A Polynomial Chaos Approach to Robust Multiobjective Optimization

Silvia Poles[1], Alberto Lovison[2]

[1] EnginSoft S.p.A., Optimization Consulting
Via Giambellino, 7 35129 Padova, Italy
s.poles@enginsoft.it
[2] University of Padua, Department of Pure and Applied Mathematics
Via Trieste, 67 35121 Padova, Italy
lovison@math.unipd.it

Abstract. Robust design optimization is a modeling methodology, combined with a suite of computational tools, which is aimed to solve problems where some kind of uncertainty occurs in the data or in the model. This paper explores robust optimization complexity in the multiobjective case, describing a new approach by means of Polynomial Chaos expansions (PCE). The aim of this paper is to demonstrate that the use of PCE may help and speed up the optimization process if compared to standard approaches such as Monte Carlo and Latin Hypercube sampling.

Keywords. Uncertainty Quantification, Multiobjective Robust Design, Monte Carlo, Latin Hypercube, Polynomial Chaos

1 Introduction

Robust design optimization has recently started to gain attention within the engineering and scientific communities since many real world optimization problems, in numerous disciplines and application areas, contain uncertainty.

This uncertainty may derive from errors in measuring, or from difficulties in sampling, or moreover can depend on events and effects in the future that are not completely known.

Often the design parameters may only be determined only up to some tolerance or, in many cases, they vary according to a probability distribution. Deterministic approaches to optimization do not consider the impact of such variations, and as a result, design solutions may be very sensitive to these variations. In this paper we focus on this source of uncertainty and propose an approach based on Polynomial Chaos expansions to quantify and control uncertainty during the optimization process.

2 A framework for robust multiobjective optimization

An ordinary multiobjective optimization (MO) problem:

$$\min_{x \in \mathbb{R}^n} \left(f_1(x), \ldots, f_k(x) \right), \tag{1}$$

Dagstuhl Seminar Proceedings 09041
Hybrid and Robust Approaches to Multiobjective Optimization
http://drops.dagstuhl.de/opus/volltexte/2009/2000

assumes that all the design parameters $x_1, \ldots, x_n$ are completely controllable, and that the functions $f_1, \ldots, f_k$ are deterministic, as in the case of computer experiments.

In real problems, $x_1, \ldots, x_n$ can be affected by uncertainty, this means that they vary over a certain range Θ following some probability distribution $\mathcal{D}_i$. In such cases, the problem parameters x_i should be substituted by random variables. For example we may have,

$$x_i \mapsto X_i(x_i) \sim \mathcal{D}_i,$$

where $\mathcal{D}_i$ could be a normal distribution $\mathcal{N}(x_i, \sigma_i)$ centered at x_i and with a fixed standard deviation σ_i. As a consequence each f_j becomes itself a random variable, defined as:

$$f_j(x_1, \ldots, x_n) \longrightarrow F_j(x_1, \ldots, x_n) := f_j(X_1(x_1), \ldots, X_n(x_n)), \qquad j = 1, \ldots, k$$

This process is illustrated in Figure 1(a), where it is shown how the randomness in an input variable x propagates through a certain f. If the deterministic parameter x becomes the floating center of a probability distribution $X(x)$ as above, the dependent variable $y = f(x)$ becomes an indexed family of random variables $F(x) := f(X(x))$, as depicted in Figure 1(b).

In this context, simply rewriting (1) substituting variables with random variables,

$$\min_{x \in \mathbb{R}^n} (F_1(x), \ldots, F_k(x))$$

does not make sense. As reported in [4], we should solve our multiobjective optimization problem redefining our task as minimizing the means of the stochastic objectives

$$\min_{x \in \mathbb{R}^n} \left(\mu_{F_1(x)}, \ldots, \mu_{F_k(x)}\right)$$

or with more sophisticated approaches as minimizing means plus κ standard deviations:

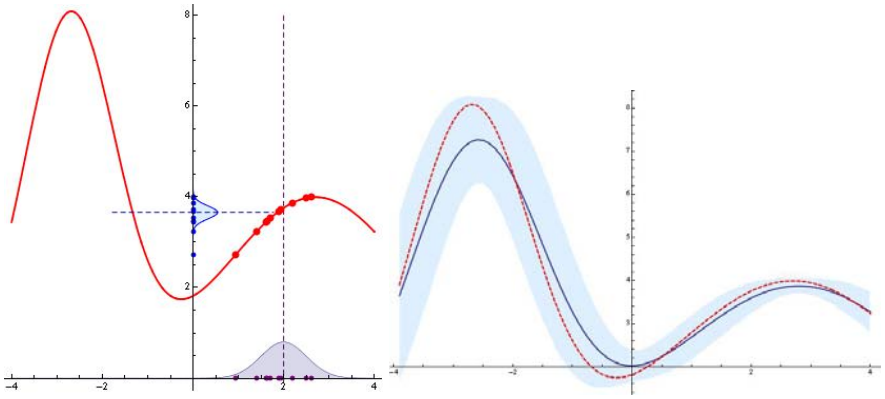
$$\min_{x \in \mathbb{R}^n} \left(\mu_{F_1(x)} + \kappa\sigma_{F_1(x)}, \ldots, \mu_{F_k(x)} + \kappa\sigma_{F_k(x)}\right)$$

This means that first of all we have to solve a problem of uncertainty quantification, estimating the statistical moments of all the responses $F_1, \ldots, F_k$.

3 Quantifying uncertainty

Quantifying uncertainty means to describe as precisely as possible the statistical features of the random variables F_j. It would be desirable to determine the full probability distribution function (PDF) of F_j, however, in most of the real world applications, it is sufficient to estimate mean and the standard deviation.

From now on, we will indicate simply F instead of F_j keeping in mind that the uncertainty quantification has to be performed for every objective function F_j in the problem at hand.



(a) (b)

Fig. 1. (a) Graph of the function $f(x)$ in (4). In abscissa a Monte Carlo sample from the distribution X, in ordinate the corresponding sample of $f(X)$. (b) The effect on the objective function when uncertainty is introduced in the input variable.

3.1 Analytical approach

The objective f can be given by a mathematical expression, like a polynomial, or could be implicitly defined by an equation, or a partial differential equation.[1]

When an analytical expression is given, the statistical moments can be determined exactly by means of symbolic integration:

$$\mu_F = E\left[f(X)\right] = \int f(x)w(x)dx$$

$$\sigma_F = \sqrt{E\left[(f(x) - m_F)^2\right]} = \sqrt{\int f(x)^2 w(x)dx - \mu_F^2}.$$

where $w(x)$ is the probability law of the random variable X. Integration may be difficult (or even impossible) for complex functions, this means that solving uncertainty quantification analytically is unworkable. Moreover, in most of engineering design optimization problems the full analytical formulations are unavailable and other approaches are needed.

3.2 Monte Carlo Sampling

The most classical methodology consists in drawing a random sample $\left\{x^{(1)}, \ldots, x^{(N)}\right\}$ from the joint distribution $\mathcal{D}_1 \otimes \cdots \otimes \mathcal{D}_n$, i.e., drawing N random numbers

[1] A typical situation is when the uncertain variable y is defined as the solution of a stochastic partial differential equation, $L(x)y = g(x)$, where L is a partial differential operator and x is a (set of) stochastic variable(s).

$\left\{x_i^{(1)}, \ldots, x_i^{(N)}\right\}$ from every distribution $\mathcal{D}_i$ and then collecting the vectors

$$\begin{aligned} x^{(1)} &:= \left(x_1^{(1)}, \ldots, x_n^{(1)}\right)^T, \\ x^{(2)} &:= \left(x_1^{(2)}, \ldots, x_n^{(2)}\right)^T, \\ \vdots \quad & \quad \vdots \\ x^{(N)} &:= \left(x_1^{(N)}, \ldots, x_n^{(N)}\right)^T. \end{aligned}$$

A sample $s_Y := \left\{y^{(1)}, \ldots, y^{(N)}\right\}$ of the uncertain dependent variable Y is computed through application of f to each one of the $x^{(j)}$:

$$y^{(j)} := f\left(x_1^{(j)}, \ldots, x_n^{(j)}\right), \qquad j = 1, \ldots, N. \tag{2}$$

Finally, the mean and standard deviation of Y are estimated by the mean and the corrected standard deviation of the sample s_Y:

$$\begin{aligned} \mu_Y \simeq \bar{\mu}_Y &:= \langle s_Y \rangle = \frac{1}{N} \sum_{j=1}^{N} y^{(j)}, \\ \sigma_Y \simeq \bar{\sigma}_Y &:= \sqrt{\frac{1}{N-1} \sum_{j=1}^{N} \left(y^{(j)} - \bar{\mu}_Y\right)^2}. \end{aligned}$$

