PCE-GPR Toolbox User Guide

Version 1.0.1

Paolo Manfredi

1 Introduction

This is a brief user guide or the PCE-GPR toolbox, which can be downloaded from

`https://doi.org/10.5281/zenodo.15348860`

or

`https://github.com/pmanfredi85/pce-gpr-toolbox`

Please note that the toolbox implementation is the result of non-professional programming work and is relatively "quick and dirty": it is not particularly optimized for efficiency, but it does the job! The PCE-GPR toolbox has already been tested on a number of numerical examples, for which it showed excellent performance.

When using this toolbox, please cite the following papers:

- P. Manfredi, A hybrid polynomial chaos expansion–Gaussian process regression method for Bayesian uncertainty quantification and sensitivity analysis, Computer Methods in Applied Mechanics and Engineering, vol. 436, no. 117693 (2025).
- P. Manfredi, Polynomial chaos vs kernel machine learning methods for uncertainty quantification: A comparative study and benchmarking with the hybrid PCE-GPR approach, Computer Methods in Applied Mechanics and Engineering, vol. 449, part A, no. 118523 (2026).

For your convenience, the corresponding BibTeX entries are provided below.

```
@article{manfredi2025cmame-pce-gpr,
  title={A hybrid polynomial chaos expansion--{G}aussian process
  regression method for {B}ayesian uncertainty quantification
  and sensitivity analysis},
  author={Manfredi, Paolo},
  journal={Computer Methods in Applied Mechanics
  and Engineering},
  volume={436},
  pages={117693},
  year={2025},
  publisher={Elsevier},
  doi={https://doi.org/10.1016/j.cma.2024.117693}
}
```

```
@article{manfredi2026cmame-benchmark,
  title={Polynomial chaos vs kernel machine learning methods for
  uncertainty quantification: A comparative study and benchmarking
  with the hybrid {PCE}-{GPR} approach},
  author={Manfredi, Paolo},
  journal={Computer Methods in Applied Mechanics
  and Engineering},
  volume={449, part~A},
  pages={118523},
  year={2026},
  publisher={Elsevier},
  doi={https://doi.org/10.1016/j.cma.2025.118523}
}
```

To cite this manual, please use

- P. Manfredi, PCE-GPR Toolbox User Guide, Version 1.0.1, Politecnico di Torino, Italy, 2025.

or the BibTeX entry below.

```
@TechReport{pce-gpr-toolbox,
  author={Manfredi, Paolo},
  title={{PCE-GPR} Toolbox User Guide}},
  institution={Politecnico di Torino, Italy},
  year={2025},
  note={Version 1.0.1}
}
```

Please report any bug or inquiry to:
Prof. Paolo Manfredi
Department of Electronics and Telecommunications
Politecnico di Torino, Italy
E-mail: `paolo.manfredi@polito.it`.

2 Software Requirements

PCE-GPR is implemented in MATLAB and builds upon the UQLab toolbox [4]. UQLab is open source and can be downloaded from `https://www.uqlab.com/`

In particular, the PCE-GPR toolbox was developed based on the UQLab release 2.0.0.

Some functionalities regarding the posterior distribution of the estimated variance and Sobol' sensitivity indices require the Generalized chi-square distribution toolbox [1]. The code was tested using version 2.3.0, which can be downloaded from

`https://github.com/abhranildas/gx2/releases/tag/v2.3.0`

We also recommend installing MATLAB's Statistics and Machine Learning Toolbox™ and Optimization Toolbox™. The code has been successfully run on MATLAB Version: 24.2.0.2833386 (R2024b) Update 4 with the Optimization Toolbox Version 24.2 and Statistics and Machine Learning Toolbox Version 24.2.

3 License

4 Theory

The toolbox implements the PCE-GPR method [3], which hybridizes polynomial chaos expansion (PCE) and Gaussian process regression (GPR) techniques. The method uses a GPR formulation that leverages special kernels, which allow the analytical conversion of the model into a PCE.

The method computes a surrogate of a mathematical function or model

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

where $\boldsymbol{x} = (x_1, \ldots, x_d)$ is a d-dimensional vector of uncertain inputs with known joint probability density function $\omega(\boldsymbol{x})$ and y is a scalar output. The model is trained using a set of L observations $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_l, y_l)\}_{l=1}^{L}$, computed as $y_l = \mathcal{M}(\boldsymbol{x}_l)$ using the actual model (1).

