University of Liege

Faculty of applied sciences

Propagation of Uncertainties in Pyrolysis Kinetic Parameters Using Polynomial Chaos Methods

Graduation Studies conducted for obtaining the Master's degree in Engineering Physics by
Martin Lacroix

Advisors:
Maarten Arnst
Joffrey Coheur

Academic year 2019-2020

Abstract

This master thesis addresses two challenges for the propagation of uncertainties related to the pyrolysis process in thermal protection materials, which are the high computational cost of numerical simulations and the correlation between input uncertainties. Due to this high computational cost, classical techniques such as Monte Carlo simulations are not applicable. In this respect, we propose exploring the so-called method of polynomial chaos, which consists in using a set of orthogonal polynomials to build a cheaper surrogate model from a limited number of runs of the reference model. First, some theoretical and computational aspects of the polynomial chaos are presented in details, then different test cases are considered in order to assess the relevance of the method in producing a surrogate model for complex pyrolysis and thermal ablation processes. In summary, the goal of this thesis is to successfully demonstrate the possibility of computing a cheap and accurate surrogate model for complex pyrolysis processes in moderately high dimensions when the uncertainties on the input parameters are correlated.

List of Abbreviations

ANCOVA	Analysis of covariance
ANOVA	Analysis of variance
GS	Gram-Schmidt
LARS	Least angle regression
MC	Monte Carlo
PCA	Principal component analysis
PCE	Polynomial chaos expansion
SD	Standard deviation
SRE	Squared relative error

Acknowledgements

I would first like to thank my thesis advisor, Professor Arnst, for the useful advice, remarks and engagement through the learning process of this master thesis. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to thank Joffrey Coheur who were involved in this research project, answering the many questions I had. Without his help and participation, this work could not have been successfully conducted.

Finally, I would like to thank my family, as well as the other engineers and students with whom I spent these few years at the University of Liège. Evidently, they contributed to my success and to make this experience unforgettable.

Martin Lacroix

Contents

List of Figures

Chapter 1

Introduction

1.1 Atmospheric Entry

Atmospheric entry is the phase during which a natural or artificial object enters the atmosphere of a planet and reaches layers dense enough to cause mechanical and thermal effects. Since entering an atmosphere from space at high velocity will cause very high levels of heating, the heat shield is intended to protect the spacecraft during this critical phase. As soon as the atmosphere becomes denser, the high kinetic energy of the vehicle is dissipated into thermal energy, bringing the temperature of its surface to several thousand degrees (Figure 1).

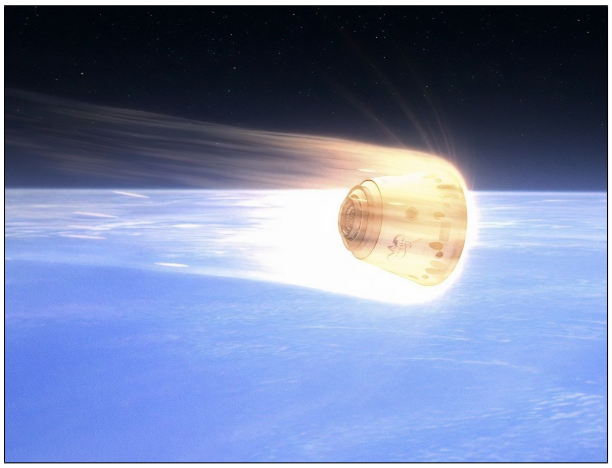

Figure 1: Artist rendering of Dragon capsule re-entry. The PICA-X heat shield is based on a variant of a phenolic impregnated carbon ablator material designed to protect the spacecraft during atmospheric entry. The Dragon capsule entered the Earth's atmosphere at around 7 km/s, heating the exterior of the shield to up to 1850 degrees Celsius. However, just a few inches of the PICA-X material keeps the interior of the capsule at room temperature [1].

Multiple approaches for the thermal protection of spacecraft are currently in use, for instance, passive or active cooling of spacecraft surfaces. In this thesis, we focus on the so-called thermal ablative heat shields. The main working principle of an ablative heat shield is to lift the shock layer gas away from the shield surface thanks to the reinjection of gases from physical or chemical decomposition, opposing the convection of hot gases in the boundary layer. This blocking process significantly reduces the heat flow and the transfer of gases to the wall. In particular, the pyrolysis is a process of chemically decomposing organic materials at elevated temperatures in the absence of oxygen, and simultaneously involves the change of physical state and chemical composition. A part of the material turns into gas and migrates to the surface, participating in the blocking process. In addition to the blocking process, another important propriety of ablative materials is their low thermal conductivity as well as their ability to absorb much higher heat fluxes than non-ablative materials.

The objective of uncertainty analysis is to investigate and mitigate the effect of uncertainties on the quantity of interest. Evidently, thermal protection systems involve a large number of physical properties whose impact on the behaviour of the shield can be more or less significant. Regarding the fact that many of these parameters are not known exactly, it is important to evaluate the effects of their uncertainties on the predictions of the behaviour of the shield. For instance, the different possible values of the parameters can be propagated through a numerical model to obtain the corresponding production of gas by the ablative material, providing a better understanding of the uncertainty on the output. Consequently, such analysis allows the assessment of the reliability of the shield, reducing the need for high security margin in its conception.

1.2 Objectives

It is clear that the proper functioning of the heat shield is one of the most important factors in the success of the atmospheric entry of the spacecraft. Numerical simulations of thermal protection systems require the development of models for the chemical reactions occurring in the material at elevated temperatures, such as the degradation of the material caused by pyrolysis process. However, models typically used for the production of pyrolysis gases are highly empirical and require the determination of several parameters [2]. The identification is based on experiments performed on small samples of the ablative material composing the thermal shield. Previous work performed the inference on the parameters using optimization algorithms [3], or using Bayesian inference methods such as Metropolis-Hastings algorithms [4][5], or more recently gradient-based methods for the sampling of the probability density function. Those algorithms were implemented in the Bayesian inference and uncertainty propagation toolbox `Pybitup`, a software developed by J. Coheur in the Computational and Stochastic Modelling research unit at the University of Liège.

Building on these previous studies, this work intends to propagate the uncertainties inferred for the input parameters through numerical models and to develop further the uncertainty propagation component of `Pybitup`. Indeed, as statistical studies typically require the runs of a large number of numerical simulations to estimate statistical moments or to analyse the effect of the input parameters on the output, uncertainty propagation is facing several challenges, among which is the large computation time required for such simulations. In order to reduce this computational cost, we propose to use the method of polynomial chaos, consisting of building a cheaper surrogate model of a stochastic system by using polynomial approximations of the input-to-output map. In fact, PCE methods are already widely used in computational and aerospace engineering [6][7]. However, in applications, most often, labeled probability density functions, such as Gaussian probability density functions, are assigned to uncertain input parameters, and the uncertain input parameters are assumed to be statistically independent of each other. In contrast, here, we intend to assign a Bayesian posterior inferred from experimental data to the uncertain input parameters. Such a Bayesian posterior is typically non-Gaussian and entails statistical dependence between the uncertain input variables. Thereby, the surrogate model must take into account the the non-Gaussiannity and the statistical dependence of its input variables.

Significant work has already been conducted on the computation of a PCE for independent random variables in low dimensions [8]. However, classical PCE methods may struggle to perform efficiently when the number of dimensions becomes high, and different improvements have recently been proposed in the literature to help remedy this problem. In addition, classical PCE methods assume a stochastic independence between the input variables, and consequently, lose their reliability when the variables are highly correlated. The first part of this thesis is therefore devoted to a literature review of recent methods developed to compute a PCE for high dimensional problems with arbitrary, non-Gaussian and multivariate probability density functions. The second part of this work is the implementation of a PCE library in `Pybitup` and the use the algorithm thus implemented to construct a surrogate model for different test cases.

1.3 Outline

- Chapter 2 provides theoretical aspects of the PCE, first by introducing the concept of orthogonal polynomials as well as different methods to compute an orthogonal basis according to an arbitrary inner product. In the second part, we present some standard and advanced algorithms to compute the coefficients of the PCE. In the last part, some interesting post processing techniques such as the computation of sensitivity indices are discussed.

- Chapter 3 first focuses on Monte Carlo integration methods and highlights their advantages for the estimation of high dimensional integrals. Some variance reduction techniques such as the use of low discrepancy sequences are then explored in order to improve the convergence of the method. The second part of the chapter provides different algorithms for computing a quadrature rule in low and moderately high dimensions.

- Chapter 4 gives a general overview of the structure of the code `Pybitup` and describes the two main classes implemented for the storage and the computation of the PCE. We give additional details about the homogeneous block representation of the polynomials as well as some important algorithms.

- Chapter 5 is dedicated to the application of the methodology to different test cases. A first introductory example aims to assess the efficiency of the code for simple problems by comparing the output of the PCE to analytical solutions. Afterwards, a larger scale application is conducted in order to compare the different algorithms thus implemented as well as to conduct statistical studies with the surrogate model.

Chapter 2

Polynomial Chaos Expansion

2.1 Introduction

The polynomial chaos is a method that enables a fast construction of surrogate models that can be efficiently used for uncertainty quantification of a model output when there is probabilistic uncertainty in the input parameters. The PCE consists of expanding the function in terms of a series of orthogonal polynomials weighted by some coefficients. Let $\mathbf{X}$ be a random vector with values in the sample space Ω and let $Y = y(\mathbf{X})$, where y is a square integrable function defined on Ω. In addition, let $\{P_j(\mathbf{x})\}$ be a set of orthogonal polynomials in Ω with respect to the probability density function of $\mathbf{X}$, the PCE reads

$$y(\mathbf{x}) = \sum_{j=1}^{\infty} y_j P_j(\mathbf{x}), \tag{2.1}$$

where the y_j are weighting coefficients obtained, for instance, by projecting y onto the polynomials. Originally, the generalized PCE was introduced in [8] by generalizing the Cameron–Martin theorem to various continuous and discrete distributions using orthogonal polynomials from the so-called Askey-scheme. However, the method has notable limitations such as the assumption of stochastic independence between the input parameters or the exponential increase of the computational cost for high dimensional problems. Recently, generalizations towards arbitrary distributions resulted in the arbitrary or data-driven PCE [9]. These techniques are still under active development, but are particularly relevant when one has an inaccurate knowledge of the statistical distributions in play.

One of the main challenges when constructing a PCE is the computation of the coefficients. Nowadays, orthogonal polynomials and their associated quadrature rule can be obtained for various distributions using a three term recurrence relation [10], and the computation of the integrals involved in spectral projections or least squares methods are straightforward, provided the number of random variables remains small. However, many problems involve high dimensional data, consequently, the sparse PCE constructed by adaptive algorithms based on least angle regression have been proposed to reduce the complexity of the surrogate model in high dimensions [11].

2.2 Polynomial Basis

An orthogonal polynomial basis in the real vector space of domain Ω equipped with the inner product $\langle \cdot, \cdot \rangle$ is defined as the set of polynomials such that

$$\langle P_j, P_i \rangle = \int_{\Omega} P_j(\mathbf{x}) P_i(\mathbf{x}) \, d\mu(\mathbf{x}) = ||P_j||^2 \, \delta_{ij}, \tag{2.2}$$

where $d\mu(\mathbf{x}) = f_X(\mathbf{x}) \, d\mathbf{x}$ and f_X is a probability density function of $\mathbf{X}$. Furthermore, all the orthogonal polynomials generated by `Pybitup` are normalized according to their inner product so that $||P_j|| = 1 \; \forall \; j$.

2.2.1 Independent Variables

A basis of orthogonal polynomials of d independent variables can be constructed by tensor product of the one-dimensional orthogonal bases $\{P_j^k(x_k)\}$ relative to each of the variables x_k considered, namely

$$\{P_j(\mathbf{x})\} = \bigotimes_{k=1}^{d} \{P_j^k(x_k)\}. \tag{2.3}$$

Defining the multi-index matrix $\mathbf{I}$, where each column corresponds to a variable in the model, and each element indicates the index of a one-dimensional polynomial in the basis related to its variable. Each row thus corresponds to a particular combination of one-dimensional polynomials related to different variables. The tensor product is

$$P_j(\mathbf{x}) = \prod_{k=1}^{d} P_{\mathbf{I}_{jk}}^k(x_k) \tag{2.4}$$

and the joint weight function is the product of each individual weight functions $f_X^k(x_k)$ relative to their corresponding variable, one has then

$$f_X(\mathbf{x}) = \prod_{k=1}^{d} f_X^k(x_k). \tag{2.5}$$

For instance, assume one wants to perform a tensor product of the two following one-dimensional polynomial bases of two independent variables:

$$\mathbf{x} = \Big[x_1, x_2\Big] \quad \text{with} \quad \{P_j(\mathbf{x})\} = \{P_j^1(x_1)\} \otimes \{P_j^2(x_2)\}. \tag{2.6}$$

If each basis contains two polynomials, the multi-index matrix as well as the three firsts polynomials of the resulting two-dimensional basis are given by

$$\mathbf{I} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 1 \\ 2 & 2 \end{bmatrix} \quad \Leftrightarrow \quad \begin{cases} P_1(\mathbf{x}) = P_1^1(x_1)P_1^2(x_2), \\ P_2(\mathbf{x}) = P_1^1(x_1)P_2^2(x_2), \\ P_3(\mathbf{x}) = P_2^1(x_1)P_1^2(x_2). \end{cases} \tag{2.7}$$

Full tensor product bases of high dimensions may contain a large number of polynomials, some truncation techniques are thus explored in future sections. In the following, we will describe the so-called three term recurrence relationship for constructing orthogonal bases. Indeed, provided the distribution is univariate, any set of polynomials forming an orthogonal basis satisfies the relation

$$P_{j+1}(x) = (x - a_j)P_j(x) - b_j P_{j-1}(x), \tag{2.8}$$

where the recurrence coefficients a_i and b_i are strictly positive and given by

$$a_j = \frac{\langle xP_j, P_j \rangle}{\langle P_j, P_j \rangle}, \qquad b_j = \frac{\langle P_j, P_j \rangle}{\langle P_{j-1}, P_{j-1} \rangle}. \tag{2.9}$$

In addition, the second equation allows a recursive computation of the L^2-norms:

$$||P_j||^2 = \langle P_j, P_j \rangle = b_j \langle P_{j-1}, P_{j-1} \rangle . \tag{2.10}$$

These coefficients have been tabulated for a number of weight functions in the Wiener-Askey scheme [12]. Since one wants the first polynomial to be zero order, the first elements in the relation are $P_1(x) = 1$ and $b_1 = 1$.

2.2.2 Dependent Variables

When the variables are dependent, the joint weight function can no longer be considered as the product of independent one-dimensional weight functions and the formulation presented previously is not valid. The most straightforward way to deal with variable dependencies is to use the Gram-Schmidt orthogonalization [13][14]. This procedure takes a non-orthogonal set of linearly independent functions and constructs an orthogonal basis over an arbitrary domain with respect to an arbitrary weight function. Given an original set of linearly independent functions $\{B_i(\mathbf{x})\}$, the orthogonal basis $\{P_i(\mathbf{x})\}$ is computed iteratively by

$$P_i(\mathbf{x}) = B_i(\mathbf{x}) - \sum_{j=1}^{i-1} \frac{\langle B_i, P_j \rangle}{\langle P_j, P_j \rangle} P_j(\mathbf{x}). \tag{2.11}$$

It turns out that the classical GS algorithm suffers from numerical instability, round-off errors can accumulate and destroy orthogonality of the resulting vectors. Indeed, if an error is made in computing $P_2(\mathbf{x})$ so that $\langle P_1, P_2 \rangle = \varepsilon$ is non-zero, this error will not be corrected in any of the computations that follows. Thereby, the modified GS is proposed to help remedy this issue [15]. The idea is to treat the vectors simultaneously rather than sequentially in order to suffer from round-off instability at a significantly less degree. Both methods are presented in Algorithms 1 and 2.

Algorithm 1: Classical GS

Compute $\{\mathbf{b}_i\} = \{B_i(\mathbf{x})\}$

for $i = 1 : n$ do
  $\mathbf{a}_i = \mathbf{b}_i$
  for $j = 1 : i - 1$ do
    $\mathbf{a}_i = \mathbf{a}_i - \langle \mathbf{p}_j, \mathbf{b}_i \rangle \, \mathbf{p}_j$
  end
  $\mathbf{p}_i = \mathbf{a}_i / \sqrt{\langle \mathbf{a}_i, \mathbf{a}_i \rangle}$
end

Output $\{\mathbf{p}_i\} = \{P_i(\mathbf{x})\}$

Algorithm 2: Modified GS

Compute $\{\mathbf{b}_i\} = \{B_i(\mathbf{x})\}$

for $i = 1 : n$ do
  $\mathbf{a}_i = \mathbf{b}_i$
end

for $i = 1 : n$ do
  $\mathbf{p}_i = \mathbf{a}_i / \sqrt{\langle \mathbf{a}_i, \mathbf{a}_i \rangle}$
  for $j = i + 1 : n$ do
    $\mathbf{a}_j = \mathbf{a}_j - \langle \mathbf{p}_i, \mathbf{a}_j \rangle \, \mathbf{p}_i$
  end
end

Output $\{\mathbf{p}_i\} = \{P_i(\mathbf{x})\}$

The GS procedure can also be interpreted as a matrix factorization known as the QR factorization [16]. Let $\{\mathbf{b}_j\}$ be a discrete polynomial basis of column vectors obtained by storing the response of the $B_j(\mathbf{x})$ at a discrete set of points $\{\mathbf{x}_j\}$, and $\{\mathbf{p}_j\}$ the corresponding orthonormal basis resulting from GS process evaluated at the same set of points, one can assemble the two following matrices:

$$\begin{aligned} \mathbf{B} &= \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b}_n \end{bmatrix}, \\ \mathbf{Q} &= \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_n \end{bmatrix}. \end{aligned} \tag{2.12}$$

Since $\{\mathbf{p}_j\}$ forms an orthonormal basis, $\mathbf{Q}$ is an orthogonal matrix. Consequently, the GS orthogonalization can be rewritten into an equivalent matrix form

$$\mathbf{B} = \mathbf{QR}, \tag{2.13}$$

where $\mathbf{R}$ is an upper triangular matrix whose coefficients are

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & r_{nn} \end{bmatrix} \quad \Leftrightarrow \quad \begin{cases} r_{ij} = \langle \mathbf{p}_i, \mathbf{b}_j \rangle & \text{if} \quad i \leq j, \\ r_{ij} = 0 & \text{if} \quad i > j. \end{cases} \tag{2.14}$$

This relation allows the use of optimized libraries such as `Numpy` to perform the QR factorization and obtain an orthonormal basis from the resulting matrices. To this end, assume one dispose of a quadrature rule $\{\mathbf{x}_j, w_j\}$ for the computation of the inner products involved in the equation (2.11), one can compute the Vandermonde matrix $\mathbf{A}$ and the diagonal weight matrix by

$$\begin{aligned} &\mathbf{A} : A_{ij} = B_j(\mathbf{x}_i), \\ &\mathbf{W} = \operatorname{diag}(\mathbf{w}), \end{aligned} \tag{2.15}$$

where $\mathbf{w}$ is the vector of quadrature weights. Under this assumption that the quadrature rule forms a proper discrete l^2-norm on the span of $\mathbf{A}$, namely

$$\sum_{j=1}^{m} w_j P^2(\mathbf{x}_j) > 0 \quad \forall\ P \in \{P_i(\mathbf{x})\}, \tag{2.16}$$

the columns of the Vandermonde matrix weighted by $\mathbf{w}$ are linearly independent [14] and there is a unique QR factorization such that

$$\mathbf{W}^{1/2}\mathbf{A} = \mathbf{QR} \tag{2.17}$$

and is effectively performing the operation (2.11). One has then

$$P_j(\mathbf{x}) = \sum_{i=1}^{j} B_i(\mathbf{x})(R^{-1})_{ij}. \tag{2.18}$$

2.2.3 Hyperbolic Truncation

The exponential increase of the number of polynomials in tensor product bases may become computationally expensive in high dimensions. This issue can be first addressed by truncating the basis and selecting the polynomials according to the q-norm of their multi-index vectors [17]. The q-norm is defined as

$$||\mathbf{I}_j||_q = \left[\sum_{k=1}^{d} \left(I_{jk} \right)^q \right]^{1/q}, \tag{2.19}$$

where d is the dimension of the polynomials. Unlike adaptive algorithms selecting relevant polynomials according to their correlation with the reference model, hyperbolic truncation schemes allow one to reduce the number of coefficients to be computed in a PCE without evaluating the response of the function to be expanded. The truncation scheme of order p consists of selecting the polynomial basis

$$\{P_j(\mathbf{x})\} : ||\mathbf{I}_j||_q \leq p, \tag{2.20}$$

with $q > 0$. Decreasing q favours low-order interactions, which may be more significant than higher order interaction in the model of interest. An example of different q-norm based truncations in two dimensions is presented in Figure 2.

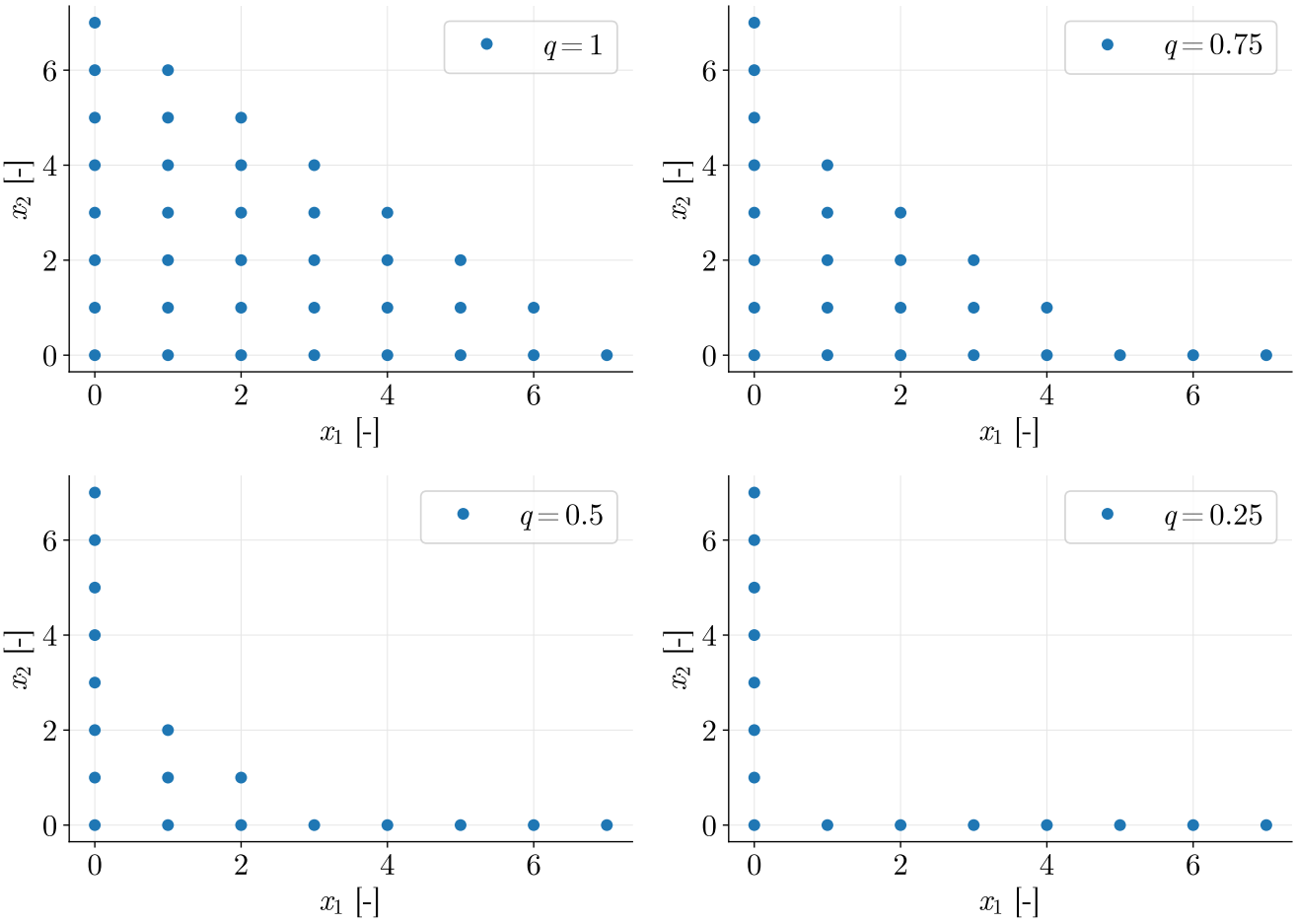

Figure 2: Two dimensional hyperbolic truncation sets of order 7 for different value of q. The two axes represent the exponents of the random variables in the system.

2.3 Computation of the Coefficients

A wide variety of numerical and analytical methods have been proposed to compute the weighting coefficients of the PCE provided the polynomial basis is known. As a reminder, the mapping function y of an arbitrary second-order random variable $Y = y(\mathbf{X})$, i.e. having a non-zero and finite variance, can be expanded as an infinite series of orthogonal polynomials. Practically, the series is truncated after the n-th term:

$$y(\mathbf{x}) = \sum_{j=1}^{\infty} y_j P_j(\mathbf{x}) \simeq \sum_{j=1}^{n} y_j P_j(\mathbf{x}). \tag{2.21}$$

2.3.1 Spectral Projection

The orthogonality property of the polynomials can be exploited to find an analytical expression for the coefficients by performing a spectral projection [8]. Multiplying both sides of the equation (2.21) by $P_i\, f_X$ and integrating over the domain Ω leads to

$$\begin{aligned} & \int_\Omega y(\mathbf{x}) P_i(\mathbf{x})\, d\mu(\mathbf{x}) = \int_\Omega \sum_{j=1}^{n} y_j P_j(\mathbf{x}) P_i(\mathbf{x})\, d\mu(\mathbf{x}) \\ \Leftrightarrow \quad & \int_\Omega y(\mathbf{x}) P_i(\mathbf{x})\, d\mu(\mathbf{x}) = \sum_{j=1}^{n} y_j \langle P_j, P_i \rangle = y_i \langle P_i, P_i \rangle \\ \Leftrightarrow \quad & y_j = \frac{1}{\langle P_j, P_j \rangle} \int_\Omega y(\mathbf{x}) P_j(\mathbf{x})\, d\mu(\mathbf{x}). \end{aligned} \tag{2.22}$$

This integral can then be computed numerically even without explicit knowledge of the weight function by using a MC integration or a quadrature rule. Moreover, if Y is a one-dimensional random variable with cumulative distribution function F_Y, the previous result can be developed a little further. The probability integral transform states that if X is a continuous random variable with cumulative distribution function F_X, then F_X maps to a uniform distribution in $[0, 1]$. Since Y and X are fully dependent by definition, F_Y and F_X map to the same uniform random variable U in $\Omega = [0, 1]$, allowing the following change of variables:

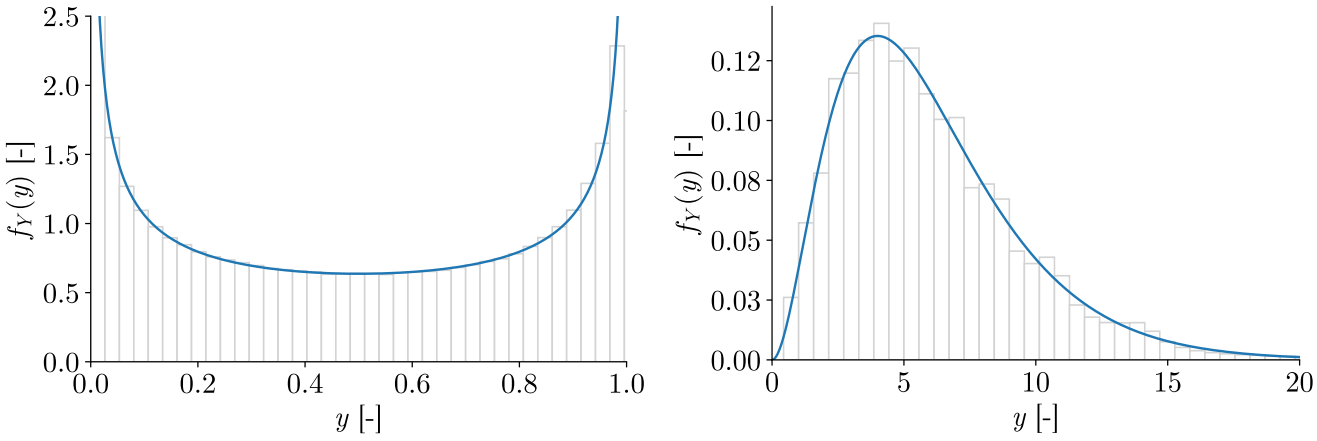

Figure 3: Beta distribution expanded in terms of an exponential distribution on the left. Gamma distribution expanded in terms of a uniform distribution on the right.

$$\begin{cases} y = F_Y^{-1}(u), \\ x = F_X^{-1}(u), \end{cases} \qquad \begin{cases} u = F_X(x) = F_Y(y), \\ du = \dfrac{\partial F_X}{\partial x}(x)\,dx = f_X(x)\,dx. \end{cases} \tag{2.23}$$

Injecting this change of variable in the second integral of equation (2.22) leads to

$$y_j = \frac{1}{\langle P_j, P_j \rangle} \int_0^1 F_Y^{-1}(u) P_j \left[F_X^{-1}(u) \right] \, du, \tag{2.24}$$

which allows the expression of any second order random variable in terms of a substitution variable provided the reciprocals of their cumulative distribution functions exist. Two examples are presented in Figure 3.

