4.1.3 Statistical estimation

Finally, we estimate for all cases studies the response PDF a-posteriori (AP) by resampling. To this end, we first model their dependencies through a C-vine copula  $\hat{C}^{(\mathcal{V})}$ . The vine is inferred from the data as detailed in A.3. Afterwards, resampling involves the following steps:

- sample  $n_{\text{AP}}$  points  $\mathcal{Z}_{\text{AP}} = \{\mathbf{z}^{(l)}, l = 1, \dots, n_{\text{AP}}\}$  from  $\mathbf{Z} \sim U([0, 1]^d)$ . We opt for Sobol’ quasi-random low-discrepancy sequences Sobol’ (1967), and set  $n_{\text{AP}} = 10^6$ ;
- map  $\mathcal{Z}_{\text{AP}} \mapsto \mathcal{U}_{\text{AP}} \subset [0, 1]^d$  by the inverse Rosenblatt transform of  $\hat{C}^{(\mathcal{V})}$ ;
- map  $\mathcal{U}_{\text{AP}} \mapsto \mathcal{X}_{\text{AP}}$  by the inverse probability integral transform of each marginal CDF  $\hat{F}_i$ .  $\mathcal{X}_{\text{AP}}$  is a sample set of input observations with copula  $\hat{C}^{(\mathcal{V})}$  and marginals  $\hat{F}_i$ ;
- evaluate the set  $\mathcal{Y}_{\text{AP}} = \{y_{\text{PC}}^{(l)} = \mathcal{M}_{\text{PC}}(\mathbf{x}), \mathbf{x} \in \mathcal{X}_{\text{AP}}\}$  of responses to the inputs in  $\mathcal{X}_{\text{AP}}$ .

The PDF of  $Y$  is estimated on  $\mathcal{Y}_{\text{AP}}$  by kernel density estimation.

## 4.2 Combined-cycle power plant

The first real data set we consider consists of 9,568 data points collected from a combined-cycle power plant (CCPP) over 6 years (2006-2011). The CCPP generates electricity by gas and steam turbines, combined in one cycle. The data comprise 4 ambient variables and the energy production  $E$ , measured over time. The four ambient variables are the temperature  $T$ , the pressure  $P$  and the relative humidity  $H$  measured in the gas turbine, and the exhaust vacuum  $V$  measured in the steam turbine. All five quantities are hourly averages. The data are available online Lichman (2013).

The data were analysed in Tüfekci (2014) with 13 different ML techniques, including regression trees, different types of neural networks (NNs), and support vector regression (SVR), to predict the energy output based on the measured ambient variables. The authors assessed the performance of each method by  $5 \times 2$ -fold randomised cross-validation, yielding a total of 10 pairs of training and validation sets. Each set contained 4,784 instances. The best learner among those tested by the authors was a bagging reduced error pruning (BREP) regression tree. Specifically, the model was obtained as the average response (“bagging”) of 10 pruned regression trees, each one built on a subset of the full data obtained by bootstrapping. Each pruned regression tree was obtained as a regression tree that was then iteratively simplified by reduced error pruning. The final model yielded a mean MAE of 3.22 MWh (see their Table 10, row 4). The lowest MAE of this model over the 10 validation sets,

	MAE	min. MAE	mean-min	rMAE (%)
<i>aPCEonX</i>	$3.11 \pm 0.03$	3.05	0.06	$0.68 \pm 0.007$
BREP Tüfekci (2014)	$3.22 \pm \text{n.a.}$	2.82	0.40	n.a.

Table 1: **Errors on CCPP data.** MAE yielded by the *aPCEonX* (first row) and by the BREP regression tree in Tüfekci (2014) (second row). From left to right: average MAE  $\pm$  its standard deviation over all 10 validation sets (in MWh), its minimum (error on the “best set”), difference between the average and the minimum MAEs, and rMAE.

corresponding to the “best” validation set, was indicated to be 2.82 MWh. Besides providing an indicative lower bound to the errors, the minimum gives, when compared to the means, an indication of the variability of the performance over different partitions of the data. The actual error variance over the 10 validation sets was not provided in the mentioned study.

We analyze the very same 10 training sets by PCE. The results are reported in Table 1. The average MAE yielded by the *aPCEonX* is slightly smaller than that of the BREP model. More importantly, the difference between the average and the minimum error, calculated over the 10 validation sets, is significantly lower with our approach, indicating a lower sensitivity of the results to the partition of the data, and therefore a higher reliability in the presence of random observations. The average error of the PCE predictions relative to the observed values is below 1%.

Finally, we estimate the PDF of the hourly energy produced by the CCPP following the procedure described in Section 4.1.3. The results are shown in Figure 11. Reliable estimates of the energy PDF aid for instance energy production planning and management.

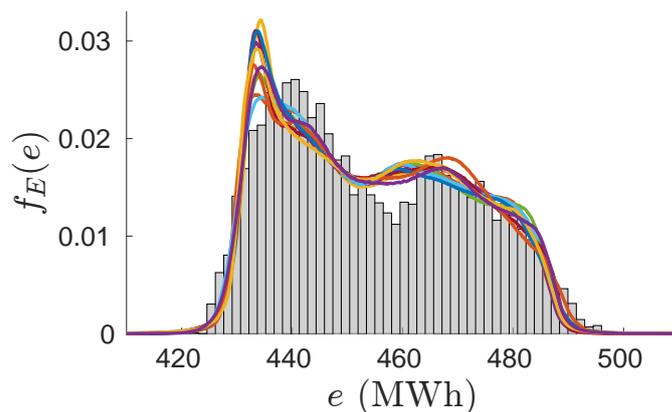


Figure 11: **Estimated PDF of the energy produced by the CCPP.** The bars indicate the histogram obtained from the observed CCPP energy output. The coloured lines show the PDFs of the PCE metamodells built on the 10 training sets, for input dependencies modelled by C-vines.

### 4.3 Boston Housing

The second real data set used to validate the PCE-based ML method concerns housing values in the suburbs of Boston, collected in 1970. The data set, downloaded from Lichman (2013), was first published in Harrison and Rubinfeld (1978), and is a known reference in the machine learning and data mining communities.

The data comprise 506 instances, each having 14 attributes. One attribute (the proximity of the neighborhood to the Charles river) is binary-valued and is therefore disregarded in our analysis. Of the remaining 13 attributes, one is the median housing value of owner-occupied homes in the neighbourhood, in thousands of \$ (MEDV). The remaining 12 attributes are, in order: the per capita crime rate by town (CRIM), the proportion of residential land zones for lots over 25,000 sq.ft. (ZN), the proportion of non-retail business acres per town (INDUS), the nitric oxides concentration, in parts per 10 million (NOX), the average number of rooms per dwelling (RM), the proportion of owner-occupied units built prior to 1940 (AGE), the weighted distances to five Boston employment centres (DIS), the index of accessibility to radial highways (RAD), the full-value property-tax rate per \$10,000 (TAX), the pupil-teacher ratio by town (PTRATIO), the index  $1,000(\text{Bk} - 0.63)^2$ , where Bk is the proportion of black residents by town, and the lower status of the population (LSTAT).

The data were analysed in previous studies with different regression methods to predict the median house values MEDV on the basis of the other attributes Can (1992); Gilley and Pace (1995); Quinlan (1993); R Kelley Pace (1997). The original publication itself Harrison and Rubinfeld (1978) was concerned with determining whether the demand for clean air affected housing prices. The data were analysed with different supervised learning methods in Quinlan (1993). Among them, the best predictor was shown to be an NN model combined with instance-based learning. The NN consisted of a single hidden layer and an output unit. The weights were regularised by a penalty coefficient. The number of hidden units and the penalty coefficients were first optimised by three-fold cross validation on the first training set, and each weight was then finally selected as the best of 3 values randomly sampled within  $[-0.3, +0.3]$ . The final NN yielded  $\text{MAE} = 2,230\$$  (rMAE: 12.9%) on a 10-fold cross-validation.

