Polynomial Chaos: A Tutorial and Critique from a Statistician's Perspective

Anthony O'Hagan
University of Sheffield, UK

March 24, 2013

Abstract

This article is written in the spirit of helping recent efforts to build bridges between the community of researchers in fields such as applied mathematics and engineering, where the term UQ began, and the community of statisticians who work on problems of uncertainty in the predictions of mechanistic models. It is addressed to researchers and practitioners in both communities.

The first purpose of the article is to explain polynomial chaos, one of the key tools of the first community, in terms that will be readily understood by a statistician in the second community. The second purpose is to explain to researchers in the first community some aspects of PC, both in theory and in practice, that a statistician might regard as deficiencies or limitations.

1 Introduction

The coming together of SIAM (the Society for Industrial and Applied Mathematics) and ASA (the American Statistical Association) to organise a conference in 2012 on Uncertainty Quantification was a major initiative to bring together two disparate communities concerned with broadly the same problem, namely to quantify uncertainty in the predictions of mechanistic simulation models. The conference coincided with an initiative in 2011–12 by SAMSI (the Statistical and Applied Mathematical Sciences Institute) with the same purpose. One of the two communities is founded in applied mathematics and various fields such as engineering and physics that use large science-based simulation models. I will call this the AM community. The other community is founded in statistics, where the application of statistical methods to quantify uncertainties in the predictions of mechanistic models has been a significant research area for at least a decade (with precursors going back much further than that). I will refer to this as the Stat community.

It is the AM community that is responsible for the term UQ (Uncertainty Quantification). To a member of the Stat community, however, UQ seems a

curiously inappropriate term because (a) it sounds as though it should cover the whole of Statistics, since one way of viewing the science of Statistics is precisely as the science of quantifying uncertainty, whereas (b) the uncertainties that are studied in the AM community do not even cover the range of uncertainties that statisticians would recognise in the predictions of models. Nevertheless, UQ is the *de facto* term that has been accepted by the SIAM/ASA joint initiative.

The principal aspect of uncertainty quantification that is studied in the AM community is propagation of input uncertainty. We write the simulator as a function η. Given input x, the simulator produces output $y = \eta(x)$. Typically, both x and y are vectors. Letting the input be uncertain, we represent this with a random variable X, in which case the output is also a random variable $Y = \eta(X)$. The problem of uncertainty propagation (UP) is to characterise the distribution of Y for a given distribution of X. A simple approach to the UP problem is to use Monte Carlo sampling, which involves drawing a large random sample $\{x_i : i = 1, 2, \ldots, N\}$ of values of X from its given distribution, running the simulator for each input x_i and so obtaining a sample $\{y_i = \eta(x_i) : i = 1, 2, \ldots, N\}$ from the distribution of Y. However, science-based simulators are often complex and take appreciable computing time to evaluate, so that the large number N of simulator runs required by Monte Carlo is impractical.

Another approach involves building a surrogate $\hat{\eta}$, also known as a metamodel, such that $\hat{\eta}(x)$ is an approximation to $\eta(x)$ for any x. Then we can approximate the distribution of Y by a Monte Carlo sample $\{\hat{y}_i = \hat{\eta}(x_i) : i = 1, 2, \ldots, N\}$. This will require a sufficiently large number of runs of the original simulator η to build the surrogate $\hat{\eta}$, but the chief idea of surrogates is that $\hat{\eta}$ is much simpler than η and so $\hat{\eta}(x)$ can be evaluated orders of magnitude more quickly than $\eta(x)$. Thus it is becomes feasible to create a very large Monte Carlo sample. The principal drawback to the use of surrogates is that we only obtain an approximation to the distribution of Y.

The AM and Stat communities have developed different tools to facilitate more efficient UP. Whereas the most widely used tool in the Stat community is Gaussian process emulation, the AM community principally uses polynomial chaos expansions. Both approaches can be viewed as based on the use of surrogates, but with quite different structures. The coming together of the two communities, and resulting sharing of methodologies, should benefit the field of UQ. In particular, we hope that the relative strengths and weaknesses of the two kinds of tools can be explored with a view to identifying the contexts in which each should be preferred.