Currently, only independent Gaussian or uniform inputs are supported, which allows expressing their joint distribution as the product of univariate distributions, i.e.,

$$\omega(\boldsymbol{x}) = \prod_{j=1}^{d} \omega(x_j). \tag{2}$$

The GPR model reads

$$y = \mathcal{M}_{\mathrm{GPR}}(\boldsymbol{x}) = \sum_{l=1}^{L} \alpha_k r(\boldsymbol{x}, \boldsymbol{x}_l), \tag{3}$$

where $r(\boldsymbol{x}, \boldsymbol{x}')$ is a special correlation function constructed from Hermite/Legendre polynomials, according to the probability distribution of the input parameters $\boldsymbol{x}$. The correlation function depends either on one hyperparameter ρ (isotropic kernel) or d hyperparameters $\boldsymbol{\rho} = (\rho_1, \dots, \rho_d)$ (anisotropic kernel).

The GPR model (3) can be converted into a PCE model in the form

$$y \approx \mathcal{M}_{\mathrm{PCE}}(\boldsymbol{x}) = \sum_{\boldsymbol{\kappa} \in \mathcal{K}} c_{\boldsymbol{\kappa}} \psi_{\boldsymbol{\kappa}}(\boldsymbol{x}), \tag{4}$$

where $\psi_{\boldsymbol{\kappa}}$ are multivariate orthonormal polynomials from the Wiener-Askey scheme [6]. The multivariate polynomials are constructed as the product of univariate ones using the multi-indices $\boldsymbol{\kappa} = (\kappa_1, \dots, \kappa_d)$, i.e.,

$$\psi_{\boldsymbol{\kappa}} = \prod_{j=1}^{d} \psi_{\kappa_j}(x_j). \tag{5}$$

The multi-indices define the degree of the multivariate polynomial in each input dimension. The univariate polynomials are defined by the distribution of the corresponding input: Hermite polynomials for Gaussian inputs and Legendre polynomials for uniform inputs.

The PCE coefficients $c_{\boldsymbol{\kappa}}$ are analytically obtained from the GPR coefficients $\boldsymbol{\alpha}$ as

$$c_{\boldsymbol{\kappa}} = \lambda_{\boldsymbol{\kappa}} \sum_{l=1}^{L} \alpha_l \psi_{\boldsymbol{\kappa}}(\boldsymbol{x}_l), \tag{6}$$

where $\lambda_{\boldsymbol{\kappa}}$ are scale factors related to the orthogonal polynomials. It should be noted that (4) is an approximation of (3) given the finite degree of the polynomial basis functions. However, the coefficients can be computed up to an arbitrary degree.

Moreover, a covariance matrix is associated to the PCE coefficients, accounting in Bayesian terms for the estimation uncertainty related to observing a limited amount of data. Hence, (4) can be interpreted as a "stochastic" model with coefficients that follow a multivariate Gaussian distribution.

Relevant statistical information for y, like mean, variance, and Sobol' sensitivity indices, is obtained analytically from the PCE coefficients with the inclusion of prediction uncertainty and confidence levels. For example, the mean of y corresponds to the coefficient of the zero-degree polynomial, i.e.,

$$\mu_y = c_{\mathbf{0}}, \tag{7}$$

while the variance of y is computed as the sum of the remaining PCE coefficients squared, i.e.

$$\sigma_y^2 = \sum_{\boldsymbol{\kappa} \in \mathcal{K} \setminus \mathbf{0}} c_{\boldsymbol{\kappa}}^2. \tag{8}$$

The Sobol' sensitivity indices, which describe the relative contribution of the individual inputs to the output variance, are computed as the sum of suitable subsets of the coefficients squared [5].

We refer to [3] for the exhaustive theoretical background, while some relevant information is also provided in the illustrative MATLAB Live Script included with this manual.