2.3.2 Least Squares Regression

An alternative to the spectral projection is the point collocation method [18], which imposes the expanded function to be satisfied at a set of points. The algorithm thus requires a number of collocation points equal to the number of polynomials to solve the linear system of equations. When the number of points is larger than the number of polynomials in the basis, the collocation PCE finds a least squares solution to the overdetermined system of equations. Considering the random variable $\mathbf{X}$ in the sample space Ω, and $Y = y(\mathbf{X})$, where y is a square integrable function in Ω. Let $\{P_i(\mathbf{x})\}$ be an orthogonal basis with respect to the weight function f_X in Ω, the least squares solution is the set of coefficients that minimizes the error

$$\text{Error} = \int_\Omega \left[y(\mathbf{x}) - \sum_{j=1}^{n} y_j P_j(\mathbf{x}) \right]^2 d\mu(\mathbf{x}). \tag{2.25}$$

If one dispose of a sample $\{\mathbf{x}_j\}$ of realisations of $\mathbf{X}$, and if the weight function is the corresponding probability density function, it will be shown in a further chapter that the error can be computed by a MC integration as follows:

$$\text{Error} = \frac{1}{m} \sum_{j=1}^{m} \left[y(\mathbf{x}_j) - \sum_{i=1}^{n} y_i P_i(\mathbf{x}_j) \right]^2. \tag{2.26}$$

The optimal value of y_i is then obtained by cancelling the partial derivatives of the error with respect to the PCE coefficients, leading to a linear system of equations:

$$\begin{aligned} & \sum_{j=1}^{m} \frac{\partial}{\partial y_k} \left[y(\mathbf{x}_j) - \sum_{i=1}^{n} y_i P_i(\mathbf{x}_j) \right]^2 = 0 \\ \Leftrightarrow \quad & \sum_{j=1}^{m} \left\{ -2y(\mathbf{x}_j) P_k(\mathbf{x}_j) + \frac{\partial}{\partial y_k} \left[\sum_{i=1}^{n} y_i P_i(\mathbf{x}_j) \right]^2 \right\} = 0 \\ \Leftrightarrow \quad & \sum_{j=1}^{m} \left\{ -2y(\mathbf{x}_j) P_k(\mathbf{x}_j) + 2P_k(\mathbf{x}_j) \sum_{i=1}^{n} y_i P_i(\mathbf{x}_j) \right\} = 0 \\ \Leftrightarrow \quad & \sum_{j=1}^{m} P_k(\mathbf{x}_j) \sum_{i=1}^{n} y_i P_i(\mathbf{x}_j) = \sum_{j=1}^{m} y(\mathbf{x}_j) P_k(\mathbf{x}_j) \\ \Leftrightarrow \quad & \mathbf{A}^\top \mathbf{A} \mathbf{y} = \mathbf{A}^\top \mathbf{b}, \end{aligned} \tag{2.27}$$

where $\mathbf{A}$ is the Vandermonde matrix, $\mathbf{y}$ is the vector of unknown PCE coefficients and $\mathbf{b}$ is the vector of model response given by

$$\begin{aligned} &\mathbf{A} : A_{ij} = P_j(\mathbf{x}_i), \\ &\mathbf{b} : b_j = y(\mathbf{x}_j). \end{aligned} \tag{2.28}$$

Similarly, the error can also be estimated with a quadrature rule $\{w_j, \mathbf{x}_j\}$ leading to

$$\text{Error} = \sum_{j=1}^{m} \Big[y(\mathbf{x}_j) - \sum_{i=1}^{n} y_i P_i(\mathbf{x}_j) \Big]^2 w_j. \tag{2.29}$$

Then, the coefficients are obtained by cancelling the partial derivatives of the error with respect to the coefficients, leading to the following linear system of equations:

$$\begin{aligned} &\sum_{j=1}^{m} P_k(\mathbf{x}_j) \sum_{i=1}^{n} y_i P_i(\mathbf{x}_j) w_j = \sum_{j=1}^{m} y(\mathbf{x}_j) P_k(\mathbf{x}_j) w_j \\ \Leftrightarrow \quad &\mathbf{A}^\top \mathbf{W} \mathbf{A} \mathbf{y} = \mathbf{A}^\top \mathbf{W} \mathbf{b}, \end{aligned} \tag{2.30}$$

where $\mathbf{W}$ is the diagonal weight matrix. Furthermore, if the quadrature weights are all positive, solving (2.30) is equivalent to finding a least squares solution to the following overdetermined system:

$$\mathbf{W}^{1/2} \mathbf{A} \mathbf{y} = \mathbf{W}^{1/2} \mathbf{b}. \tag{2.31}$$

The convergence of the solution can be improved by computing the error with a quasi-MC integration. Either by generating a low-discrepancy sequence for the distribution related to f_X using the inverse probability transform [19], or by generating a uniform grid and use f_X as a weight function. In the latter case, it will be shown that

$$\mathbf{W} : W_{ij} = \frac{v}{m} f_X(\mathbf{x}_j) \delta_{ij}, \tag{2.32}$$

where v is the volume of the domain:

$$v = \int_\Omega d\mathbf{x}. \tag{2.33}$$

2.3.3 Least Angle Regression

In addition to compute the PCE coefficients, the LARS algorithm can efficiently select the relevant polynomials to be used in the surrogate model, acting as a truncation technique. The principle of the algorithm is to move the coefficient estimates in the direction where the polynomial is the most correlated with the remaining residual [20][17]. First, the constant predictor is removed from the model, which amounts to remove the first column of $\mathbf{A}$. The input data are then standardized and we define

$$\begin{aligned} &\mathbf{b}' : b'_j = b_j - \bar{b}, \\ &\mathbf{A}' : A'_{ij} = \frac{A_{ij} - \bar{A}_j}{\sigma_j}, \end{aligned} \tag{2.34}$$

where $\bar{b}$ is the mean of $\mathbf{b}$, $\bar{\mathbf{A}}$ is the column-wise mean of $\mathbf{A}$ and $\boldsymbol{\sigma}$ the column-wise standard deviation of $\mathbf{A}$. In the following, the prime symbol of the standardized quantities will be omitted for readability reason. In the initial step, the model is empty and the residual is $\mathbf{r}_1 = \mathbf{b}$, one then select the index of the most correlated polynomial:

$$j_1 = \text{argmax}_j \left|\langle \mathbf{A}_j, \mathbf{r}_1 \rangle\right| = \text{argmax}_j \left|\mathbf{A}_j^\top \mathbf{r}_1\right| \tag{2.35}$$

where $\mathbf{A}_j = \text{col}_j(\mathbf{A})$ is the response of the predictor j in the standardized Vandermonde matrix. The direction $\mathbf{d}_i$ at the i-th step is obtained by solving

$$\mathbf{A_J}^\top \mathbf{A_J}\, \mathbf{d}_i = \mathbf{A_J}^\top \mathbf{r}_{i-1}, \tag{2.36}$$

where $\mathbf{A_J} = \text{col}_\mathbf{J}(\mathbf{A})$ is the matrix formed by the columns of indices $\mathbf{J} = [j_1, j_2, \ldots, j_i]$ corresponding to the predictors in the model at the i-th step. The coefficient vector is then updated with

$$\mathbf{y}_i(\alpha) = \mathbf{y}_{i-1} + \alpha \mathbf{d}_i, \tag{2.37}$$

where $\alpha \in [0, 1]$ represents how far the estimate of $\mathbf{y}$ moves in the direction $\mathbf{d}_i$ before another estimator enters the model. One chooses α at the i-th step by finding the smallest value of α such that the angle between the remaining residual

$$\mathbf{r}_i(\alpha) = \mathbf{r}_{i-1} - \alpha \mathbf{A} \mathbf{d}_i \tag{2.38}$$

and one of the variables not in the model equals the angle between $\mathbf{r}_i(\alpha)$ and a predictor in the model. Practically, α is chosen such that

$$\langle \mathbf{r}_i(\alpha), \mathbf{A}_k \rangle = \langle \mathbf{r}_i(\alpha), \mathbf{A}_j \rangle, \tag{2.39}$$

with $j \in \mathbf{J}$ being the index of a predictor in the model at the i-th step and k the index of a predictor out of the model. The solution to this equation is

$$\alpha_k^+ = \frac{\langle \mathbf{r}_{i-1}, \mathbf{A}_j \rangle - \langle \mathbf{r}_{i-1}, \mathbf{A}_k \rangle}{\langle \mathbf{r}_{i-1}, \mathbf{A}_j \rangle - \langle \mathbf{A}\mathbf{d}_i, \mathbf{A}_j \rangle}. \tag{2.40}$$

Similarly, the angle between $\mathbf{r}_i(\alpha)$ and $-\mathbf{A}_k$ equals the angle between $\mathbf{r}_i(\alpha)$ and $\mathbf{A}_j$ when

$$\alpha_k^- = \frac{\langle \mathbf{r}_{i-1}, \mathbf{A}_j \rangle + \langle \mathbf{r}_{i-1}, \mathbf{A}_k \rangle}{\langle \mathbf{r}_{i-1}, \mathbf{A}_j \rangle + \langle \mathbf{A}\mathbf{d}_i, \mathbf{A}_j \rangle}. \tag{2.41}$$

Finally, $\mathbf{J}$ is updated for the next step and the new predictor enters the model:

$$\begin{aligned} &\alpha = \min\left\{\alpha_k^+, \alpha_k^-\right\} \in [0, 1] \quad \text{for} \quad k \notin \mathbf{J}, \\ &j_{i+1} = k \quad \Rightarrow \quad \mathbf{J} = \begin{bmatrix} j_1 & j_2 & \ldots & j_{i+1} \end{bmatrix}. \end{aligned} \tag{2.42}$$

Once the desired number of predictors has entered the model, the LARS algorithm ends. The last step is to rescale the vector of PCE coefficients thus obtained to the original non-standardized problem:

$$y'_1 = \bar{b} - \sum_{j=2}^{n} \frac{y_j}{\sigma_j} \quad \text{and} \quad y'_j = y_j/\sigma_j \quad \text{for} \quad j > 1. \tag{2.43}$$

Generally, the coefficients corresponding to the selected polynomials are recomputed by a least squares algorithm to improve the accuracy of the LARS [11]. A summary of the procedure is presented in algorithm 3.

Algorithm 3: Least angle regression

Standardize $\mathbf{A}$ and $\mathbf{b}$
Initialize $\mathbf{d} = \mathbf{0}$ and $\mathbf{r} = \mathbf{b}$
$\mathbf{J} = j_1 = \text{argmax}_j \, |\langle \mathbf{A}_j, \mathbf{r}_1 \rangle|$

while $\alpha < 1$ or (not enough predictors in $\mathbf{J}$) do

$\mathbf{d}[\mathbf{J}] = \left(\mathbf{A_J}^\top \mathbf{A_J}\right)^{-1} \mathbf{A_J}^\top \mathbf{r}$
$\alpha = \min\left\{\alpha_k^+, \alpha_k^-\right\}$ with $k \notin \mathbf{J}$

Add k to the array $\mathbf{J}$
$\mathbf{y} = \mathbf{y} + \alpha \mathbf{d}$
$\mathbf{r} = \mathbf{r} - \alpha \mathbf{A} \mathbf{d}$

end

Rescale the output $\mathbf{y}$ to the original problem

2.4 Sensitivity Analysis

The PCE provides an analytical expression for some of its statistical moments [21]. Indeed, due to the orthogonality of the polynomial basis, a particular property of the PCE is that the total variance and expectation of the truncated expansion of $Y = y(\mathbf{X})$ can be directly obtained from the coefficients by

$$\begin{aligned} \mathrm{E}(Y) &= \int_\Omega y(\mathbf{x}) \, d\mu(\mathbf{x}) = y_1, \\ \mathrm{Var}(Y) &= \int_\Omega \left[y(\mathbf{x}) - \mathrm{E}(Y)\right]^2 d\mu(\mathbf{x}) = \sum_{j=2}^{n} y_j^2 \, \langle P_j, P_j \rangle, \end{aligned} \tag{2.44}$$

meaning that the total variance of the output of the function is decomposed into a sum of squared coefficients related to one or more random variables, i.e. into a sum of variances associated with each group of input variables. If the polynomials are normalized, the expression of the variance reduces to

$$\mathrm{Var}(Y) = \sum_{j=2}^{n} y_j^2. \tag{2.45}$$

2.4.1 Sobol Indices

The analysis of variance decomposition consists in identifying the shares of variance of the output Y associated with the different random variables [22]. The first order Sobol indices quantify the variance of the conditional expectation, normalised by the total variance, of the output given the value of an input. The k-th index is thus given by

$$S_k = \frac{\mathrm{Var}\left[\mathrm{E}\left(Y|X_k\right)\right]}{\mathrm{Var}(Y)} = \sum_j \frac{y_j^2}{\mathrm{Var}(Y)} \quad \forall j : \mathbf{I}_j \in \mathbf{k}, \tag{2.46}$$

where $\mathbf{I}$ is the multi-index matrix, $\{y_j\}$ corresponds to the set of coefficients of polynomials depending only on the variable x_k and $\mathbf{k}$ is the corresponding subset of multi-indices. Following the same reasoning, the total order indices quantify the complementary of the variance of the conditional expectation, normalised by the total variance, given the values of all outputs but the one considered. One has thus

$$S_{Tk} = 1 - \frac{\mathrm{Var}\left[\mathrm{E}\left(Y|\mathbf{X}_{\sim k}\right)\right]}{\mathrm{Var}(Y)} = \sum_j \frac{y_j^2}{\mathrm{Var}(Y)} \quad \forall j : \mathbf{I}_j \in \mathbf{k}_T, \tag{2.47}$$

where $\mathbf{X}_{\sim k}$ denotes the vector of all random variables but X_k, the set $\{y_j\}$ corresponds to the coefficients of polynomials depending at least on the variable x_k and $\mathbf{k}_T$ is the corresponding subset of multi-indices. While the ANOVA decomposition allows explaining the variance provided the variables are independent, the ANCOVA decomposition is an extension to the case of correlated variables [21][23]. The variance of the model is decomposed in a first part related to the model structure and a second part related to the dependence of the variables. In the following, the PCE will be developed as

$$y(\mathbf{x}) = \sum_{j=1}^{n} y_j P_j(\mathbf{x}) = \sum_{j=1}^{m} a_j x_{\mathbf{I}_j}, \tag{2.48}$$

where $x_{\mathbf{I}_j}$ is the monomial of multi-index $\mathbf{I}_j$. Moreover, we define the function $g(x_{\mathbf{k}})$ as the part of (2.48) depending only on a particular monomial and its powers, with $\mathbf{k}$ the corresponding subset of multi-index vectors:

$$\begin{aligned} g(x_{\mathbf{k}}) &= \sum_j a_j x_{\mathbf{I}_j} \quad \forall j : \mathbf{I}_j \in \mathbf{k}, \\ g(x_{\sim\mathbf{k}}) &= \sum_{j=1}^{m} a_j x_{\mathbf{I}_j} - g(x_{\mathbf{k}}). \end{aligned} \tag{2.49}$$

The variance of the model response can be expressed as the covariance between the exact model and its polynomial representation:

$$\begin{aligned} \mathrm{Var}(Y) = \mathrm{Cov}(Y,Y) &= \mathrm{Cov}\left(Y, \sum_{j=1}^{m} a_j X_{\mathbf{I}_j}\right) = \mathrm{Cov}\left(Y, \sum_{\mathbf{k}} g(X_{\mathbf{k}})\right) \\ &= \sum_{\mathbf{k}} \left\{ \mathrm{Var}\left[g(X_{\mathbf{k}})\right] + \mathrm{Cov}\left[g(X_{\mathbf{k}}), g(X_{\sim\mathbf{k}})\right] \right\}, \end{aligned} \tag{2.50}$$

where the first contribution to the sum is the ANOVA structural part and a second part is related to the dependence structure of the variables. As a reminder, the covariance between two continuous random variables $Y = y(\mathbf{X})$ and $G = g(\mathbf{X})$ is obtained by computing the integral

$$\mathrm{Cov}(Y, G) = \int_{\Omega} \Big[y(\mathbf{x}) - E(Y)\Big]\Big[g(\mathbf{x}) - E(G)\Big]\, d\mu(\mathbf{x}). \tag{2.51}$$

This separation of contribution is also known as the ANCOVA decomposition, the following triplet of sensitivity indices has been proposed to describe the contributions of each variable in the expansion:

$$\begin{aligned} S_S(\mathbf{k}) &= \mathrm{Var}\Big[g(X_{\mathbf{k}})\Big] / \mathrm{Var}(Y), \\ S_T(\mathbf{k}) &= \mathrm{Cov}\Big[Y, g(X_{\mathbf{k}})\Big] / \mathrm{Var}(Y), \\ S_C(\mathbf{k}) &= \mathrm{Cov}\Big[g(X_{\mathbf{k}}), g(X_{\sim\mathbf{k}})\Big] / \mathrm{Var}(Y), \end{aligned} \tag{2.52}$$

where S_S is the structural contribution index, S_T is the total contribution index and S_C is the correlative contribution index. Finally, the ANCOVA indices verify

$$\begin{aligned} &S_T(\mathbf{k}) = S_S(\mathbf{k}) + S_C(\mathbf{k}), \\ &\sum_{\mathbf{k}} S_T(\mathbf{k}) = 1. \end{aligned} \tag{2.53}$$

For example, one can have the particular functional decomposition

$$\begin{aligned} y(\mathbf{x}) &= 3 + x_1 + x_2 + x_2^2 + x_1 x_2 \\ &= g(1) + g(x_1) + g(x_2) + g(x_1 x_2) \\ &= \textstyle\sum_{\mathbf{k}} g(x_{\mathbf{k}}), \end{aligned} \tag{2.54}$$

where the random input vector $\mathbf{X}$ follows a bivariate normal distribution of mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, contains three different monomials and so three ANCOVA indices. The estimation of the triplet of sensitivity indices is presented in Table 1.

$$\boldsymbol{\mu} = \begin{bmatrix} 0 & 0 \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}. \tag{2.55}$$

Index	S_S [-]	S_C [-]	S_T [-]
X_1	0.29	0.23	0.52
X_2	0.1	0.07	0.17
X_1X_2	0.16	0.15	0.31
Total	0.55	0.45	1

Table 1: Triplet of ANCOVA sensitivity indices resulting from the example of functional decomposition with correlated normal random variables.

2.4.2 Shapley Indices

The main drawback of the sensitivity indices derived previously is their difficulty of interpretation when the variables are correlated. Indeed, an index is computed for each group of variables and the contribution from the covariance may be negative. The notion of Shapley value originated in game theory, where this latter gives players a fair distribution of winnings [24][25]. In the context of sensitivity analysis, the goal is to assign to each random variable an index associated with the uncertainty that it brings in the output variable. The Shapley value of the j-th random variable is given by

$$S_j = \frac{1}{d}\sum_{\mathbf{k}} \binom{d-1}{|\mathbf{k}|}^{-1} \Big[\mathcal{V}(\mathbf{k}\cup X_j) - \mathcal{V}(\mathbf{k})\Big] \qquad \forall\, \mathbf{k} \in \{\mathbf{X}/X_j\}, \tag{2.56}$$

where d is the dimension, $\mathbf{k}$ corresponds to a particular subgroup of random variables not containing X_j and $|\mathbf{k}|$ is its size. Finally, the cost function is given by

$$\mathcal{V}(\mathbf{k}) = \mathrm{Var}\Big[\mathrm{E}(Y|\mathbf{k})\Big] / \mathrm{Var}(Y) \tag{2.57}$$

and measures the part of variance of Y caused by the uncertainty of the inputs in $\mathbf{k}$. The cost function thus satisfies the following propriety

$$\begin{aligned} &\mathcal{V}(\varnothing) = 0, \\ &\mathcal{V}(\mathbf{X}) = \mathrm{Var}(Y). \end{aligned} \tag{2.58}$$

Another interesting propriety of the Shapley indices is that they are positive and sum to one, S_j can therefore be interpreted as the part of the total variance due to the j-th variable. For example, the Shapley value related to X_1 in the function (2.54) is

$$S_1 = \frac{1}{2}\left(\mathrm{Var}(Y) - \mathrm{Var}\Big[\mathrm{E}(Y|X_2)\Big] + \mathrm{Var}\Big[\mathrm{E}(Y|X_1)\Big] \right) \mathrm{Var}(Y)^{-1}. \tag{2.59}$$

Using the same distributions for the input random vector, the Shapley indices relative to the different variables of this toy problem are

$$S_1 = 0.38 \quad \text{and} \quad S_2 = 0.62. \tag{2.60}$$

2.5 Conclusion

In conclusion, we presented two methods to generate an orthogonal polynomial basis for the construction of the PCE. The three term recurrence relation builds analytical polynomials which are orthogonal with respect to a product of labelled and independent probability density functions, the GS orthogonalization builds orthogonal polynomials with respect to an arbitrary probability density function from a training sample. In particular, this method allows the construction of the basis with respect to Bayesian posteriors coming from the inverse problem in `Pybitup`. Once the polynomial basis has been obtained, we propose three methods for computing the PCE coefficients. The spectral projection is the classical approach by projecting y onto the polynomials, the least squares regression and the LARS estimate the coefficients by minimizing a cost function.

The accuracy of their estimate depends on the quality of the polynomial basis at a less significant degree than the spectral projection, and the LARS allows the selection of relevant polynomials in order to reduce the complexity of the surrogate model when the number of dimensions is high. As these algorithms require the computation of definite integrals, it is important to continue the discussion on the different integration methods available as well as on the difficulties encountered in higher dimensions or when one dispose of a chain of samples generated by Bayesian inference. This topic is discussed in the next chapter.

The last part of the chapter was dedicated to the post processing of the PCE by exploiting the functional decomposition of the variance. The computation of Sobol sensitivity indices is straightforward, but limited to independent input variables. The ANCOVA sensitivity indices are a generalisation of the Sobol indices to dependent variables, but can be difficult to interpret as the contribution from the covariance may be negative. Finally, the Shapley indices provide a clear interpretation of the shares of variance of the output associated with the different random variables, but may become difficult to estimate when the number of input variables increases.

Chapter 3

Integration Methods

3.1 Introduction

Various Gaussian quadrature rules are available for computing one-dimensional integrals or multidimensional integrals for independent variables. For instance, a well known example is the Gauss-Hermite quadrature rule, which approximates the value of an integral in $\mathbb{R}$ according to the weight function

$$f_X(x) = \frac{e^{-x^2/2}}{\sqrt{2\pi}}. \tag{3.1}$$

It is clear that f_X is nothing but the probability density function of a normal distribution of zero mean and unitary variance. This quadrature rule can thus be used to compute the PCE coefficients of a random variable $Y = y(\mathbf{X})$ related to some independent normal random parameters $\mathbf{X}$. For instance, by estimating the value of the error in (2.25) or the inner product in (2.22). Others Gaussian quadrature rules related to different distributions can be computed from three term recurrence relations [10], but suffer from a curse of dimensionality in addition to be limited to independent random variables in multiple dimensions. A common alternative for high dimensional integrals is the MC integration [26]. Indeed, the promised advantage of MC integration against most deterministic methods is the independence of the computational cost to the dimensions of the integral, but the slow convergence of the method may still require a large number of points to reach the desired accuracy. Numerous variance reduction techniques such as the quasi-MC have been developed in order to increase this convergence rate.

Recently, different sparse quadrature rules have been developed in order to overcome the exponential increase of the number of points in Gaussian quadratures and to provide an alternative to classical MC integration techniques. Indeed, it will be observed in a further chapter that the embedded quadrature based on approximate Fekete points [27][28] allows the construction of an accurate PCE with a smaller number of points than MC integrations. As Fekete points are obtained from the resolution of an expensive optimization problem, we will seek an approximation of them through the factorization of the Vandermonde matrix. In addition, these new methods allow the generation of an embedded quadrature rule for dependent random variables.

3.2 Monte Carlo Integration

Let $\mathbf{X}$ be a random vector of joint probability density functions f_X defined in the sample space Ω, and let $\{\mathbf{x}_j\}$ be a set of some realisations of $\mathbf{X}$. Finally, let $Y = y(\mathbf{X})$, where y is a square integrable function in Ω. By the law of large numbers [29], one can write

$$\int_\Omega y(\mathbf{x})\, d\mu(\mathbf{x}) = E\Big[y(\mathbf{X})\Big] = \lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} y(\mathbf{x}_j). \tag{3.2}$$

Afterwards, assume one wants to compute this integral using the samples of a random vector $\mathbf{Z}$ with an arbitrary probability density function f_Z defined over the same domain. Following the same reasoning, one can write

$$\int_\Omega y(\mathbf{x})\, d\mu(\mathbf{x}) = \int_\Omega \frac{y(\mathbf{x}) f_X(\mathbf{x})}{f_Z(\mathbf{x})} f_Z(\mathbf{x})\, d\mathbf{x} = E\left[\frac{y(\mathbf{Z})\, f_X(\mathbf{Z})}{f_Z(\mathbf{Z})}\right]. \tag{3.3}$$

For instance, this expression allows the use of a uniform grid of points or samples uniformly distributed on Ω to estimate the integral, leading to $f_Z(\mathbf{x}) = 1/v$ and to the simplification of (3.3) into the classical MC integration formula:

$$\int_\Omega y(\mathbf{x})\, d\mu(\mathbf{x}) = v\, E\Big[y(\mathbf{Z}) f_X(\mathbf{Z})\Big] = \lim_{n\to\infty} \frac{v}{n} \sum_{j=1}^{n} y(\mathbf{z}_j) f_X(\mathbf{z}_j). \tag{3.4}$$

where $\{\mathbf{z}_j\}$ are uniformly distributed points in Ω and v is the volume of the domain. However, if most of the contributions to the integral comes from a small region of the integration volume, there will be only a few significant points there, which can lead to large statistical errors. The result can thus be greatly improved if the sampling points are chosen such that

$$f_Z(\mathbf{x}) \propto y(\mathbf{x}) f_X(\mathbf{x}), \tag{3.5}$$

which concentrates the points where the function to be integrated is the largest. This method is the so-called importance sampling [26]. To simplify the importance sampling process in one dimension, one can use the probability integral transform with a change of variable to obtain the following relation:

$$\int_\Omega y(x)\, d\mu(x) = E\left\{ \frac{y\left[F_Z^{-1}(U)\right] f_X\left[F_Z^{-1}(U)\right]}{f_Z\left[F_Z^{-1}(U)\right]} \right\}, \tag{3.6}$$

where U is a uniform random variable in $[0, 1]$. Thereby, MC integration allows the computation of definite integrals when an analytical solution is impossible, or when a closed form of f_X is not known explicitly.

3.2.1 Quasi-Monte Carlo

The convergence rate can be improved by different variance reduction techniques. In particular, the so called quasi-MC integration uses deterministic sequences where the points are maximally self-avoiding. The simplest approach to obtain this property is to uniformly partitioning each dimension of a domain, but this approach suffers from several drawbacks for the numerical integration in high dimension. Thereby, different low-discrepancy sequences (Figure 4) were proposed in the literature to help remedy this problem [19]. The Halton sequence is a generalisation of the van der Corput sequence in multiple dimensions. This latter relies on the fact that a positive integer x can be expressed in a base b with a sequence of digits uniquely determined by

$$x = \sum_{j=1}^{n} d_j(x)\, b^{j-1}, \tag{3.7}$$

where $d_j(x) \in [0, b-1]$. By introducing the radical inverse function ψ, defined in the base b as the function that converts any positive integers x to a fractional value in $[0, 1[$ according to the following relation:

$$\psi_b(x) = 0.\, d_1(x) d_2(x) \ldots d_n(x). \tag{3.8}$$

The points of the van der Corput sequence on the interval $[0, 1]$ are then obtained by the radical inverse in base 2 for different integers:

$$x_j = \psi_2(j), \tag{3.9}$$

and the Halton sequence is obtained by generating these sequences for different bases in each dimension. Since the bases must all be relatively prime to each other, a common choice is to use the first prime numbers:

$$\mathbf{x}_j = \begin{bmatrix} \phi_2(j) & \phi_3(j) & \dots & \phi_p(j) \end{bmatrix}. \tag{3.10}$$

A second example of simple and efficient quasi-random sequence is the additive recurrence R-sequence, this latter is based on irrational numbers [30]. The j-th point of the sequence is computed by

$$\mathbf{x}_j = \Big\{ 0.5 + \boldsymbol{\alpha}(1 + j) \Big\}, \tag{3.11}$$

where $\{a\}$ denotes the decimal part of a and $\boldsymbol{\alpha}$ is a vector such that

$$\boldsymbol{\alpha} : \alpha_k = \phi_d^{-(1+k)}, \tag{3.12}$$

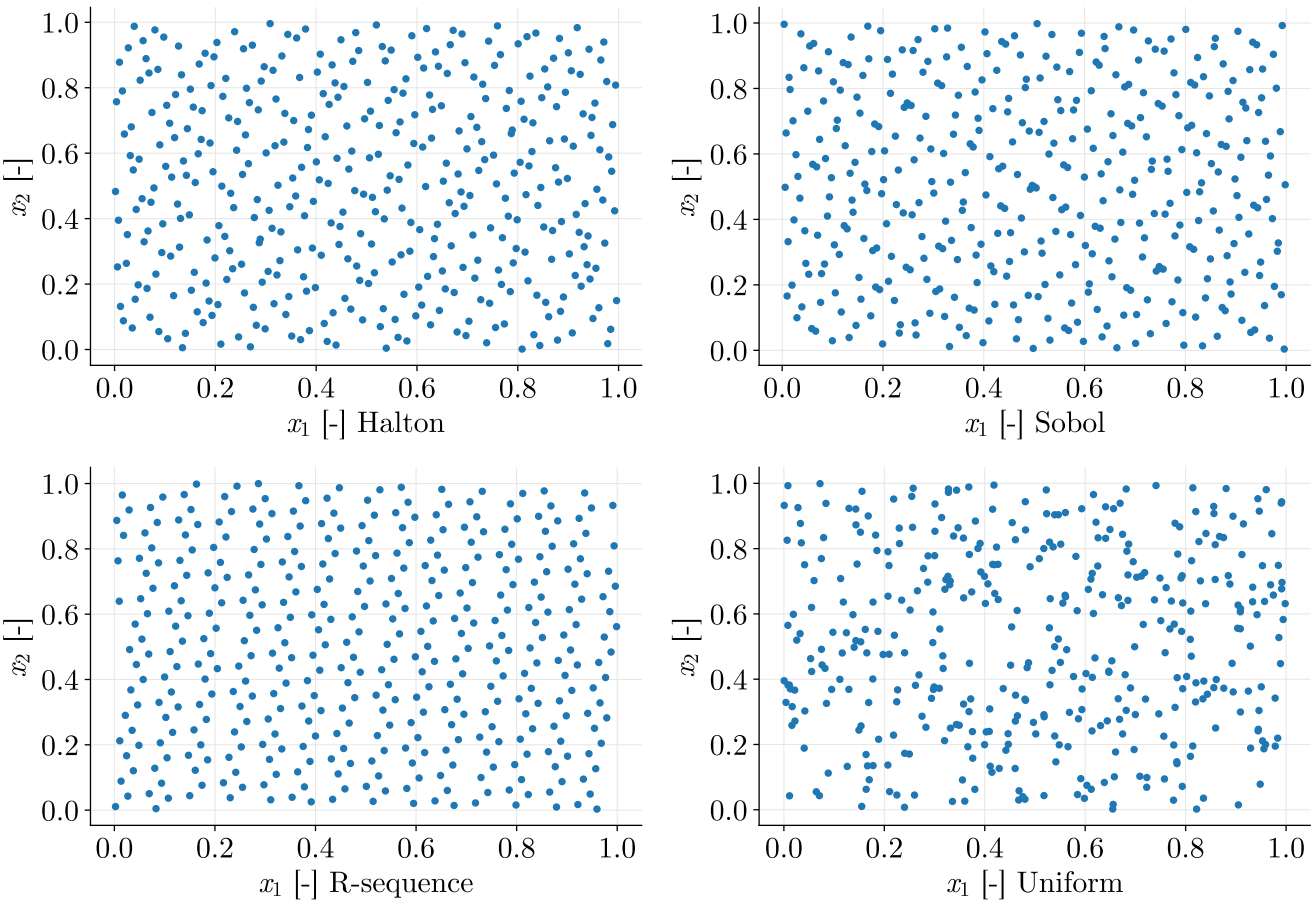

Figure 4: Comparison between Halton, Sobol, R-sequences of quasi-random numbers and a pseudo-random uniform distribution on the unit square in two dimensions.

where d is the number of dimensions of the domain, $k \in [1, d]$ and ϕ_d is the generalized golden ratio defined as the unique positive solution to

$$x^{d+1} = x + 1. \tag{3.13}$$

Finally, another possibility is the so-called Sobol sequence [31], which constructs the points based on successive partitions of the interval and reorder the coordinates in each dimension. These different methods are illustrated in Figure 4.