The theory and methods of polynomial chaos (PC) are not well understood by most statisticians, and the primary purpose of this article is to present a tutorial on PC methods for members of the Stat community. However, it also serves as a critique from the perspective of a statistician, so that I also identify herein a number of apparent deficiencies or limitations which I hope the AM community will be able to respond to. These issues are raised throughout the main development in sections 2 and 3, generally in the form of Remarks, and then more fully in the final section 4.

It would no doubt also be useful for someone from the AM community to present a comparable tutorial and critique on Gaussian process emulators.

2 Polynomial Chaos expansions

2.1 Functions of random variables, the univariate case

A PC expansion (PCE) is a way of representing an arbitrary random variable of interest as a function of another random variable with a given distribution, and of representing that function as a polynomial expansion. I will begin by looking at the first component of this idea — representing one random variable as a function of another.

Let X be the random variable of interest, and let Ξ be the random variable in terms of which X will be expressed. In PC theory, Ξ is called the *germ*. For the moment, I will assume that both X and Ξ are scalar (i.e. univariate) random variables. The choice of germ is a modelling choice. For instance, we may in some cases choose to represent X as a function of a uniform random variable, and in other cases as a function of a standard normal variable.

Formally,the basic idea is to write

$$X = f(\Xi) \ , \tag{1}$$

and we seek an appropriate function $f(.)$, such that if Ξ has the given germ distribution then X will have its required distribution.

The first point to note about the representation (1) is that not only is this always possible but indeed there are always many such representations for given distributions of X and Ξ.

Example 1 *A canonical example uses the inverse CDF transform. Let the CDF of X be F_x and let Ξ be a uniform random variable on $[0,1]$, then as long as X is a continuous random variable its CDF will be invertible and hence*

$$X = F_x^{-1}(\Xi)$$

is a solution.

Example 2 *It is now a simple step to use an arbitrary germ Ξ with invertible CDF F_ξ. Since $F_\xi(\Xi)$ is a uniform random variable, we can write*

$$X = F_x^{-1}(F_\xi(\Xi)) \ , \tag{2}$$

so that $f = F_x^{-1} \circ F_\xi$.

Example 3 *To see that there are always many solutions, we only need to partition the range of Ξ into k segments with equal probabilities k^{-1} and do the same with X. We can then map any segment of Ξ into any segment of X, and the various permutations produce $k!$ solutions. This will of course very rarely lead to sensible constructions, but it makes the point that there are always many functions f that solve the basic requirement (1).*

Example 4 *In particular instances, of course, there may be much simpler solutions then those obtained with the inverse CDF transform. If X has a χ^2_1 distribution and the distribution of Ξ is standard normal, then we just have $X = \Xi^2$. It is easy to see that this is different from the general inverse-CDF solution (2) because CDFs are monotone, and therefore (2) is always a monotone transformation.*

Before progressing further it is useful to explore the meaning of a PC representation, and in particular the sense in which the two sides of (1) are equal. In Example 4, for instance, we have $X = \Xi^2$ and also $X = F_x^{-1}(F_\xi(\Xi))$, where F_ξ is the standard normal CDF and F_x is the χ^2_1 CDF. Clearly, if we interpret these literally as equations relating predefined random variables X and Ξ they cannot both be true. In statistical notation it would be more correct to write $X \sim f(\Xi)$ instead of (1), which is read 'X is distributed as $f(\Xi)$', meaning that X has the same distribution as $f(\Xi)$. Such a statement holds simultaneously for both representations in Example 4. Nevertheless, the equals sign is appropriate when we interpret the PC representation as a construction. That is, if we wish to construct a random variable X with a desired distribution using a random variable Ξ that has a given germ distribution, then in Example 4 $X = \Xi^2$ and $X = F_x^{-1}(F_\xi(\Xi))$ are both valid constructions. The different constructions yield *different* random variables X, but they both have the desired distribution.

This is indeed the appropriate interpretation when PC is used in UP. Suppose that the simulator η is fast, so that Monte Carlo is feasible. The input X has a given distribution and we need to sample from that distribution. Computer systems generally have inbuilt algorithms to sample from a uniform distribution over [0.1], and sometimes other standard distributions such as the standard normal. If the distribution of X is not one for which an inbuilt algorithm is available we need a transformation of the form (1). The interpretation is that the distribution we can sample from is the germ distribution, and so if Ξ is sampled using that inbuilt routine and we compute $X = f(\Xi)$ then X will be a sample value from the required distribution. Indeed, the inverse CDF transform of Example 1 is a standard way of sampling from an arbitrary distribution given a system-generated uniform random variable.