We model the data by PCE and quantify the performance by  $10 \times 10$  randomised cross-validation. The results are summarised in Table 2. The errors are comparable to the NN model with instance-based learning in Quinlan (1993). While the latter yields the lowest absolute error, the *aPCEonX* achieves a smaller relative error. In addition, it does not require the fine parameter tuning that affects most NN models. Finally, we estimate the PDF of the median house value as described in Section 4.1.3. The results are shown in Figure 12.

	MAE (\$)	rMAE (%)
<i>aPCEonX</i>	$2483 \pm 337$	$12.6 \pm 2.0$
NN Quinlan (1993)	$2230 \pm \text{n.a.}$	$12.9 \pm \text{n.a.}$

Table 2: **Errors on Boston Housing data.** MAE and rMAE yielded by the *aPCEonX* (first row) and by the NN model with instance-based learning from Quinlan (1993) (second row).

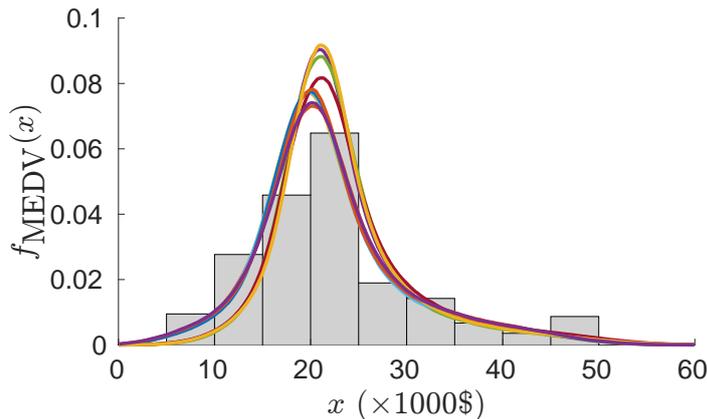


Figure 12: **Estimated PDF of the variable MEDV.** The bars indicate the sample PDF, as an histogram obtained using 50 bins from \$0 to \$50k (the maximum house value in the data set). The coloured lines show the PDFs of the PCE metamodells built on 10 of the 100 training sets (one per randomisation of the data), for input dependencies modelled by C-vines. The dots indicate the integrals of the estimated PDFs for house values above \$49k.

#### 4.4 Wine quality

The third real data set we consider concerns the quality of wines from the *vinho verde* region in Portugal. The data set consists of 1,599 red samples and 4,898 white samples, collected between 2004 and 2007. The data are available online at <http://www3.dsi.uminho.pt/pcortez/wine/>. Each wine sample was analysed in laboratory for 11 physico-chemical parameters: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol. In addition, each sample was given a quality score  $Q$  based on blinded sensory tests from three or more sensory assessors. The score is the median of the grades (integers between 0 and 10) assigned by each assessor.

The data were previously analysed in Cortez et al. (2009) to predict the wine quality score on the basis of the 11 physico-chemical parameters. The study compared various ML algorithms, namely multiple regression, a single-layer NN, and linear support vector machine (SVM, Schölkopf and Smola (2002)) regression. The NN consisted of a single hidden layer with logistic transfer function and an output node with linear function. The number of hidden nodes and the weights were automatically calibrated by cross-validation. For linear

SVM regression, a Gaussian kernel was selected, with kernel width tuned by cross-validation. The remaining parameters were fixed to arbitrary values to reduce the computational cost. SVM regression outperformed the other methods (at the expense of a considerably higher computational cost), yielding the lowest MAE, as assessed by means of  $20 \times 5$ -fold randomised cross-validation.

We model the data by *aPCEonX*, and round the predicted wine ratings, which take continuous predicted values, to their closest integer. The performance is then assessed through the same cross-validation procedure used in Cortez et al. (2009). The results are reported in Table 3. The MAE of *aPCEonX* is comparable to that of the SVM regressor, and always lower than the best NN.

	red wine:		white wine:	
	MAE	rMAE (%)	MAE	rMAE (%)
<i>aPCEonX</i>	$0.44 \pm 0.03$	$8.0 \pm 0.6$	$0.50 \pm 0.02$	$8.8 \pm 0.3$
SVM in Cortez et al. (2009)	$0.46 \pm 0.00$	n.a.	$0.45 \pm 0.00$	n.a.
Best NN in Cortez et al. (2009)	$0.51 \pm 0.00$	n.a.	$0.58 \pm 0.00$	n.a.

Table 3: **Errors on wine data.** MAE and rMAE yielded on red and white wine data by the *aPCEonX*, by the SVM in Cortez et al. (2009), and by the best NN model in Cortez et al. (2009).

Finally, our framework enables the estimation of the PDF of the wine rating as predicted by the PCE metamodels. The resulting PDFs are shown in Figure 13. One could analogously compute the conditional PDFs given by fixing any subset of inputs to given values (*e.g.*, the residual sugar or alcohol content, which can be easily controlled in the wine making process). This may help, for instance, predicting the wine quality for fixed physico-chemical parameters, or choosing the latter so as to optimize the wine quality or to minimize its uncertainty. This analysis goes beyond the scope of the present work.

## 5 Discussion and conclusions

We proposed an approach to machine learning (ML) that capitalises on polynomial chaos expansion (PCE), an advanced regression technique from uncertainty quantification. PCE is a popular spectral method in engineering applications, where it is often used to replace expensive-to-run computational models subject to uncertain inputs with an inexpensive metamodel that retains the statistics of the output (*e.g.*, moments, PDF). Our paper shows that PCE can also be used as an effective regression model in purely data-driven problems, where only input observations and corresponding system responses - but no computational

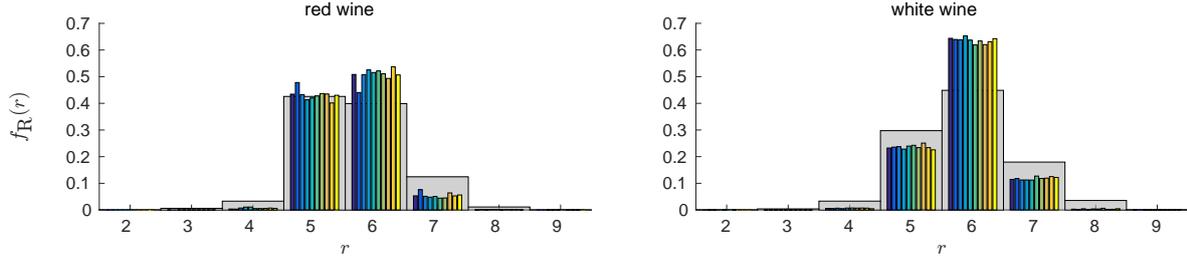


Figure 13: **Estimated PDF of the wine rating.** For each panel (left: red wine; right: white wine), the grey bars indicate the sample PDF of the median wine quality score assigned by the assessors. The coloured bars show the predicted PDFs obtained by resampling from the PCE metamodels built on 10 of the 100 total training sets, for input dependencies modelled by C-vines.

model of the system - are available. This setup poses various challenges that PCE does not have to deal with in classical uncertainty quantification problems. Namely, the training set is fixed and cannot be enriched, the input joint distribution (needed to build the spectral basis and to draw statistical estimates) is unknown and has to be entirely inferred from the available data, and the data may be noisy.

We tested the performance of PCE on simulated data first, and then on real data by cross-validation. The reference performance measure was the mean absolute error of the PCE metamodel over all test data. The simulations also allowed us to assess the ability of PCE to estimate the statistics of the response (its mean, standard deviation, and PDF) in the considered data-driven scenario. Both the pointwise and the statistical errors of the methodology were low, even when relatively few observations were used to train the model. Importantly, high performance was still obtained in the presence of strong noise in the simulated data. PCE can thus be used for denoising, a feature that had not been previously investigated. Finally, the numerical experiments demonstrated the superiority of PCE, which uses as expansion basis the set of orthonormal polynomials with respect to the product of the input marginals, over methods that perform sparse regression over a non-orthonormal basis, such as LASSO Tibshirani (1996), SALSA Kandasamy and Yu (2016), or PCE built enforcing a different basis.