3.2.2 Quadratic Resampling

The moment matching method consists in correcting a sample set of realisations of $\mathbf{X}$ in order to match the actual distribution moments, for instance, precomputed with more accuracy on a larger sample set. The quadratic resampling generalizes this method to correlated random variables [32]. Assume that the exact expectation $E(\mathbf{X})$ and the covariance matrix $\mathbf{\Sigma}$ of the distribution are known, one can compute an empirical mean and covariance matrix of the sample $\mathbf{x}$ as

$$\begin{aligned} \bar{\mathbf{x}} &= \frac{1}{n} \sum_{j=1}^{n} \mathbf{x}_j, \\ \mathbf{\Sigma}' &= E\left[(\mathbf{x} - \bar{\mathbf{x}}) (\mathbf{x} - \bar{\mathbf{x}})^\top \right]. \end{aligned} \tag{3.14}$$

Due to the finite number of points in the sample, the empirical moments are different from the exact distribution moments and can lead to large statistical errors. We first define the gain matrix:

$$\mathbf{H} = \sqrt{\mathbf{\Sigma}} \sqrt{\mathbf{\Sigma}'}^{-1}. \tag{3.15}$$

Then, the method of quadratic resampling consists in computing the corrected sample $\mathbf{x}'$ with first and second order moments equals to $E(\mathbf{X})$ and $\mathbf{\Sigma}$ that can be used in place of the original sample in the quadrature formula. This latter is exact for any polynomial of order two or less:

$$\mathbf{x}' = (\mathbf{x} - \bar{\mathbf{x}}) \, \mathbf{H}^\top + E(\mathbf{X}). \tag{3.16}$$

3.3 Quadrature Rules

While there exist many powerful algorithms for the numerical evaluation of one-dimensional integrals such as trapezoidal, Simpson or Gaussian quadrature rules, it is often difficult to find computationally efficient methods for multidimensional integrals even in standard domains. As a reminder, if $\mathbf{x}_j$ are the quadrature points, w_j their associated weights and m the number of points, a quadrature rule allows the estimation of a definite integral with the following relation:

$$\int_\Omega y(\mathbf{x}) \, d\mu(x) = \sum_{j=1}^{m} w_j \, y(\mathbf{x}_j). \tag{3.17}$$

3.3.1 Recurrence Relations

When the polynomial basis is obtained by a three term recurrence relation (2.8), a one-dimensional quadrature rule can be computed using the coefficients and the Golub-Welsch algorithm [10] [33], then generalized to multidimensional spaces by tensor product provided the random variables are independents. Let x_j be a point of the rule, the three term recurrence relation can be identified with

$$\begin{bmatrix} x_j P_1(x_j) \\ x_j P_2(x_j) \\ x_j P_3(x_j) \\ \vdots \\ x_j P_n(x_j) \end{bmatrix} = \begin{bmatrix} a_1 & 1 & 0 & 0 & \dots & 0 \\ b_2 & a_2 & 1 & 0 & \dots & 0 \\ 0 & b_3 & a_3 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & a_n \end{bmatrix} \begin{bmatrix} P_1(x_j) \\ P_2(x_j) \\ P_3(x_j) \\ \vdots \\ P_n(x_j) \end{bmatrix} \tag{3.18}$$

$$\Leftrightarrow \quad x_j \mathbf{P}(x_j) = \mathbf{T}\mathbf{P}(x_j).$$

Namely, the points x_j are the eigenvalues of $\mathbf{T}$. If the polynomials are not orthonormals, $\mathbf{T}$ is not symmetric and one can perform a diagonal similarity transformation $\mathbf{D}\mathbf{T}\mathbf{D}^{-1} = \mathbf{J}$ which yields to a symmetric tridiagonal matrix $\mathbf{J}$, also known as the Jacobi matrix:

$$\mathbf{J} = \begin{bmatrix} a_1 & \sqrt{b_2} & 0 & 0 & \dots & 0 \\ \sqrt{b_2} & a_2 & \sqrt{b_3} & 0 & \dots & 0 \\ 0 & \sqrt{b_3} & a_3 & \sqrt{b_4} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & a_n \end{bmatrix}. \tag{3.19}$$

As the matrices $\mathbf{T}$ and $\mathbf{J}$ are similar, they share the same spectrum and the same characteristic polynomial. The orthonormal version of (3.18) is thus

$$x_j \mathbf{P}(x_j) = \mathbf{J}\mathbf{P}(x_j), \tag{3.20}$$

where $\mathbf{P}(x)$ are orthonormal polynomials, and the quadrature rule verifies by definition:

$$\sum_{j=1}^{n} w_j P_i(x_j) P_k(x_j) = \langle P_i, P_k \rangle = \delta_{ik}. \tag{3.21}$$

Introducing the Vandermonde matrix and the diagonal weight matrix, the previous equality can be rewritten as follows:

$$\begin{aligned} & \mathbf{A}\mathbf{W}\mathbf{A}^\top = \mathbb{I} \\ \Leftrightarrow \quad & \mathbf{W}^{-1} = \mathbf{A}\mathbf{A}^\top \\ \Leftrightarrow \quad & \frac{1}{w_j} = \sum_i P_i(x_j)^2 = ||\mathbf{P}(x_j)||^2. \end{aligned} \tag{3.22}$$

On the other hand, computing the orthonormal eigenvectors of $\mathbf{J}$ gives

$$\begin{aligned} & \mathbf{J}\mathbf{v}_j = x_j \mathbf{v}_j \\ & \mathbf{v}_j = \begin{bmatrix} v_{j1} & v_{j2} & \dots & v_{jn} \end{bmatrix}^\top, \end{aligned} \tag{3.23}$$

and since the first polynomial of the orthonormal basis is given by $P_1(x) = 1$, it can be shown there exists a constant c such that

$$\begin{aligned} & \mathbf{v}_j = c\mathbf{P}(x_j) \\ \Leftrightarrow \quad & v_{j1} = cP_1(x_j) = c. \end{aligned} \tag{3.24}$$

Injecting $c = v_{j1}$ in the equation (3.24) leads to

$$\mathbf{P}(x_j) = \mathbf{v}_j / v_{j1}, \tag{3.25}$$

and the quadrature weight w_j associated with the point x_j can be computed using equation (3.22) and the fact that the eigenvectors $\mathbf{v}_j$ have a unit norm:

$$w_j = ||\mathbf{P}(x_j)||^{-2} = ||\mathbf{v}_j / v_{j1}||^{-2} = v_{j1}^2. \tag{3.26}$$

Finally, a multidimensional quadrature rule is obtained by performing the tensor product of the one-dimensional quadrature points and weight vectors, where k is the index of the variable and d is the dimension:

$$\{w_j, \mathbf{x}_j\} = \bigotimes_{k=1}^{d} \left\{ w_j^k, \mathbf{x}_j^k \right\}, \tag{3.27}$$

3.3.2 Approximate Fekete Points

The previous multidimensional quadrature rule was obtained from the tensor product assuming the random variables were independent. When the random variables are dependent, selecting a set of points for multidimensional polynomial interpolation is still an open problem. For instance, the Fekete points are a good interpolation set which may be defined for any compact set and dimension, but requires an expensive and challenging multivariate optimization [28][34]. Therefore, we will seek for approximating Fekete points by solving the discrete optimization problem of extracting a square submatrix $\mathbf{F}$ of maximum volume from the Vandermonde matrix [27], namely

$$\mathbf{F} = \mathrm{row}_{\mathbf{J}}(\mathbf{A}) : \mathbf{J} = \mathrm{argmax}_{\mathbf{J}} \left(\det \left[\mathrm{row}_{\mathbf{J}}(\mathbf{A}) \right] \right), \tag{3.28}$$

where $\mathbf{J}$ is the index vector of the selected rows, and corresponds to the selected points among an original set containing more points than the number of polynomials in the basis. The corresponding quadrature weights are then obtained by solving

$$\mathbf{F}^\top \mathbf{w} = \mathbf{m}, \tag{3.29}$$

where $\mathbf{m}$ is the statistical moment vector defined by

$$\mathbf{m} : m_j = \int_\Omega P_j(\mathbf{x}) \, d\mu(\mathbf{x}). \tag{3.30}$$

The solution to this problem can be easily computed by performing a QR factorization of the Vandermonde matrix with column pivoting:

$$\mathbf{A}^\top = \mathbf{QRP}, \tag{3.31}$$

where $\mathbf{P}$ is a pivot matrix, then solving the linear system

$$\mathrm{col}(\mathbf{R})\mathbf{w} = \mathbf{Q}^\top \mathbf{m}, \tag{3.32}$$

where $\mathrm{col}(\mathbf{R})$ is the square submatrix formed by the n first columns of $\mathbf{R}$. The solution $\mathbf{w}$ is the optimal quadrature weight vector for the set of approximate Fekete points corresponding to the points whose indices are the n first elements of the pivot vector. An example of such embedded quadrature is presented in Figure 5. If the polynomial basis is not orthogonal, for example, due to approximation errors in the GS algorithm, one has $m_j \neq \delta_{1j}$ and $\mathbf{A}$ may be ill-conditioned. The following step aims to remedy this problem by successive orthogonalizations (Algorithm 4). This step amounts to a change of basis from the original polynomials basis $\mathbf{q}$ to the discrete orthonormal basis $\mathbf{p}$ with respect to the inner product

$$\langle \mathbf{p}_1, \mathbf{p}_2 \rangle = \sum_{j=1}^{m} \mathbf{p}_1(\mathbf{x}_j)\mathbf{p}_2(\mathbf{x}_j), \tag{3.33}$$

and leads to the following vector of moments:

$$m_j = \langle 1, \mathbf{p}_j \rangle = \sum_{i=1}^{m} A_{ij} = \delta_{1j}, \tag{3.34}$$

where $\mathbf{A}$ is the orthogonal Vandermonde matrix. In practice, two iterations suffice to reach an epsilon machine accuracy unless the original matrix is severely ill-conditioned. The main drawback of quadrature rules such as the one obtained from approximate Fekete points is the risk of catastrophic cancellation of the weights. Indeed, the rounding errors are amplified by a factor

$$c = \sum_{j=1}^{n} |w_j| \tag{3.35}$$

which is known as the condition number [14]. If the weights are positive, $c = 1$ by definition. However, c may be very high for quadrature rules containing negative weights.

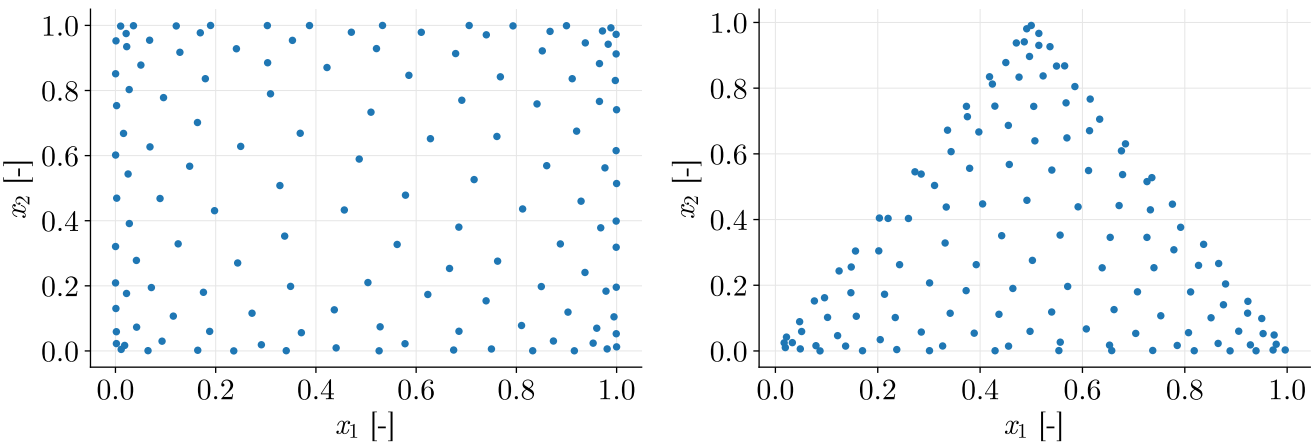

Figure 5: Approximate Fekete points on the unit square and a triangle using rescaled Legendre polynomials, the original set of points is a uniform sampling.

Algorithm 4: Iterative refinement

for $k = 1 : s$ do
  Compute the decomposition $\mathbf{QR} : \mathbf{A}_k = \mathbf{QR}$
  Update $\mathbf{A}_{k+1} = \mathbf{Q}$
end

3.3.3 Embedded Positive Quadrature Rule

Different algorithms were recently proposed in the literature [35][36] to extract an embedded quadrature rule with positive weights from an existing quadrature rule containing a higher number of points with strictly positive weights. Indeed, we dispose of a long chain of samples generated by `Pybitup`, this latter can be considered as MC samples and so, as a quadrature rule with a large number of points and positive weights from which the embedded quadrature may be extracted. First, let $\{\mathbf{x}_j, w_j\}$ be any positive quadrature rule for n polynomials in Ω with $m > n$ points, for instance a MC integration:

$$\mathbf{A}^\top \mathbf{w} = \mathbf{m}. \tag{3.36}$$

The null space of $\mathbf{A}^\top$ is the collection of all nonzero vectors $\mathbf{z}$ such that

$$\mathbf{A}^\top \mathbf{z} = \mathbf{0}. \tag{3.37}$$

By definition, any linear combination of a vector of the null space with the solution is also a solution of the system. Thus, taking α such that

$$\alpha = \frac{w_j}{z_j} : j = \text{argmin}_j \left| \frac{w_j}{z_j} \right| \quad \text{and} \quad w_k \geq \alpha z_k \quad \forall\, k \tag{3.38}$$

allows the computation of a new solution to (3.36) with $w'_j = 0$ as follows:

$$\mathbf{w}' = \mathbf{w} - \alpha \mathbf{z}. \tag{3.39}$$

The j-th point as well as the associated column of $\mathbf{A}^\top$ can then be removed from the system. Repeating this operation until the remaining quadrature uses at most as many points as there are equality constraints imposed generates the desired quadrature rule that still allow to integrate the polynomials. A similar solution can be obtained by solving the following optimisation problem:

$$\min \mathbf{c}^\top \mathbf{w} \quad \text{subject to} \quad \begin{cases} \mathbf{A}^\top \mathbf{w} = \mathbf{m}, \\ w_j \geq 0 \quad \forall\, j, \end{cases} \tag{3.40}$$

where $\mathbf{c} = [1, \ldots, 1]$. Indeed, some linear optimization methods such as the simplex algorithm directly provide a sparse optimal solution by performing a sequence of pivoting operations on basic feasible solutions.

3.4 Conclusion

In conclusion, this chapter introduced several integration techniques in order to estimate the integrals involved in the computation of the PCE coefficients. When the polynomial basis has been obtained by a three term recurrence relation, this latter provides an accurate quadrature rule directly generated by the eigenvectors of a tridiagonal matrix. However, Gaussian quadrature rules suffer from a curse of dimensionality and are limited to independent variables. A well-known alternative is the MC integration, whose convergence is independent of dimension. This method allows a robust estimation of the integral based on a sample of realisation of the random variables, but its slow convergence may require a large number of integration points to reach the desired accuracy, variance reduction techniques such as quasi-MC or importance sampling were thereby proposed to improve the convergence. The last method proposed aims at combining the advantages of the two previous integration techniques by extracting an embedded quadrature rule from a MC sample set. The quadrature weights are computed so that the equalities satisfied by the MC integration are still satisfied by the embedded quadrature rule containing a smaller number of points. This method is particularly useful when the computational cost of numerical simulations does not allow an evaluation of the reference model at a large number of integration points.

Finally, it is clear that an essential element determining the efficiency of an algorithm is the way this latter is implemented in the code. In the following chapter, we thus continue the discussion on the numerical implementation of the different algorithms presented in the chapters 2 and 3 in `Pybitup`.

Chapter 4

Implementation Details

4.1 Introduction

Several implementations of polynomial chaos solvers already exist in different programming languages, all with their advantages and limitations. For instance, `Chaospy` [37] is an efficient open-source Python library providing numerous one-dimensional integration algorithms for the computation of a PCE, but is not designed for high dimensional problems or dependent variables. `UQLab` is a highly accessible uncertainty quantification toolbox developed in Matlab. However, both are licensed programs and Matlab has some limitations in terms of performance, parallelization tools, oriented-object programming and compatibility.

The PCE library developed for this work is coupled with `Pybitup`, the Bayesian inference and uncertainty propagation toolbox developed by J. Coheur. As Python loops are inherently slower than their low-level counterpart, the library mainly relies on `Numpy` and `Scipy` packages integrating `C` and Fortran parallel codes for efficient matrix operations. Once computed, the polynomials and the PCE coefficients are stored in a compact homogeneous block representation [38] within two classes of similar structure presented in Figure 6, one containing the orthogonal polynomial basis and the other containing the surrogate model. The compact homogeneous block representation allows a memory efficient storage as well as the use of simple matrix operations to manipulate the polynomials. It is important to note that `Polynomial_basis` and `Expansion` share the same exponent table and dimension, but store a different coefficient matrix.

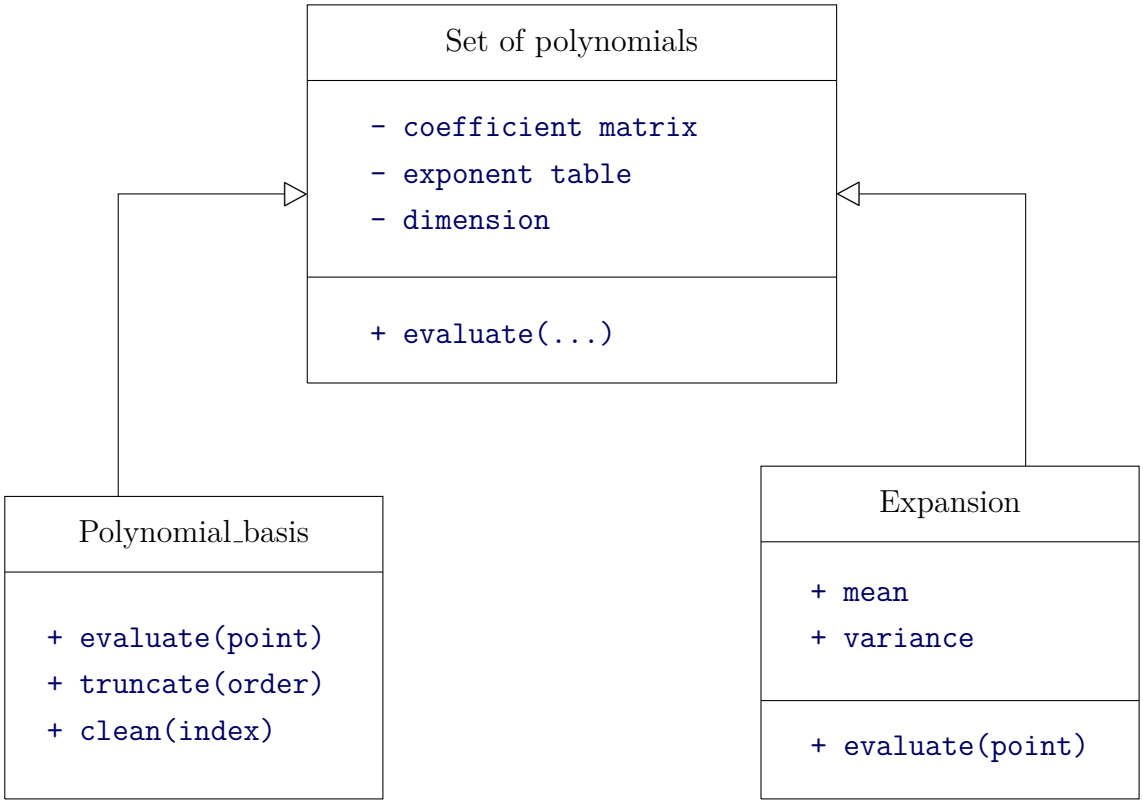

Figure 6: Abstract representation of the PCE related classes implemented in `Pybitup`.

4.2 Polynomial_basis Class

The compact homogeneous block representation is adapted for sparse polynomials, this latter uses an exponent table $\mathbf{E}$ containing the power of individual monomials, which is nothing but the transpose of the multi-index matrix, and a coefficient matrix $\mathbf{H}$ to represent the polynomial basis. For instance, the following set of polynomials:

$$
\begin{array}{l|l}
P_1(\mathbf{x}) = 1 & P_5(\mathbf{x}) = 1 + 5x_2 + x_1 + x_1x_2 \\
P_2(\mathbf{x}) = 3 + x_2 & P_6(\mathbf{x}) = 2 + 4x_1 + 7x_1x_2 + x_1^2 \\
P_3(\mathbf{x}) = 1 + x_1 & P_7(\mathbf{x}) = 6 + 3x_2 + 9x_2^2 + 3x_1^2 + x_2^3 \\
P_4(\mathbf{x}) = 2 + 4x_1 + x_2^2 & P_8(\mathbf{x}) = 3 + 2x_1 + x_2^2 + 4x_1^2 + x_1x_2^2
\end{array}
$$

will have the homogeneous block representation presented in Figure 7. It is important to note that a truncated basis may lead to a non-square or non-triangular coefficient matrix. Moreover, we define the density of $\mathbf{H}$ as the fraction of nonzero elements:

$$\rho = \frac{\sum_i \sum_j 1 \ \forall \ (i,j) : H_{ij} \neq 0}{\sum_i \sum_j 1 \ \forall \ (i,j)}. \tag{4.1}$$

When a high dimensional basis is computed by the tensor product of one-dimensional polynomials such as Legendre, Hermite or Laguerre polynomials, ρ tends to zero and $\mathbf{H}$ becomes sparse, meaning that the explicit storage of the whole matrix is memory inefficient. This issue can be addressed by storing $\mathbf{H}$ in a compressed sparse row format consisting of an array containing the column indices, a second array of corresponding nonzero values, and a third array pointing to row starts of the two previous ones. This sparse structure is generated by the `Scipy` library. As the Vandermonde matrix is an important element in most of the algorithms for computing the PCE, the polynomials must be efficiently evaluated at a set of points to form

$\mathbf{E}^\top =$

0	0
0	1
1	0
0	2
1	1
2	0
0	3
1	2

$\mathbf{H} =$

1							
3	1						
1	-	1					
2	-	4	1				
1	5	1	-	1			
2	-	4	-	7	1		
6	3	-	9	-	3	1	
3	-	2	1	-	4	-	1

Figure 7: Example of compact homogeneous block representation of an arbitrary third order polynomial basis of two variables. The zero coefficients are not explicitly stored.

$$\mathbf{A} : A_{ij} = P_j(\mathbf{x}_i). \tag{4.2}$$

In accordance with the homogeneous block representation, the computation of the Vandermonde matrix is straightforward:

$$\mathbf{A} = \mathbf{A}'\mathbf{H}^\top \quad \text{with} \quad A'_{ij} = \prod_{k=1}^{d} x_{ik}^{E_{kj}}, \tag{4.3}$$

where x_{ik} is the value of the k-th component of the i-th point and d is the dimension. An example of vectorized implementation of (4.3) is presented in Listing 1. Provided the polynomials share the same exponent table, additions and subtractions are performed by summing or subtracting their respective coefficient vectors. In addition, the explicit multiplication of polynomials is not necessary due to the use of quadrature-based algorithms for evaluating the integrals and will therefore not be detailed here. Furthermore, removing a row from $\mathbf{H}$ is equivalent to truncating the corresponding polynomial from the orthogonal basis.

Finally, another important algorithm implemented in `Pybitup` is the tensor product of one-dimensional bases, which is used for constructing orthogonal polynomials for independent variables:

$$\{P_j(\mathbf{x})\} = \bigotimes_{k=1}^{d} \{P_j^k(x_k)\}, \tag{4.4}$$

where k denotes the index of the variable in the model. This operation is performed first by rewriting each one-dimensional basis according to the same exponent table, which is simply the multi-index matrix containing all the possible combinations of the monomials up to the desired q-norm. The coefficient matrix $\mathbf{H}$ of the tensor product basis is then computed with a Hadamard product:

$$\operatorname{row}_j(\mathbf{H}) = \bigodot_{k=1}^{d} \operatorname{row} E_{jk}\left(\mathbf{H}^k\right), \tag{4.5}$$

where $\operatorname{row} E_{jk}\left(\mathbf{H}^k\right)$ is the E_{jk}-th row of the coefficient matrix relative to $\{P_i^k(x_k)\}$. Similarly, the computation of a polynomial basis by the GS orthogonalisation algorithm presented in (2.18) is straightforward. Indeed, since both the initial set of linearly independent functions $\{B_j(\mathbf{x})\}$ and the orthogonal polynomial basis share the same exponent table, the coefficient matrix can be obtained by

$$\operatorname{row}_j(\mathbf{H}) = \sum_{i=1}^{j} \operatorname{row}_i\left(\mathbf{H}'\right) R_{ij}^{-1}, \tag{4.6}$$

where $\mathbf{H}'$ is the coefficient matrix relative to the initial set of functions and $\mathbf{R}$ is the upper triangular matrix obtained by QR decomposition of $\mathbf{W}^{1/2}\mathbf{A}$. If $\{B_j(\mathbf{x})\}$ is chosen such that $\mathbf{H}'$ is the identity, the previous equation reduces to

$$H_{jk} = \sum_{i=1}^{n} \delta_{ik} R_{ij}^{-1} \quad \Leftrightarrow \quad \mathbf{H} = \left(\mathbf{R}^{-1}\right)^\top \tag{4.7}$$

and $\mathbf{R}$ can be inverted by solving a linear system of equations taking advantage of the fact this matrix is upper triangular using `Lapack` packages.

```
def eval(self,point):

    A = 1
    point = np.reshape(np.transpose(point),(self.dim,-1)).T
    for i in range(self.dim): A *= np.power(point[:,i,None],self.E[i])
    A = np.transpose(self.H.dot(A.T))
    return A
```

Listing 1: Computation of the Vandermonde matrix, where d is the dimension of the polynomial basis. The points are stored in a two dimensional array with $\mathbf{x}_j$ along the rows. The explicit use of a loop on d aims to reduce the memory usage.

4.3 Expansion Class

The expansion class takes a polynomial basis and a list of PCE coefficients as input arguments to build and store the surrogate model. At this point, it is important to make the difference between the dimension of the polynomial basis, which is the number of individual random variables $\mathbf{X}$ in the system, and the dimension of the response, which is the number of deterministic variables $\mathbf{r}$ such as the location in space or the time:

$$y(\mathbf{r}, \mathbf{x}) = \sum_{j=1}^{n} y_j(\mathbf{r}) P_j(\mathbf{x}) = \sum_{j=1}^{m} a_j(\mathbf{r})\, x_{\mathbf{I}_j}, \tag{4.8}$$

where $x_{\mathbf{I}_j}$ is the monomial of multi-index $\mathbf{I}_j$. If the output $Y = y(\mathbf{X})$ depends only on stochastic parameters, the PCE coefficients are constants and the expansion class stores a unique polynomial. However, if the output depends on n deterministic parameters, the coefficients a_j are discretized in a $n+1$-dimensional array and the expansion class stores a polynomial for each discretization of the response. Therefore, the structure of the expansion and polynomial basis classes differ only by the dimension of their coefficient matrix. For instance, a one-dimensional response leads to $n = 1$ with $\mathbf{H}$ as a two-dimensional matrix computed by

$$\mathbf{H} = \mathbf{K}^{\top}\mathbf{H}', \tag{4.9}$$

where $\mathbf{H}'$ is the coefficient matrix of the polynomial basis and $\mathbf{K}$ is the 2-dimensional matrix of PCE coefficients defined by

$$\mathbf{K} : K_{ij} = y_i(r_j). \tag{4.10}$$

When $n > 1$, the computation of the coefficient matrix can be performed by flattening $\mathbf{K}$ as a two-dimensional matrix and solving (4.9). Afterwards, the resulting $\mathbf{H}$ is reshaped in the adequate $n+1$-dimensional array format.