5 Main Functions

The toolbox consists of four main functions, which are listed and summarized below in the order they are normally used:

- `train_pce_gpr`: trains the associated GPR model (3) using input-output pairs of training data;
- `get_pce_gpr_coeff`: retrieves the PCE coefficients in (4) for a given set of basis functions, as well as the correlation matrix defining their estimation uncertainty in Bayesian terms;
- `predict_pce_gpr`: evaluates the GPR model (3) and (if the PCE coefficients have been previously computed) the PCE model (4) at a given set of points, as well as the corresponding covariance matrix of the predictions;
- `get_pce_gpr_stats`: retrieves probabilistic statistical information (mean, variance, Sobol' indices) from the PCE coefficients in terms of expected value, variance, and posterior distribution.

Before documenting the above functions, we define or recall common notation that will be frequently used throughout this report:

- d: number of input uncertain variables;
- L: number of training samples;
- $\mathcal{K}$: set of multi-indices $\boldsymbol{\kappa}$ for the PCE basis functions;

The cardinality of $\mathcal{K}$ (i.e., the number of PCE basis functions) is denoted as $|\mathcal{K}|$.

5.1 Train a PCE-GPR Model: `train_pce_gpr`

The first step in the typical modeling workflow is to train the PCE-GPR model based on a set of input-output data. To this end, a set $\{\boldsymbol{x}_l\}_{l=1}^{L}$ with L configurations of the d uncertain inputs is generated, typically—but not necessarily—according to their distribution. The corresponding observations (outputs) $\{y_l\}_{l=1}^{L}$ are computed using the reference simulator (1) that we would like to surrogate and, together with the input points, form the training dataset $\mathcal{D}_{\text{train}}$.

The training is performed using the function `train_pce_gpr`, whose inputs are described in Table 1. We use ■ to denote a mandatory input and □ to

Table 1: Description of `train_pce_gpr` inputs.

Type	Name	Value	Description
■	`x_train`	$L \times d$ Double	Input training samples $\boldsymbol{x}_l$
■	`y_train`	$L \times 1$ Double	Output training observations y_l
■	`distr`	Structure array of length d	Distribution of input parameters
	`.Type`	String	Distribution type
		`'norm'`	Gaussian distribution
		`'unif'`	Uniform distribution
	`.Parameters`	1×2 Double	Distribution parameters
		$[\mu, \sigma]$	Mean μ and standard deviation σ, if `Type='norm'`
		$[a, b]$	Lower bound a and upper bound b, if `Type='unif'`
□	`'IsotropicKernel'`	Logical	Flag indicating kernel type
		`false` (default)	Anisotropic kernel
		`true`	Isotropic kernel
□	`'Noise'`	Logical	Flag to activate noise on training data
		`false` (default)	Noiseless interpolation
		`true`	Noisy regression
□	`'EstimationMethod'`	String	Hyperparameter estimation method
		`'ML'` (default)	Maximum likelihood
		`'CV'`	Cross-validation (default: leave-one-out)
□	`'UQLabOpts'`	Structure	Structure with advanced UQLab-specific options

denote an optional name-value argument. Input-output pairs are provided using variables `x_train` and `y_train`, respectively. The former is a matrix with the $L \times d$ samples $\boldsymbol{x}_l$ and the latter is a column vector collecting the corresponding L observations y_l. Moreover, `distr` is a structure array specifying, for each input variable, the type of the distribution $\omega(x_j)$ and related parameters. Currently, only Gaussian (normal) and uniform distributions are supported.

By default, the function considers an anisotropic kernel, no noise on the observations, and a maximum likelihood estimation of the hyperparameters. Hence, the resulting model interpolates the training observations. The above settings can be changed, by means of the optional name-value arguments, to make the model consider an isotropic kernel (with a single hyperparameter ρ for all input dimensions), noise on the observations (thus operating in regression mode), and cross-validation for the hyperparameter estimation. In case of noisy regression, the observations y_l are assumed to be corrupted by a zero-mean Gaussian noise with standard deviation σ_n, which is estimated along with the kernel hyperparameters.

Some advanced UQLab-specific options regarding the hyperparameter estimation can be modified using the structure `UQLabOpts`, following the standard syntax of the Kriging module in UQLab [2]. The main options that can be modified regard the number of folds in the cross-validation scheme (the default is leave-one-out) and the optimization settings, such as hyperparameter initial values and bounds, iteration number, tolerance, optimization method, as well as other optimizer-specific options. Other UQLab settings, not related to the hyperparameter estimation, cannot be modified, as they are constrained by the PCE-GPR formulation (e.g., trend function and correlation family). Further information in this regard is provided in the function header, while we refer to [2] for the syntax.