It is well known that statistics obtained via Monte Carlo are reliable, but are very poor in accuracy. It is possible to prove that the statistical moments, and as a result the mean and the standard deviation, of a random sample converge to the exact moments of the full distribution of Y as $\frac{1}{\sqrt{N}}$, i.e.,

$$|\bar{\mu}_Y(N) - \mu_Y| \leqslant C \frac{1}{\sqrt{N}}, \qquad N = \text{sample size}. \tag{3}$$

This means that to halve the estimation error it is necessary a four times larger sample, and to reduce the error of an order of magnitude (1/10) it is necessary to take a sample 100 times larger.

Example 1. Let $Y = f(X)$, where

$$f(x) := 20 \left(\frac{e^{-\frac{(x+2.7)^2}{2}}}{\sqrt{2\pi}} + \frac{e^{-\frac{(x-2.7)^2}{2(2^2)}}}{\sqrt{2\pi}2} \right), \tag{4}$$

$$X \sim \mathcal{N}(2, 0.8), \qquad \text{i.e.,} \quad p(x) = \frac{e^{-\frac{(x-2)^2}{2(0.8^2)}}}{\sqrt{2\pi}0.8}. \tag{5}$$

Symbolic computation lead to exact mean and standard deviation for Y:

$$\mu_Y = E\left[f(X)\right] = \int f(x)p(x)dx \simeq 3.5209849377883446,$$

$$\sigma_Y = \sqrt{E\left[(f(x)-\mu_Y)^2\right]} = \sqrt{\int f(x)^2 p(x)dx - \mu_Y^2} \simeq 0.5073131752261475.$$

Performing 20 replications of Monte Carlo with increasing sample sizes $N = 8$, 16,..., 8192, 16384 brings the results reported in Table 1, reported also in log–log scales in Figure 3. Summarizing the simulation results, we can notice that in order to obtain an error of 1% we need a sample of approximately 256 points for the mean and 8192 points for the standard deviation, while to reach the 0.1% accuracy we need 16384 points for the mean. The 0.1% accuracy is not reached within our simulation. Approximately, we should need an 800000 large sample.

nData	Average Error Mean	Average Error StDev
8	0.13523321821576068	0.1610272359435758
16	0.11711965574510605	0.0864659032341357
32	0.059222020366550845	0.04713102966918838
64	0.057784528160956625	0.05253445820159186
128	0.03838430554585017	0.034136016860713006
256	0.022344036432755	0.020111546424191107
512	0.02061047535304621	0.015098327730704386
1024	0.009641651869702561	0.010470922893152998
2048	0.010312701913743005	0.008504295088329538
4096	0.007855294629528543	0.006264945824679183
8192	0.003879754790574208	0.004073598991190089
16384	0.0029156602991482483	0.0034092125680575958

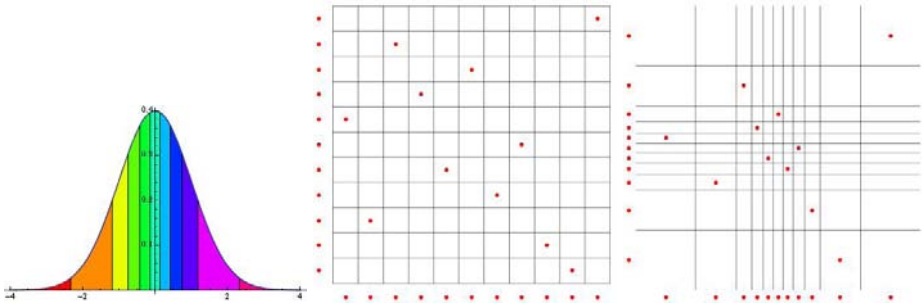
Table 1. Monte Carlo simulations replicated 20 times for the problem in 4

4 Latin Hypercube Sampling

If our purpose is to speed up the convergence of statistical moments of a random sample to the actual statistical moments of the probability distribution from which the sample is taken, there exists a smarter strategy: a Latin Hypercube Sampling (LHS) (see [5]).