Finally, the discretized response of the PCE model at a given set of points is obtained by performing the operation (4.3) for the expansion class. As an illustrative example, consider the following random variables:

$$Y(\mathbf{r}) = y(\mathbf{r}, \mathbf{X}) \qquad \text{with} \quad \begin{cases} \mathbf{X} = \begin{bmatrix} X_1 & X_2 \end{bmatrix}, \\ \mathbf{r} = \begin{bmatrix} r_1 & r_2 \end{bmatrix}, \end{cases} \tag{4.11}$$

where the output Y thus depends on two deterministic variables, for instance, r_1 is the time and r_2 is the position. In addition, consider the following PCE composed of 7 monomials and their coefficients:

$$y(\mathbf{r}, \mathbf{x}) = a_1(\mathbf{r})x_1 + a_2(\mathbf{r})x_1x_2 + \ldots + a_7(\mathbf{r})x_1^3x_2^2. \tag{4.12}$$

The resulting coefficients matrix $\mathbf{H}$ is 3-dimensional, with one dimension for the monomials and two for the discretization of a_j along r_1 and r_2 (Figure 8).

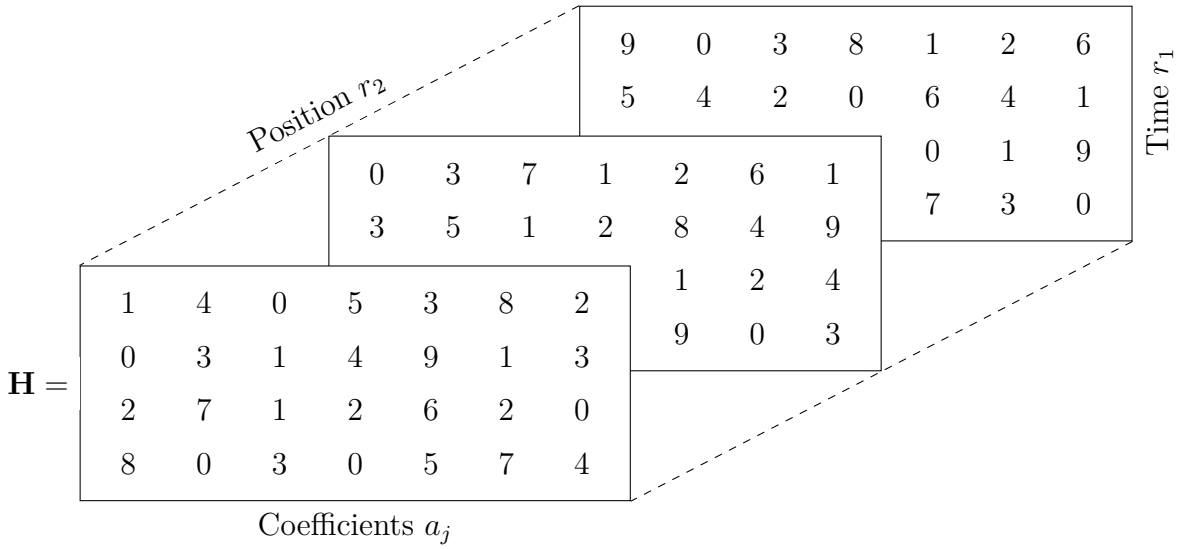

Figure 8: Fictive coefficient matrix stored in an `Expansion` class related to a PCE discretized in two deterministic dimension: the time and the position.

```
def __init__(self,K,poly):

    shape = (poly[:].shape[1],)+K.shape[1:]
    K = K.reshape(poly[:].shape[0],-1)

    self.E = np.copy(np.atleast_2d(poly.E))
    self.H = poly[:].T.dot(K).reshape(shape).T
    self.dim = self.E.shape[0]
```

Listing 2: Computation of the coefficient matrix in the expansion class for an arbitrary number of dimensions of the response.

4.4 Quadrature Rules

With the exception of the quadrature rule obtained from the three term recurrence relation, which is obtained by computing the eigenvalues and the first element of the eigenvectors of a tridiagonal matrix, the Vandermonde matrix is mandatory for the computation of the quadrature rules, as well as in all the algorithms computing the PCE coefficients in `Pybitup`. For instance, the quadrature rule based on approximate Fekete points can be obtained in a straightforward and efficient way by applying the QR decomposition functions provided by `Scipy` and `Numpy` on the Vandermonde matrix. An example of implementation is presented in Listing 3.

Due to the large number of algorithms implemented in `Pybitup` for computing the different elements of the PCE, the majority of the algorithms is not presented in this section for readability reason. However, it may be interesting to provide an example of a positive embedded quadrature rule. Listing 4 displays the algorithm used to extract a sparse embedded quadrature rule from a MC sample by iterative weight cancellation. This algorithm mainly suffers from the computation time required to compute the null space of the transpose of the Vandermonde matrix at each iteration. In addition, round-off errors accumulate at each iteration. However, this quadrature generally provides a better accuracy than the approximate Fekete points thanks to its positive quadrature weights.

```
def fekquad(point,poly):

    nbrPoly = poly[:].shape[0]

    # Reconditioning of A and QR factorization

    A = poly.eval(point)
    A,R = np.linalg.qr(A)
    m = np.sum(A,axis=0)/A.shape[0]

    Q,R,P = linalg.qr(A.T,pivoting=1,mode='economic')
    R = R[:,:nbrPoly]
    q = np.dot(Q.T,m)

    # Computes the weights and Fekete points

    index = P[:nbrPoly]
    weight = linalg.solve_triangular(R,q)
    return index,weight
```

Listing 3: Extraction of an embedded quadrature rule for a MC sample by preforming a QR factorisation of the Vandermonde matrix. The `index` output correspond to the indices of the Fekete points in the original sample. Note that `poly` is a `polynomial_basis`.

```
def newquad(point,poly,weight=0):

    def null(At,Jinv,z):

        for i in range(50):
            F = At.dot(z)
            z -= np.dot(Jinv,F)
            z = z/np.linalg.norm(z)
            if np.max(np.abs(F))<1e-16: break

        if i==49: return 0
        else: return z

    # First null space and initialization

    A = poly.eval(point)
    nbrPts = A.shape[0]
    A,R = np.linalg.qr(A)
    end = nbrPts-A.shape[1]
    index = np.arange(nbrPts)
    if not np.any(weight): weight = np.ones(nbrPts)/nbrPts
    z = np.ones(nbrPts)

     # Iteratively updates the quadrature rule

    for i in range(end):

        z = null(A.T,A,z)
        if not np.any(z): break

        # Selects the coefficient to cancel a weight

        wz = weight/z
        idx = np.argmin(np.abs(wz))
        alp = wz[idx]

        # Updates the weights and the matrices

        weight -= alp*z
        z = np.delete(z,idx)
        index = np.delete(index,idx)
        weight = np.delete(weight,idx)
        A,R = linalg.qr_delete(A,R,idx,overwrite_qr=1,check_finite=0)

    return index,weight
```

Listing 4: Extraction of an embedded quadrature rule from a MC sample by iterative cancellation of the weights. The `index` output correspond to the indices of the quadrature points in the original sample. Note that `poly` is a `polynomial_basis`.

4.5 Conclusion

In conclusion, we presented the so-called homogeneous block representation for the storage of the polynomials, allowing the use of the `sparse_matrix` class and its inbuilt functions provided by `Scipy`. In particular, the computation of the Vandermonde matrix and the basis by GS orthogonalization, which are central elements in most of the data-driven PCE related algorithms, is straightforward. The PCE is then stored in two classes of similar structure, the first containing the orthogonal polynomial basis, and the second containing the surrogate model thus obtained.

Afterwards, we presented some examples of algorithm for computing a quadrature rule, taking advantage of the various `Numpy` and `Scipy` functions at our disposal. It is clear that most of the algorithms involved in the PCE were omitted for readability reason, in particular, the reader may refer to the documentation or to the code itself for more information about the implementation of the spectral projection, the least squares collocation, the LARS, the sensitivity indices, the probability distribution classes and their three term recurrence coefficients, the PCA whitening, the MC integration or other quadrature rules.

Chapter 5

Application Examples

5.1 Introduction

Solving test cases is an essential step in the development of numerical algorithms, either to demonstrate the validity of the methods, or to analyse and compare the behaviour of different algorithms. To this end, the Ishigami function is commonly used as a test function to benchmark global sensitivity analysis methods because it exhibits strong nonlinearity and non-monotonicity [39]. In addition, this latter provides an analytical solution for its statistical moments and sensitivity indices.

The one- and two-reactions pyrolysis processes implemented in `Pybitup` for parameter estimation are the firsts practical applications related to thermal protection systems we would like to address for testing the PCE. Indeed, in addition to being directly related to the context of atmospheric entry of spacecraft, the random variables representing the different input parameters have been identified as non-Gaussian and highly correlated, this test case thereby aims to access the relevance of the sparse PCE for moderately high dimensional problems with dependent variables. Finally, the one-dimensional ablation test case implemented in `PATO` is a large scale problem involving a sample of solid material heated at a constant temperature until it undergoes a pyrolysis reaction [40]. As the numerical model is computationally expensive to evaluate, this latter cannot be used to perform statistical and sensitivity analysis of the output. Consequently, the computation of a cheaper surrogate model becomes a necessity.

It is important to distinguish the two types of test cases explored in this chapter: the first involving the Ishigami function is an introduction to classical PCE applications, we consider that the uncertain input variables are statistically independent and described in terms of labelled probability density functions. The pyrolysis and ablation examples consider a purely data-driven PCE where the uncertainty in the input variables is described in terms of a chain of samples. The orthogonal polynomials and their coefficients are thus computed using only a part of the available sample set as training data.

5.2 Ishigami Function

The Ishigami function is a 3-dimensional function with a particular dependence on its third input variable, whose analytical expression is given by

$$y(\mathbf{x}) = \sin(x_1) + a\sin^2(x_2) + bx_3^4\sin(x_1), \tag{5.1}$$

where a and b are constant parameters. For computer experiment purposes, we assume that the random variables are uniform and independent, the output is thus

$$Y = y(\mathbf{X}) \quad \text{with} \quad \mathbf{X} = \mathcal{U}(-\pi, \pi)^3. \tag{5.2}$$

Under such conditions, it has been shown in [39] that the analytical expressions of the expectation and the variance of the output are given by

$$\begin{aligned} &\mathrm{E}(Y) = a/2, \\ &\mathrm{Var}(Y) = \frac{1}{2} + \frac{a^2}{8} + \frac{b^2\pi^8}{18} + \frac{b\pi^4}{5}, \end{aligned} \tag{5.3}$$

and the joint probability density function of the input vector is simply the product of each one-dimensional probability density function:

$$f_X(\mathbf{x}) = \begin{cases} (2\pi)^{-3} & \text{in } \Omega = [-\pi, \pi]^3, \\ 0 & \text{otherwise.} \end{cases} \tag{5.4}$$

Moreover, the first and total Sobol sensitivity indices are

$$\begin{aligned} S_1 &= \frac{1}{\mathrm{Var}(Y)} \frac{1}{2} \left(1 + \frac{b\pi^4}{5}\right)^2, \\ S_2 &= \frac{1}{\mathrm{Var}(Y)} \frac{a^2}{8}, \\ S_3 &= 0, \end{aligned} \qquad \begin{aligned} S_{T1} &= S_{T3} + S_1, \\ S_{T2} &= S_2, \\ S_{T3} &= \frac{1}{\mathrm{Var}(Y)} \frac{8b^2\pi^8}{255}. \end{aligned}$$

Consequently, an orthogonal basis can be constructed by the tensor product of Legendre polynomials as well as the associated Gauss-Legendre quadrature rule for the computation of the coefficients by spectral projection. For the sake of comparison, we define the squared relative error of a surrogate model y' on the reference model y as

$$\mathrm{SRE} = \frac{\int_\Omega \left[y(\mathbf{x}) - y'(\mathbf{x})\right]^2 \, d\mu(\mathbf{x})}{\int_\Omega y(\mathbf{x})^2 \, d\mu(\mathbf{x})}. \tag{5.5}$$

This error is computed by a MC integration. It is important to note that the error must be estimated with a sample different from the training data in order to avoid an underestimation of the error due to overfitting of the PCE. Indeed, the surrogate model may provide a correct solution at the integration points but deviate from it between the points. As shown in Figure 9, the estimate of the surrogate model converges to the reference solution when the order of the polynomials increases.

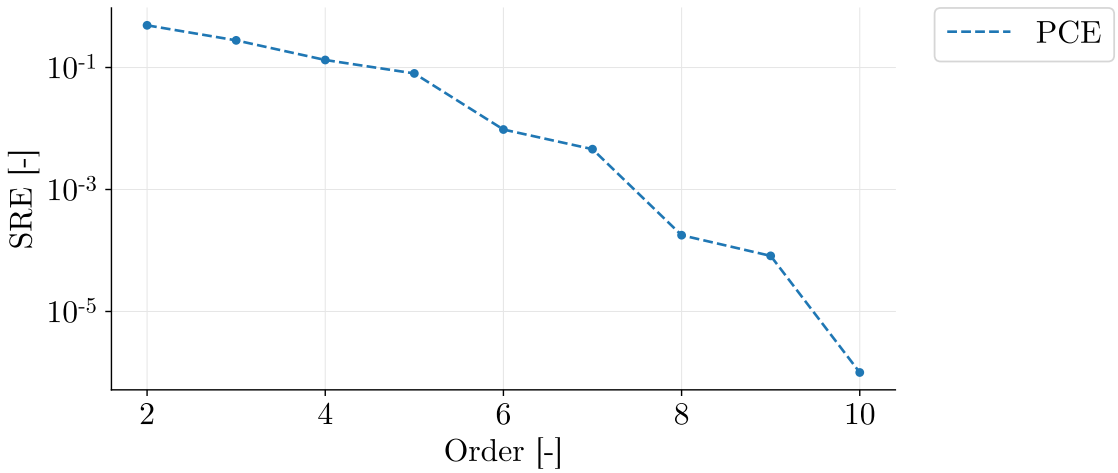

Figure 9: Error of a PCE using Legendre polynomials, for the Ishigami function. The coefficients are computed by a Gauss-Legendre quadrature rule.

Finally, the PCE provides an accurate approximation of the Sobol sensitivity indices. The total index of the random variable X_1 is the highest, meaning that it has the most significant impact on the total variance. Since X_3 appears only in the term multiplied by $\sin(X_1)$, this latter has no effect at first order, but its total effect is nonzero due to the interactions with X_1. The second random variable X_2 has no interactions with the others, which implies that its first order index is equal to its total order index.

$S_1 = 0.314,$	$S_2 = 0.442,$	$S_3 = 0,$
$S_{T1} = 0.558,$	$S_{T2} = 0.442,$	$S_{T3} = 0.243.$

5.3 One-Reaction Pyrolysis

The second example consists in a simplified case of pyrolysis process involving a single solid phase. Consider a sample of solid phase S producing a gaseous species G, where k is the reaction rate and F is the mass fraction of gas involved:

$$S \xrightarrow{k} FG. \tag{5.6}$$

Assuming the sample is heated at a constant rate τ, the time evolution of the temperature of the sample is a linear relation given by

$$T(t) = T_0 + \tau t \tag{5.7}$$

and the production of gas is governed by the following equation:

$$g = \frac{-d\beta}{dt} = \frac{FA}{\tau}\beta^n \exp\left(\frac{-E}{RT}\right), \tag{5.8}$$

where n is the reaction order, A is the Arrhenius pre-exponential factor, E is the activation energy, R is the perfect gas constant and β is a function to be determined. First, the time derivative of β can be rewritten as

$$\frac{d\beta}{dt} = \frac{\partial\beta}{\partial T}\frac{dT}{dt} = \tau\frac{d\beta}{dT}. \tag{5.9}$$

Injecting this result into the equation (5.8) leads to

$$\int_1^\beta \frac{d\beta}{\beta^n} = \frac{-FA}{\tau}\int_{T_0}^T \exp\left(\frac{-E}{RT}\right) dT. \tag{5.10}$$

In order to avoid dealing with different solutions depending on the value of n, we will consider the case where $n \neq 1$. The left-hand side becomes

$$\int_1^\beta \frac{d\beta}{\beta^n} = \frac{1-\beta^{1-n}}{n-1}. \tag{5.11}$$

Introducing the following change of variable:

$$z = \frac{E}{RT} \quad \Leftrightarrow \quad dz = \frac{-E}{RT^2} \tag{5.12}$$

and integrating by part, the integral of the right-hand side becomes

$$\int_{T_0}^{T} \exp\left(\frac{-E}{RT}\right) dT = \frac{-E}{R}\left[\frac{e^{-z}}{z} - E_1(z)\right]_{z_0}^{z} \tag{5.13}$$

where $E_1(z)$ is the exponential integral function

$$E_1(z) = \int_z^{\infty} \frac{e^{-z}}{z}\, dz. \tag{5.14}$$

The expression of β is then obtained by replacing these two results in (5.10):

$$\begin{aligned} \beta &= \left\{1 - \frac{1-n}{\tau}\left[AT\exp\left(\frac{-E}{RT}\right) - \frac{AE}{R}E_1\left(\frac{E}{RT}\right) + C_0\right]\right\}^{1/(1-n)} \\ C_0 &= \frac{AE}{R}E_1\left(\frac{E}{RT}\right) - AT\exp\left(\frac{-E}{RT}\right). \end{aligned} \tag{5.15}$$

Finally, the production of gas is computed by injecting β into (5.8), leading to

$$g = \frac{FA}{\tau}\beta^n \exp\left(\frac{-E}{RT}\right). \tag{5.16}$$

5.3.1 Inverse Problem

The solution is evaluated for different temperatures and the variables $[A, E, n, F]$ play the role of random parameters $\mathbf{X}$, the PCE is thus composed of 4-dimensional polynomials and the coefficients are discrete functions of the temperature. These random parameters need to be identified based on experimental data available in the literature. The parameter estimation can be seen as an inverse problem consisting in finding the model parameters $\mathbf{x}$ given experimental observations $\mathbf{d}$ such that

$$y(\mathbf{x}) + \varepsilon = \mathbf{d}, \tag{5.17}$$

where ε accounts for measurement noise. One of the possible approach is to interpret the parameters as a probability distribution $f_X(\mathbf{x} \,|\, \mathbf{d})$ [41] that can be obtained by writing the Bayes theorem:

$$f_X(\mathbf{x} \,|\, \mathbf{d}) = \frac{f_X(\mathbf{d} \,|\, \mathbf{x})\, f_Y(\mathbf{x})}{\int_\Omega f_X(\mathbf{d} \,|\, \mathbf{x})\, f_Y(\mathbf{x})\, d\mathbf{x}}, \tag{5.18}$$

where f_Y is the prior distribution and contains a priori beliefs on the parameters. Due to the processing noise, the function $f_X(\mathbf{d} \,|\, \mathbf{x})$ measures the probability of observing the experimental data $\mathbf{d}$ through the model given $\mathbf{X} = \mathbf{x}$. Practically, The sample is obtained by a Metropolis–Hastings algorithm, which consists in generating a collection of points according to a posterior distribution using a Markov process. First, a random candidate $\mathbf{x}_2$ is generated according to a proposal distribution $f_Z(\mathbf{x}_2 \,|\, \mathbf{x}_1)$, where $\mathbf{x}_1$ is the current state. The next step is to decide whether to accept the candidate [42][43]. The candidate is accepted with a probability

$$\alpha = \min\left[1, \frac{f_X(\mathbf{x}_2 \,|\, \mathbf{d})\, f_Z(\mathbf{x}_1 \,|\, \mathbf{x}_2)}{f_X(\mathbf{x}_1 \,|\, \mathbf{d})\, f_Z(\mathbf{x}_2 \,|\, \mathbf{x}_1)}\right]. \tag{5.19}$$

If the candidate is accepted, this latter is added to the output samples and the current state is updated to $\mathbf{x}_2$, otherwise the current state remains at $\mathbf{x}_1$ for the next iteration. An overview of the resulting distribution is presented in Figures 10 and 11.

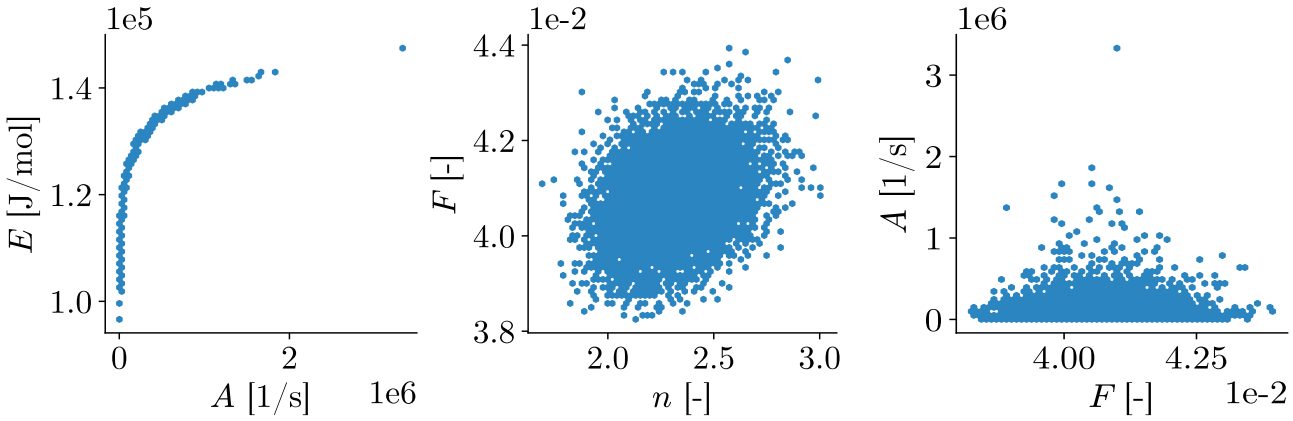

Figure 10: Overview of the correlation between different input parameters in the one-reaction pyrolysis test case computed by Metropolis-Hastings algorithm.

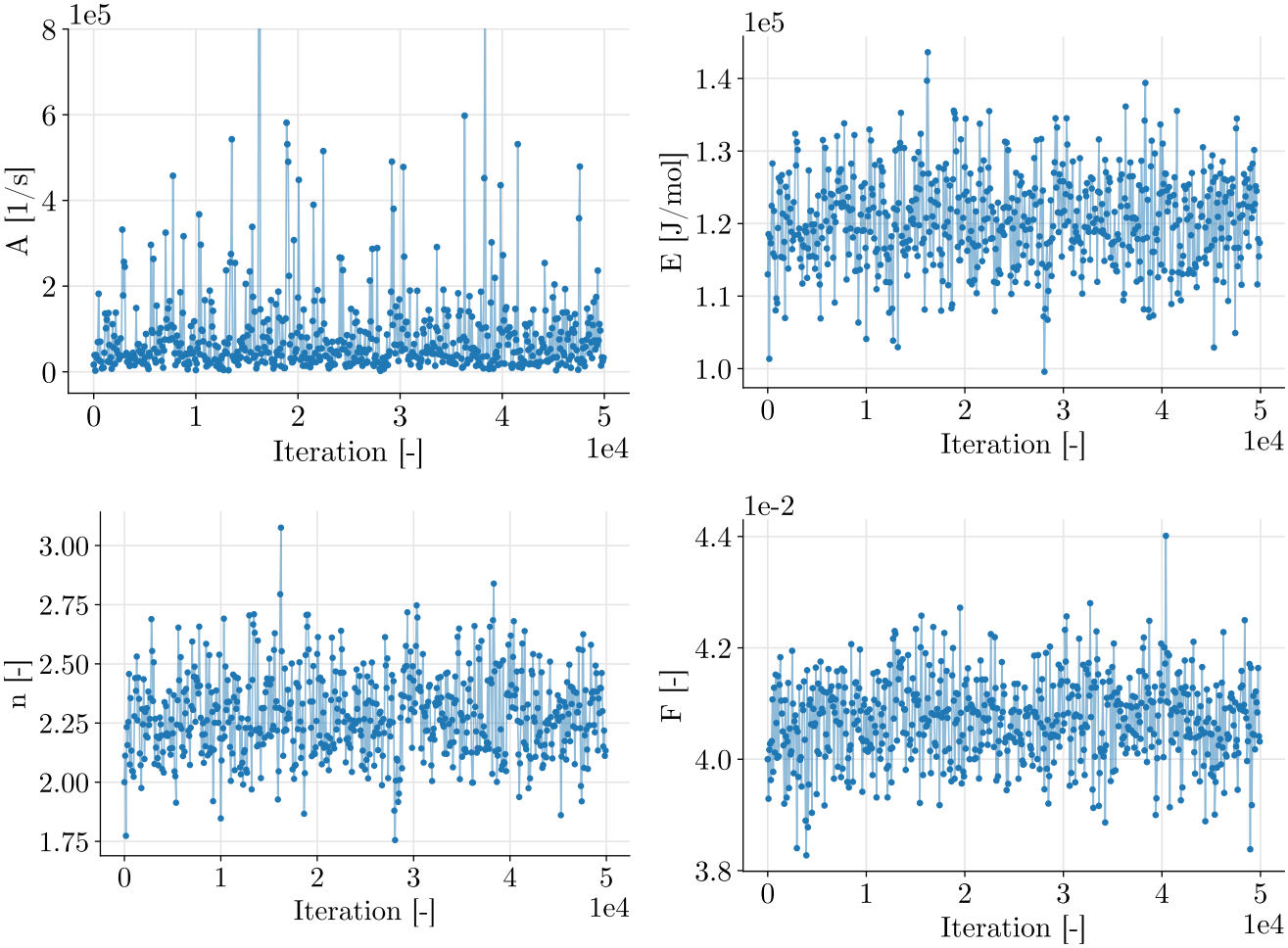

Figure 11: Markov chains generated by Metropolis-Hastings algorithm to obtain the distribution of the random parameters in the one-reaction pyrolysis test case.

5.3.2 Monte Carlo

Since the computation of the polynomial basis by GS orthogonalisation only requires to compute the response of the polynomials at the integration points, this method is independent of the computational cost to evaluate the response of the reference model. Once the basis has been computed using a part of the original samples, a straightforward way to compute the PCE coefficients is to perform a MC integration using the sample on which the polynomials have been orthogonalized. However, its slow convergence may require a large number of points to achieve the desired accuracy. Figures 13 and 12 show the error of the surrogate model computed with different samples than the ones used to compute the PCE. Figure 14 shows the statistical moments of the gas production.

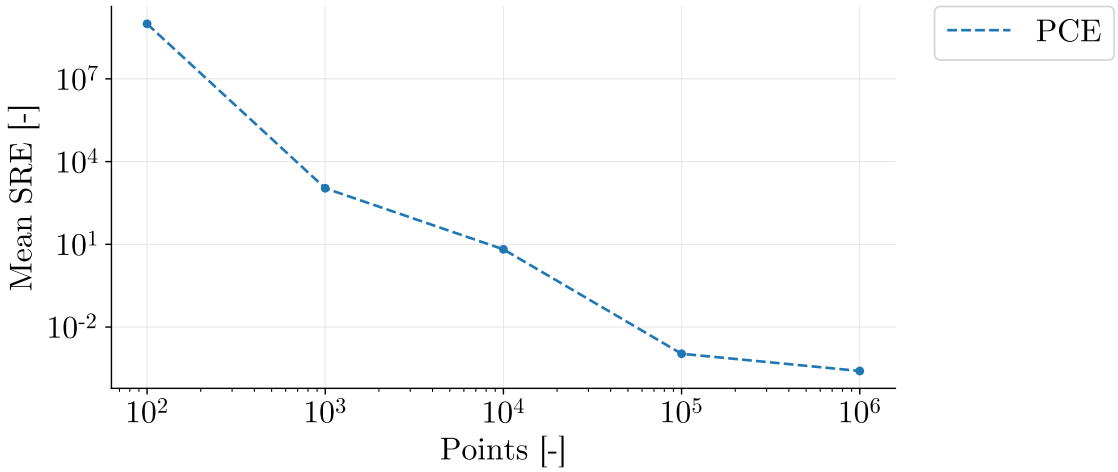

Figure 12: Mean error of a 3rd order PCE computed by a MC integration with different number of points for the one-reaction pyrolysis test case.

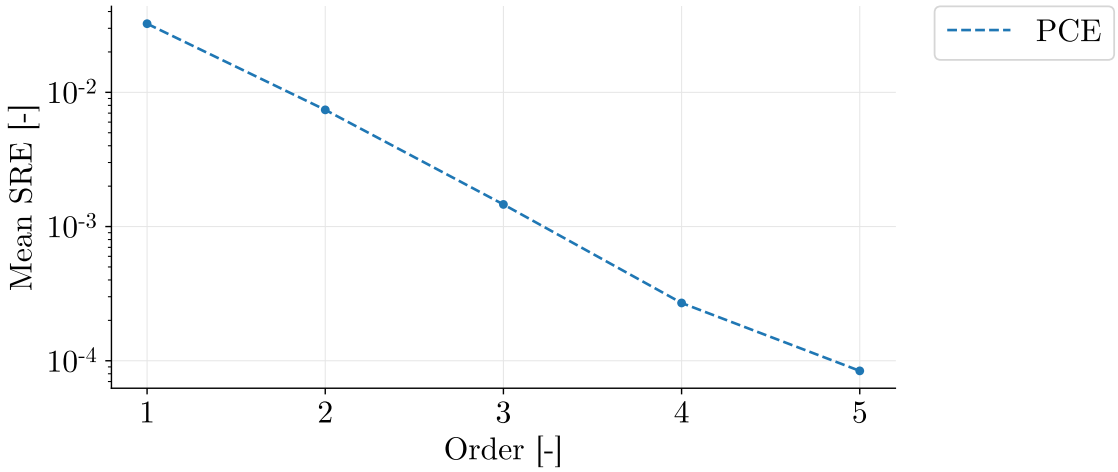

Figure 13: Mean error of a PCE computed by a MC integration of 2×10^5 points at different orders for the one-reaction pyrolysis test case.

Increasing the number of MC points provides a better approximation of the integrals involved in the algorithms constructing the PCE. However, it can be seen that the gas production cannot be exactly represented by a PCE of order 3 as the error converges to a nonzero value when increasing the number of points. On the contrary, increasing the order of the PCE allows a further reduction the error.