Table 2: Description of `train_pce_gpr` output.

Name	Value	Description
`M_pce_gpr`	Structure	Structure with model parameters
`.x_train`	$L \times d$ Double	Standardized training points (a shifted and rescaled version of the input)
`.y_train`	$L \times 1$ Double	Training observations (an input)
`.distr`	Structure array of length d	Structure defining the distribution of the inputs (an input)
`.basisfun`	Cell array of length d	Array of function handles to the PCE basis functions for each input dimension
`.kernel`	Function handle	Handle to the correlation function $r(\boldsymbol{x}, \boldsymbol{x}' \vert \boldsymbol{\rho})$, embedding the estimated value of the hyperparameter(s)
`.L`	$L \times L$ Double	Cholesky factor $\boldsymbol{L}$ of matrix $\tilde{\boldsymbol{R}}$
`.rho`	Double or $1 \times d$ Double	Kernel hyperparameter(s) $\boldsymbol{\rho}$. A vector for an anisotropic kernel, a scalar for an isotropic kernel
`.tau`	Double	Noise ratio $\tau = \sigma_n^2/(\sigma^2 + \sigma_n^2)$
`.sigma`	Double	Kernel standard deviation σ
`.sigma_n`	Double	Noise standard deviation σ_n
`.alpha`	$L \times 1$ Double	Vector of GPR coefficients $\boldsymbol{\alpha}$
`.model`	`uq_model` class object	Model returned by UQLab

The function `train_pce_gpr` trains the associated GPR model (3) and returns the structure described in Table 2. The input samples are internally shifted and rescaled to match the standardized normal distribution $\mathcal{N}(0,1)$ or uniform distribution $\mathcal{U}(-1,1)$, in order to leverage the classical orthogonal polynomials as in the PCE framework. Users should keep in mind that the values of the training points stored in the output structure are the standardized ones, for the ease of further internal use.

The structure contains relevant information regarding the trained GPR model, particularly the model coefficients $\boldsymbol{\alpha}$, the kernel hyperparameter(s) $\boldsymbol{\rho}$, the kernel standard deviation σ, the noise standard deviation σ_n, and the ratio τ between the kernel variance and the noise variance, defined as

$$\tau = \frac{\sigma_n^2}{\sigma^2 + \sigma_n^2} \tag{9}$$

The kernel variance σ^2 is used in the calculation of posterior covariances.

The handle to the multivariate correlation function $r(\boldsymbol{x}, \boldsymbol{x}')$ is provided in the form

```
R = kernel(x,xp)
```

where `x` and `xp` are matrices with d-dimensional lists of points for the input pairs $\boldsymbol{x}$ and $\boldsymbol{x}'$. The returned matrix contains the correlation function evaluated at all pair-wise combinations of these inputs. The correlation function already embeds the estimated value of the hyperparameter(s) $\boldsymbol{\rho}$. The output structure also contains the Cholesky factor of matrix $\tilde{\boldsymbol{R}} = (1-\tau)\boldsymbol{R} + \tau\boldsymbol{I}$, where $\boldsymbol{I}$ is the indentity matrix and $\boldsymbol{R}$ is the correlation matrix evaluated at the (standardized) training points, such that $\tilde{\boldsymbol{R}} = \boldsymbol{L}\boldsymbol{L}^\mathsf{T}$. The Cholesky factor is useful as $\tilde{\boldsymbol{R}}$ is the matrix to be inverted in most post-processing calculations.

Finally, the handles to the univariate PCE basis functions are also provided. Specifically, the structure provides a cell array with d function handles to the univariate basis functions in (5) in the form

```
Psi = basisfun(k,x)
```

where `k` is the polynomial degree κ_j and `x` is the input x_j at which the basis function is evaluated. The function returns $\psi_{\kappa_j}(x_j)$.

5.2 Retrieve the PCE Coefficients: `get_pce_gpr_coeff`

This function is used to convert the GPR model into a PCE model, by analytically calculating the respective coefficients via (6). The inputs and outputs of the function are described in Table 3.