The message of the examples presented above is that, given distributions for X and Ξ there is no unique function f to satisfy (1). There are in principle many different solutions, different functions f that would take a germ random variable Ξ with its specified distribution and construct a random variable X with the desired distribution. Furthermore, additional solutions are available using different germ distributions. As we explore PC expansions and their use in uncertainty propagation, we will identify the considerations that arise in choosing a germ distribution and then choosing a representation f.

2.2 PCEs, the 1D case

A PCE takes a representation (1) and expands the function f in a polynomial series. The specific polynomial basis used is a set of orthogonal polynomials with

respect to the distribution of the germ. In order to define those polynomials, it is helpful to introduce the notation

$$\langle g_1, g_2 \rangle = \int g_1(\xi) g_2(\xi) p_\xi(\xi) d\xi \tag{3}$$

for the inner product of any two functions g_1 and g_2 with respect to the probability density function p_ξ of Ξ.

Then the polynomial basis comprises polynomials $\psi_0 = 1, \psi_1, \psi_2, \ldots$, where ψ_j is a polynomial of order j and where they satisfy the orthogonality condition that for all $j \neq k$

$$\langle \psi_j, \psi_k \rangle = 0 \ .$$

It is worth noting that all of the random variables $\psi_j(\Xi)$ for $j \geq 1$ have zero mean (by virtue of ψ_j being orthogonal to ψ_0), and so $\langle \psi_j, \psi_j \rangle$ is the variance of $\psi_j(\Xi)$ and orthogonality says that the covariance between any two different $\psi_j(\Xi)$s is zero — they are uncorrelated. A related interpretation that may be useful is that the inner product defines a metric on the space of random variables $g(\Xi)$ with finite variance. (The same metric is used in Bayes linear theory.)

Remark 5 *There is an arbitrariness in how each ψ_j is scaled. In principle we could construct a unique orthonormal basis set by insisting that $\langle \psi_j, \psi_j \rangle = 1$ for all j (so that the $\psi_j(\Xi)$s have variance 1), and this would simplify subsequent formulae. However, another convention which simplifies implementations is to require the leading coefficient of each polynomial to be 1, so we do not impose orthonormality. Note also that other orthogonal/orthonormal bases could be used that are not polynomial systems (e.g. Fourier-type bases), and these might be useful in some special applications.*

By construction, we get an orthogonal basis set starting from any initial germ distribution provided that this distribution has moments of all orders (otherwise it would not be possible to define the infinite set of polynomials).

Example 6 *If the germ is uniformly distributed we get Legendre polynomials. The classic Legendre polynomials correspond to a uniform germ on $[-1, 1]$, but can be adapted to the case more usually met in Statistics of uniformity on $[0, 1]$.*

Example 7 *If the germ is standard normally distributed we get the Hermite polynomials, the first few of which are $\psi_0(\xi) = 1$, $\psi_1(\xi) = \xi$, $\psi_2(\xi) = \xi^2 - 1$, $\psi_3(\xi) = \xi^3 - 3\xi$, $\psi_4(\xi) = \xi^4 - 6\xi^2 + 3$.*

Example 8 *If the germ is an exponential random variable on $[0, \infty)$ we get the Laguerre polynomials.*

It should be remembered that although these last three examples are widely used for the cases of bounded, unbounded and half-bounded domains there are many other orthogonal basis sets that could be constructed. On an unbounded domain, one could no doubt construct (with some difficulty!) a set of orthogonal

polynomials with respect to a logistic distribution (but not with respect to a Cauchy or t distribution because these do not have moments of all orders) and these would be different from the Hermite polynomials. The key point in PC theory is that the polynomial basis is linked with the germ distribution. The choice of germ dictates the polynomial basis functions.

Remark 9 *The use of Hermite polynomials with a normal germ is reminiscent of other expansions in statistics, such as the Gram-Charlier A series, but that expands a density function in terms of the standard normal density and a Hermite polynomial expansion. It is very important to remember that the PCE is an expansion to construct a random variable with a desired distribution as a function of another given random variable. It is not an expansion of the density function of X or even of its CDF.*

Now using the orthogonal basis polynomials we write

$$X = f(\Xi) = \sum_{j=0}^{\infty} x_j \psi_j(\Xi) \ . \tag{4}$$

In PC theory the combination of x_j (the mode strength) and ψ_j (the mode function) is called the j-th *mode*. Given f and the ψ_js there is a unique expansion (4) in which the mode strengths are given by

$$x_j = \langle f, \psi j \rangle \, / \, \langle \psi_j, \psi_j \rangle \ . \tag{5}$$

Any expansion of X in the form (4) is called a PCE. Since there are many possible functions f for given X and Ξ distributions, there are many possible PCEs of a given X using a given germ. They will differ only in the series of mode strengths.