The applications to real data showed a comparable, and sometimes slightly superior, performance to that of other ML methods used in previous studies, such as different types of neural networks and support vector machines. In general, the quality of the approximations will depend on the magnitude of the truncation error, which is dictated by the speed of the decay of the spectrum. The amount of data available also limits in practice the attainable expansion order, as already observed in Oladyshkin and Nowak (2018).

PCE offers several advantages over other established ML regression techniques. First, the framework performs well on very different tasks, with only little parameter tuning needed to adapt the methodology to the specific data considered. The quadratic nature of the PCE

regression problem enables direct optimisation, compared for instance to the cumbersome iterative training/calibration procedure in NNs. For PCE, only the total degree  $p$ , the  $q$ -norm parameter and the interaction degree  $r$  are to be specified. Automatic calibration of these parameters within pre-assigned ranges is straightforward. Specifically, as a single analysis takes only a few seconds to a minute to be completed on a standard laptop (depending on the size of the data), it is possible to repeat the analysis over an array of allowed  $(p, q, r)$  values, and to retain the PCE with minimum error in the end. In our analyses we tuned the parameters  $p$  and  $r$  in this way, while setting  $q = 0.75$  (not optimized, in order to reduce the total computational cost). This feature distinguishes PCE from the above mentioned ML methods, which instead are known to be highly sensitive to their hyperparameters and require an appropriate and typically time consuming calibration Claesen and Moor (2015). Indeed, it is worth noting that, in the comparisons we made, all PCE metamodels were built by using the very same procedure, including the automatic hyperparameter calibration. When compared to the best NNs or SVM found in other studies, which differed significantly from each other in their construction and structure, the PCE metamodels exhibited a comparable performance.

Second, PCE delivers not only accurate pointwise predictions of the output for any given input, but also statistics thereof in the presence of input uncertainties. This is made possible by combining the PCE metamodel with a proper probabilistic characterisation of the input uncertainties through marginal distributions and copulas, and then by resampling from the obtained input model. The result is not trivial: theory motivates the spectral convergence of PCE in classical UQ settings (see Section 2.2). In the data-driven setup considered here, however, the joint PDF of the input is unknown and has to be inferred (see also Section 3.5), a step which generally harms convergence. As a further difficulty, we addressed the worst-case scenario where parametric inference is not possible due to missing information, and proceeded instead by kernel density estimation. Despite these challenges, quite accurate statistics were obtained. The methodology works well also in the presence of several inputs (as tested on simulated problems of dimension up to 10) and of sample sets of comparably small size. It is worth mentioning in this respect that the same resampling approach could be applied in combination with any technique for function approximation (including neural networks and support vector machines). In the particular case of independent inputs, the PCE representation encodes the moments of the output and its Sobol sensitivity indices directly in the polynomial coefficients Sudret (2008).

Third, the analytical expression of the output yielded by PCE in terms of a simple polynomial of the input makes the model easy to interpret, compared for instance to other popular methods for ML regression.

Fourth, the polynomial form makes the calibrated metamodel portable to embedded devices (*e.g.*, drones). For this kind of applications, the demonstrated robustness to noise in

the data is a particularly beneficial feature.

Fifth and last, PCE needs relatively few data points to attain acceptable performance levels, as shown here on various test cases. This feature demonstrates the validity of PCE metamodelling for problems affected by data scarcity, also when combined with complex vine copula representations of the input dependencies.

One limitation of PCE-based regression as presented here is its difficulty in dealing with data of large size or consisting of a very large number of inputs. Both features lead to a substantially increased computational cost needed to fit the PCE parameters and (if statistical estimation is wanted and the inputs are dependent) to infer the copula. Various solutions can be nevertheless envisioned. In the presence of very large training sets, the PCE may be initially trained on a subset of the available observations, and subsequently refined by enriching the training set with points in the region where the observed error is larger. Regarding copula inference, which is only needed for an accurate quantification of the prediction's uncertainty, a possible solution is to employ a Gaussian copula. The latter involves a considerably faster fitting than the more complex vine copulas, and still yielded in our simulations acceptable performance. Alternatively, one may reduce the computational time needed for parameter estimation by parallel computing, as done in Wei et al. (2016). Recently, a technique to efficiently infer vine copulas in high-dimensional problems has been proposed in Müller and Czado (2018).

Finally, the proposed methodology has been shown here on data characterised by continuous input variables only. PCE construction in the presence of discrete data is equally possible, and the Stiltjes orthogonalisation procedure is known to be quite stable in that case Gautschi (1982). The a-posteriori quantification of the output uncertainty, however, generally poses a challenge. Indeed, it involves the inference of a copula among discrete random variables, which requires a different construction Genest and Nešlehová (2007). Recently, however, methods have been proposed to this end, including inference for R-vines Panagiotelis et al. (2012, 2017). Further work is foreseen to integrate these advances with PCE metamodelling.

## Acknowledgments

Emiliano Torre gratefully acknowledges financial support from RiskLab, Department of Mathematics, ETH Zurich and from the Risk Center of the ETH Zurich. Declarations of interest: none.

## References

Aas, K. (2016). Pair-copula constructions for financial applications: A review. *Econometrics* 4(4), 43.

- Aas, K., C. Czado, A. Frigessi, and H. Bakken (2009). Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics* 44(2), 182–198.
- Applegate, D. L., R. E. Bixby, V. Chvátal, and W. J. Cook (2006). *The Traveling Salesman Problem: A Computational Study*. New Jersey: Princeton University Press.
- Bedford, T. and R. M. Cooke (2002). Vines – a new graphical model for dependent random variables. *The Annals of Statistics* 30(4), 1031–1068.
- Bishop, C. (2009). *Pattern recognition and machine learning*. Springer.
- Blatman, G. and B. Sudret (2011). Adaptive sparse polynomial chaos expansion based on Least Angle Regression. *J. Comput. Phys* 230, 2345–2367.
- Can, A. (1992). Specification and estimation of hedonic housing price models. *Regional Science and Urban Economics* 22(3), 453–474.
- Chan, S. and A. H. Elsheikh (2018). A machine learning approach for efficient uncertainty quantification using multiscale methods. *Journal of Computational Physics* 354, 493–511.
- Claesen, M. and B. D. Moor (2015). Hyperparameter search in machine learning. *arXiv 1502.02127*.
- Cortez, P., A. Cerdeira, F. Almeida, T. Matos, and J. Reis (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems* 47, 547–553.
- Czado, C. (2010). *Pair-Copula Constructions of Multivariate Copulas*, pp. 93–109. Berlin, Heidelberg: Springer Berlin Heidelberg.
- Dißmann, J., E. C. Brechmann, C. Czado, and D. Kurowicka (2013). Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics and Data Analysis* 59, 52–69.
- Efron, B., T. Hastie, I. Johnstone, and R. Tibshirani (2004). Least angle regression. *Annals of Statistics* 32, 407–499.
- Eldred, M., C. Webster, and P. Constantine (2008). Evaluation of non-intrusive approaches for Wiener-Askey generalized polynomial chaos. *AIAA 2008-1892* 46.
- Ernst, O. G., A. Mugler, H.-J. Starkloff, and E. Ullmann (2012). On the convergence of generalized polynomial chaos expansions. *ESAIM: M2AN* 46(2), 317–339.
- Forman, G. and I. Cohen (2004). Learning from little: Comparison of classifiers given little training. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi (Eds.), *Knowledge Discovery in Databases: PKDD 2004*, Berlin, Heidelberg, pp. 161–172. Springer Berlin Heidelberg.