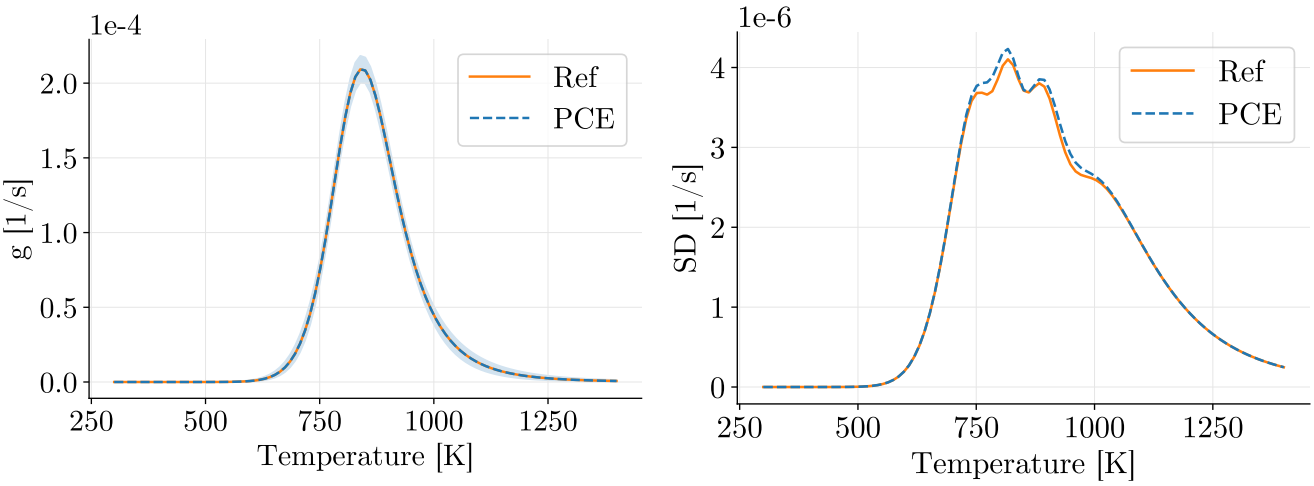

Figure 14: Range containing 98% of the outputs around the mean for a 4th order PCE computed by MC integration on the left, and its standard deviation on the right. The reference curve is the gas production of the one-reaction pyrolysis test case.

5.3.3 Quadrature Rule

Another approach to compute the PCE coefficients when an evaluation of the reference model is computationally expensive is to use an embedded quadrature rule selecting a small number of integration points. It is important to note that an embedded quadrature rule generated from a basis of order n integrates a polynomial of the same order. Since the integrand is the product of a polynomial of the basis and the function to be expanded, one must generate quadrature at least of order $2n$.

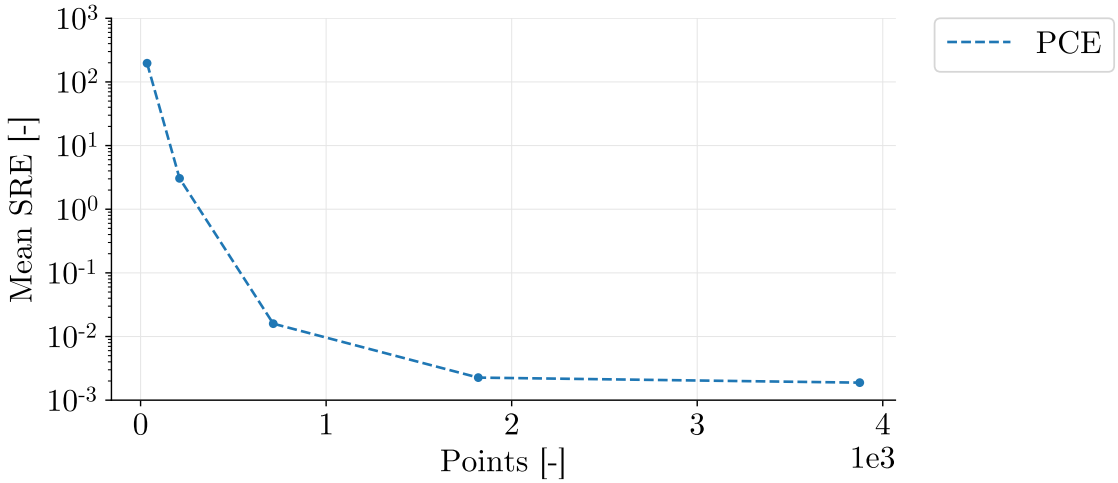

Figure 15: Mean error of a 3rd order PCE computed by an embedded quadrature rule with different number of points for the the one-reaction pyrolysis test case.

This method has the advantage of requiring less points than the MC integration in low and moderately high dimensions, but suffers from the fast increase of the number of polynomials with the dimension. Figure 15 shows the mean error of a PCE when the coefficients are computed using approximate Fekete points. Figure 16 shows the statistical moments estimated by the PCE at different temperatures.

It can be first observed that only 4×10^3 integration points are sufficient to obtain the same accuracy than a MC integration with 2×10^5 points, leading to a significant reduction of the computation cost since the reference model must be evaluated for reduced number of points. It is important to note that the computation time of the weight cancellation algorithm strongly depends on the size original sample since a null space must be computed at each iteration. Moreover, rounding errors can accumulate and decrease the accuracy of the quadrature thus obtained. Pivoting operations in the simplex algorithm become expensive when the number of points is large, making the selection of a quadrature rule based on approximate Fekete points much faster than the two other methods for large sample sets.

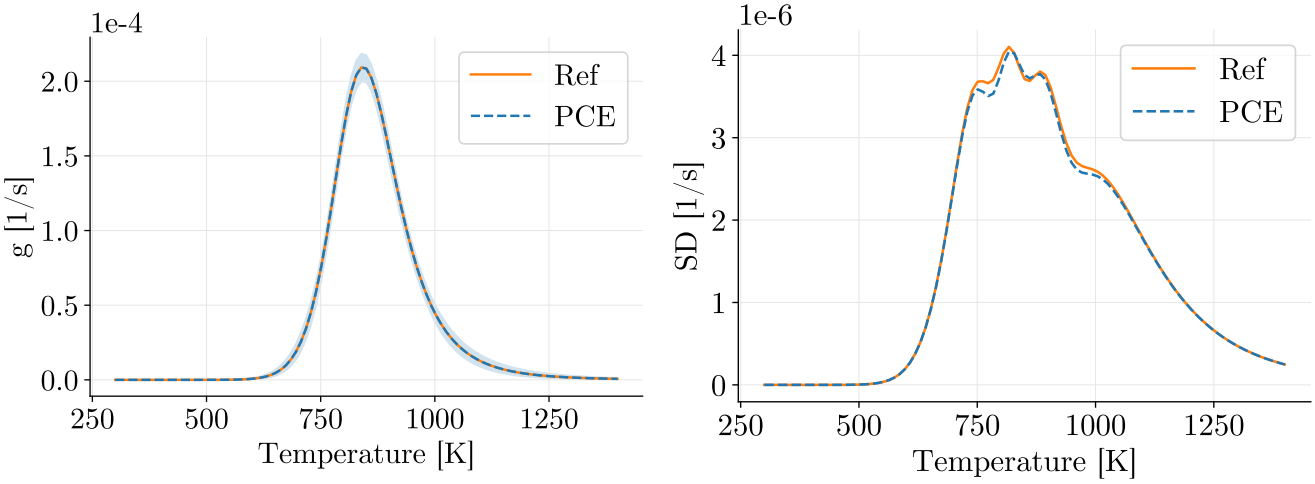

Figure 16: Range containing 98% of the outputs around the mean for a 3rd order PCE computed by an embedded quadrature rule on the left, and its standard deviation on the right. The reference curve is the gas production of the one-reaction pyrolysis test case.

5.3.4 Decorrelation Methods

The correlated random variables and their non-Gaussian joint probability density prevent the use of analytical polynomials. This issue can be addressed by the parameterization proposed in [4] to map the original set of random parameters $\mathbf{X}$ to a random vector $\mathbf{X}'$ of non-dimensional variables which can be approximated by Gaussian random variables. The latter is expressed as follows, where $\bar{A}$, $\bar{E}$, $\bar{n}$, $\bar{F}$ and $\bar{T}$ are scaling factors, note that A undergoes a non-linear transformation while the other parameters are simply rescaled:

$$\begin{cases} A' = \ln A - E(R\bar{T})^{-1}, \\ E' = E/\bar{E}, \end{cases} \quad \begin{cases} n' = n/\bar{n}, \\ F' = F/\bar{F}, \end{cases} \tag{5.20}$$

It can be observed in Figure 17 that the resulting random variables are now linearly correlated, meaning that a change of coordinate system is sufficient to decorrelate them. In particular, the PCA whitening is a decorrelation method for linearly separable and zero-centered random variables, generating a linear mapping between the correlated and decorrelated random vectors through a whitening matrix. Practically, the PCA will project the random vector into a new coordinate system where each variable has a maximum variance along a principal axis [44]. The whitening matrix $\mathbf{K}$ satisfies the relation

$$\mathbf{K}^\top \mathbf{K} = \mathbf{\Sigma}^{-1}, \tag{5.21}$$

where $\mathbf{\Sigma}$ is the covariance matrix. In fact, there are infinitely many possible $\mathbf{K}$ satisfying this condition, in particular, the PCA uses the following whitening matrix:

$$\mathbf{K} = \mathbf{S}^{-1/2}\mathbf{U}^\top, \tag{5.22}$$

where $\mathbf{S}$ is a diagonal matrix of singular values of $\mathbf{\Sigma}$ and $\mathbf{U}$ is the matrix containing the eigenvectors of $\mathbf{\Sigma}$, which can be obtained by a singular value decomposition:

$$\mathbf{\Sigma} = \mathbf{U}\mathbf{S}\mathbf{V}^\top. \tag{5.23}$$

Note that since $\mathbf{\Sigma}$ is symmetric and positive semi-definite, the left and right eigenvectors are equals. An important remark is that the PCA depends on the scaling of the variables, this can be solved by standardizing the input to obtain unitary variances. The linear mapping between the parametrized samples $\mathbf{x}'$ and the corresponding whitened samples $\mathbf{z}$ is then obtained by

$$\mathbf{z} = \mathbf{K}\mathbf{x}'. \tag{5.24}$$

Finally, an orthogonal polynomial basis for $\mathbf{Z} \sim \mathcal{N}(0,1)^4$ can be constructed by the tensor product of the well known Hermite polynomials and the corresponding quadrature rule is derived from the three term recurrence coefficients. Let f be the mapping

$$f : \mathbf{X} \to \mathbf{Z}. \tag{5.25}$$

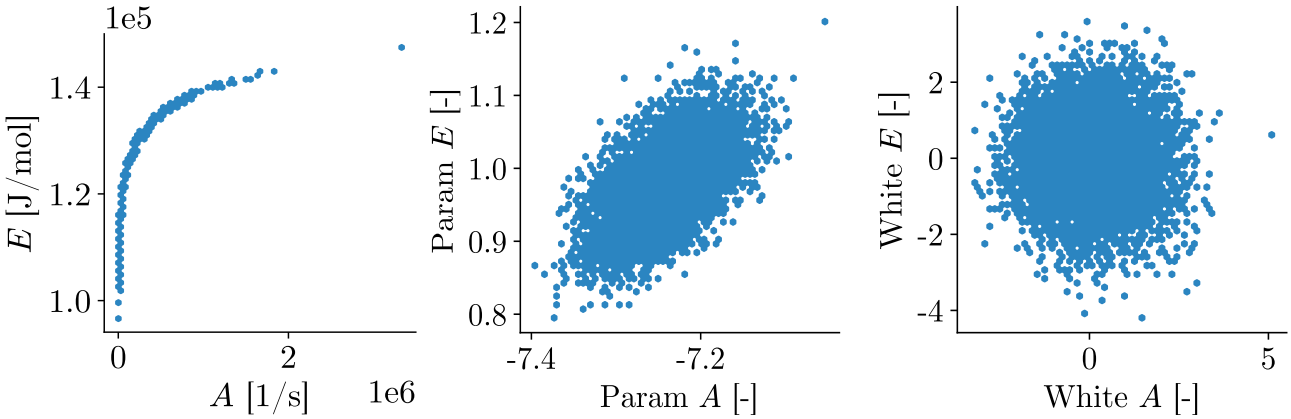

Figure 17: Overview of the correlation between A and E on the left, their parametrized image at the middle, and their whitened image by PCA on the right.

If the training data for the computation of the PCE coefficients are such that the response at a point $\mathbf{z}_j$ is the response of the pyrolysis process at $\mathbf{x}_j = f^{-1}(\mathbf{z}_j)$, the resulting PCE model $\hat{g}$ will thus take $\mathbf{z}_j$ as input parameter and generate the response corresponding to

$$\hat{g}(\mathbf{z}_j) \simeq g\left[f(\mathbf{z}_j)\right] = g(\mathbf{x}_j). \tag{5.26}$$

The resulting statistical moments and the convergence of the PCE computed by Gauss-Hermite quadrature at different temperatures and orders are presented in Figures 18 and 19. It is important to note that if the empirical mean and covariances are only an approximation of the actual moments, or if the original samples are not linearly separable, the resulting samples $\mathbf{z}$ may not exactly match the uncorrelated Gaussian distribution and thus generate some errors in the PCE.

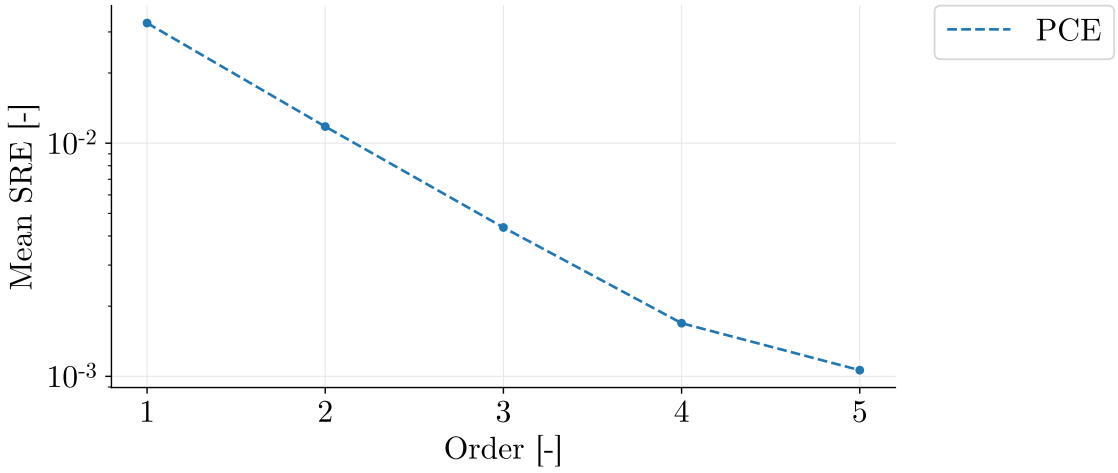

Figure 18: Mean error of a PCE computed by a Gauss-Hermite quadrature rule at different orders for the one-reaction pyrolysis test case.

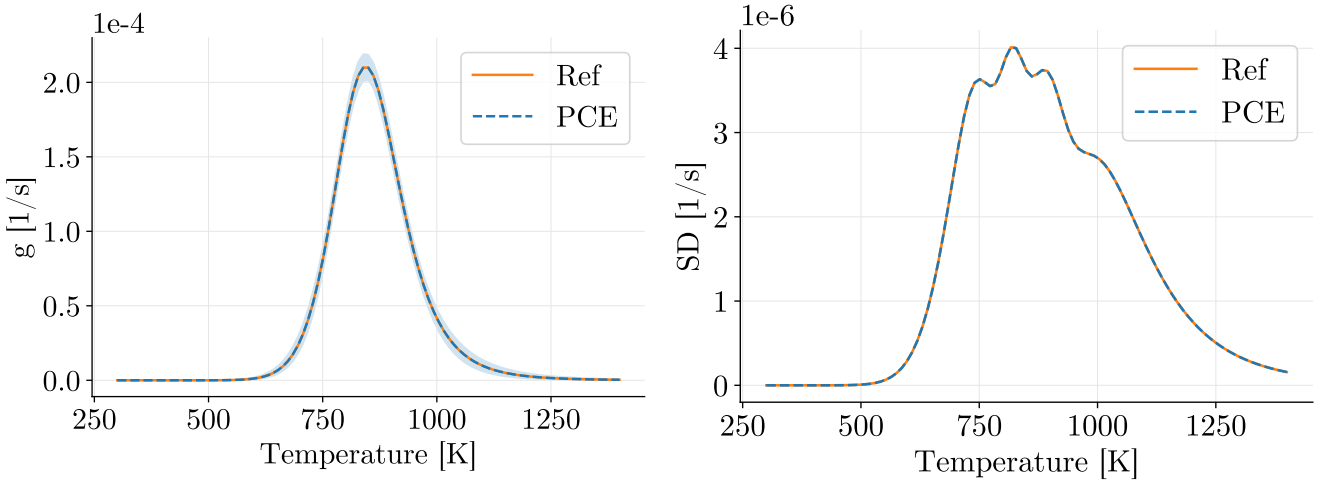

Figure 19: Range containing 98% of the outputs around the mean for a 6th order PCE computed by a Gauss-Hermite quadrature on the left, and its standard deviation on the right. The reference curve is the gas production in the one-reaction pyrolysis test case.

Since the whitened random variables are independent and follow a Gaussian probability distribution, a quasi-MC integration for computing the orthogonal polynomial basis and the associated PCE coefficients may be considered as an alternative to overcome the exponential increase of the number of points in Gaussian quadrature rules with the dimension. In particular, one can perform an inverse probability transform or the Box–Muller transform of the Halton or Sobol sequences to generate a low-discrepancy sequence for the normal distribution [19].

5.4 Two-Reactions Pyrolysis

The following test case is a pyrolysis reaction where two different solid phases are involved in the production of a single gas species. The reaction scheme considered is

$$\begin{cases} S_1 \xrightarrow{k_1} F_1 G, \\ S_2 \xrightarrow{k_2} F_2 G, \end{cases} \tag{5.27}$$

and the production of gas is given by an equation of the form

$$g = \rho \alpha_1 F_1 \frac{d\xi_1}{dt} + \rho \alpha_2 F_2 \frac{d\xi_2}{dt}, \tag{5.28}$$

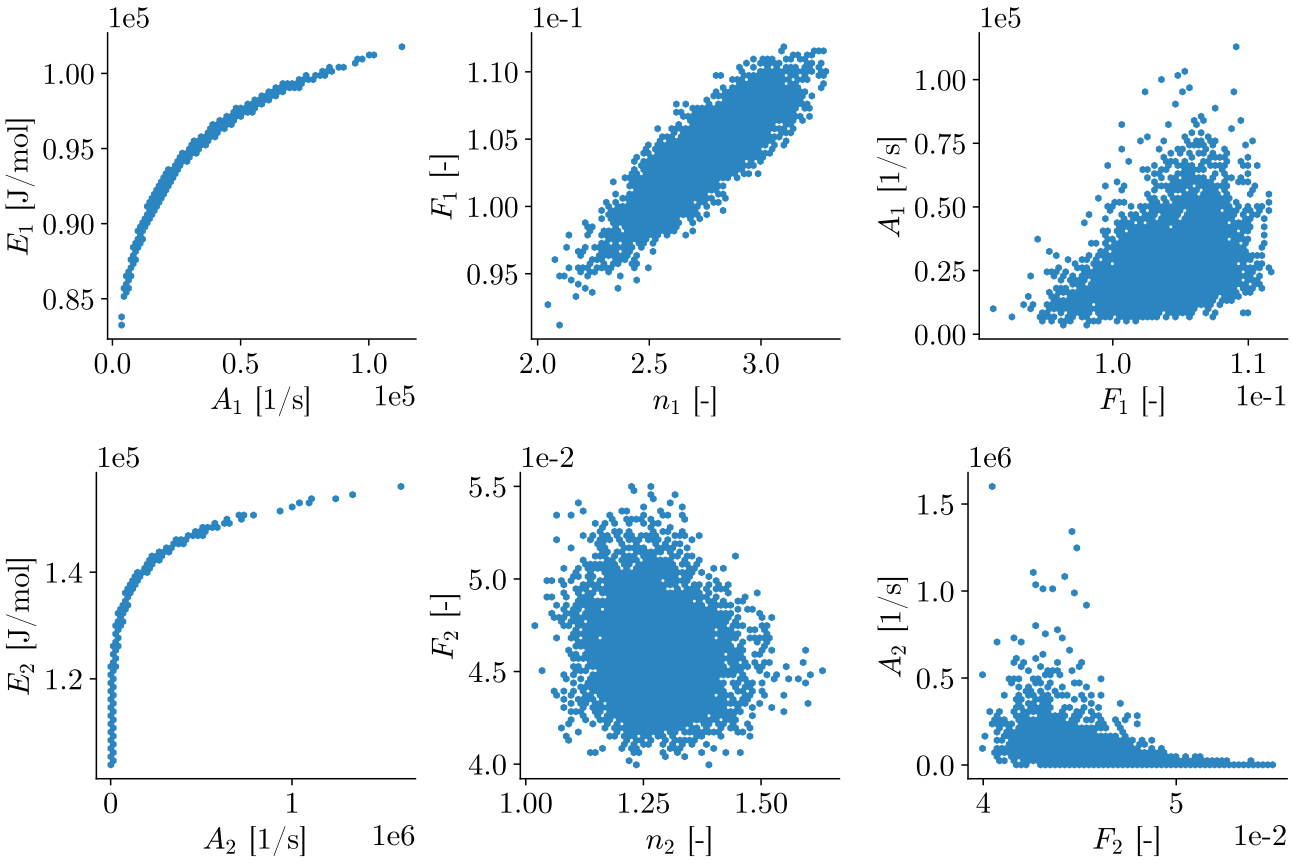

Figure 20: Overview of the correlation between different parameters in the two-reaction pyrolysis test case computed by Metropolis-Hastings algorithm.

where the advancement of the reaction $\xi \in [0, 1]$ follows an empirical relation with $f(\xi)$ assumed to be a non-linear function of the advancement:

$$\frac{d\xi_j}{dt} = f(\xi_j)k_j(T). \tag{5.29}$$

The stochastic parameters considered are the two sets of reaction variables $[A, E, n, F]$ and the deterministic parameter is the temperature, leading to a 8-dimensional problem. As in the previous example, the different parameter are non-Gaussian and highly correlated. An overview of the input distributions is presented in Figure 20.

5.4.1 Sparse Polynomial Chaos

The classical PCE of order p uses all the available monomials up to a total degree p, meaning that the total number of polynomials in the expansion, and so the number of unknown coefficients to estimate, is given by

$$n = \binom{d+p}{p} = \frac{(d+p)!}{d!\,p!}, \tag{5.30}$$

which may become considerably large when the dimension of the problem increases, hence the use of hyperbolic truncation schemes based on the q-norm to reduce the complexity of the expansion. Indeed, decreasing q leads to a drastic reduction of the number of polynomials, and so of the number of unknown coefficients as shown in Figure 21. The accuracy of the PCE for different q-norms is presented in Figure 22. It can be observed that taking $q = 0.7$ allows a reduction of the number of polynomials while preserving a good accuracy of the surrogate model. A part of the samples at our disposal has been used to compute the polynomials by GS orthogonalization, this set of points has been then reused to compute the associated coefficients by MC integration.

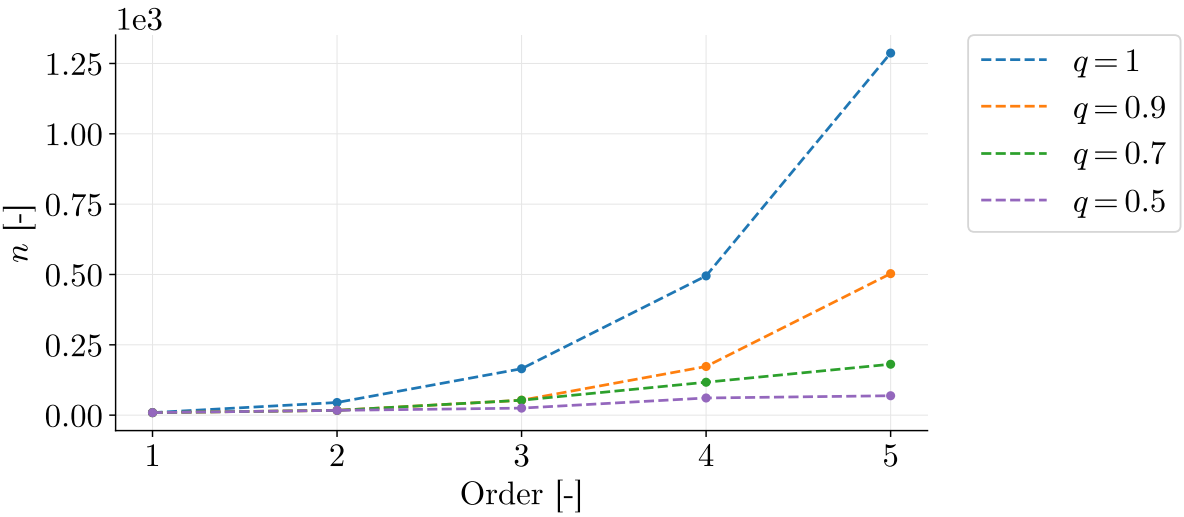

Figure 21: Number of polynomials in the orthogonal basis as a function of the order and the hyperbolic q-norm truncation for a 8-dimensional PCE.

A further reduction in the number of polynomials in the expansion using the LARS algorithm aims at eliminating irrelevant predictors to improve the stability of the PCE, reducing the computational cost and the number of unknowns to estimate in the expansion. At each iteration, a polynomial is selected by the LARS algorithm according to its correlation with the response of the reference model. It is important to note that a different set of polynomials is selected at each discretization of the response, meaning that the set of polynomials selected at a temperature T_1 may be different from the one selected at a temperature T_2. An estimation of the error is presented in Figure 23 for different number of iterations. It can be observed that only 60 iterations are sufficient to capture the behaviour of the reference solution with an optimal accuracy, it is also possible that large number of iterations may destabilizes the surrogate model by selecting irrelevant polynomials. The statistical moments of gas production are presented in Figure 24.

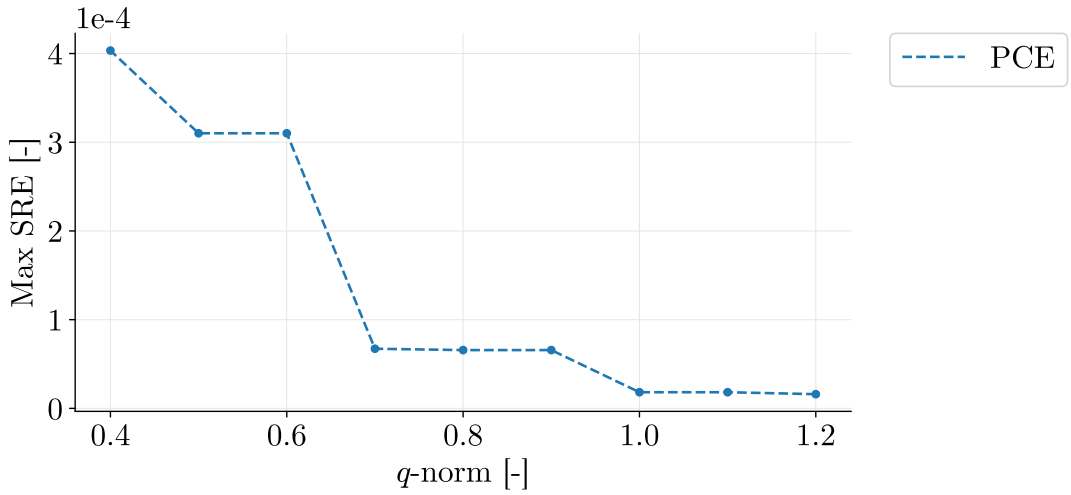

Figure 22: Mean error of a 4th order PCE computed by MC integration and a large number of integration points for different q-norm hyperbolic truncation schemes.

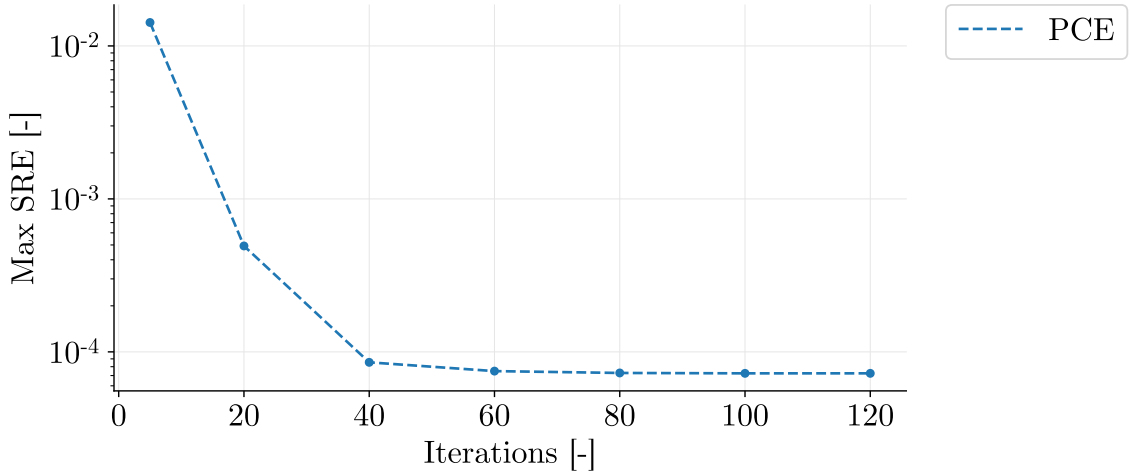

Figure 23: Maximum error of a 4th order PCE with a hyperbolic truncation of $q = 0.7$. The polynomials are first selected by the LARS algorithm and the resulting coefficients are updated by a least squares regression for a better accuracy.