Latin Hypercubes is a sampling strategy which has the following good marginal property: projecting the sample along each single variable produced a stratified sampling. More precisely, if $\left\{x^{(1)}, \ldots, x^{(N)}\right\}$ is a LHS, and every variable is divided in N strata with equal probability, every single stratum will be occupied by exactly one point. In figure 2(a) it is shown how to divide $(-\infty, +\infty)$ into equal

strata according to a standard normal distribution. A stratified sampling of N points will put a point in each slice. An LHS in two dimension of N points can be obtained dividing every variable into N equal slices and putting the points in such a way that every column would contain only one point and the same for every row, as shown in Figure 2(b-c). Doing so, the projections on each variable will result in a stratified sampling.



(a) (b) (c)

Fig. 2. Latin Hypercube Samplings. In panel (a) $\mathbb{R} = (-\infty, +\infty)$ is subdivided in slices with the same probability, according to a standard normal distribution. In panel (b) the two variables are uniformly distributed while in panel two (c) the two variables are normally distributed.

Latin hypercube sampling has been designed specifically to produce better accuracy than Monte Carlo in uncertainty quantification. In fact it could be proved that the error in estimating the statistical moments of the uncertain variable scales as $\frac{1}{N}$, where N is the sample size, i.e.,

$$|\bar{\mu}_Y(N) - \mu_Y| \leqslant C\frac{1}{N}, \qquad N = \text{sample size}. \tag{6}$$

This means that to halve the estimation error it suffices to double the sample size, while, in order to reduce the error of an order of magnitude (1/10) it suffices to take a sample 10 times larger.

Example 2. Let $Y = f(X)$, as in the example above. Performing 20 replications of Latin hypercube sampling with increasing sizes $N = 8, 16, \ldots, 8192, 16384$ brings the results reported in Table 2, reported also in log–log scales in Figure 3. Summarizing, we can notice that in order to obtain an error of 1% we need a sample of approximately 16 points for the mean and 128 points for the standard deviation, while to reach the 0.1% accuracy we need 64 and 2048 points, respectively.

nData	Average Error Mean	Average Error StDev
8	0.03838708340684853	0.0823640656227671
16	0.030567950211235484	0.040710534223220365
32	0.007571863928251643	0.01648128041162316
64	0.003418078142074843	0.006587126182655906
128	0.0024618618972161777	0.004848681021425105
256	0.0018873455437873998	0.003133046085056787
512	5.648087501843202E-4	0.0012508917015900789
1024	4.2176395159760905E-4	0.001042445154700511
2048	2.665184159121647E-4	4.888817764855613E-4
4096	1.1563639765082012E-4	2.1436136917031833E-4
8192	4.724986318238589E-5	7.123204436129126E-5
16384	3.0972434058185175E-5	4.675409217411719E-5

Table 2. Latin hypercube sampling replicated 20 times for the problem in 4

5 Polynomial Chaos

There exists also a methodology far more efficient than Monte Carlo and Latin hypercube samplings for estimating statistical moments of uncertain dependent variables. This methodology originates from the work of Norbert Wiener (see [7]) and is called Polynomial Chaos Expansion. This methodology consists essentially in expanding the uncertain variable in a suitable series and then determine analytically (and thus exactly) the statistical moments of the truncated expansion. The expansion itself is referred to as the "chaos", while the maximum degree of the expansion is called the "chaos order". As a result, it can be proved that the estimate of the statistical moments converge to true values at exponential rate, i.e., the error in the estimates scales as $\exp(-N)$, where N is the sample size.

Consider for the moment a one dimensional problem. Assume that X is distributed according a probability density function $w(x)$. Let us fix a family of polynomials

$$\mathcal{P} = \{p_0(x), p_1(x), \dots, p_k(x), \dots\}, \qquad \text{where } p_i(x) \text{ has degree } i,$$

and assume that $f(x)$ is a linear combination of a finite subset of this family,

$$f(x) := \sum_{i=0}^{k} \alpha_i p_i(x).$$

Formally, the m–th statistical moment of Y could be written as

$$\begin{aligned}\langle y^m \rangle = \int (f(x))^m w(x)dx = \int \left(\sum_{i=0}^{k} \alpha_i p_i(x)\right)^m w(x)dx = \\ = \int \sum \alpha_{i_1} \dots \alpha_{i_m} p_{i_1}(x) \dots p_{i_m}(x) w(x)dx = \\ = \sum \alpha_{i_1} \dots \alpha_{i_m} \int p_{i_1}(x) \dots p_{i_m}(x) w(x)dx. \quad (7)\end{aligned}$$

Computing symbolically the integral of any product of the polynomials $p_0(x), \ldots, p_k(x), \ldots$ it would be possible to compute any statistical moment of Y.