The function requires to provide the list of multi-indices $\boldsymbol{\kappa}$ in set $\mathcal{K}$, which can be obtained using the auxiliary function `PCEindex` (described later). It returns an updated model structure that includes the vector of PCE coefficients $c_{\boldsymbol{\kappa}}$ as well as their correlation and covariance matrices, which describe the uncertainty in the estimation of such coefficients.

Table 3: Description of function `get_pce_gpr_coeff`.

Name	Value	Description
Inputs:		
`M_pce_gpr`	Structure	PCE-GPR model structure
`kvec`	$\lvert\mathcal{K}\rvert \times d$ Double	Matrix with the list of multi-indices $\boldsymbol{\kappa}$ describing the PCE basis
Outputs:		
`M_pce_gpr`	Structure	Updated PCE-GPR model structure, including the additional fields indicated below
`.kvec`	$\lvert\mathcal{K}\rvert \times d$ Double	List of PCE multi-indices (an input)
`.pce_coeff`	$\lvert\mathcal{K}\rvert \times 1$ Double	Vector of PCE coefficients $c_{\boldsymbol{\kappa}}$
`.corr_pce_coeff`	$\lvert\mathcal{K}\rvert \times \lvert\mathcal{K}\rvert$ Double	Correlation matrix $\boldsymbol{C}$ of the PCE coefficients
`.cov_pce_coeff`	$\lvert\mathcal{K}\rvert \times \lvert\mathcal{K}\rvert$ Double	Covariance matrix $\boldsymbol{\Sigma} = (\sigma^2+\sigma_n^2)\boldsymbol{C}$ of the PCE coefficients
`c`	$\lvert\mathcal{K}\rvert \times 1$ Double	Vector of PCE coefficients (same as field `.pce_coeff`)

5.3 Make Predictions: `predict_pce_gpr`

Table 4: Description of function `predict_pce_gpr`.

Name	Value	Description
Inputs:		
`M_pce_gpr`	Structure	PCE-GPR model structure
`x`	$N \times d$ Double	Matrix with the list of points $\boldsymbol{x}$ at which to evaluate predictions
Outputs:		
`y_gpr`	$N \times 1$ Double	Predictions at the N input points obtained with the GPR model
`y_pce`	$N \times 1$ Double	Predictions at the N input points obtained with the PCE model
`cov_gpr`	$N \times N$ Double	Covariance matrix of the GPR model predictions
`cov_pce`	$N \times N$ Double	Covariance matrix of the PCE model predictions

This function is used to evaluate the PCE-GPR model and make predictions at a list of input points $\boldsymbol{x}$. The inputs and outputs are described in Table 4.

The function uses both the GPR and PCE models to make predictions along with the corresponding covariance matrices. Note that the PCE prediction is non-empty only if the PCE coefficients have been previously computed with `get_pce_gpr_coeff`. It is an approximation of the GPR prediction within the finite degree of the multi-index set $\mathcal{K}$. The covariance matrices reflect the prediction uncertainty due to observing a limited amount of samples in the training and is used, e.g., to assess the confidence level of such predictions.

5.4 Extract Statistical Information: `get_pce_gpr_stats`

Table 5: Description of function `get_pce_gpr_stats`.

Name	Value	Description
Inputs:		
`M_pce_gpr`	Structure	PCE-GPR model structure
Ouputs:		
`mu`	Double	Mean of the output y
`sigma2`	Double	Variance of the output y
`St`	$d \times 1$ Double	Sobol' indices of the output y
`var_mu`	Double	Predictive variance of the mean `mu`
`var_sigma2`	Double	Predictive variance of the variance `sigma2`
`var_St`	$d \times 1$ Double	Predictive variances of the Sobol' indices `St`
`pdf_mu`	`NormalDistribution` object	Posterior distribution of the mean `mu`
`pdf_sigma2`	`GeneralizedChiSquareDistribution` object	Posterior distribution of the variance `sigma2`
`pdf_St`	Array of $d \times 1$ `GeneralizedChiSquareDistribution` objects	Posterior distributions of the Sobol' indices `St`

This function, described in Table 5, is used to extract probabilistic statistical information from the PCE-GPR model. It requires to compute the PCE coefficients first, using the function `get_pce_gpr_coeff`.