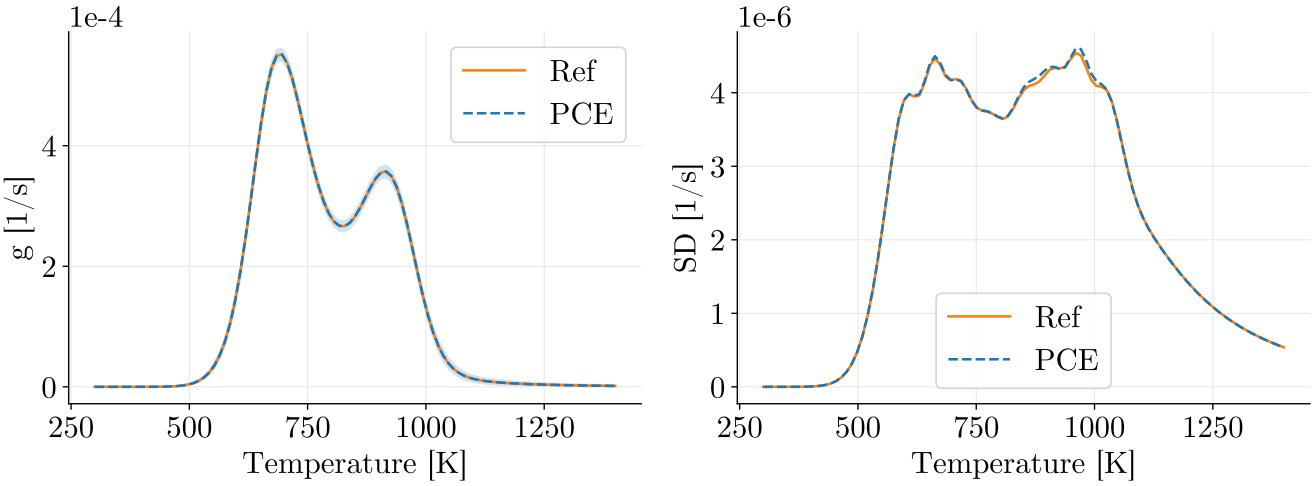

Figure 24: Range containing 98% of the outputs around the mean for a 4th order PCE computed by LARS on the left, and its standard deviation on the right. The reference curve is the gas production of the two-reaction pyrolysis test case.

5.5 One-Dimensionnal Ablation

The porous material analysis toolbox PATO is a modular analysis platform for multiphase porous reactive materials based on OpenFoam and developed by NASA. Such complex models are used to analyse the behaviour of ablative thermal shields with high fidelity, but are typically expensive to compute. Thereby, the primary interest of the polynomial chaos is to reduce this computational cost in order to allow statistical studies. The one-dimensional ablation test case presented in [40][7] and Figure 25 consists of a sample of a theoretical ablative composite of 5 cm heated on its side for 1 minute at atmospheric pressure with adiabatic boundary condition on the other side. The heat flux will heat the virgin material until it undergoes a pyrolysis reaction, then decomposes into a porous matrix. The gas produced by the pyrolysis and the eroded char is then reinjected into the boundary layer of the incoming flow.

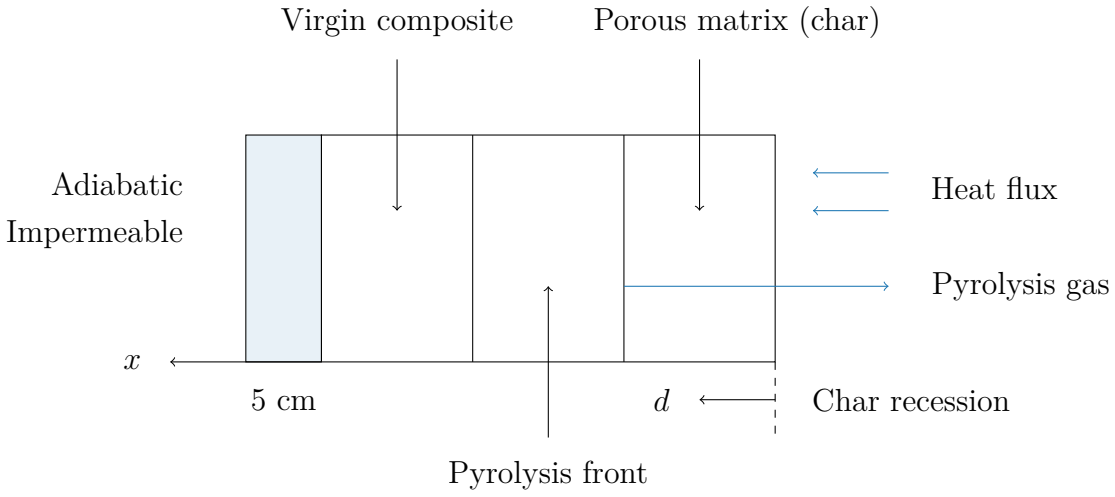

Figure 25: Progressive decomposition of the theoretical ablative virgin composite into a pyrolysis char due to the inflow heat flux at its surface.

5.5.1 Independent Parameters

The process involves 2 pyrolysis reactions and so 8 kinetic parameters. As we do not have any information about the actual distributions in play, the stochastic parameters will first be described either by the uniform or normal probability density functions presented in Figure 26 and Table 2. Finally, the reference model provides different outputs such as the recession depth d of the char and virgin material respectively defined as the length of the sample with less than 2% and more than 98% of resin, the blow rate of pyrolysis gas and the spatio-temporal temperature profile in the sample.

As we consider stochastic independence between the input parameters, a straightforward way to obtain an orthogonal basis is to compute the tensor product of Hermite and Legendre polynomials. However, the reference model is expensive to evaluate, the PCE coefficients must therefore be computed with a minimal number of points and the tensor product of Gauss-Hermite and Gauss-Legendre quadrature rules is not efficient. Indeed, the number of points in tensor product Gaussian quadrature rules increases as

$$n = (p+1)^d, \tag{5.31}$$

Parameter	Distribution
F_1 [-]	$\mathcal{U}(0.1, 0.4)$
F_2 [-]	$\mathcal{U}(0.1, 0.3)$
A_1 [1/s]	$\mathcal{N}(12, 1) \times 10^3$
A_2 [1/s]	$\mathcal{N}(50, 2) \times 10^7$

Parameter	Distribution
E_1 [J/mol]	$\mathcal{N}(71, 4) \times 10^3$
E_2 [J/mol]	$\mathcal{N}(17, 0.6) \times 10^4$
n_1 [-]	$\mathcal{N}(3, 0.2)$
n_2 [-]	$\mathcal{N}(3, 0.2)$

Table 2: Kinetic parameters of the pyrolysis reactions involved in the one-dimensional ablation test case with their probability distributions.

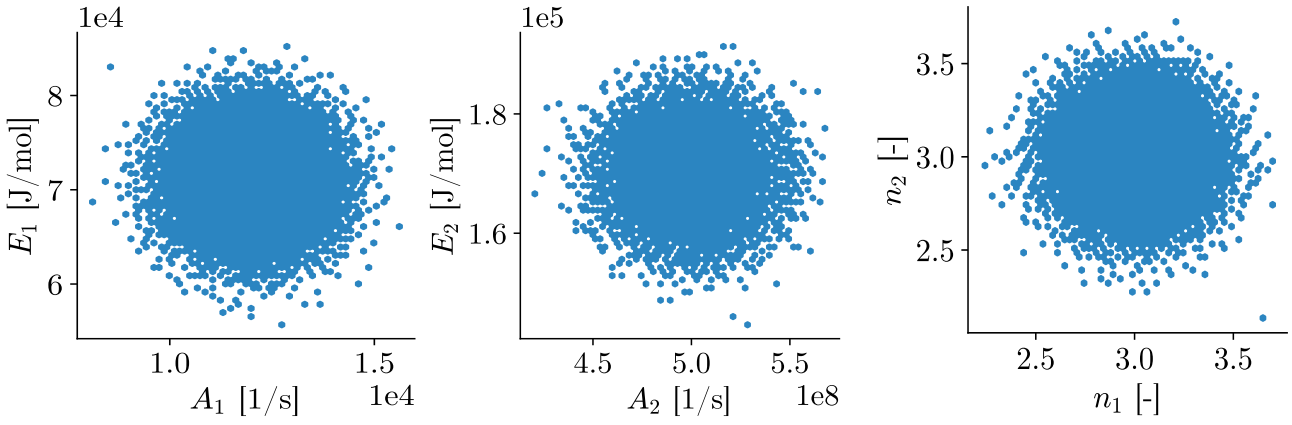

Figure 26: Overview of the input parameter distributions for the one-dimensional ablation test case. We first consider the random variables as statistically independent.

where p is the order of the quadrature and d is the dimension. To remedy this problem, we propose to compute the basis by GS orthogonalization and generate an embedded quadrature rule in order to estimate the PCE coefficients. Figure 27 highlights the equivalent accuracy of the surrogate model computed by a embedded quadrature rule and the MC sample from which it has been extracted. The error has been evaluated using different samples than the ones used to build the surrogate model. It is important to note that the number of points in the embedded quadrature is smaller than in the MC sample and depends on the number of polynomials in the basis, which increases with the dimension at a significantly less degree than for Gaussian quadrature rules.

It can be first observed that the error of the char recession depth is higher than for the other outputs of the model. Table 3 shows that an important part of the variance is due to the parameter n which is related to the decomposition speed of the virgin material into char, for large n, the virgin material decomposes slowly and the char takes longer to appear. The high relative variance due to the strong dependence to n of this output requires higher order polynomials and a larger number of quadrature points to be accurately represented.

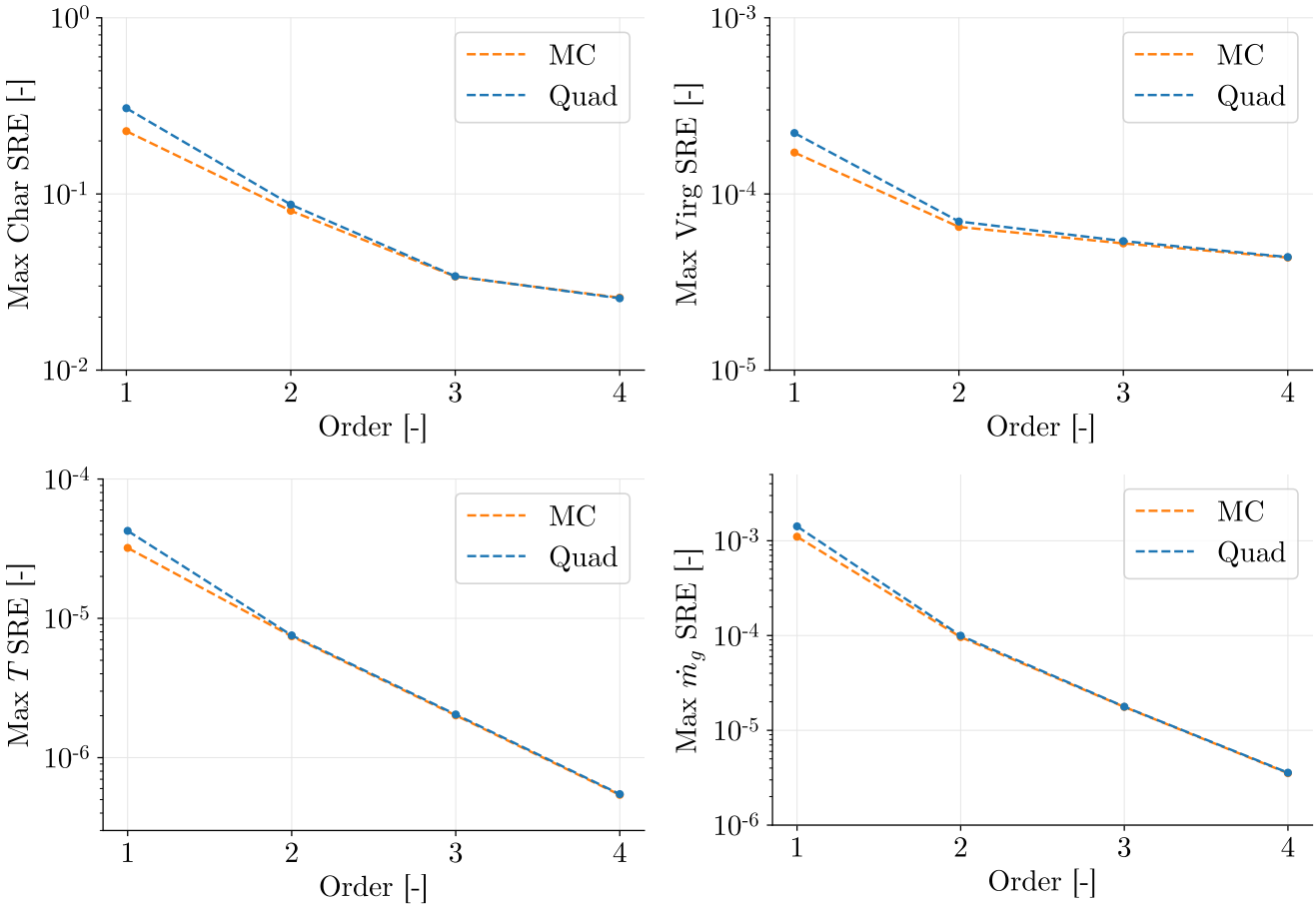

Figure 27: Maximum error of a PCE computed by a MC integration of 2×10^4 points, and an embedded quadrature rule extracted from this MC sample at different orders. The reference solution is the one-dimensional ablation test case.

Figure 28 shows that A and E have an opposite effect on the output. As predicted by its small Sobol index, the variation of A has a lower influence than n on the char recession. This suggests that the range of possible values for A has been assumed too small. Indeed, by assuming statistical independence between the inputs, we limited the sample space in order to avoid forbidden combinations of the parameters.

Char	E_1	E_2	n_1	n_2
S [-]	0.12	0.03	0.5	0.08
S_T [-]	0.16	0.06	0.58	0.12
Char	F_1	F_2	A_1	A_2
S [-]	0.09	0.04	0.008	7×10^{-4}
S_T [-]	0.13	0.06	0.014	0.01

Table 3: First and total Sobol sensitivity indices relative to the char recession depth.

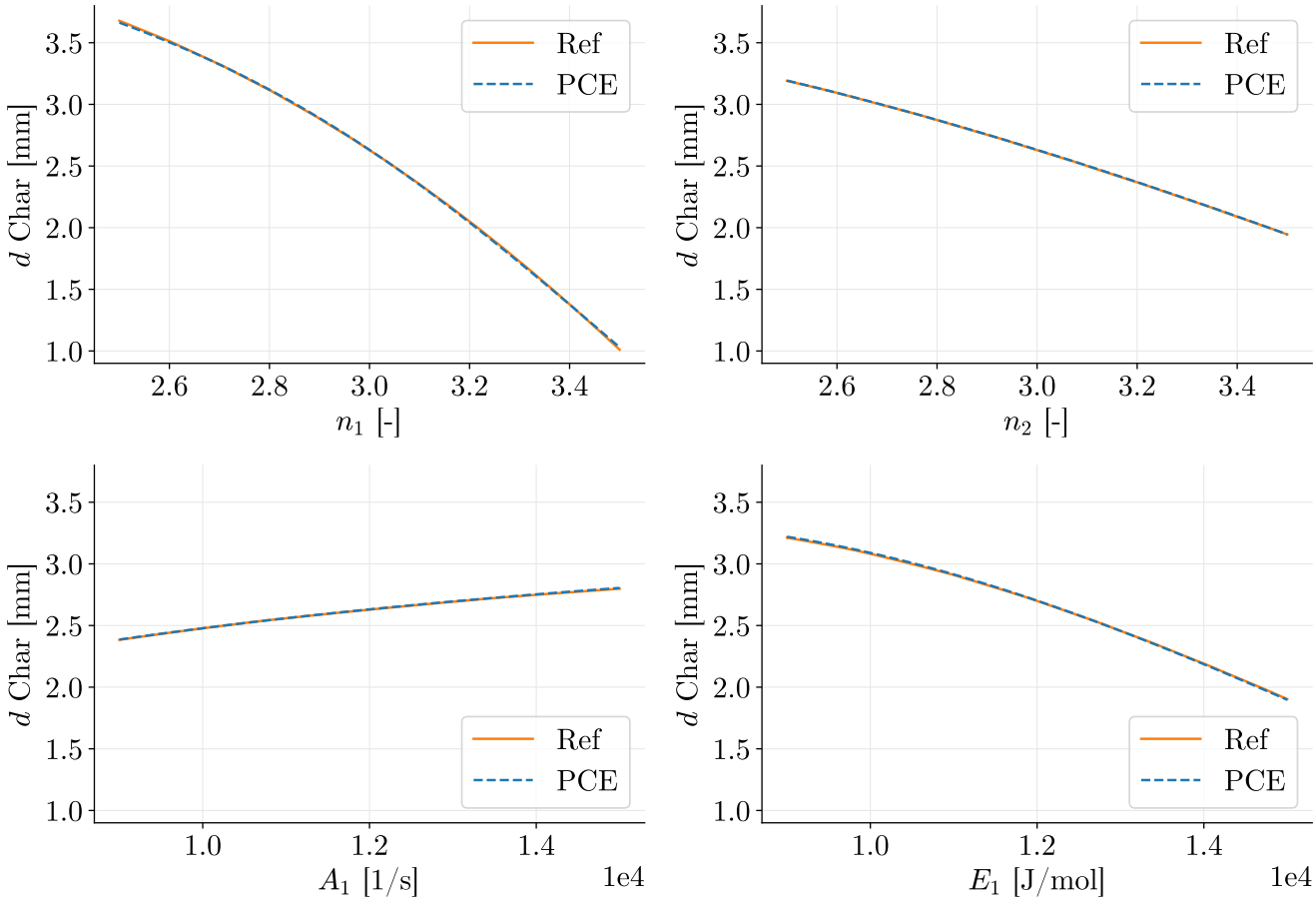

Figure 28: Char recession depth of the one-dimensional ablation test case after 1 min as a function of n, A and E. The other parameters are fixed at their mean value.

The difference between the embedded quadrature and the MC integration for the char recession may be explained by the location of the selected points in the domain. Indeed, the quadrature rule tends to select the points at the boundaries of the domain while the MC integration selects the points mainly in the center of the domain. Figure 29 shows that the error mainly depends on n_1 and rapidly increases for higher values of this parameter, since we decided to take into account the maximum error to assess the convergence of the solution, the few points at large n_1 considerably increase the error.

Figure 30 displays the statistical moments of the different outputs of the ablation process, the temperature of the material is recorded at each time step by 8 probes disposed at different distances from the heated surface, allowing the representation of a spatio-temporal temperature profile. The average temperature profile is displayed in Figure 31 and its standard deviation is presented in Figure 32 for each probe.

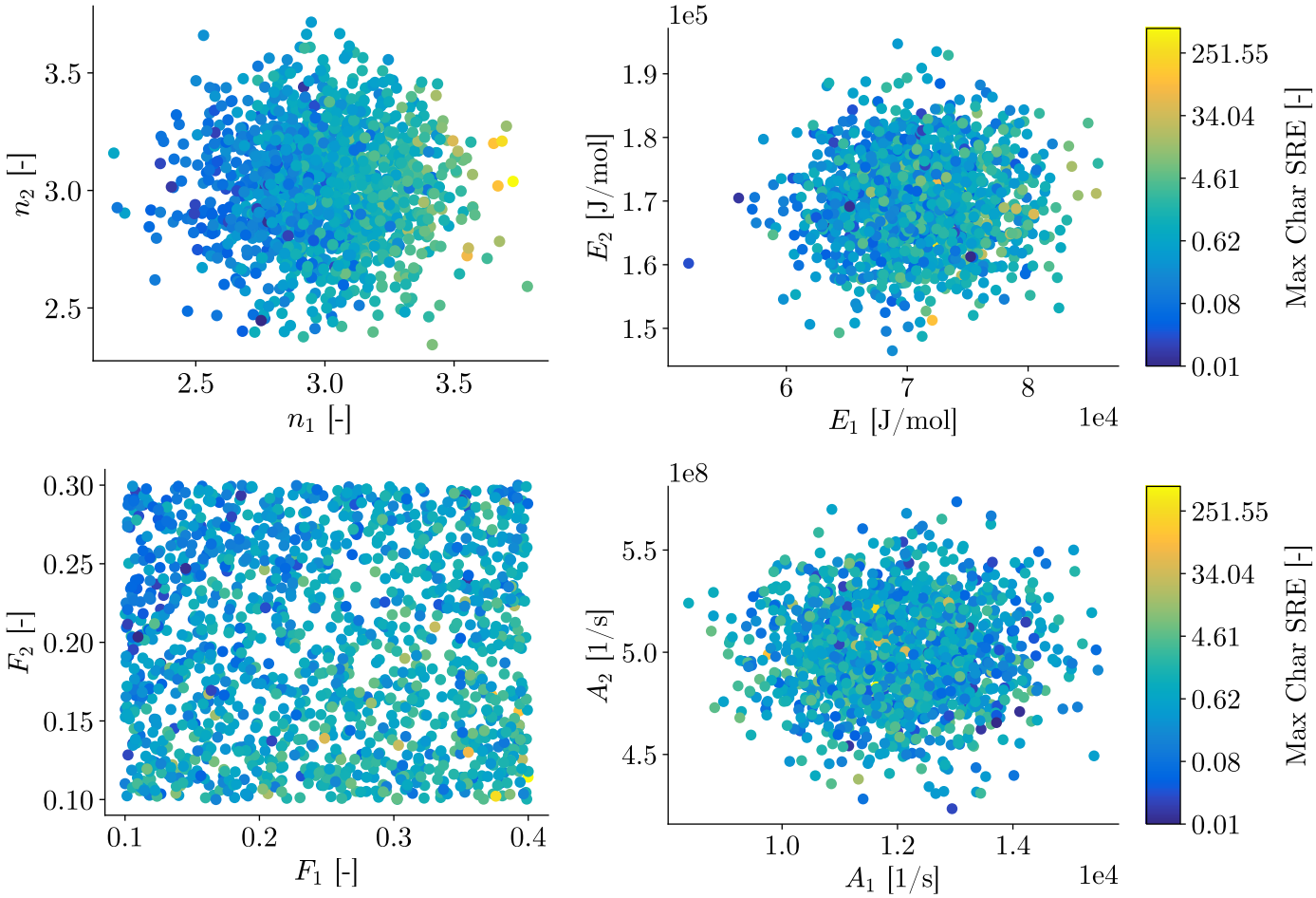

Figure 29: Maximum error of a 3rd order PCE computed by an embedded quadrature rule of $1e3$ integration points for the char recession depth in the one-dimensional ablation test case. The coloured dots correspond to the integration points projected in a two dimensional space.

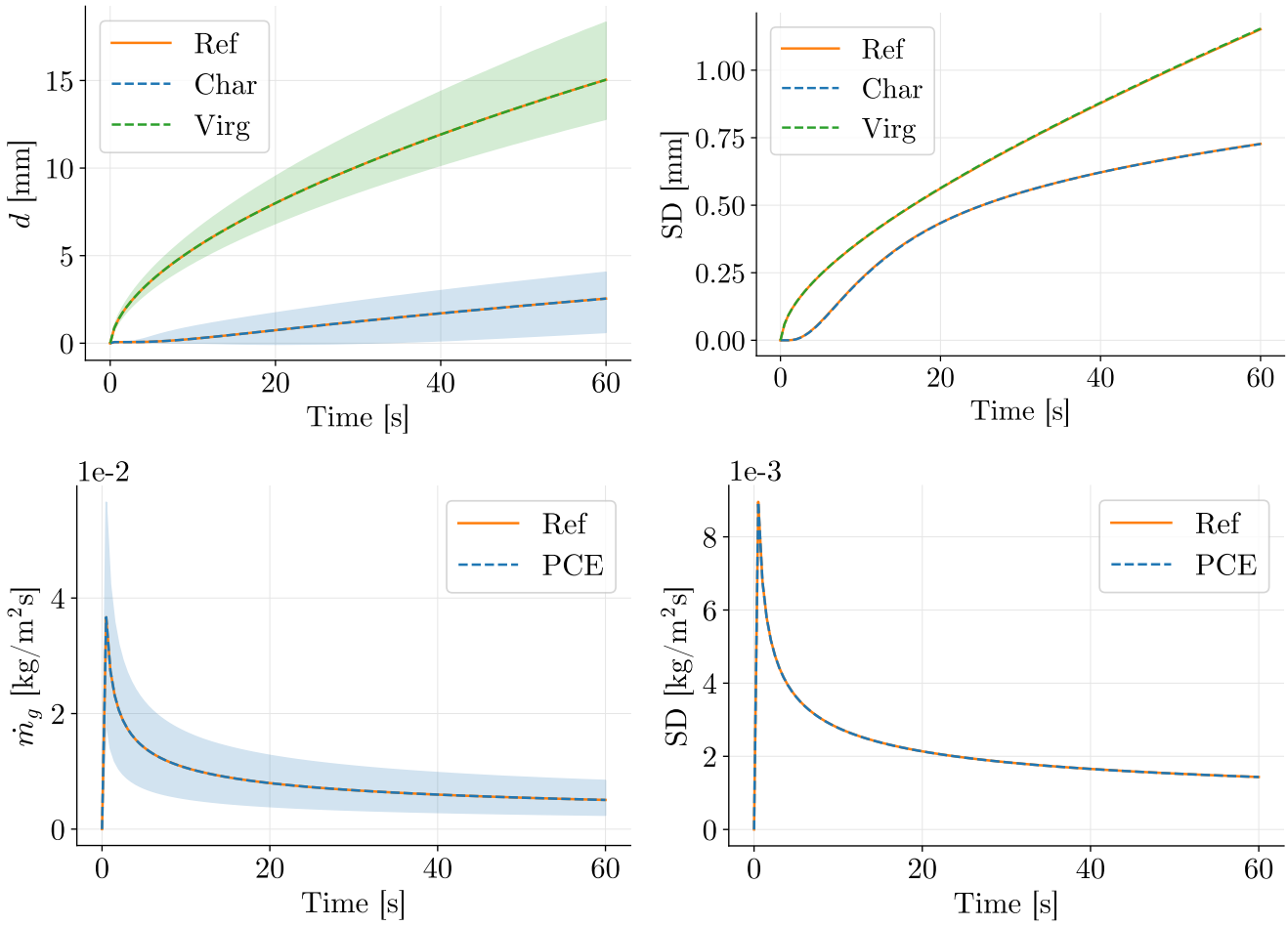

Figure 30: Range containing 98% of the gas blowing rate and the recession depths around the mean for a 4th order PCE computed by an embedded quadrature rule of 3×10^3 points on the left, and their standard deviation on the right. The reference curve is the reference solution of the one-dimensional ablation test case.

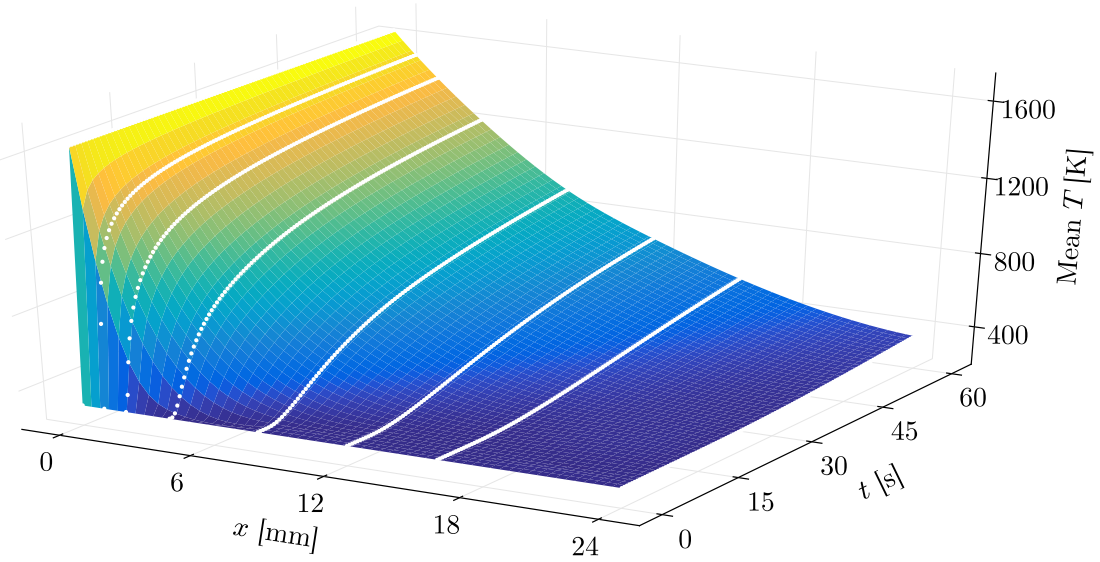

Figure 31: Mean of the temperature profile in the material obtained with a 4th order PCE computed by an embedded quadrature rule of 3×10^3 points. The coloured surface is the reference solution and the white dots are the response of the PCE at the probes.

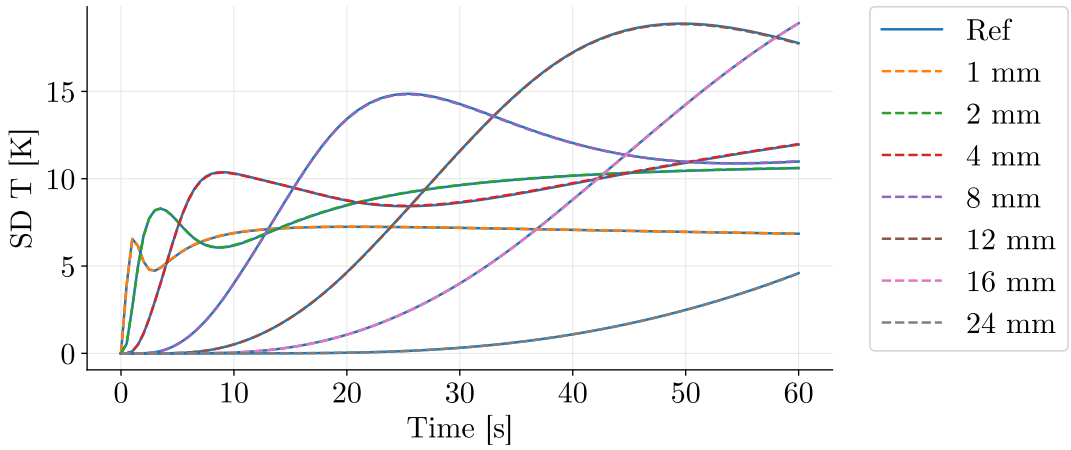

Figure 32: Standard deviation of the temperature profile in the material obtained with a 4th order PCE computed by an embedded quadrature rule of 3×10^3 integration points.

5.5.2 Correlated Parameters

We propose to analyse the effect of a correlation by generating a sample using the Metropolis-Hasting algorithm implemented in `Pybitup` for the two-reaction pyrolysis process with respect to fake experimental data. Although the differential equations involved in both algorithms are similar, it is important to note that the one-dimensional ablation process is more complex and that the resulting distributions may be different from actual experimental data. The distributions are presented in Figure 33.