The key point to realize an effective procedure is to notice that for every probability density function $w(x)$ there exists a special family of *orthogonal polynomials* for which the w–scalar product

$$\langle p_i(x), p_j(x) \rangle_w := \int p_i(x) p_j(x) w(x) dx = 0, \qquad \text{whenever } i \neq j.$$

In particular, if $w(x)$ is Gaussian, $w(x) = \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$, the family of orthogonal polynomials is formed by the Hermite polynomials:

$$\begin{aligned}
He_0(x) &= 1, \\
He_1(x) &= x, \\
He_2(x) &= x^2 - 1, \\
He_3(x) &= x^3 - 3x, \\
He_4(x) &= x^4 - 6x^2 + 3, \\
He_5(x) &= x^5 - 10x^3 + 15x, \\
He_6(x) &= x^6 - 15x^4 + 45x^2 - 15, \\
He_7(x) &= x^7 - 21x^5 + 105x^3 - 105x, \\
&\vdots \quad \vdots
\end{aligned}$$

Their $w(x)$–norm is:

$$\|He_k\|_w^2 = \langle He_k(x), He_k(x) \rangle_w := \int He_k(x)^2 w(x) dx = k!.$$

Moreover, they satisfy the recursion formula, which helps in writing the infinite sequence,

$$He_{n+1}(x) = x He_n(x) - n He_{n-1}(x). \tag{8}$$

The special family of polynomials orthogonal with respect to the probability distribution $w(x)$ simplifies dramatically the formula for the statistical moments (7). Indeed, consider the mean of $f(X)$, i.e., the first statistical moment

$$\mu_Y = \langle y^1 \rangle = \int (f(x)) w(x) dx = \int \left(\sum_{i=0}^{k} \alpha_i p_i(x) \right) w(x) dx = \\ = \sum_{i=0}^{k} \alpha_i \int p_i(x) w(x) dx. \tag{9}$$

Being $p_0(x) = 1$, $\int p_i(x) w(x) dx \equiv 0$ for every $i > 0$, thus

$$\mu_Y = \alpha_0.$$

Computing the second statistical moment, we get, by the orthogonality of the $p_i(x)$,

$$\begin{aligned}\langle y^2 \rangle = \int (f(x))^2 w(x) dx = \int \left(\sum_{i=0}^{k} \alpha_i p_i(x) \right)^2 w(x) dx = \\ = \sum_{\substack{i_1=0,\dots,k \\ i_2=0,\dots,k}} \alpha_{i_1} \alpha_{i_2} \int p_{i_1}(x) p_{i_2}(x) w(x) dx = \\ = \sum_{i=0,\dots,k} \alpha_i^2 \int p_i(x)^2 w(x) dx = \sum_{i=0,\dots,k} \alpha_i^2 \left\| p_i(x) \right\|^2 . \quad (10)\end{aligned}$$

Therefore, the standard deviation is computed straightforwardly:

$$\sigma_Y = \sqrt{\sigma_Y^2} = \sqrt{\langle Y^2 \rangle - \mu_Y^2}.$$

Things can be even simpler taking normalized polynomials, i.e., multiplying by a suitable constant every polynomial such that

$$\left\| p_i(x) \right\|^2 = \langle p_i(x), p_i(x) \rangle = 1.$$

In the case of normal distributions, one can consider the normalized Hermite polynomials:

$$\widetilde{He}_k(x) := \frac{He_k(x)}{\sqrt{k!}}, \qquad \Longrightarrow \left\| \widetilde{He}_k(x) \right\| = 1, \qquad \forall k = 1, 2, \dots .$$

Doing so the variance of Y can be written as:

$$\sigma_Y^2 = \sum_{i=1}^{k} \alpha_i^2.$$

The formulas result very simple and effective.

Moreover, orthogonality guarantees that every polynomial expression of maximum degree k can be written uniquely as an expansion in the orthogonal family, i.e., the coefficients α_i are well defined.

For uniform, beta exponential and gamma probability distributions there exist families of orthogonal polynomials, and corresponding chaoses. The correspondence is summarized in Table 5.