- Gautschi, W. (1982). On generating orthogonal polynomials. *SIAM J. Sci. Stat. Comput.* 3(3), 289–317.
- Genest, C. and J. Nešlehová (2007). A primer on copulas for count data. *The Astin Bulletin* 37, 475–515.
- Ghanem, R. and P.-D. Spanos (1990). Polynomial chaos in stochastic finite elements. *J. Applied Mech.* 57, 197–202.
- Gilley, O. W. and R. K. Pace (1995). Improving hedonic estimation with an inequality restricted estimator. *The Review of Economics and Statistics* 77(4), 609–621.
- Haff, I. H., K. Aas, and A. Frigessi (2010). On the simplified pair-copula construction – simply useful or too simplistic? *Journal of Multivariate Analysis* 101, 1296–1310.
- Harrison, D. and D. L. Rubinfeld (1978). Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management* 5, 81–102.
- Ishigami, T. and T. Homma (1990). An importance quantification technique in uncertainty analysis for computer models. In *Proc. ISUMA '90, First Int. Symp. Unc. Mod. An.*, pp. 398–403. University of Maryland.
- Joe, H. (1996). Families of  $m$ -variate distributions with given margins and  $m(m-1)/2$  bivariate dependence parameters. In L. Rüschendorf, B. Schweizer, and M. D. Taylor (Eds.), *Distributions with fixed marginals and related topics*, Volume 28 of *Lecture Notes–Monograph Series*, pp. 120–141. Institute of Mathematical Statistics.
- Joe, H. (Ed.) (2015). *Dependence modeling with copulas*. CRC Press.
- Kandasamy, K. and Y. Yu (2016). Additive approximations in high dimensional nonparametric regression via the SALSA. Volume 48, pp. 69–78.
- Kasiviswanathan, K. S., K. P. Sudheer, and J. He (2016). *Quantification of Prediction Uncertainty in Artificial Neural Network Models*, pp. 145–159. Cham: Springer International Publishing.
- Kirk, J. (2014). Traveling salesman problem – genetic algorithm. <https://ch.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm>.
- Kurowicka, D. and R. M. Cooke (2005). Distribution-free continuous Bayesian belief nets. In A. Wilson, N. Limnios, S. Keller-McNulty, and Y. Armijo (Eds.), *Modern Statistical and Mathematical Methods in Reliability*, Chapter 10, pp. 309–322. World Scientific Publishing.
- Kurz, M. (2015). Vine copulas with matlab. <https://ch.mathworks.com/matlabcentral/fileexchange/46412-vine-copulas-with-matlab>.

- Lebrun, R. and A. Dutfoy (2009). Do Rosenblatt and Nataf isoprobabilistic transformations really differ? *Prob. Eng. Mech.* 24, 577–584.
- Lichman, M. (2013). UCI machine learning repository.
- Mentch, L. and G. Hooker (2016). Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *Journal of Machine Learning Research* 17, 1–41.
- Morales-Nápoles, O. (2011). Counting vines. In D. Kurowicka and H. Joe (Eds.), *Dependence Modeling: Vine Copula Handbook*, Chapter 9, pp. 189–218. World Scientific Publisher Co.
- Müller, D. and C. Czado (2018). Selection of sparse vine copulas in high dimensions with the Lasso. *Statistics and Computing*, 1–19.
- NAVARRO, M., J. WITTEVEEN, and J. BLOM (2014). Polynomial chaos expansion for general multivariate distributions with correlated variables. *ArXiv 1406.5483v1*.
- Nelsen, R. (2006). *An introduction to copulas* (Second ed.). Springer Series in Statistics. Springer-Verlag New York.
- Oladyshkin, S. and W. Nowak (2012). Data-driven uncertainty quantification using the arbitrary polynomial chaos expansion. *Reliability Engineering and System Safety* 106, 179–190.
- Oladyshkin, S. and W. Nowak (2018). Incomplete statistical information limits the utility of high-order polynomial chaos expansions. *Reliability Engineering & System Safety* 169, 137–148.
- Panagiotelis, A., C. Czado, and H. Joe (2012). Pair copula constructions for multivariate discrete data. *J. Amer. Statist. Assoc.* 107, 1063–1072.
- Panagiotelis, A., C. Czado, H. Joe, and J. Stöber (2017). Model selection for discrete regular vine copulas. *Computational Statistics & Data Analysis* 106, 138–152.
- Parzen, E. (1962). On estimation of a probability density function and mode. *The Annals of Mathematical Statistics* 33(3), 1065–1076.
- Paulson, J. A., E. A. Buehler, and A. Mesbah (2018). Arbitrary polynomial chaos for uncertainty propagation of correlated random variables in dynamic systems. *IFAC-PapersOnLine* 50(1), 3548–3553.
- Puig, B., F. Poirion, and C. Soize (2002). Non-Gaussian simulation using Hermite polynomial expansion: convergences. *Prob. Eng. Mech.* 17, 253–264.
- Quinlan, J. R. (1993). Combining instance-based and model-based learning. In *Proceedings on the Tenth International Conference of Machine Learning*, pp. 236–243.

- R Kelley Pace, O. W. G. (1997). Using the spatial configuration of the data to improve estimation. *Journal of Real Estate Finance and Economics* 14(3), 333–340.
- Rosenblatt, M. (1952). Remarks on a multivariate transformation. *The Annals of Mathematical Statistics* 23, 470–472.
- Rosenblatt, M. (1956). Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics* 27(3), 832–837.
- Saltelli, A., K. Chan, and E. Scott (Eds.) (2000). *Sensitivity analysis*. J. Wiley & Sons.
- Schepsmeier, U. (2015). Efficient information based goodness-of-fit tests for vine copula models with fixed margins. *Journal of Multivariate Analysis* 138(C), 34–52.
- Schölkopf, B. and A. Smola (2002). *Learning with kernels*. MIT Press.
- Sklar, A. (1959). Fonctions de répartition à  $n$  dimensions et leurs marges. *Publications de l'Institut de Statistique de L'Université de Paris* 8, 229–231.
- Snoek, J., H. Larochelle, and R. P. Adams (2012). Practical Bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, USA, pp. 2951–2959. Curran Associates Inc.
- Sobol', I. (1993). Sensitivity estimates for nonlinear mathematical models. *Mathematical Modeling & Computational Experiment* 1, 407–414.
- Sobol', I. M. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics* 7(4), 86–112.
- Soize, C. and R. Ghanem (2004). Physical systems with random uncertainties: chaos representations with arbitrary probability measure. *SIAM J. Sci. Comput.* 26(2), 395–410.
- Stöber, J., H. Joe, and C. Czado (2013). Simplified pair copula constructions — limitations and extensions. *Journal of Multivariate Analysis* 119, 101–118.
- Stuart, A. and K. Ord (1994). *Kendall's advanced theory of statistics Vol. 1 – Distribution theory*. Arnold.
- Sudret, B. (2008). Global sensitivity analysis using polynomial chaos expansions. *Reliab. Eng. Sys. Safety* 93, 964–979.
- Sudret, B., S. Marelli, and C. Lataniotis (2015). Sparse polynomial chaos expansions as a machine learning regression technique. International Symposium on Big Data and Predictive Computational Modeling, invited lecture.

- Tibshirani, R. (1996). Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society* 58(1), 267–288.
- Todor, R.-A. and C. Schwab (2007). Convergence rates for sparse chaos approximations of elliptic problems with stochastic coefficients. *IMA J. Numer. Anal* 27, 232–261.
- Torre, E., S. Marelli, P. Embrechts, and B. Sudret (2019). A general framework for data-driven uncertainty quantification under complex input dependencies using vine copulas. *Probabilistic Engineering Mechanics* 55, 1–16.
- Tüfekci, P. (2014). Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems* 60, 126–140.
- Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.
- Wan, X. and G. E. Karniadakis (2006). Beyond Wiener-Askey expansions: handling arbitrary PDFs. *J. Sci. Comput.* 27, 455–464.
- Wei, Z., D. Kim, and E. M. Conlon (2016). Parallel computing for copula parameter estimation with big data: A simulation study. *arXiv 1609.05530*.
- Wiener, N. (1938). The homogeneous chaos. *Amer. J. Math.* 60, 897–936.
- Witten, I. H., E. Frank, M. A. Hall, and C. J. Pal (2016). *Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques* (4th ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- Xiu, D. and G. Karniadakis (2002). The Wiener-Askey polynomial chaos for stochastic differential equations. *SIAM J. Sci. Comput.* 24(2), 619–644.