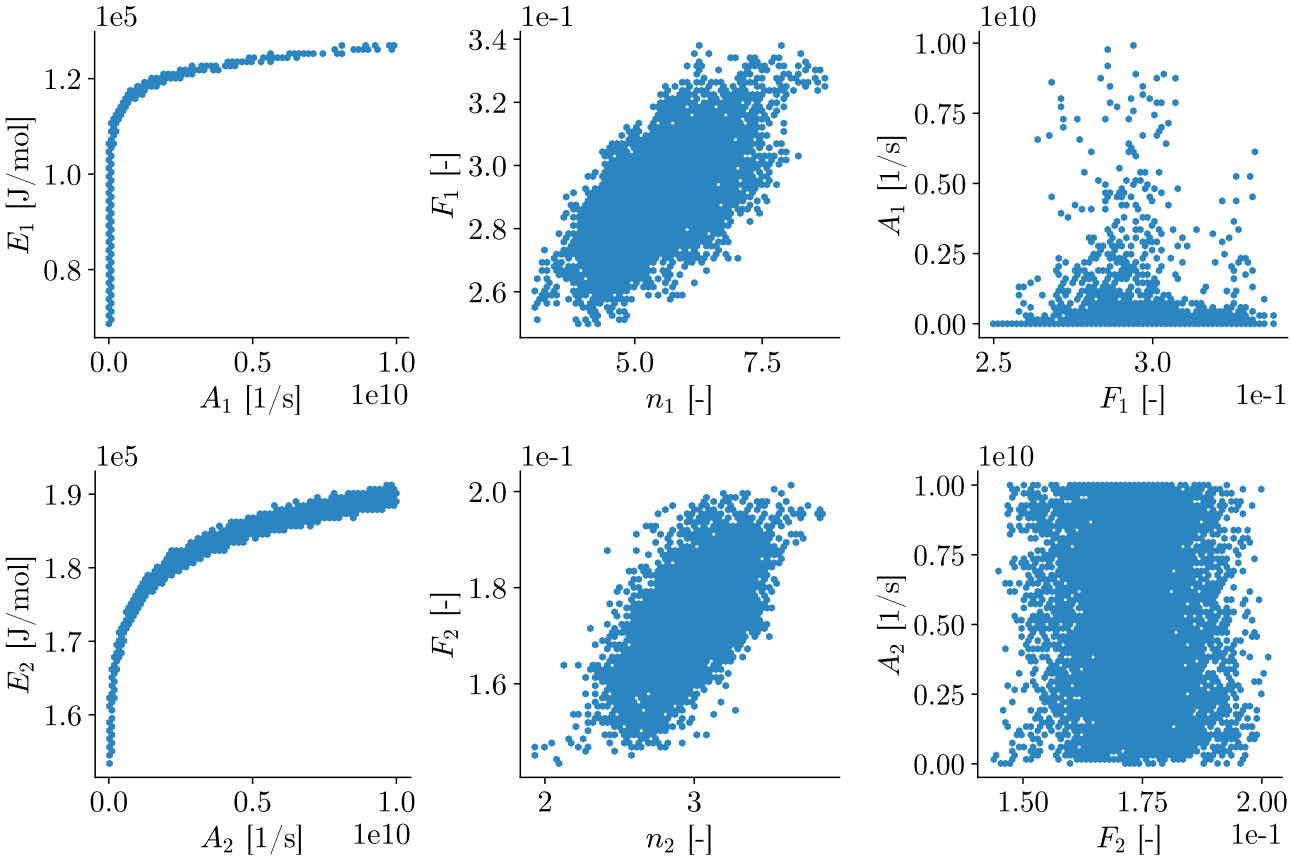

Figure 33: Overview of the correlation between different input parameters of the one-dimensional ablation test case obtained from a posterior distribution.

In this test case, we will use a positive embedded quadrature rule, this latter provides a more accurate estimation of PCE coefficients compared to approximate Fekete points thanks to its lower condition number. As a reminder, the condition number of a quadrature rule $\{\mathbf{x}_j, w_j\}$ is given by

$$c = \sum_j = 1^n |w_j| \tag{5.32}$$

and represents the amplification of rounding errors in the estimation of the integral. By definition, $c = 1$ for a positive quadrature rule, but the approximate Fekete points generally contains some quadrature points of negative weights. A comparison between the MC integration is presented in Figure 34. The statistical moments of the different outputs are presented in Figures 35 and 36. It is clear that the total standard deviations of the outputs is smaller than in the previous case. This can be explained first by the smaller variance of F, then by the fact that A and E have opposite effects on the pyrolysis reaction. As the correlation imposes A to increase when E increases, their effect on the solution tends to cancel out and the variance of the output decreases despite of the fact that both parameters display a large marginal variance. This observation suggests that there exists a nonlinear relation between A and E leading to a constant value involved in the governing equations of the ablation process, as shown in Figure 37.

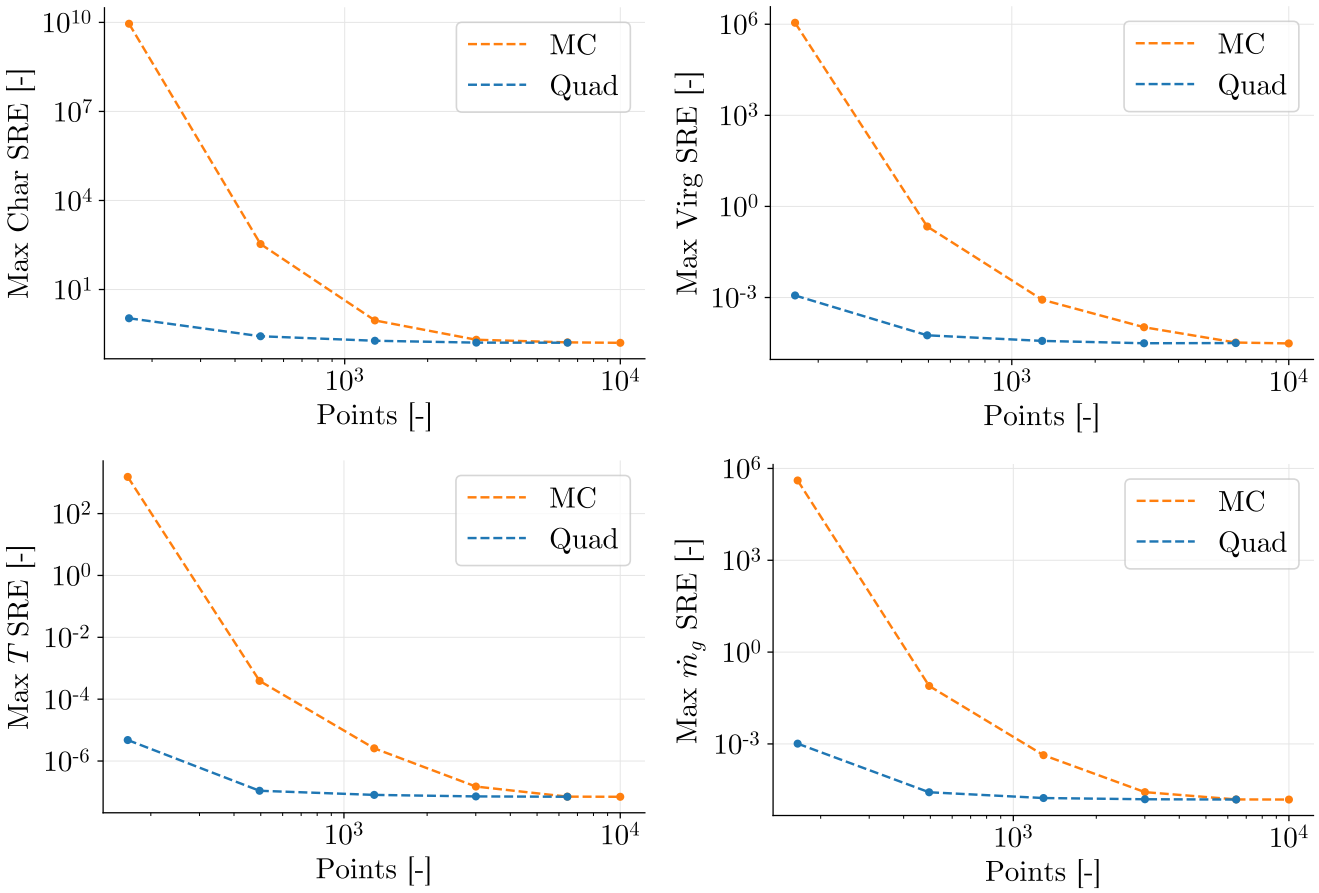

Figure 34: Maximum error of a 3rd order PCE computed by an embedded quadrature rule and MC integration for different number of points. The reference solution is the one-dimensional ablation test case.

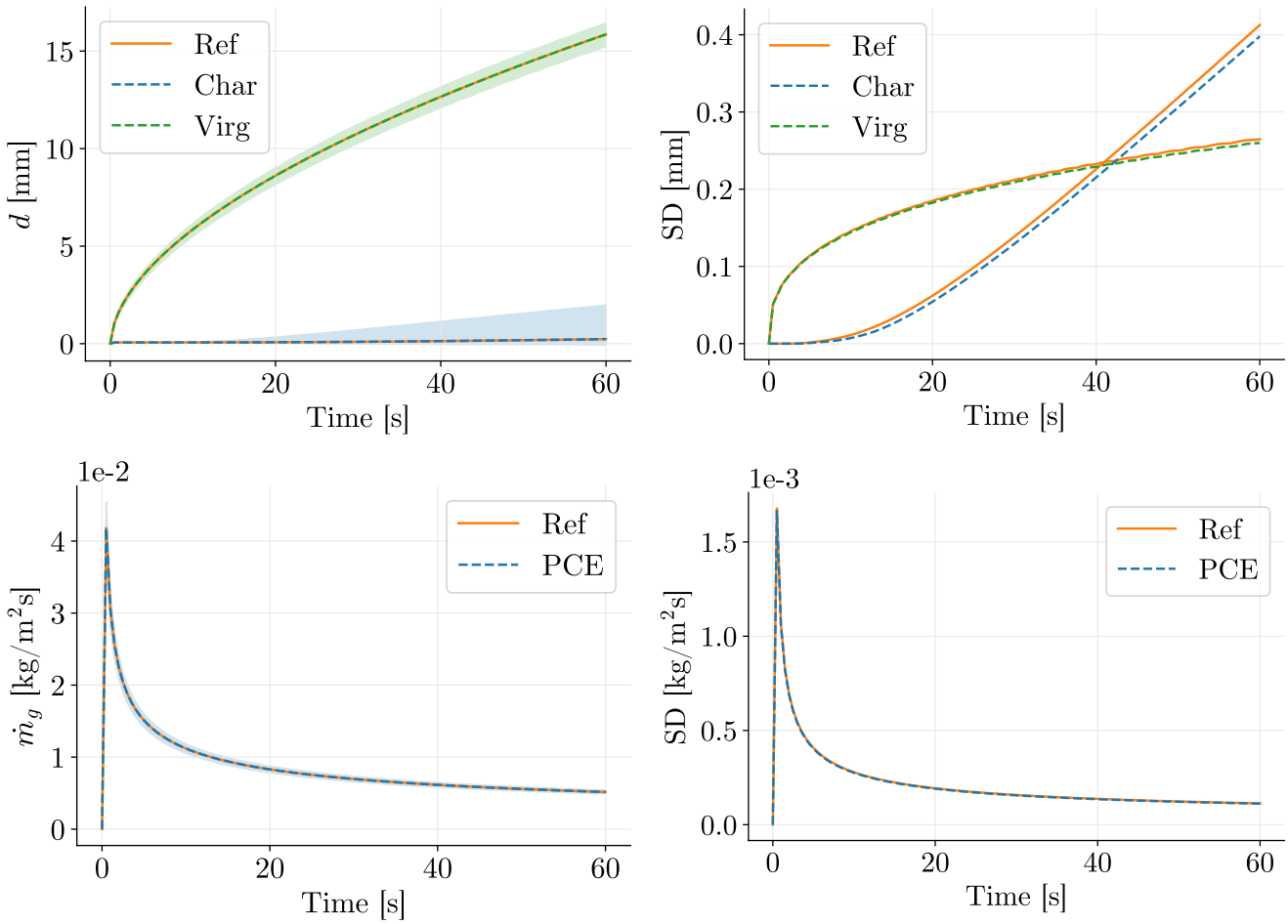

Figure 35: Range containing 98% of the gas blowing rate and recession depths around the mean for a 4th order PCE computed by a positive embedded quadrature rule of 3×10^3 integration points and 20 iterations of the LARS algorithm. The reference curves are the reference solutions of the one-dimensional ablation test case.

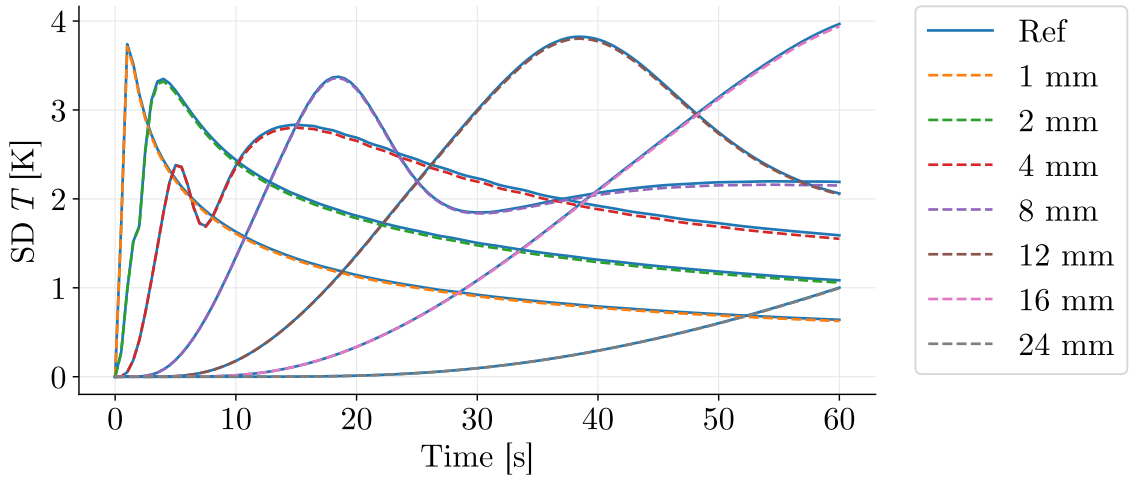

Figure 36: Standard deviation of the temperature profile in the material obtained with a 4th order PCE computed by a positive embedded quadrature rule of 3×10^3 integration points and 20 iterations of the LARS algorithm.

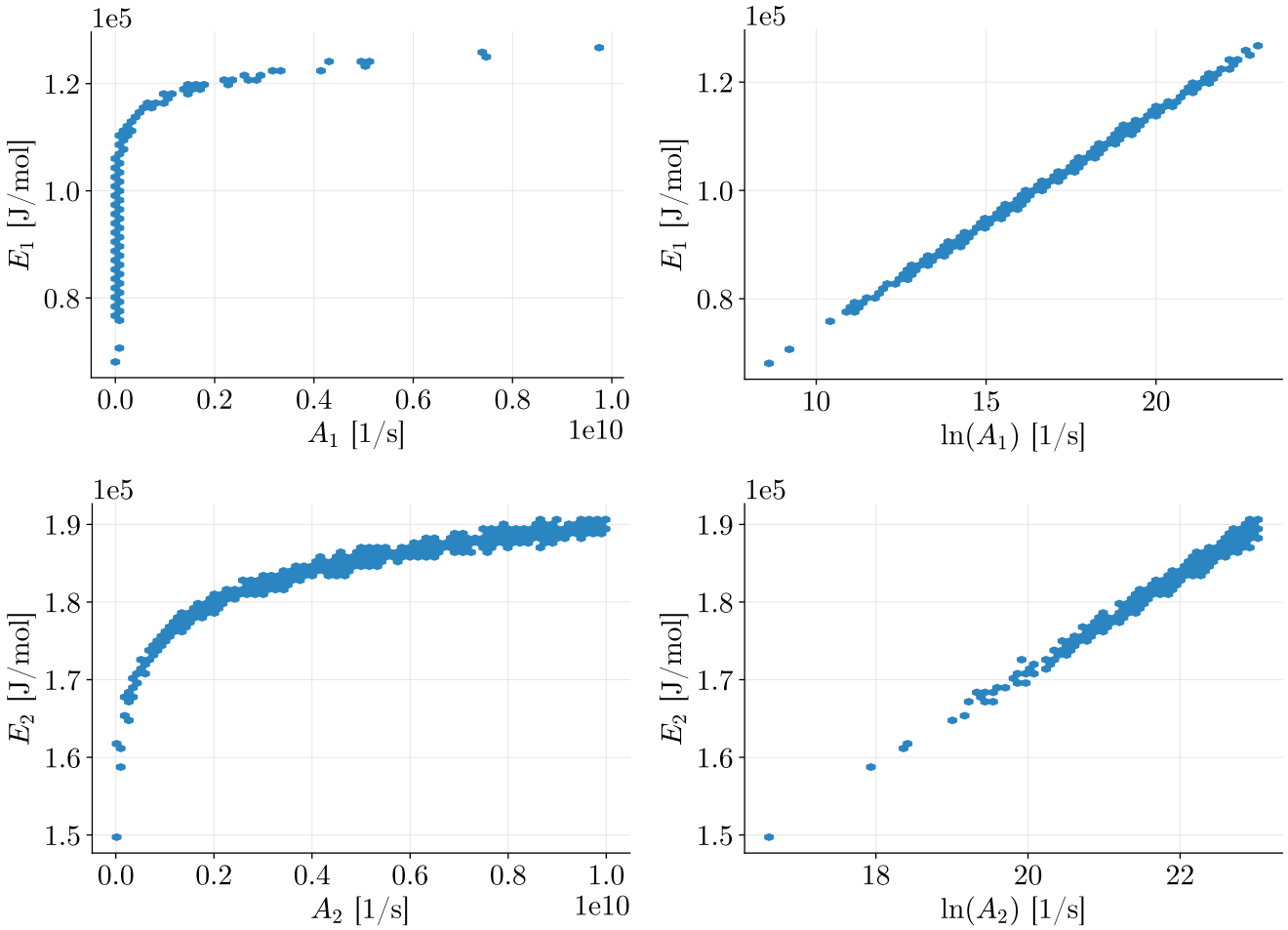

Figure 37: Correlation between the natural logarithm of the Arrhenius pre-exponential factor and the activation energy for the one-dimensional ablation test case obtained by a Bayesian posterior.

5.5.3 Extremal Parameters

It was previously shown that extremal values of the input parameters are typically difficult to estimate by the PCE as they are located at the boundaries of the validity domain of the surrogate model. Nevertheless, the model response related to a large part of the sample space, including extremal values, can be correctly represented provided one remains within the validity domain determined by the training data. An example is presented in Table 4 and Figure 38.

Parameter	A_1 [1/s]	E_1 [kJ/mol]	n_1 [-]	A_2 [1/s]	E_2 [J/mol]	n_2 [-]
Max $(A, E)_1$	9.9×10^9	1.2×10^5	6.6	9.4×10^8	1.7×10^5	3.1
Min $(A, E)_1$	5322	6.8×10^4	3.1	3.1×10^9	1.8×10^5	2.8

Table 4: Parameter combinations containing the maximal and minimal values of A_1 and E_1 in the one-dimensional ablation test case, with $F_1 = 0.27$ and $F_2 = 0.18$.

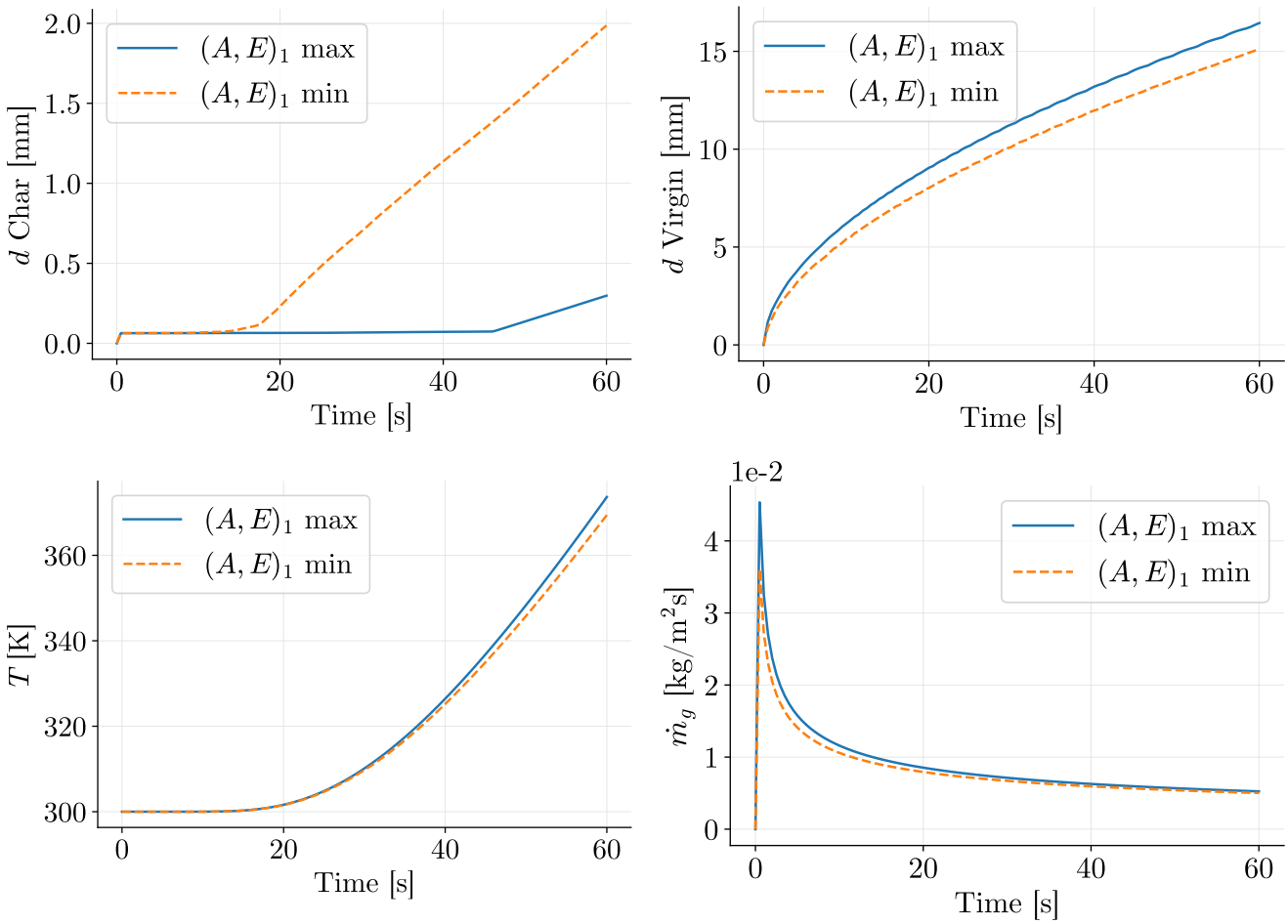

Figure 38: Outputs of the one-dimensional ablation test case for extremal values of the Arrhenius pre-exponential factor and the activation energy of the first reaction.

Chapter 6

Conclusion

6.1 Summary

The first part of this work was a literature review about the so-called polynomial chaos expansion. On the one hand, related to the construction of an orthogonal polynomial basis by recurrence relation or GS orthogonalization, and on the other hand, related to the computation of the PCE weighting coefficients by spectral projection, leasts-squares collocation or LARS algorithm. In addition, different methods were proposed to take advantage of the functional decomposition provided by the PCE. For instance, the computation of the mean, the variance and the Sobol sensitivity indices is straightforward. It was shown that orthogonal bases obtained by three term recurrence relation provide an accurate quadrature rule for the computation of inner products and are probably the best choice when such relations are available, however, Gaussian quadrature rules suffer from the curse of dimensionality and are limited to independent variables. In contrast, the GS orthogonalization allows the computation of orthogonal polynomials with respect to an arbitrary weight function regardless of the correlation between the random variables, but the quality of the basis thus obtained depends on the accuracy of the technique used to perform the integrations.

Once the polynomial basis has been obtained, the PCE coefficients can be computed by spectral projection, which assumes a perfect orthogonality between the polynomials forming the basis. The least squares method will determine the coefficients that minimize the error with respect to the reference solution. Finally, the LARS algorithm will select the polynomials which are the most correlated with the reference model by minimizing the angle with a residual. All these methods require the evaluation of inner products and so the evaluation of integrals. When Gaussian quadrature rules are not feasible, due to the high dimension or the correlation, different integration techniques have been explored. For instance, MC integration is the most robust and simplest solution, but may lead to a large number of integration points due to its low convergence, which is not always feasible when the reference model is costly to evaluate. Nevertheless, this method remains a good alternative. The second possibility is the embedded quadrature rule, aiming at extracting a quadrature rule with a minimum number of points from an already existing quadrature rule having a larger number of points. This technique suffers from the increase in the number of dimensions to a lesser extent than Gaussian quadrature rules. We have shown in the last chapter that the embedded quadrature rule allows the computation of a PCE with far fewer integration points than MC integration. Indeed, Chapter 5 was dedicated to the application of the methodology to different test cases:

- A surrogate model was first computed for the Ishigami function as an introduction to strong nonlinear and non-monotonic functions. In addition, the estimation of the sensitivity indices and statistical moments have been shown as equivalent to their analytical solution.
- The second problem was a pyrolysis process of one and two reactions. This test case allowed the comparison and the validation of the different tools at our disposal to compute a surrogate model for correlated input parameters in addition to reduce the complexity of the PCE in higher dimension by limiting the number of polynomials in the surrogate model.

In particular, the embedded quadrature rules allowed to reach, with far fewer evaluation of the reference model, an accuracy comparable to using a large sample set in a reference Monte Carlo approach. Afterwards, decorrelation and whitening methods were used to compute a classical PCE with Hermite polynomials and a Gauss-Hermite quadrature rule.

- The last application was a large scale problem of one-dimensional ablation, a sample of solid material is heated at a constant temperature until it undergoes a pyrolysis reaction. As the numerical model is expensive to evaluate, an embedded quadrature rule was used in order to compute an accurate PCE with a minimum number of integration points. Finally, the surrogate model was used for statistical studies of the output related to different sets of input parameters and to quantify the impact of the correlation between the different variables of the problem.

In conclusion, the goal of this thesis was to successfully demonstrate the possibility of computing a cheap and accurate surrogate model for complex problems in moderately high dimensions when the input parameters are correlated, in particular, in the context of pyrolysis reactions occurring in thermal ablation shields. To this end, a PCE library has been implemented in `Pybitup` and different algorithms proposed in the literature were compared. As thermal protection systems involve a large number of physical parameters whose exact value are not known exactly, the need for a cheaper surrogate model arises when one wants to investigate and mitigate the effect of uncertainties on the output of a numerical model whose execution time is long.

6.2 Future Work

- First, the PCE developed during this thesis is still limited to a moderately high dimensionality of the input uncertainty, more complex algorithms have been proposed in the literature to address different types of problems or to improve the accuracy of the PCE. For instance, the global behaviour of the model could be approximated by a sparse PCE while the local variations of the output could be modeled by a Gaussian random process. Thus, some methods such as the PCE-Kriging [45] seek to exploit sparsity. In the same way, an improvement of the algorithms extracting the embedded quadrature rule should be considered, in particular, the simplex and the iterative weight cancellation suffer from their computational cost and rounding errors when the training sample set is large, but provide a better accuracy than other quadrature rules containing negative weights.

- Moreover, future works can focus on more elaborated test cases for statistical studies. Indeed, the ablation process presented in this paper is limited to one spatial dimension, but other algorithms of higher fidelity are available in the porous material analysis toolbox. Consequently, the higher number of input parameters as well as the complex behaviour of the response may highlight some limitations of the methods and the requirement of more advanced PCE meta-modeling techniques. To this end, different approaches such as the use machine learning, which are not directly related to the PCE, have been proposed in the literature to improve the efficiency of the method.

Although significant advantages of the PCE on machine learning techniques were highlighted in [46], where it was shown that a PCE purely trained on data can yield to an accuracy comparable to machine learning regression models, a recent paper shows that the integration of an artificial neural network with the polynomial chaos (PCE-ANN) can provide similar results using much less computational effort [47].

- Finally, as one of the main challenges when dealing with high dimensional data is the computational cost, other improvements of the code may be achieved by transcription the Python script into lower level languages such as `C` and `C++`, as they are intrinsically faster and allow the use of efficient libraries for high performance parallel computing such as `MPI` and `OpenMP`. However, it could be more interesting to improve the efficiency of the code with `Cython` [48], a programming language aiming to give `C`-like performance with code written mostly in Python using optional additional `C`-inspired syntax. `Cython` thus provides the benefits of both low and high-level languages.