The function returns the mean μ_y and the variance σ_y^2 of the output y, computed via (7) and (8), respectively, as well as the Sobol' sensitivity indices. The function also returns the predictive variance and the posterior distribution of the above quantities. Indeed, following the Bayesian interpretation of the GPR model, the predictions are random variables with an associated expected value, variance, and distribution. The posterior distributions are described using distribution objects. Specifically, the posterior distribution of the mean μ_y is Gaussian and handled using the MATLAB's native

`NormalDistribution` class. The posterior distribution of the variance σ_y^2 and of the Sobol' indices is a generalized chi-square instead, which is handled using a custom `GeneralizedChiSquareDistribution` class that is defined within the PCE-GPR toolbox and relies on the generalized chi-square toolbox [1].

6 Auxiliary Functions

This section describes auxiliary functions that are used internally by the four main functions described in the previous section. Most of these are not meant to be used as stand-alone functions.

6.1 Function `PCEindex`

Table 6: Description of function `PCEindex`.

Name	Type	Description
Inputs:		
`p`	Double	Maximum degree of the polynomials (an integer)
`d`	Double	Number of input dimensions (an integer)
`u` (optional)	Double	Hyperbolic truncation factor, which must be in the interval $(0, 1]$; default: $u = 1$
Outputs:		
`indices`	$\lvert\mathcal{K}\rvert \times d$ Double	List of multi-indices $\boldsymbol{\kappa}$ for the multivariate PCE basis

This function computes the multi-index set $\mathcal{K}$ for the PCE basis functions in d dimensions using a hyperbolic truncation with maximum degree p and hyperbolic truncation factor u. Hence, the function returns the list of multi-indices $\boldsymbol{\kappa}$ such that

$$\lVert\boldsymbol{\kappa}\rVert_u = \left(\sum_{j=1}^{d} \kappa_j^u\right)^{1/u} \leq p \tag{10}$$

If u not provided, $u = 1$ is used. This corresponds to the so-called total-degree truncation, which has $\lvert\mathcal{K}\rvert = (p + d)!/(p!d!)$ terms. The inputs and outputs of the function are described in Table 6.

6.2 Function `standardize_input`

This function is used to standardize input data $\boldsymbol{x}$ by shifting and rescaling them as appropriate, based on the corresponding distribution specified by the structure `distr`. The standardization is applied to each individual input. If

Table 7: Description of function `standardize_input`.

Name	Value	Description
Inputs:		
`x`	$N \times d$ Double	Inputs $\boldsymbol{x}$ to standardize
`distr`	Structure array of length d	Structure defining the input distributions (see Table 1)
Outputs:		
`x_standard`	$N \times d$ Double	Standardized version of the inputs $\boldsymbol{x}$

normally distributed, the input is standardized as

$$x_s = \frac{x - \mu}{\sigma}, \tag{11}$$

where μ and σ are the mean and standard deviation of the distribution, respectively. If uniformly distributed, the input is standardized as

$$x_s = \frac{x - \frac{a+b}{2}}{\frac{b-a}{2}}, \tag{12}$$

where a and b are the lower and upper bound of the distribution, respectively. The inputs and outputs of the function are described in Table 7.

6.3 Function `evalBasisFunctions_heterogeneous`

Table 8: Description of function `evalBasisFunctions_heterogeneous`.

Name	Value	Description
Inputs:		
`basisfun`	Cell array of length d	Array of function handles to the PCE basis functions for each input dimension (see Table 2)
`kvec`	$\lvert\mathcal{K}\rvert \times d$ Double	Matrix with the list of multi-indices $\boldsymbol{\kappa}$ of the PCE basis
`x`	$N \times d$ Double	Matrix with the list of points $\boldsymbol{x}$ at which to evaluate the basis functions
Outputs:		
`Psi`	$N \times \lvert\mathcal{K}\rvert$ Double	Matrix with the $\lvert\mathcal{K}\rvert$ basis functions evaluated at the N input points

This function is used to evaluate the multivariate PCE basis functions at given points $\boldsymbol{x}$. A cell array of function handles is used to specify the heterogeneous basis functions (Hermite or Legendre polynomials) for each input dimension, along with the list of multi-indices $\boldsymbol{\kappa}$ in set $\mathcal{K}$. The cell array is consistent with the definition of the PCE-GPR output structure in Table 2 and defines the generic type of the basis functions (Hermite or Legendre polynomials). The inputs $\boldsymbol{x}$ must be standardized using `standardize_input`.