In the case of multiple variables, assuming they are independently distributed, it is possible to write multivariate chaoses considering the tensorial products of the univariate polynomials corresponding to the distributions of every single variable.

For instance, in the two dimensional case, for two identical normal distributions, the family of orthogonal polynomials orthogonal associated to the joint

Distribution	Probability Density	Orthogonal Polynomials	Support Range
Normal	$\frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$	Hermite $He_n(x)$	$(-\infty, +\infty)$
Uniform	$1/2$	Legendre $P_n(x)$	$(-1, +1)$
Beta	$\frac{(1-x)^\alpha (1-x)^\beta}{2^{\alpha+\beta+1} B(\alpha+1,\beta+1)}$	Jacobi $P_n^{(\alpha,\beta)}(x)$	$(-1, +1)$
Exponential	e^{-x}	Laguerre $L_n(x)$	$(0, +\infty)$
Gamma	$\frac{x^\alpha e^{-x}}{\Gamma(\alpha+1)}$	Generalized Laguerre $L_n^{(\alpha)}(x)$	$(0, +\infty)$

Table 3. Wiener–Askey scheme on relations between probability distributions and orthogonal polynomials.

probability distribution $w_1(x_1) \otimes w_2(x_2)$ starts with:

$$\begin{aligned}
p_0(x_1, x_2) &= 1, \\
p_1(x_1, x_2) &= He_1(x_1), \\
p_2(x_1, x_2) &= He_1(x_2), \\
p_3(x_1, x_2) &= He_2(x_1), \\
p_4(x_1, x_2) &= He_1(x_1) He_1(x_2), \\
p_5(x_1, x_2) &= He_2(x_2), \\
&\vdots
\end{aligned}$$

thus, for writing a polynomial chaos in two variables of order 2, we have 6 parameters α_i, $i = 1, \ldots, 5$:

$$f(x_1, x_2) = \sum_{i=1}^{5} \alpha_i p_i(x_1, x_2).$$

More generally, to write a polynomial chaos in d variables of order k, the number of free parameters is

$$\# \left\{ \alpha_i \middle| \ \deg(f) = k, x = (x_1, \ldots, x_d) \right\} = \binom{k+d}{d} = \frac{(k+d)!}{k!d!}.$$

To have an idea on how the minimum sample size scales with dimensionality and chaos order, see Table 5.

5.1 Polynomial Chaos applied to a general function: Chaos Collocation

It is clear that if the function f to be studied is a polynomial the problem is finished, because the previous paragraph offers a methodology which will produce analytically exact values for mean and standard deviation.

$N \geqslant \frac{(k+d)!}{k!d!}$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$
$d=1$	2	3	4	5	6	7	8	9
$d=2$	3	6	10	15	21	28	36	45
$d=3$	4	10	20	35	56	84	120	165
$d=4$	5	15	35	70	126	210	330	495
$d=5$	6	21	56	126	252	462	792	1287
$d=6$	7	28	84	210	462	924	1716	3003
$d=7$	8	36	120	330	792	1716	3432	6435
$d=8$	9	45	165	495	1287	3003	6435	12870
$d=9$	10	55	220	715	2002	5005	11440	24310
$d=10$	11	66	286	1001	3003	8008	19448	43758
$d=11$	12	78	364	1365	4368	12376	31824	75582
$d=12$	13	91	455	1820	6188	18564	50388	125970
$\vdots$								

Table 4. Number of parameters in a chaos of order k in d variables, i.e., minimum size of a sample to be employed for chaos collocation.

In a more general case, assuming that the function f can be expanded in series with respect to the sequence of orthogonal polynomials, i.e., when f is analytical, the polynomial chaos method will produce an infinite series:

$$\begin{aligned} f(x) &= \sum_{i=0}^{\infty} \alpha_i p_i(x), \\ \mu_Y &= \alpha_0, \\ \sigma_Y^2 &= \sum_{i=1}^{\infty} \alpha_i^2 \left\| p_i(x) \right\|_w^2 . \end{aligned} \tag{11}$$