## A Mutually dependent inputs modelled through copulas

In Section 2.3 we illustrated PCE for an input  $\mathbf{Z}$  with independent components being uniformly distributed in  $[0, 1]$ .  $\mathbf{Z}$  was obtained from the true input  $\mathbf{X}$  by assuming either that the latter had independent components as well, and thus defining  $Z_i = F_i(X_i)$ , or that a transformation  $\mathcal{T}$  to perform this mapping was available.

This section recalls known results in probability theory allowing one to specify  $F_{\mathbf{X}}$  in terms of its marginal distributions and a dependence function called the *copula* of  $\mathbf{X}$ . We focus in particular on regular vine copulas (R-vines), for which the transformation  $\mathcal{T}$  can be computed numerically. R-vines provide a flexible class of dependence models for  $f_{\mathbf{X}}$ . This

will prove beneficial to the ability of the PCE models to predict statistics of the output accurately, compared to simpler dependence models. However, and perhaps counter-intuitively, the pointwise error may not decrease accordingly (see Section 2.1 for details), especially when  $\mathcal{T}$  is highly non-linear. Examples on simulated data and a discussion are provided in Section 3.

## A.1 Copulas and Sklar's theorem

A  $d$ -copula is defined as a  $d$ -variate joint CDF  $C : [0, 1]^d \rightarrow [0, 1]$  with uniform marginals in the unit interval, that is,

$$C(1, \dots, 1, u_i, 1, \dots, 1) = u_i \quad \forall u_i \in [0, 1], \quad \forall i = 1, \dots, d.$$

Sklar's theorem Sklar (1959) guarantees that any  $d$ -variate joint CDF can be expressed in terms of  $d$  marginals and a copula, specified separately.

**Theorem (Sklar).** *For any  $d$ -variate CDF  $F_{\mathbf{X}}$  with marginals  $F_1, \dots, F_d$ , a  $d$ -copula  $C_{\mathbf{X}}$  exists, such that*

$$F_{\mathbf{X}}(\mathbf{x}) = C_{\mathbf{X}}(F_1(x_1), F_2(x_2), \dots, F_d(x_d)). \quad (1)$$

Besides,  $C_{\mathbf{X}}$  is unique on  $\text{Ran}(F_1) \times \dots \times \text{Ran}(F_d)$ , where  $\text{Ran}$  is the range operator. In particular,  $C_{\mathbf{X}}$  is unique on  $[0, 1]^d$  if all  $F_i$  are continuous, and it is given by

$$C_{\mathbf{X}}(u_1, u_2, \dots, u_d) = F_{\mathbf{X}}(F_1^{-1}(u_1), F_2^{-1}(u_2), \dots, F_d^{-1}(u_d)). \quad (2)$$

Conversely, for any  $d$ -copula  $C$  and any set of  $d$  univariate CDFs  $F_i$ ,  $i = 1, \dots, d$ , the function  $F : \mathbb{R}^d \rightarrow [0, 1]$  defined by

$$F(x_1, x_2, \dots, x_d) := C(F_1(x_1), F_2(x_2), \dots, F_d(x_d)) \quad (3)$$

is a  $d$ -variate CDF with marginals  $F_1, \dots, F_d$ .

Throughout this study it is assumed that  $F_{\mathbf{X}}$  has continuous marginals  $F_i$ . The relation (3) allows one to model any multivariate CDF  $F$  by modelling separately  $d$  univariate CDFs  $F_i$  and a copula function  $C$ . One first models the marginals  $F_i$ , then transforms each  $X_i$  into a uniform random variable  $U_i = F_i(X_i)$  by the so-called probability integral transform (PIT)

$$\mathcal{T}^{\text{PIT}} : \mathbf{X} \mapsto \mathbf{U} = (F_1(X_1), \dots, F_d(X_d))^{\top}. \quad (4)$$

Finally, the copula  $C$  of  $\mathbf{X}$  is obtained as the joint CDF of  $\mathbf{U}$ . The copula models the dependence properties of the random vector. For instance, mutual independence is achieved by using the independence copula

$$C(\mathbf{u}) = \prod_{i=1}^d u_i. \quad (5)$$

Table A.4 provides a list of 19 different parametric families of pair copulas implemented in the VineCopulaMatlab toolbox Kurz (2015), which was also used in this study. Details on copula theory and on various copula families can be found in Nelsen (2006) and in Joe (2015).

Sklar's theorem can be re-stated in terms of probability densities. If  $\mathbf{X}$  admits joint PDF  $f_{\mathbf{X}}(\mathbf{x}) := \frac{\partial^d F_{\mathbf{X}}(\mathbf{x})}{\partial x_1 \dots \partial x_M}$  and copula density  $c_{\mathbf{X}}(\mathbf{u}) := \frac{\partial^d C_{\mathbf{X}}(\mathbf{u})}{\partial u_1 \dots \partial u_M}$ ,  $\mathbf{u} \in [0, 1]^d$ , then

$$f_{\mathbf{X}}(\mathbf{x}) = c(F_1(x_1), F_2(x_2), \dots, F_d(x_d)) \cdot \prod_{i=1}^d f_i(x_i). \quad (6)$$

Once all marginal PDFs  $f_i$  and the corresponding CDFs  $F_i$  have been determined (see Section 2.3.2), each data point  $\mathbf{x}^{(j)} \in \mathcal{X}$  is mapped onto the unit hypercube by the PIT (4), obtaining a transformed data set  $\mathbf{U}$  of *pseudo-observations* of  $\mathbf{U}$ . The copula of  $\mathbf{X}$  can then be inferred on  $\mathbf{U}$ .

## A.2 Vine copulas

In high dimension  $d$ , specifying a  $d$ -copula which properly describes all pairwise and higher-order input dependencies may be challenging. Multivariate extensions of pair-copula families (e.g. Gaussian or Archimedean copulas) are often inadequate when  $d$  is large. In Joe (1996) and later in Bedford and Cooke (2002) and Aas et al. (2009), an alternative construction by multiplication of 2-copulas was introduced. Copula models built in this way are called *vine copulas*. Here we briefly introduce the vine copula formalism, referring to the references for details.

Let  $\mathbf{u}_{\bar{i}}$  be the vector obtained from the vector  $\mathbf{u}$  by removing its  $i$ -th component, *i.e.*,  $\mathbf{u}_{\bar{i}} = (u_1, \dots, u_{i-1}, u_{i+1}, \dots, u_d)^\top$ . Similarly, let  $\mathbf{u}_{\overline{\{i,j\}}}$  be the vector obtained by removing the  $i$ -th and  $j$ -th component, and so on. For a general subset  $\mathcal{A} \subset \{1, \dots, d\}$ ,  $\mathbf{u}_{\overline{\mathcal{A}}}$  is defined analogously. Also,  $F_{\overline{\mathcal{A}}|\mathcal{A}}$  and  $f_{\overline{\mathcal{A}}|\mathcal{A}}$  indicate in the following the joint CDF and PDF of the random vector  $\mathbf{X}_{\overline{\mathcal{A}}}$  conditioned on  $\mathbf{X}_{\mathcal{A}}$ . In the following,  $\mathcal{A} = \{i_1, \dots, i_k\}$  and  $\overline{\mathcal{A}} = \{j_1, \dots, j_l\}$  form a partition of  $\{1, \dots, d\}$ , *i.e.*,  $\mathcal{A} \cup \overline{\mathcal{A}} = \{1, \dots, d\}$  and  $\mathcal{A} \cap \overline{\mathcal{A}} = \emptyset$ .