The function returns a matrix $\boldsymbol{\Psi}$ with the basis functions evaluated at the N input points, with entries

$$\Psi_{i\boldsymbol{\kappa}} = \psi_{\boldsymbol{\kappa}}(\boldsymbol{x}_i), \tag{13}$$

for $i = 1, \dots, N$ and $\boldsymbol{\kappa} \in \mathcal{K}$. The inputs and outputs of the function are described in Table 8.

6.4 Functions `orthonormal_hermite` and `orthonormal_legendre`

Table 9: Description of functions `orthonormal_hermite` and `orthonormal_legendre`.

Name	Type	Description
Inputs:		
`k`	Double	Degree of the polynomial (an integer)
`N` (optional)	Double	Maximum degree of the returned polynomial (an integer)
Outputs:		
`P`	$(k+1) \times 1$ Double or $(N+1) \times 1$ Double	Array with the polynomial coefficients

These functions return the coefficients of the k-th univariate Hermite or Legendre orthonormal polynomial $P_k(x)$. The inputs and outputs are described in Table 9. Compared to classical Hermite and Legendre polynomials, the coefficients are normalized so that the L^2 norm of each polynomial is one.

The functions return an array that can be used, e.g., with MATLAB's function `polyval` to evaluate the polynomials at given points. The optional input N specifies the maximum degree of the polynomial. If $N > k$, the coefficient vector is zero padded.

6.5 Function `kernel_heterogeneous`

This function evaluates the heterogeneous kernel, with given hyperparameters $\boldsymbol{\rho}$, at input pairs $(\boldsymbol{x}, \boldsymbol{x}')$. The heterogeneous kernel is constructed as the product of univariate Hermite or Legendre correlation functions, i.e.,

$$r(\boldsymbol{x}, \boldsymbol{x}'|\boldsymbol{\rho}) = \prod_{j=1}^{d} r(x_j, x'_j|\rho_j) \tag{14}$$

Table 10: Description of function `kernel_heterogeneous`.

Name	Value	Description
Inputs:		
`x`	$M \times d$ Double	Matrix with the list of first input points $\boldsymbol{x}$
`xp`	$N \times d$ Double	Matrix with the list of second input points $\boldsymbol{x}'$
`rho`	Double or $1 \times d$ Double	Kernel hyperparameter(s). It must be a scalar (ρ) for isotropic kernels or a vector ($\boldsymbol{\rho}$) for anisotropic kernels
`distr`	Structure array of length d	Structure defining the input distributions (see Table 1)
Outputs:		
`Omega`	$M \times N$ Double	Matrix $\boldsymbol{\Omega}$ with the kernel evaluated at all the pair-wise combinations of the inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$

given the distribution of the inputs. The inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$ must be standardized using `standardize_input`.

The function returns a matrix $\boldsymbol{\Omega}$ with the kernel evaluated at all pair-wise combinations of the M and N input points, with entries

$$\Omega_{ij} = r(\boldsymbol{x}_i, \boldsymbol{x}'_j|\boldsymbol{\rho}), \tag{15}$$

for $i = 1, \ldots, M$ and $j = 1, \ldots, N$. The inputs and outputs of the function are described in Table 10.

6.6 Functions `kernel_hermite` and `kernel_legendre`

Table 11: Description of functions `kernel_hermite` and `kernel_legendre`.

Name	Value	Description
Inputs:		
`x`	$M \times 1$ Double	Matrix with the list of first input points x
`xp`	$N \times 1$ Double	Matrix with the list of second input points x'
`rho`	Double	Kernel hyperparameter ρ
Outputs:		
`Omega`	$M \times N$ Double	Matrix $\boldsymbol{\Omega}$ with the kernel evaluated at all pair-wise combinations of the inputs x and x'

These functions, described in Table 11, evaluate the univariate Hermite or Legendre correlation functions, with given hyperparameter ρ, at input pairs (x, x'). The inputs x and x' must be standardized with `standardize_input`.