In numerical terms, it is necessary to arrest the expansion at a certain order, and the values for mean and standard deviation, exact for the truncated chaos, will be only an approximation of the true mean and standard deviation:

$$\begin{aligned} f(x) &\cong \sum_{i=0}^{k} \alpha_i p_i(x), \\ \mu_Y &\cong \alpha_0, \\ \sigma_Y^2 &\cong \sum_{i=1}^{k} \alpha_i^2 \left\| p_i(x) \right\|_w^2 . \end{aligned} \tag{12}$$

Another problem consists in determining the coefficients, problem that can be faced in two ways. Exploiting the orthogonality of the polynomials, coefficients

can be determined by Galerkin projections:

$$\begin{aligned} \langle f(x), p_i(x) \rangle &= \left\langle \sum_j \alpha_j p_j(x), p_i(x) \right\rangle = \alpha_i \left\| p_i(x) \right\|^2, \\ \alpha_i &= \frac{\langle f(x), p_i(x) \rangle}{\left\| p_i(x) \right\|^2}. \end{aligned} \tag{13}$$

The problem is shifted to which of computing effectively the integral $\langle f(x), p_i(x) \rangle = \int f(x) p_i(x) w(x) dx$. The best way to tackle it is via Gaussian integration formulas. In one dimension, a sample of $k+1$ points is needed to apply the formula. The points will be the Gaussian nodes associated to the weighting function $w(x)$, i.e., they will be the zeros of the orthogonal polynomial of degree $k+1$. Higher dimension requires even smarter strategy. The best solution is to employ Smolyak sparse grids, in order to keep reasonable the size of the sample employed for integration.

A simpler strategy for determining the αs consists in a nonlinear regression, a procedure called *chaos collocation*. αs are chosen in order to minimize the sum of the squares of the differences between $f(x)$ and a chaos of order k, over an assigned set of points $\{\bar{x}_1, \ldots, \bar{x}_N\}$:

$$\begin{aligned} \alpha_i : \qquad & \min_{\alpha_i} \sum_{j=1}^{N} \left| f(\bar{x}_j) - \sum_{i=1}^{k} \alpha_i p_i(\bar{x}_j) \right|^2, \\ & \{\bar{x}_1, \ldots, \bar{x}_N\}, \qquad \text{arbitrary sample} \end{aligned} \tag{14}$$

The size N of the sample has to be at least equal or larger than the number of parameters α, i.e., $N \geqslant \frac{(k+d)!}{k!d!}$, as noticed above.

Even if the points can be chosen arbitrarily, it seems an attractive idea to employ Latin hypercube samplings, because they can be of arbitrary size N, independently on d, and they are such that along every variable x_i there will be a point for each one of the strata with equal probability.

5.2 Polynomial Chaos Accuracy

In fact, it can be proved that the approximation is very good, i.e., as already mentioned above, the error scales as $\exp(-N)$. This means that in order to halve the error, it suffices to take a sample only $\log(2)$ times larger, and to reduce the error of an order of magnitude, it suffices to consider a sample $\log(10) \simeq 3$ times larger.

Example 3. Let $Y = f(X)$, as in (4) above. Uncertainty quantification via polynomial chaos has been performed as follows. Let the order of the chaos expansion be $k = 1, 2, 3, \ldots, 14$. For every value k it has been extracted a sample of size twice as large as the number of free parameters to be determined, i.e.,

$$N_k = 2 \times \frac{(d+k)!}{d!k!} = 2(k+1).$$

Next chaos collocation has been performed by least squares. The procedure has been replicated 40 times. The results are reported in Table 3 and in Figure 3, compared with the results of Monte Carlo and Latin hypercube.

Chaos Order	Sample Size	Average Error Mean	Average Error StDev
1	4	0.09344069282015036	0.10429522822805484
2	6	0.019326055906163132	0.0697189377206368
3	8	0.011271788209210331	0.02654736033749695
4	10	8.03467578078676E-4	0.001684371972847451
5	12	7.877378937433344E-4	0.0028642320499342065
6	14	6.138461875910162E-4	0.0018260971296785306
7	16	2.663000586760944E-4	8.607678781524158E-4
8	18	3.311333063379607E-4	2.9924223556135885E-4
9	20	3.4444147888486044E-4	3.7666167066683075E-4
10	22	6.672160731088228E-5	1.358801147891109E-4
11	24	4.41927163589595E-5	1.2805374914218737E-4
12	26	8.515981157662944E-5	2.491946753026775E-4
13	28	8.775184447483708E-5	2.5355458755216276E-4
14	30	1.2559038776682741E-5	1.1160349235239675E-5

Table 5. Polynomial Chaos replicated 40 times for the problem in (4)

We notice that an error of 1% is obtained with a sample of 8 points for the mean and with 12 points for the standard deviation, while a 0.1% is reached with 12 and 20 points, respectively.