Using (6),  $f_{\overline{\mathcal{A}}|\mathcal{A}}$  can be expressed as

$$f_{\overline{\mathcal{A}}|\mathcal{A}}(\mathbf{x}_{\overline{\mathcal{A}}}|\mathbf{x}_{\mathcal{A}}) = c_{\overline{\mathcal{A}}|\mathcal{A}}(F_{j_1|\mathcal{A}}(x_{j_1}|\mathbf{x}_{\mathcal{A}}), F_{j_2|\mathcal{A}}(x_{j_2}|\mathbf{x}_{\mathcal{A}}), \dots, F_{j_l|\mathcal{A}}(x_{j_l}|\mathbf{x}_{\mathcal{A}})) \times \prod_{j \in \overline{\mathcal{A}}} f_{j|\mathcal{A}}(x_j|\mathbf{x}_{\mathcal{A}}), \quad (7)$$

where  $c_{\overline{\mathcal{A}}|\mathcal{A}}$  is an  $l$ -copula density – that of the conditional random variables  $(X_{j_1|\mathcal{A}}, X_{j_2|\mathcal{A}}, \dots, X_{j_l|\mathcal{A}})^\top$  – and  $f_{j|\mathcal{A}}$  is the conditional PDF of  $X_j$  given  $\mathbf{X}_{\mathcal{A}}$ ,  $j \in \overline{\mathcal{A}}$ . Following Joe (1996), the univariate conditional distributions  $F_{j|\mathcal{A}}$  can be further expressed in terms of any conditional pair copula  $C_{j_i|\mathcal{A} \setminus \{i\}}$  between  $X_{j|\mathcal{A} \setminus \{i\}}$  and  $X_{i|\mathcal{A} \setminus \{i\}}$ ,  $i \in \mathcal{A}$ :

$$F_{j|\mathcal{A}}(x_j|\mathbf{x}_{\mathcal{A}}) = \frac{\partial C_{j_i|\mathcal{A} \setminus \{i\}}(u_j, u_i)}{\partial u_i} \Big|_{(F_{j|\mathcal{A} \setminus \{i\}}(x_j|\mathbf{x}_{\mathcal{A} \setminus \{i\}}), F_{i|\mathcal{A} \setminus \{i\}}(x_i|\mathbf{x}_{\mathcal{A} \setminus \{i\}}))}. \quad (8)$$

An analogous relation readily follows for conditional densities:

$$\begin{aligned}
f_{j|\mathcal{A}}(x_j|\mathbf{x}_{\mathcal{A}}) &= \frac{\partial F_{j|\mathcal{A}}(x_j|\mathbf{x}_{\mathcal{A}})}{\partial x_j} \\
&= c_{j|\mathcal{A}\setminus\{i\}}(F_{j|\mathcal{A}\setminus\{i\}}(x_j|\mathbf{x}_{\mathcal{A}\setminus\{i\}}), F_{i|\mathcal{A}\setminus\{i\}}(x_i|\mathbf{x}_{\mathcal{A}\setminus\{i\}})) \\
&\quad \times f_{j|\mathcal{A}\setminus\{i\}}(x_j|\mathbf{x}_{\mathcal{A}\setminus\{i\}}).
\end{aligned} \tag{9}$$

By substituting iteratively (8) and (9) into (7), Bedford and Cooke (2002) expressed  $f_{\mathbf{X}}$  as a product of pair copula densities multiplied by  $\prod_i f_i$ . Recalling (6), it readily follows that the associated joint copula density  $c$  can be factorised into pair copula densities. Copulas expressed in this way are called vine copulas.

The factorisation is not unique: the pair copulas involved in the construction depend on the variables chosen in the conditioning equations (8)-(9) at each iteration. To organise them, Bedford and Cooke (2002) introduced a graphical model called the regular vine (R-vine). An R-vine among  $d$  random variables is represented by a graph consisting of  $d - 1$  trees  $T_1, T_2, \dots, T_{d-1}$ , where each tree  $T_i$  consists of a set  $N_i$  of nodes and a set  $E_i$  of edges  $e = (j, k)$  between nodes  $j$  and  $k$ . The trees  $T_i$  satisfy the following three conditions:

1. Tree  $T_1$  has nodes  $N_1 = \{1, \dots, d\}$  and  $d - 1$  edges  $E_1$
2. for  $i = 2, \dots, d - 1$ , the nodes of  $T_i$  are the edges of  $T_{i-1}$  :  $N_i = E_{i-1}$
3. Two edges in tree  $T_i$  can be joined as nodes of tree  $T_{i+1}$  by an edge only if they share a common node in  $T_i$  (proximity condition)

To build an R-vine with nodes  $\mathcal{N} = \{N_1, \dots, N_{d-1}\}$  and edges  $\mathcal{E} = \{E_1, \dots, E_{d-1}\}$ , one defines for each edge  $e$  linking nodes  $j = j(e)$  and  $k = k(e)$  in tree  $T_i$ , the sets  $I(e)$  and  $D(e)$  as follows:

- If  $e \in E_1$  (edge of tree  $T_1$ ), then  $I(e) = \{j, k\}$  and  $D(e) = \emptyset$ ,
- If  $e \in E_i$ ,  $i \geq 2$ , then  $D(e) = D(j) \cup D(k) \cup (I(j) \cap I(k))$  and  $I(e) = (I(j) \cup I(k)) \setminus D(e)$ .

$I(e)$  contains always two indices  $j_e$  and  $k_e$ , while  $D(e)$  contains  $i - 1$  indices for  $e \in E_i$ . One then associates each edge  $e$  with the conditional pair copula  $C_{j_e, k_e|D(e)}$  between  $X_{j_e}$  and  $X_{k_e}$  conditioned on the variables with indices in  $D(e)$ . An R-vine copula density with  $d$  nodes can thus be expressed as Aas (2016)

$$c(\mathbf{u}) = \prod_{i=1}^{d-1} \prod_{e \in E_i} c_{j_e, k_e|D(e)}(u_{j_e|D(e)}, u_{k_e|D(e)}). \tag{10}$$

Two special classes of R-vines are the drawable vine (D-vine; Kurowicka and Cooke (2005)) and the canonical vine (C-vine; Aas et al. (2009)). Denoting  $F(x_i) = u_i$  and  $F_{i|\mathcal{A}}(x_i|\mathbf{x}_{\mathcal{A}}) = u_{i|\mathcal{A}}$ ,  $i \notin \mathcal{A}$ , a C-vine density is given by the expression

$$c(\mathbf{u}) = \prod_{j=1}^{d-1} \prod_{i=1}^{d-j} c_{j, j+i|\{1, \dots, j-1\}}(u_{j|\{1, \dots, j-1\}}, u_{j+i|\{1, \dots, j-1\}}), \tag{11}$$

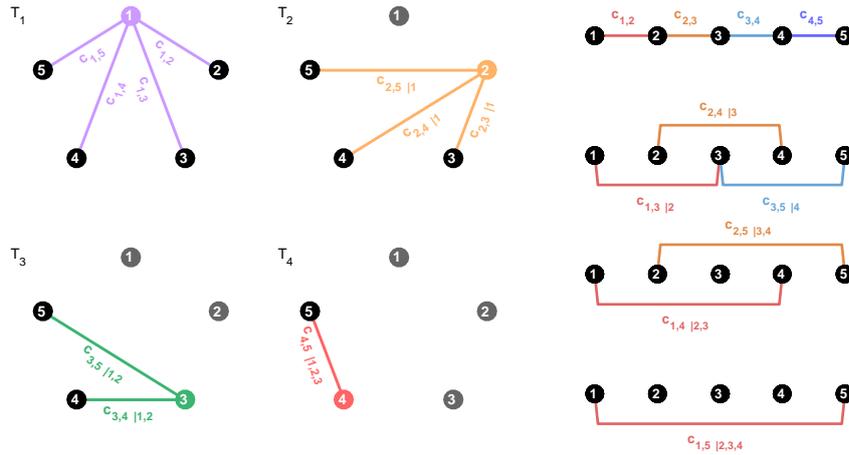


Figure 14: **Graphical representation of C- and D-vines.** The pair copulas in each tree of a 5-dimensional C-vine (left; conditioning variables are shown in grey) and of a 5-dimensional D-vine (right; conditioning variables are those between the connected nodes).

while a D-vine density is expressed by

$$c(\mathbf{u}) = \prod_{j=1}^{d-1} \prod_{i=1}^{d-j} c_{i,i+j|\{i+1,\dots,i+j-1\}}(u_{i|\{i+1,\dots,i+j-1\}}, u_{i+j|\{i+1,\dots,i+j-1\}}). \quad (12)$$

The graphs associated to a 5-dimensional C-vine and to a 5-dimensional D-vine are shown in Figure 14. Note that this simplified illustration differs from the standard one introduced in Aas et al. (2009) and commonly used in the literature.