Chapter 7

Appendix

7.1 Finding $z : A^\top z = 0$

In the following sections, we now address how to find a solution to (3.37). If the polynomial basis is orthonormal with respect to the Monte Carlo samples, the Vandermonde matrix has orthogonal columns and thus satisfies

$$\begin{aligned} \mathbf{A}^\top \mathbf{A} &= \mathbf{B} = m\mathbb{I}, \\ \mathbf{A}^{-1} &= \mathbf{A}^\top \frac{1}{m}, \end{aligned} \tag{7.1}$$

where $\mathbf{A}^{-1}$ denotes the Moore–Penrose inverse and m is the number of Monte Carlo samples. Indeed, the columns of the Vandermonde matrix form an orthonormal basis with respect to the discrete inner product

$$\langle p_i, p_k \rangle = \frac{1}{m} \sum_{j=1}^{m} p_i(x_j) p_k(x_j), \tag{7.2}$$

where the points $\{\mathbf{x}\}$ are the Monte Carlo samples for which the polynomials are orthogonormal. The elements of the resulting matrix $\mathbf{B}$ are thus given by

$$B_{ik} = \sum_{j=1}^{m} p_i(x_j) p_k(x_j) = m\, \delta_{ik}. \tag{7.3}$$

However, if the polynomial basis is not orthonormal, for instance due to the use of an orthogonal basis or another Monte Carlo sample set, the Vandermonde matrix does not have orthogonal columns and may be ill-conditioned [27]. The following step aims to remedy this problem by performing a QR factorization of $\mathbf{A}$ and use the reduced orthogonal matrix $\mathbf{Q}$ instead of $\mathbf{A}$ in (3.36) for the extraction of the quadrature:

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} \mathbf{Q} & \mathbf{Z} \end{bmatrix} \begin{bmatrix} \mathbf{R} \\ 0 \end{bmatrix} = \mathbf{QR}, \\ \mathbf{A} &\leftarrow \mathbf{Q}. \end{aligned} \tag{7.4}$$

This step amounts to a change of basis from the original polynomials basis to the orthonormal basis with respect to the discrete inner product

$$\langle p_i, p_k \rangle = \sum_{j=1}^{m} p_i(x_j) p_k(x_j). \tag{7.5}$$

Meaning that $\mathbf{Q}$ has orthogonal column vectors and thus displays the following proprieties

$$\begin{aligned} \mathbf{Q}^\top \mathbf{Q} &= \mathbb{I}, \\ \mathbf{Q}^{-1} &= \mathbf{Q}^\top. \end{aligned} \tag{7.6}$$

Note that $\mathbf{Z}$ is the null space of $\mathbf{A}^\top$ and can thus be used to find $\mathbf{z}$, however the full QR factorization becomes expensive when m is large as $[\mathbf{Q}\ \mathbf{Z}]$ is a $m \times m$ matrix. Instead, it is preferable to only compute $\mathbf{Q}$ by a thin QR factorization and find a nontrivial $\mathbf{z}$ by a Newton-Raphson algorithm. Indeed, as (3.37) is a linear system of equations, the Jacobian matrix is nothing but $\mathbf{Q}^\top$. Since this latter has orthogonal columns:

$$\begin{aligned} \mathbf{J} &= \mathbf{Q}^\top, \\ \mathbf{J}^{-1} &= \mathbf{Q}, \end{aligned} \tag{7.7}$$

where $\mathbf{J}^{-1}$ denotes the Moore–Penrose inverse of the Jacobian. The vector $\mathbf{z}$ is thus computed iteratively at each iteration of the quadrature rule extraction algorithm by

$$\begin{aligned} \mathbf{F}^k &= \mathbf{Q}^\top \mathbf{z}^k, \\ \bar{\mathbf{z}}^{k+1} &= \left(\mathbf{z}^k - \mathbf{J}^{-1}\mathbf{F}^k\right), \\ \mathbf{z}^{k+1} &= \bar{\mathbf{z}}^{k+1}/|\bar{\mathbf{z}}^{k+1}| \end{aligned} \tag{7.8}$$

until the desired accuracy is reached. Note that one must ensure the algorithm to converge to the nontrivial solution by normalizing $\mathbf{z}$ at each Newton-Raphson iteration. Moreover, the relation (7.7) is only valid for the first iteration of the quadrature rule extraction. Indeed, as we remove a row of $\mathbf{Q}$ at each iteration, this latter loses its orthogonality propriety after the first weight removal. In a further section, we present two methods for updating $\mathbf{J}^{-1}$ without explicitly recomputing the pseudo-inverse.

7.2 The Thin QR Factorization

Numerous algorithms such as Givens rotations or Householder reflections already exist for computing a QR factorization of a matrix. However, the full QR factorization is expensive as we assume the number of Monte Carlo samples used to compute the quadrature rule is much larger than the number of polynomials in the basis. Thereby, it is necessary to compute the reduced matrix $\mathbf{Q}$ of the thin QR factorization in (7.4) without actually computing the full matrix $[\mathbf{Q}\ \mathbf{Z}]$. We propose the use of the QR-Cholesky method [49] for computing the thin QR factorization of $\mathbf{A}$ as this latter allows the computation of $\mathbf{Q}$ using only $m \times n$ and $n \times n$ sized matrices. One first compute

$$\begin{aligned} \mathbf{B} &= \mathbf{A}^\top \mathbf{A}, \\ \mathbf{L}^\top \mathbf{L} &= \mathbf{B}, \end{aligned} \tag{7.9}$$

where $\mathbf{B}$ is Hermitian positive-definite by definition and $\mathbf{L}$ is the lower triangular matrix obtained by a Cholesky factorization of $\mathbf{B}$. Practically, the elements of the factorized matrix $\mathbf{L}$ can be computed [50] as follows:

$$\begin{aligned} L_{11} &= \sqrt{B_{11}}, \qquad L_{j1} = \frac{B_{j1}}{L_{11}} && \text{for } j \in [2, n], \\ L_{jj} &= \sqrt{B_{jj} - \sum_{i=1}^{j-1} L_{ji}^2} && \text{for } j \in [2, n], \\ L_{ij} &= \left(B_{ij} - \sum_{k=1}^{j-1} L_{jk} L_{ik}\right)/L_{jj} && \text{if } j \neq n, \\ & && \text{for } i \in [i+1, n]. \end{aligned} \tag{7.10}$$

Finally, the reduced $m \times n$ matrix $\mathbf{Q}$ and the corresponding right $n \times n$ matrix $\mathbf{R}$ are obtained as follows. It is important to note that $\mathbf{R}$ can easily be inverted by exploiting the fact this latter is upper triangular.

$$\begin{aligned} \mathbf{R} &= \mathbf{L}^\top, \\ \mathbf{Q} &= \mathbf{A}\mathbf{R}^{-1}. \end{aligned} \tag{7.11}$$

7.3 The Givens Rotation

In this section, we present a reminder about the so-called Givens rotation, this technique can for instance be employed for computing the full QR factorization of a matrix, but also for updating the full or thin QR factorization when a particular change in the original matrix occurs, at a much lower computational cost than recomputing the QR factorization. A Givens rotation rotates a plane about two coordinates axes and can thus be used to cancel out elements in a matrix. This latter is represented by a matrix $\mathbf{G}(i, j)$ whose elements are given by

$$\begin{aligned} G_{ii} &= G_{jj} = \cos\theta, \\ G_{ji} &= -G_{ij} = \sin\theta, \\ G_{rr} &= 1 \quad \text{for } r \neq i, j, \end{aligned} \tag{7.12}$$

where θ is the rotation angle. The left matrix multiplication $\mathbf{G}(i, j)\mathbf{A}$ only effects the i-th and j-th rows in $\mathbf{A}$. Similarly, the right transpose multiplication $\mathbf{A}\mathbf{G}(i, j)^\top$ only affects the i-th and j-th columns in $\mathbf{A}$. For instance, if one wants to cancel-out the element A_{jk} in the original matrix by rotating the column vector

$$\mathbf{v} = \begin{bmatrix} A_{ik} & A_{jk} \end{bmatrix}^\top \tag{7.13}$$

along the horizontal axis, one can find the values of $\cos\theta$ and $\sin\theta$ by computing the linear map $\mathbf{H}(i, j)\, v$ that zeroes the second element of $\mathbf{v}$. One thus needs to solve

$$\mathbf{H}(i, j)\, \mathbf{v} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}, \tag{7.14}$$

where $r = |\mathbf{v}|$ is the l^2-norm of $\mathbf{v}$. An explicit calculation of the rotation angle is not necessary as one can directly solve the previous problem for $\cos\theta$ and $\sin\theta$, leading to

$$\begin{aligned} \cos\theta &= v_1/\sqrt{v_1^2 + v_2^2}, \\ \sin\theta &= -v_2/\sqrt{v_1^2 + v_2^2}. \end{aligned} \tag{7.15}$$

Each Givens rotation can be used to cancel out an element in the subdiagonal part of $\mathbf{A}$ and successive Given rotations can be used to compute the upper triangular matrix of a full QR factorization [51]. As the Givens matrices are orthogonal by definition, their concatenation form the orthogonal matrix. Generally, the Givens rotations are not actually performed by building a whole $\mathbf{G}(i, j)$ matrix and doing a matrix product. Indeed, as the product $\mathbf{G}(i, j)A$ will only affect the i-th and j-th rows of $\mathbf{A}$, one can use the smaller matrix $\mathbf{H}(i, j)$ and replace the previous matrix product by

$$\mathbf{H}(i, j)\, \mathrm{row}_{ij}(\mathbf{A}). \tag{7.16}$$

7.4 Rank Downgrade

One way to update $\mathbf{J}^{-1}$ for the next iteration of the quadrature rule extraction algorithm is the computation of the QR factorization related to the new matrix $\mathbf{A}'$ in order to recover the orthogonality propriety, then use (7.7) to obtain the Jacobian and its pseudo inverse. However, it is clear that recomputing the QR factorization from scratch at each iteration of the process for extracting the embedded quadrature rule may become very expensive when m is large.

An alternative is the so-called rank downgrade of the thin QR factorization [52], this method allows one to update the factorization matrices related to $\mathbf{A}$ for the ones related to a new matrix $\mathbf{A}'$ where a column or a row has been removed or added. In the present case, the j-th column of $\mathbf{A}^\top$ is deleted at each iteration, we thus would like to update the decomposition for $\mathbf{A}$ having its j-th row deleted. This update can be achieved first by computing a permutation matrix $\mathbf{P}$ moving the row to be deleted at the bottom of the original matrix, one can write

$$\mathbf{PA} = \mathbf{PQR} = \begin{bmatrix} \mathbf{A}' \\ \mathbf{a}^\top \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{Q}} \\ \mathbf{q}^\top \end{bmatrix} \mathbf{R}, \tag{7.17}$$

where $\mathbf{a}$ is the row be deleted and $\mathbf{q}$ is the last row of the permuted orthogonal matrix. As adding a row of zeros at the bottom of $\mathbf{R}$ and an arbitrary column vector $\mathbf{v}$ of size n on the right side of $\mathbf{Q}$ does not change the output of the matrix product, one can write

$$\begin{bmatrix} \mathbf{A}' \\ \mathbf{a}^\top \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{Q}} & 0 \\ \mathbf{q}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} \\ 0 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{Q} & \mathbf{v} \end{bmatrix} \begin{bmatrix} \mathbf{R} \\ 0 \end{bmatrix}. \tag{7.18}$$

In addition, it has been shown in [53] that performing a Gram-Schmidt process with reorthogonalization allows the decomposition of the new matrix $\mathbf{Q}$ augmented by $\mathbf{v}$ into the following matrix product:

$$\begin{aligned} &\begin{bmatrix} \mathbf{Q} & \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{Q} & \mathbf{g} \end{bmatrix} \begin{bmatrix} \mathbb{I} & \mathbf{r} \\ 0 & \rho \end{bmatrix}, \\ &\mathbf{r} = \mathbf{Q}^\top \mathbf{v}, \qquad \rho = |\mathbf{v}'|, \\ &\mathbf{g} = \mathbf{v}'/\rho, \qquad \mathbf{v}' = \mathbf{v} - \mathbf{Q}\mathbf{r}. \end{aligned} \tag{7.19}$$

The vector $\mathbf{g}$ of size n is then decomposed into a subvector $\bar{\mathbf{g}}$ of size $n-1$ augmented by scalar σ, the relation (7.19) can thus be rewritten as

$$\begin{bmatrix} \bar{\mathbf{Q}} & 0 \\ \mathbf{q}^\top & 1 \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{Q}} & \bar{\mathbf{g}} \\ \mathbf{q}^\top & \sigma \end{bmatrix} \begin{bmatrix} \mathbb{I} & \mathbf{r} \\ 0 & \rho \end{bmatrix}. \tag{7.20}$$

Moreover, injecting the decomposition of $\mathbf{Q}$ and $\mathbf{v}$ into the equations (7.19) allows the computation of the two vectors

$$\begin{aligned}\mathbf{r} &= \begin{bmatrix}\bar{\mathbf{Q}}^\top & \mathbf{q}\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = \mathbf{q},\\ \mathbf{v}' &= \begin{bmatrix}\bar{\mathbf{g}}\\ \sigma\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix} - \begin{bmatrix}\bar{\mathbf{Q}}\\ \mathbf{q}^\top\end{bmatrix}\mathbf{r} = \begin{bmatrix}-\bar{\mathbf{Q}}\,\mathbf{q}\\ 1-\mathbf{q}^\top\mathbf{q}\end{bmatrix}.\end{aligned} \tag{7.21}$$

The value of σ can finally be deduced by injecting the value of $\mathbf{v}'$ obtained above into the equation of ρ presented in (7.19) and taking its square, leading to

$$\begin{aligned}\rho^2 &= |\bar{\mathbf{Q}}\mathbf{q}|^2 + (1-\mathbf{q}^\top\mathbf{q})^2\\ &= \mathbf{q}(\mathbb{I}-\mathbf{q}^\top\mathbf{q})\mathbf{q} + (1-\mathbf{q}^\top\mathbf{q})^2\\ &= 1-\mathbf{q}^\top\mathbf{q} = \sigma\rho.\end{aligned} \tag{7.22}$$

If ρ is nonzero, one has $\rho^2 = \rho\sigma$ so $\rho = \sigma$, otherwise one has $\mathbf{q}^\top\mathbf{q} = 1$ and since $\mathbf{q}^\top\mathbf{q} + \sigma^2 \leq 1$ it can also be deduced that $\rho = \sigma$. The relation (7.20) becomes

$$\begin{bmatrix}\bar{\mathbf{Q}} & 0\\ \mathbf{q}^\top & 1\end{bmatrix} = \begin{bmatrix}\bar{\mathbf{Q}} & \bar{\mathbf{g}}\\ \mathbf{q}^\top & \rho\end{bmatrix}\begin{bmatrix}\mathbb{I} & \mathbf{r}\\ 0 & \rho\end{bmatrix}. \tag{7.23}$$

Finally, injecting this matrix decomposition into (7.18) gives

$$\begin{bmatrix}\mathbf{A}'\\ \mathbf{a}\end{bmatrix} = \begin{bmatrix}\bar{\mathbf{Q}} & \bar{\mathbf{g}}\\ \mathbf{q}^\top & \rho\end{bmatrix}\begin{bmatrix}\mathbb{I} & \mathbf{r}\\ 0 & \rho\end{bmatrix}\begin{bmatrix}\mathbf{R}\\ 0\end{bmatrix} = \begin{bmatrix}\bar{\mathbf{Q}} & \bar{\mathbf{g}}\\ \mathbf{q}^\top & \rho\end{bmatrix}\begin{bmatrix}\mathbf{R}\\ 0\end{bmatrix}. \tag{7.24}$$

It is now possible to cancel-out the n first elements of the bottom row in the right-hand side matrix by performing n successive Givens rotations such that

$$\begin{aligned}\mathbf{H}^\top &= \mathbf{G}(n,n+1)^\top \dots \mathbf{G}(1,n+1)^\top,\\ \begin{bmatrix}\mathbf{q}^\top & \rho\end{bmatrix}\mathbf{H}^\top &= \begin{bmatrix}0 & \alpha\end{bmatrix}.\end{aligned} \tag{7.25}$$

Note that the resulting vector is normalized, meaning that $|\alpha| = 1$ since the Givens rotations do not modify the length. Thereby, applying these rotations to the right and to the left of the decomposition obtained in (7.24) leads to

$$\begin{aligned}\begin{bmatrix}\bar{\mathbf{Q}} & \bar{\mathbf{g}}\\ \mathbf{q}^\top & \rho\end{bmatrix}\mathbf{H}^\top &= \begin{bmatrix}\mathbf{Q}' & 0\\ 0 & \alpha\end{bmatrix},\\ \mathbf{H}\begin{bmatrix}\mathbf{R}\\ 0\end{bmatrix} &= \begin{bmatrix}\mathbf{R}'\\ \beta\end{bmatrix},\end{aligned} \tag{7.26}$$

where the right hand side of the second equation is upper Hessenberg due to the Givens rotations, meaning that $\mathbf{R}'$ is upper triangular. Note that the column of zero above α is forced by the orthogonality of the matrix. The thin QR factorization of the matrix $\mathbf{A}'$ is finally obtained by

$$
\begin{aligned}
&\begin{bmatrix}\mathbf{A}'\\ \mathbf{a}\end{bmatrix} = \begin{bmatrix}\bar{\mathbf{Q}} & \bar{\mathbf{q}}\\ \mathbf{q}^\top & \rho\end{bmatrix}\mathbf{H}^\top\mathbf{H}\begin{bmatrix}\mathbf{R}\\ 0\end{bmatrix},\\
&\begin{bmatrix}\mathbf{A}'\\ a\end{bmatrix} = \begin{bmatrix}\mathbf{Q}' & 0\\ 0 & \alpha\end{bmatrix}\begin{bmatrix}\mathbf{R}'\\ \beta\end{bmatrix} = \begin{bmatrix}\mathbf{Q}'\mathbf{R}'\\ \pm\beta\end{bmatrix},\\
&\Leftrightarrow \quad \mathbf{A}' = \mathbf{Q}'\mathbf{R}'.
\end{aligned}
\tag{7.27}
$$

A summary of the procedure of extracting an embedded quadrature rule with positive weights from a Monte Carlo sample set is presented in Algorithm 5.

Algorithm 5: Embedded quadrature rule

Compute $\mathbf{A} : A_{ij} = p_j(x_i)$
Compute the thin QR factorization of $\mathbf{A}$
Set $\mathbf{w} : w_j = 1/m \ \forall j$

while len($\mathbf{w}$) $> n$ do
    Set $\mathbf{J}^{-1} = \mathbf{Q}$
    Find nontrivial $\mathbf{z} : \mathbf{Q}^\top\mathbf{z} = 0$ with Newton-Raphson
    Set $\alpha = w_j/z_j : j = \text{argmin}_j\,|w_j/z_j|$
    Replace $\mathbf{w} \leftarrow \mathbf{w} - \alpha\mathbf{z}$
    Remove the j-th row of $\mathbf{A}$ and element of $\{\mathbf{w}, \mathbf{x}\}$
    Update the thin QR factorization of $\mathbf{A}$
end

Return $\{\mathbf{w}, \mathbf{x}\}$

7.5 Update the Pseudo Inverse

A second possibility to update $\mathbf{J}^{-1}$ for the next iteration of the quadrature rule extraction is to directly update the inverted matrix instead of updating the QR factorization and taking its transpose. Indeed, if the polynomials are orthonormal, one can directly obtain the Moore–Penrose inverse of $\mathbf{A}^\top$ at the first iteration by

$$
\begin{aligned}
&\mathbf{J} = \mathbf{A}^\top,\\
&\mathbf{J}^{-1} = \mathbf{A}/m
\end{aligned}
\tag{7.28}
$$

and then deduce the Moore–Penrose inverse of the Jacobian matrix $\mathbf{J}' = \mathbf{J}$ whose the j-th column has been removed from the knowledge of $\mathbf{J}$ and $\mathbf{J}^{-1}$ obtained at a previous iteration [54, 55]. One first compute

$$
\begin{aligned}
&\mathbf{v}_1 = \mathrm{col}_j(\mathbf{J}),\\
&\mathbf{v}_2^\top = \mathrm{row}_j(\mathbf{J}^{-1}),\\
&\alpha = \mathbf{v}_1^\top \mathbf{v}_2.
\end{aligned}
\tag{7.29}
$$

One then compute a permutation matrix $\mathbf{P}$ allowing to extract the submatrix $\bar{\mathbf{J}}^{-1}$ such that the following relation holds:

$$
\mathbf{P}\mathbf{J}^{-1} = \begin{bmatrix} J_j^{-1} \\ \bar{\mathbf{J}}^{-1} \end{bmatrix}, \tag{7.30}
$$

where $\mathbf{J}_j^{-1}$ denotes the j-th row of $\mathbf{J}^{-1}$. Depending on the scalar product α, the Moore–Penrose inverse of the new Jacobian matrix is finally computed by

$$
\begin{aligned}
&\mathbf{V} = \mathbb{I} + \frac{\mathbf{v}_1 \mathbf{v}_2^\top}{1-\alpha} \quad \text{if } \alpha < 1,\\
&\mathbf{V} = \mathbb{I} - \frac{\mathbf{v}_2 \mathbf{v}_2^\top}{\mathbf{v}_2^\top \mathbf{v}_2} \quad \text{if } \alpha \geq 1,\\
\Leftrightarrow \quad &\mathbf{J}'^{-1} = \bar{\mathbf{J}}^{-1}\mathbf{V}.
\end{aligned}
\tag{7.31}
$$

In both cases, one restrict to the exclusive use of at most $m \times n$ sized matrices. Note that both approaches (i.e. updating the QR factorization or the pseudoinverse) lead to similar accuracy, and the choice of the method may rely on the available pre-built linear algebra libraries or their implementation. A summary of this alternative procedure is presented in Algorithm 6.

Algorithm 6: Embedded quadrature rule

Compute $\mathbf{A} : A_{ij} = p_j(x_i)$
Set $\mathbf{w} : w_j = 1/m \ \forall j$
Set $\mathbf{J}^{-1} = \mathbf{A}/m$

while $\mathrm{len}(\mathbf{w}) > n$ do
    Find nontrivial $\mathbf{z} : \mathbf{A}^\top \mathbf{z} = 0$ with Newton-Raphson
    Set $\alpha = w_j/z_j : j = \mathrm{argmin}_j \, |w_j/z_j|$
    Replace $\mathbf{w} \leftarrow \mathbf{w} - \alpha \mathbf{z}$
    Remove the j-th row of $\mathbf{A}$ and element of $\{\mathbf{w}, \mathbf{x}\}$
    Update $\mathbf{J}^{-1} = \mathbf{A}^{\top -1}$
end

Return $\{\mathbf{w}, \mathbf{x}\}$

References

[1] *Space Exploration Technologies Corp, SpaceX*. Online: https://www.spacex.com/. 2012.

[2] J.Lachaud et al. "A Generic Local Thermal Equilibrium Model for Porous Reactive Materials Submitted to High Temperatures". In: *International Journal of Heat and Mass Transfer* (2017).

[3] Francisco Torres-Herrador et al. "A High Heating Rate Pyrolysis Model for the Phenolic Impregnated Carbon Ablator Based on Mass Spectroscopy Experiments". In: *Journal of Analytical and Applied Pyrolysis* (2019).

[4] Coheur Joffrey et al. "Bayesian Parameter Inference for PICA Devolatilization Pyrolysis at High Heating Rates". 10th VKI PhD Symposium. 2019.

[5] Francisco Torres-Herrador et al. "Competitive Kinetic Model for the Pyrolysis of the Phenolic Impregnated Carbon Ablator". In: *Aerospace Science and Technology* (2019).

[6] H. Arnst and J.P. Ponthot. "An Overview of Nonintrusive Characterization, Propagation, and Sensitivity Analysis of Uncertainties in Computational Mechanics". In: *International Journal for Uncertainty Quantification* (2014).

[7] M.Riviera, J.Lachaudd, and P.M.Congedoc. "Ablative Thermal Protection System Under Uncertainties Including Pyrolysis Gas Composition". In: *Aerospace Science and Technology* (2018).

[8] Xiu Dongbin and Karniadakis George Em. "The Wiener-Askey Polynomial Chaos for Stochastic Differential Equations". In: *SIAM Journal on Scientific Computing* (2002).

[9] S.Oladyshkin and W.Nowak. "Data-Driven Uncertainty Quantification Using the Arbitrary Polynomial Chaos Expansion". In: *Reliability Engineering System Safety* (2012).

[10] Golub et al. *Calculation of Gauss Quadrature Rules*. Tech. rep. Stanford, CA, USA, 1967.

[11] Blatman et al. "Adaptive Sparse Polynomial Chaos Expansion Based on Least Angle Regression". In: *Journal of Computational Physics* (2011).

[12] Abramowitz and Milton. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*. New York, NY, USA: Dover Publications, 1974.

[13] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 2012.

[14] J.D. Jakeman, A. Narayan, and F. Franzelin. *Polynomial Chaos Expansions for Dependent Random Variables*. Presentation: Sandia National Laboratories. 2019.

[15] Akira Imakura and Yusaku Yamamoto. "Efficient Implementations of the Modified Gram-Schmidt Orthogonalization With a Non-standard Inner Product". In: *Japan Journal of Industrial and Applied Mathematics* (2017).

[16] Peter J. Olver. “Orthogonal Bases and the QR Algorithm”. Lecture notes at University of Minnesota. 2010.

[17] Mai et al. “Hierarchical Adaptive Polynomial Chaos Expansions”. In: *Thematic Conference on Uncertainty Quantification in Computational Sciences and Engineering* (2015).

[18] Haiyan Cheng and Adrian Sandu. “Collocation Least-squares Polynomial Chaos Method”. In: Society for Computer Simulation International, 2010.

[19] Giray Öktena and Ahmet Göncüb. “Generating Low-discrepancy Sequences From the normal Distribution: Box-Muller or Inverse Transform”. In: *Mathematical and Computer Modelling* (2011).

[20] Sandamala Hettigoda. “Computation of Least Angle Regression Coefficient Profiles and LASSO Estimates.” In: *The University of Louisville's Institutional Repository* (2016).

[21] Sudret Bruno and Caniou Yann. *Analysis of Covariance Using Polynomial Chaos Expansions*. International Conference on Structural Safety and Reliability. 2013.

[22] Sudret Bruno and Chu Mai. “Computing Derivative-Based Global Sensitivity Measures Using Polynomial Chaos Expansions”. In: *Reliability Engineering and System Safety* (2014).

[23] Y. Caniou and B. Sudret. *Covariance-BasedSensitivity Indices Based on Polynomial Chaos Functional Decomposition*. Presentation: Eidgenössische Technische Hochschule Zürich. 2013.

[24] Art B. Owen. “Sobol Indices and Shapley Value”. In: *Society for Industrial and Applied Mathematics* (2013).

[25] Bertrand Iooss and Clementine Prieur. “Shapley Effects for Sensitivity Analysis with Correlated Inputs, Comparisons with Sobol Indices, Numerical Estimation and Applications”. In: *International Journal for Uncertainty Quantification, Begell House Publishers* (2019).

[26] Stefan Weinzierl. *Introduction to Monte Carlo Methods*. Lecture notes: Research School Subatomic Physics, Amsterdam. 2000.

[27] Alvise Sommariva and Marco Vianello. “Computing Approximate Fekete Points by QR Factorizations of Vandermonde Matrices”. In: *Computers and Mathematics with Applications* (2009).

[28] L. Bos et al. “Computing Multivariate Fekete and Leja Points by Numerical Linear Algebra”. In: *SIAM J. Numerical Analysis* (2010).

[29] Bengt Ringnér. *The law of the unconscious statistician*. Centre for Mathematical Sciences, Lund University. 2009.

[30] M. Roberts. *The Unreasonable Effectiveness of Quasirandom Sequences*. Online: http://extremelearning.com.au. 2019.

[31] Corrado Chisari. *The Sobol Quasirandom Sequence*. Online: https://people.sc.fsu.edu/ jburkardt/py_src/sobol/sobol.html. 2020.

[32] Jérôme Barraquand. “Monte Carlo Integration, Quadratic Resampling, and Asset Pricing”. In: *Mathematics and Computers in Simulation* (1995).

[33] Maria Cameron. *Gaussian Quadrature*. Lecture notes: University of Maryland, Department of Mathematics. 2015.

[34] Len Bos et al. "Weakly Admissible Meshes and Discrete Extremal Sets". In: *Numerical Mathematics, Theory, Methods and Applications* (2011).

[35] M. Arnst et al. "Measure Transformation and Efficient Quadrature in Reduced-Dimensional Stochastic Modeling of Coupled Problems". In: *International Journal for Numerical methods in Engineering* (2012).

[36] Laurent van den Bos et al. *Generating Nested Quadrature Rules with Positive Weights Based on Arbitrary Sample Sets*. 2020.

[37] Jonathan Feinberga and Hans Petter Langtangena. "Chaospy, an Open Source Tool for Designing Methods of Uncertainty Quantification". In: *Journal of Computational Science* (2015).

[38] Mickael Gastineau. "Storage of Multivariate Polynomials". Workshop at Observatoire de Paris. 2007.

[39] Takayuki Ishigami and Toshiteru Homma. "An Importance Quantification Technique in Uncertainty Analysis for Computer Models". In: *Proceedings First International Symposium on Uncertainty Modeling and Analysis* (1990).

[40] Jean Lachaud et al. *Ablation Workshop Test Case*. AF-SNL-NASA Ablation Workshop, Albuquerque, New Mexico. 2011.

[41] Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, Philadelphia, 2005.

[42] Ilker Yildirim. *Bayesian Inference, Metropolis-Hastings Sampling*. Lecture notes: Department of Brain and Cognitive Sciences, University of Rochester. 2012.

[43] Christian P. Robert. *The Metropolis-Hastings Algorithm*. 2015.

[44] Agnan Kessy, Alex Lewin, and Korbinian Strimmer. "Optimal Whitening and Decorrelation". In: *The American Statistician* (2018).

[45] Roland Schöbi, Bruno Sudret, and Joe Wiart. "Polynomial Chaos Based Kriging". In: *International Journal of Uncertainty Quantification* (2015).

[46] E. Torre et al. "Data-Driven Polynomial Chaos Expansion for Machine Learning Regression". In: *Journal of Computational Physics* (2019).

[47] Maysara Ghaith and Zhong Li. "Propagation of Parameter Uncertainty in SWAT, A Probabilistic Forecasting Method Based on Polynomial Chaos Expansion and Machine Learning". In: *Journal of Hydrology* (2020).

[48] *Cython C-extension for Python*. Online: https://cython.org/. 2007.

[49] Takeshi Terao, Katsuhisa Ozaki, and Takeshi Ogita. "LUCholesky QR algorithms for thin QR decomposition". In: *Parallel Computing* (2019).

[50] Michael Parker. *Digital Signal Processing 101, Second Edition: Everything You Need to Know to Get Started*. Newnes, 2017.

[51] Che-Rung Lee. *An Example of QR Decomposition*. Department of Computer Science , National Tsing Hua University. 2008.

[52] Hammarling Sven and Lucas Craig. *Updating the QR factorization and the least squares problem*. Manchester Institute for Mathematical Sciences School of Mathematics. 2008.

[53] J. Daniel et al. "Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization". In: *Mathematics of Computation* (1976).

[54] William W.Hager. "Updating the Inverse of a Matrix". In: *SIAM Review* (1989).

[55] Yi Cao. *Pseudo-Inverse Update*. MATLAB Central File Exchange. 2009.