The comparison of the performances of the three methodologies is summarized in Table 3.

	Sample size needed to reach 1% accuracy for Mean / St Dev	Sample size needed to reach 0.1% accuracy for Mean / St Dev
Monte Carlo	256 / 8192	16384 / –(≃ 800000?)
Latin hypercube	16 / 128	64 / 2048
Polynomial Chaos	8 /12	12 / 20

Table 6. Summary of the performances of Monte Carlo, Latin hypercube and polynomial chaos for the problem (4).

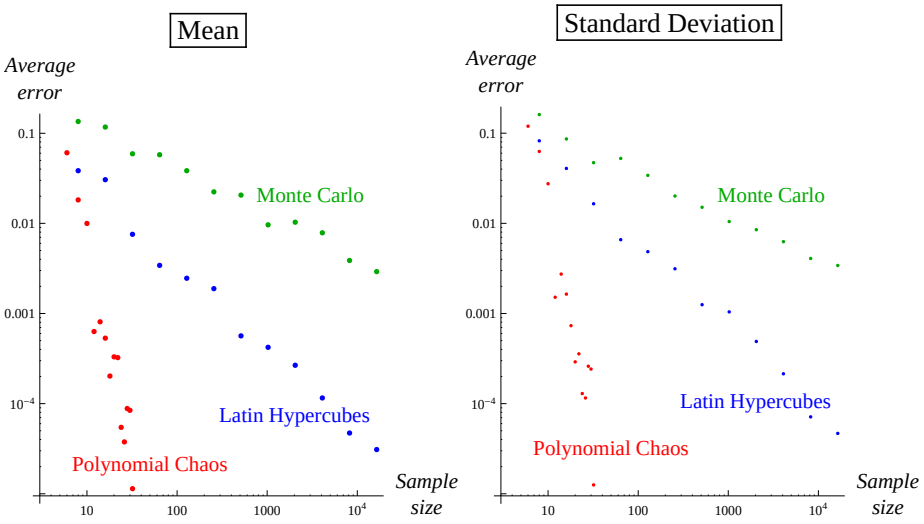


Fig. 3. Polynomial Chaos replicated 40 times compared with results from Latin hypercube and Monte Carlo for the problem in 4. The plot is represented in Log–Log scales. The convergence to zero is exponential, i.e., the slope in log–log scale tends to $-\infty$.

6 Conclusions and future works

In this paper we present some preliminary studies on the use of PCE for uncertainty quantification for solving multiobjective optimization problems when objective functions are not known analytically. This study shows that we can have a considerable reduction in terms of number of required evaluations when applying PCE instead of other classical approaches such as Monte Carlo or Latin hypercube samplings. This is an important result especially when the multiobjective optimization problem requires the evaluation of time consuming functions. A line of future work is to deepen into the study and application of the PCE in order to understand how constraints and boundaries can be included in the problem and how to deal practically with functions that, formally, cannot be expanded in series.

References

1. R Askey and J Wilson 1985 Some basic hypergeometric polynomials that generalize Jacobi polynomials *Memoirs Amer. Math. Soc.* **AMS** Providence RI 319
2. R Ghanem and J Red–Horse 1999 Propagation of probabilistic uncertainty in complex physical systems using a stochastic finite element approach *Phisica D* **133** 137–144
3. R Ghanem and P Spanos 1991 *The stochastic finite element method: a spectral approach* Springer

4. A. C. Mattson and A. Messac 2005 Pareto frontier based concept selection under uncertainty, with visualization *Optimization and Engineering* **6**(1) Special Issue on Multidisciplinary Design Optimization
5. M D McKay, R J Beckmkan and W J Conover 1979 A comparison of three methods for selecting values of input variables in the analysis of output from a computer code *Technometrics* **21** 239–245
6. Xiu D and Karniadakis G 2002 The Wiener–Askey polynomial chaos for stochastic differential equations *SIAM J. Sci. Comput.* **24**(2) 619–644
7. N Wiener 1938 The homogeneous chaos *Amer. J. Math.* **60** 897–936