The function returns a matrix $\boldsymbol{\Omega}$ with the kernel evaluated at all pair-wise combinations of the M and N inputs, with entries

$$\Omega_{ij} = r(x_i, x'_j|\rho), \tag{16}$$

for $i = 1, \ldots, M$ and $j = 1, \ldots, N$.

6.7 Function `qf_mean_var`

Table 12: Description of function `qf_mean_var`.

Name	Type	Description
Inputs:		
`mu`	$K \times 1$ Double	Mean vector $\boldsymbol{m}$ of the normal random variables
`Cov`	$K \times K$ Double	Covariance matrix $\boldsymbol{C}$ of the normal random variables
`Q`	$K \times K$ Double	Symmetric matrix $\boldsymbol{Q}$ of quadratic coefficients
Outputs:		
`m`	Double	Expected value μ_{qf} (mean) of the quadratic form
`v`	Double	Variance σ^2_{qf} of the quadratic form

This function is used to evaluate the mean μ_{qf} and variance σ^2_{qf} of the quadratic form

$$y = \boldsymbol{x}^{\mathsf{T}} \boldsymbol{Q} \boldsymbol{x}, \tag{17}$$

where $\boldsymbol{Q}$ is a symmetric matrix and $\boldsymbol{x}$ is a vector of K normal random variables with mean vector $\boldsymbol{m}$ and covariance matrix $\boldsymbol{C}$. The inputs and outputs are described in Table 12. This function is used to evaluate the mean and variance of the output variance and of the Sobol' indices.

6.8 Class `GeneralizedChiSquareDistribution`

This file defines the `GeneralizedChiSquareDistribution` class that describes a generalized chi-square random variable $\tilde{\chi}(\boldsymbol{w}, \boldsymbol{k}, \boldsymbol{\lambda}, s, m)$. The properties and methods of the class are listed in Table 13.

The class has similar methods as other classes natively available in MATLAB, such as `NormalDistribution` and `UniformDistribution`. The class relies on the generalized chi-square toolbox [1] and is used to handle the posterior distribution of the output variance and of the Sobol' indices.

References

[1] Abhranil Das. Generalized chi-square distribution, version 2.3.0. `https://github.com/abhranildas/gx2/releases/tag/v2.3.0`. Accessed: 2025-02-27.

Table 13: Description of class `GeneralizedChiSquareDistribution`.

Name	Description
Properties:	
`w`	Vector $\boldsymbol{w}$ of weights
`k`	Vector $\boldsymbol{k}$ of degrees of freedom
`lambda`	Vector $\boldsymbol{\lambda}$ of non-centrality parameters
`m`	Mean m of normal term
`s`	Standard deviation s of normal term
Methods:	
`.pdf`	Computes the probability density function of the distribution
`.cdf`	Computes the cumulative distribution function of the distribution
`.icdf`	Computes the inverse cumulative distribution function of the distribution
`.ci`	Computes confidence intervals with a given significance level α
`.random`	Generate random samples from the distribution
`.mean`	Returns the mean of the distribution
`.var`	Returns the variance of the distribution
`.std`	Returns the standard deviation of the distribution

[2] C. Lataniotis, D. Wicaksono, S. Marelli, and B. Sudret. UQLab user manual – Kriging (Gaussian process modeling). Technical report, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2022. Report UQLab-V2.0-105.

[3] Paolo Manfredi. A hybrid polynomial chaos expansion–gaussian process regression method for bayesian uncertainty quantification and sensitivity analysis. *Computer Methods in Applied Mechanics and Engineering*, 436:117693, 2025.

[4] Stefano Marelli and Bruno Sudret. UQLab: A framework for uncertainty quantification in Matlab. In *Vulnerability, uncertainty, and risk: quantification, mitigation, and management*, pages 2554–2563. 2014.

[5] Bruno Sudret. Global sensitivity analysis using polynomial chaos expansions. *Reliability Engineering & System Safety*, 93(7):964–979, 2008.

[6] Dongbin Xiu and George Em Karniadakis. The Wiener–Askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing*, 24(2):619–644, 2002.