### A.3 Vine copula inference in practice

We consider the purely data-driven case, typical in machine learning applications, where  $\mathbf{X}$  is only known through a set  $\mathcal{X}$  of independent observations. As remarked in Section 2.3.2, inference on  $C_{\mathbf{X}}$  can be performed on  $\mathbf{u}$ , obtained from  $\mathcal{X}$  by probability integral transform of each component after the marginals  $f_i$  have been assigned. In this setup, building a vine copula model on  $\mathbf{u}$  involves the following steps:

1. Selecting a vine structure (for C- and D-vines: selecting the order of the nodes);
2. Selecting the parametric family of each pair copula;
3. Fitting the pair copula parameters to  $\mathbf{u}$ .

Steps 1-2 form the representation problem. Concerning step 3, algorithms to compute the likelihood of C- and D-vines Aas et al. (2009) and of R-vines Joe (2015) given a data set  $\mathbf{u}$  exist, enabling parameter fitting based on maximum likelihood. In principle, the vine copula that best fits the data may be determined by iterating the maximum fitting approach over all possible vine structures and all possible parametric families of comprising pair copulas. In practice however, this approach is computationally infeasible in even moderate dimension

$d$  due to the large number of possible structures Morales-Nápoles (2011) and of pair copulas comprising the vine.

Taking a different approach first suggested in Aas et al. (2009) and common to many applied studies, we first solve step 1 separately. The optimal vine structure is selected heuristically so as to capture first the pairs  $(X_i, X_j)$  with the strongest dependence (which then fall in the upper trees of the vine). The Kendall's tau Stuart and Ord (1994) is selected as such a measure of dependence, defined by

$$\tau_{ij} = \mathbb{P}((X_i - \tilde{X}_i)(X_j - \tilde{X}_j) > 0) - \mathbb{P}((X_i - \tilde{X}_i)(X_j - \tilde{X}_j) < 0), \quad (13)$$

where  $(\tilde{X}_i, \tilde{X}_j)$  is an independent copy of  $(X_i, X_j)$ . If the copula of  $(X_i, X_j)$  is  $C_{ij}$ , then

$$\tau_K(X_i, X_j) = 4 \int \int_{[0,1]^2} C_{ij}(u, v) dC_{ij}(u, v) - 1. \quad (14)$$

For a C-vine, ordering the variables  $X_1, \dots, X_M$  in decreasing order of dependence strength corresponds to select the central node in tree  $T_1$  as the variable  $X_{i_1}$  which maximises  $\sum_{j \neq i_1} \tau_{i_1 j}$ , then the node of tree  $T_2$  as the variable  $X_{i_2}$  which maximises  $\sum_{j \notin \{i_1, i_2\}} \tau_{i_2 j}$ , and so on. For a D-vine, this means ordering the variables  $X_{i_1}, X_{i_2}, \dots, X_{i_d}$  in the first tree so as to maximise  $\sum_{k=1}^{d-1} \tau_{i_k i_{k+1}}$ , which we solve as an open travelling salesman problem (OTSP) Applegate et al. (2006). An open source Matlab implementation of a genetic algorithm to solve the OTSP is provided in Kirk (2014). An algorithm to find the optimal structure for R-vines has been proposed in Dißmann et al. (2013).

For a selected vine structure, the vine copula with that structure that best fits the data is inferred by iterating, for each pair copula forming the vine, steps 2 and 3 over a list of pre-defined parametric families and their rotated versions. The families considered for inference in this paper are listed in Table A.4. The rotations of a pair copula  $C$  by 90, 180 and 270 degrees are defined, respectively, by

$$\begin{aligned} C^{(90)}(u, v) &= v - C(1 - u, v), \\ C^{(180)}(u, v) &= u + v - 1 + C(1 - u, 1 - v), \\ C^{(270)}(u, v) &= u - C(u, 1 - v). \end{aligned} \quad (15)$$

(Note that  $C^{(90)}$  and  $C^{(270)}$  are obtained by flipping the copula density  $c$  around the horizontal and vertical axis, respectively; some references provide the formulas for actual rotations:  $C^{(90)}(u, v) = v - C(v, 1 - u)$ ,  $C^{(270)}(u, v) = u - C(1 - v, u)$ ). Including the rotated copulas, 62 families were considered in total for inference in our study.

To facilitate inference, we rely on the so-called *simplifying assumption* commonly adopted for vine copulas, namely that the pair copulas  $C_{j(e), k(e)|D(e)}$  in (10) only depend on the variables with indices in  $D(e)$  through the arguments  $F_{i(e)|D(e)}$  and  $F_{j(e)|D(e)}$  Czado (2010). While being exact only in particular cases, this assumption is usually not severe Haff et al. (2010). Estimation techniques for non-simplified vine models have also been proposed Stöber et al. (2013).

For each pair copula composing the vine, its parametric family is selected as the family that minimises the Akaike information criterion (AIC)

$$\text{AIC} = 2 \times (k - \log L), \quad (16)$$

where  $k$  is the number of parameters of the pair copula and  $\log L$  is its log-likelihood. The process is iterated over all pair copulas forming the vine, starting from the unconditional copulas in the top tree (sequential inference). Finally (and optionally), one keeps the vine structure and the copula families obtained in this way (that is, the parametric form of the vine), and performs global likelihood maximisation.

#### A.4 Rosenblatt transform of R-vines

Suppose that the probabilistic input model  $F_{\mathbf{X}}$  is specified in terms of marginals  $F_i$  and a copula  $C_{\mathbf{X}}$ , the polynomial chaos representation can be more conveniently achieved by first mapping  $\mathbf{X}$  onto a vector  $\mathbf{Z} = \mathcal{T}(\mathbf{X})$  with independent components.

The most general map  $\mathcal{T}$  of a random vector  $\mathbf{X}$  with dependent components onto a random vector  $\mathbf{Z}$  with mutually independent components is the Rosenblatt transform Rosenblatt (1952)

$$\mathcal{T} : \mathbf{X} \mapsto \mathbf{W}, \text{ where } \begin{cases} Z_1 = F_1(X_1) \\ Z_2 = F_{2|1}(X_2|X_1) \\ \vdots \\ Z_d = F_{d|1,\dots,d-1}(X_d|X_1, \dots, X_{d-1}) \end{cases} . \quad (17)$$

One can rewrite (see also Lebrun and Dutfoy (2009))  $\mathcal{T} = \mathcal{T}^{(\text{II})} \circ \mathcal{T}^{\text{PIT}} : \mathbf{X} \mapsto \mathbf{U} \mapsto \mathbf{Z}$ , where  $\mathcal{T}^{\text{PIT}}$ , given by (4), is known once the marginals have been computed, while  $\mathcal{T}^{(\text{II})}$  is given by

$$\mathcal{T}^{(\text{II})} : \mathbf{U} \mapsto \mathbf{Z}, \text{ with } Z_i = C_{i|1,\dots,i-1}(U_i|U_1, \dots, U_{i-1}). \quad (18)$$

Here,  $C_{i|1,\dots,i-1}$  are conditional copulas of  $\mathbf{X}$  (and therefore of  $\mathbf{U}$ ), obtained from  $C_{\mathbf{X}}$  by differentiation. The variables  $Z_i$  are mutually independent and have marginal uniform distributions in  $[0, 1]$ . The problem of obtaining an isoprobabilistic transform of  $\mathbf{X}$  is hence reduced to the problem of computing derivatives of  $C_{\mathbf{X}}$ .

Representing  $C_{\mathbf{X}}$  as an R-vine solves this problem. Indeed, algorithms to compute (18) have been established (see Schepsmeier (2015), and Aas et al. (2009) for algorithms specific to C- and D-vines). Given the pair-copulas  $C_{ij}$  in the first tree of the vine, the algorithms first compute their derivatives  $C_{i|j}$ . Higher-order derivatives  $C_{i|ijk}$ ,  $C_{i|ijkh}$ ,  $\dots$  are obtained from the lower-order ones by inversion and differentiation. Derivatives and inverses of continuous pair copulas are generally computationally cheap to compute numerically, when analytical solutions are not available. The algorithms can be trivially implemented such that  $n$  sample points are processed in parallel.

ID	Name	CDF	Parameter range
1	AMH	$\frac{uv}{1 - \theta(1-u)(1-v)}$	$\theta \in [-1, 1]$
2	AsymFGM	$uv(1 + \theta(1-u)^2v(1-v))$	$\theta \in [0, 1]$
3	BB1	$\left(1 + \left((u^{-\theta_2} - 1)^{\theta_1} + (v^{-\theta_2} - 1)^{\theta_1}\right)^{1/\theta_1}\right)^{-1/\theta_2}$	$\theta_1 \geq 1, \theta_2 > 0$
4	BB6	$1 - \left(1 - \exp\left\{-\left[(-\log(1 - (1-u)^{\theta_2}))^{\theta_1} + (-\log(1 - (1-v)^{\theta_2}))^{\theta_1}\right]^{1/\theta_1}\right\}\right)^{1/\theta_2}$	$\theta_1 \geq 1, \theta_2 \geq 1$
5	BB7	$\varphi(\varphi^{-1}(u) + \varphi^{-1}(v))$ , where $\varphi(w) = \varphi(w; \theta_1, \theta_2) = 1 - (1 - (1+w)^{-1/\theta_1})^{1/\theta_2}$	$\theta_1 \geq 1, \theta_2 > 0$
6	BB8	$\frac{1}{\theta_1} \left(1 - \left(1 - \frac{(1 - (1 - \theta_1 u)^{\theta_2})(1 - (1 - \theta_1 v)^{\theta_2})}{1 - (1 - \theta_1)^{\theta_2}}\right)^{1/\theta_2}\right)$	$\theta_1 \geq 1, \theta_2 \in (0, 1]$
7	Clayton	$(u^{-\theta} + v^{-\theta} - 1)^{-1/\theta}$	$\theta > 0$
8	FGM	$uv(1 + \theta(1-u)(1-v))$	$\theta \in (-1, 1)$
9	Frank	$-\frac{1}{\theta} \log\left(\frac{1 - e^{-\theta} - (1 - e^{-\theta u})(1 - e^{-\theta v})}{1 - e^{-\theta}}\right)$	$\theta \in \mathbb{R} \setminus \{0\}$
10	Gaussian	$\Phi_{2,\theta}(\Phi^{-1}(u), \Phi^{-1}(v))$ <sup>(a)</sup>	$\theta \in (-1, 1)$
11	Gumbel	$\exp(-((-\log u)^\theta + (-\log v)^\theta)^{1/\theta})$	$\theta \in [1, +\infty)$
12	Iterated FGM	$uv(1 + \theta_1(1-u)(1-v) + \theta_2uv(1-u)(1-v))$	$\theta_1, \theta_2 \in (-1, 1)$
13	Joe/B5	$1 - ((1-u)^\theta + (1-v)^\theta + (1-u)^\theta(1-v)^\theta)^{1/\theta}$	$\theta \geq 1$
14	Partial Frank	$\frac{uv}{\theta(u+v-uv)}(\log(1 + (e^{-\theta} - 1)(1 + uv - u - v)) + \theta)$	$\theta > 0$
15	Plackett	$\frac{1 + (\theta - 1)(u + v) - \sqrt{(1 + (\theta - 1)(u + v))^2 - 4\theta(\theta - 1)uv}}{2(\theta - 1)}$	$\theta \geq 0$
16	Tawn-1	$(uv)^{A\left(\frac{\log v}{\log(uv)}; \theta_1, \theta_3\right)}$ , where $A(w; \theta_1, \theta_3) = (1 - \theta_3)w + [w^{\theta_1} + (\theta_3(1-w))^{\theta_1}]^{1/\theta_1}$	$\theta_1 \geq 1, \theta_3 \in [0, 1]$
17	Tawn-2	$(uv)^{A\left(\frac{\log v}{\log(uv)}; \theta_1, \theta_2\right)}$ , where $A(w; \theta_1, \theta_2) = (1 - \theta_2)(1-w) + [(\theta_2 w)^{\theta_1} + ((1-w))^{\theta_1}]^{1/\theta_1}$	$\theta_1 \geq 1, \theta_2 \in [0, 1]$
18	Tawn	$(uv)^{A(w; \theta_1, \theta_2, \theta_3)}$ , where $w = \frac{\log v}{\log(uv)}$ and $A(w; \theta_1, \theta_2, \theta_3) = (1 - \theta_2)(1-w) + (1 - \theta_3)w + [(\theta_2 w)^{\theta_1} + (\theta_3(1-w))^{\theta_1}]^{1/\theta_1}$	$\theta_1 \geq 1, \theta_2, \theta_3 \in [0, 1]$
19	t-	$t_{2;\nu,\theta}(t_\nu^{-1}(u), t_\nu^{-1}(v))$ <sup>(b)</sup>	$\nu > 1, \theta \in (-1, 1)$

Table A.4: **Distributions of bivariate copula families used in vine inference.** Copula IDs (reported as assigned in the VineCopulaMatlab toolbox), distributions, and parameter ranges. (a)  $\Phi$  is the univariate standard normal distribution, and  $\Phi_{2;\theta}$  is the bivariate normal distribution with zero means, unit variances and correlation parameter  $\theta$ . (b)  $t_\nu$  is the univariate  $t$  distribution with  $\nu$  degrees of freedom, and  $t_{\nu,\theta}$  is the bivariate  $t$  distribution with  $\nu$  degrees of freedom and correlation parameter  $\theta$ .

## B PCE with R-vine input models

If the inputs  $\mathbf{X}$  to the system are assumed to be mutually dependent, a possible approach to build a basis of orthogonal polynomials by tensor product is to transform  $\mathbf{X}$  into a random vector  $\mathbf{Z}$  with mutually independent components. The PCE metamodel can be built afterwards from  $\mathbf{Z}$  to  $Y$ . This approach, indicated by *lPCEonZ* in the text, comprises the following steps:

1. Model the joint CDF  $F_{\mathbf{X}}$  of the input by inferring its marginals and copula. Specifically:
  - (a) infer the marginal CDFs  $F_i$ ,  $i = 1, \dots, d$ , from the input observations  $\mathcal{X}$  (e.g., by KDE);
  - (b) map  $\mathcal{X}$  onto  $\mathbf{U} = \{\mathcal{T}^{\text{PIT}}(\hat{\mathbf{x}}^{(j)}), j = 1, \dots, n\} \subset [0, 1]^d$  by (4);
  - (c) model the copula  $C_{\mathbf{X}} \equiv C_{\mathbf{U}}$  of the input by inference on  $\mathbf{U}$ ; R-vine copulas are compatible with this framework;
  - (d) define  $F_{\mathbf{X}}$  from the  $F_i$  and  $C_{\mathbf{X}}$  using (1).
2. Map  $\mathbf{U}$  onto  $\mathcal{Z} = \{\mathcal{T}^{(\Pi)}(\hat{\mathbf{u}}^{(j)}), j = 1, \dots, n\}$  by the Rosenblatt transform (18). If the inferred marginals and copula are accurate, the underlying random vector  $\mathbf{Z}$  has approximately independent components, uniformly distributed in  $[0, 1]$ .
3. Build a PCE metamodel on the training set  $(\mathcal{Z}, \mathcal{Y})$ . Specifically, obtain the basis of  $d$ -variate orthogonal polynomials by tensor product of univariate ones. The procedure used to build each  $i$ -th univariate basis depends on the distribution assigned to  $Z_i$  (if  $Z_i \sim U([0, 1])$ , use Legendre polynomials; if  $Z_i \sim \hat{F}_i$  obtained by KDE, use arbitrary PCE).

The PCE metamodel obtained on the transformed input  $\mathbf{Z} = \mathcal{T}(\mathbf{X})$  can be seen as a transformation of  $\mathbf{X}$ ,

$$Y_{\text{PC}}(\mathbf{Z}) = (Y_{\text{PC}} \circ \mathcal{T})(\mathbf{X}). \quad (1)$$