For practical reasons, PCEs are often truncated to a finite number of terms, hence we consider

$$X_p = f_p(\Xi) = \sum_{j=0}^{p} x_j \psi_j(\Xi) \ . \tag{6}$$

It will of course no longer be true that the constructed X_p has the required distribution. Instead we suppose it to have a distribution that approximates to that required of X.

2.3 Practicalities

Consider now the question of how to choose and construct a PCE in practice to represent a given X distribution. The first step is to select (the distribution for) a germ Ξ. As we have seen, orthogonal polynomial systems are known for some specific choices of Ξ, but will otherwise be much more difficult to construct. In practice, then, the choice will typically be between uniform, normal or exponential distributions.

The next step is to devise a suitable representation function f. Several examples have been given to show that this is always possible and moreover that there will always be many solutions. In practice, though, apart from the general solution (2) it will not be straightforward to find representations. Considerable research effort has in the past been spent on efficient ways to sample from given distributions using uniform pseudo-random numbers, but many good sampling methods are iterative or implicit and I cannot see how those could be adapted for PC representations. Even the CDF solution requires the CDF of X to be known explicitly, which may not always be the case in practice.

Having chosen Ξ and f, the PC expansion is completed by deriving the coefficients x_j (the mode strengths), which means applying equation (5). The denominator $\langle \psi_j, \psi_j \rangle$ will in general be available from the construction of the orthogonal polynomials, but the numerator is an integral of the form (3) and unless f is particularly simple it will generally be necessary to evaluate these integrals numerically. When f itself requires numerical computation, as is often the case when the inverse CDF solution is used, the computation of mode strengths can be a non-trivial task.

In practice, as noted above, the PCE will be truncated to a finite number of terms, and so the number p is the final choice to be made in developing a PCE. Ideally p should be large enough for the approximation to be good. In practice, this is achieved in the usual way by increasing p until the mode weights appear to stabilise and new weights are small. However, this is really assessing whether f_p approximates well to f, and does not guarantee that the distribution of X_p approximates well to that of X. This could be checked for any given p by Monte Carlo sampling.

Truncation is where it matters what PC representation we use for a given X. A good representation f in practice is one for which the truncated representation f_p with small p will be a good approximation.

Example 10 *The representation of a χ^2_1 variable as $X = \Xi^2$, will have an exact polynomial expansion with $p = 2$, whereas the inverse-CDF representation may require large p to be any use at all.*

Remark 11 *It seems to me that achieving sufficiently accurate finite PCEs of the form (6) must be a major challenge. Furthermore, this implies that PC methodology is, at least in some of its aspects, an art.*

2.4 Multivariate PC theory

In general both X and Ξ can be vectors. If X is a vector, then the mode strengths x_j must be vectors. If Ξ is a vector, then the polynomial $\psi_j(\Xi)$ is multivariate and it is convenient to write it as Ψ_j. In the case where the components of Ξ are independent Ψ_j is a tensor product of the polynomial bases for each Ξ_i, and specifically when Ξ comprises m iid germs, we can let Ψ

have index $\mathbf{j} = (j_1, j_2, \ldots, j_m)$ and write

$$\Psi_{\mathbf{j}}(\xi) = \prod_{i=1}^{m} \psi_{j_i}(\xi_i) \ . \tag{7}$$

It is usual to truncate the PCE now to include all mode functions (7) with total order $\sum_{i=1}^{m} j_i$ less than or equal to p.

Remark 12 *Even this truncated expansion has $\binom{m+p}{m}$ terms. This number is of order m^p and so grows rapidly with p and m, which seems to me a potential difficulty when it comes to doing PC in many dimensions.*

The dimension of Ξ does not need to match that of X. Here are some examples to illustrate the possibilities.

Example 13 *A scalar X may usefully be expanded in terms of a vector Ξ. A simple example is the χ^2_2 distribution which has the expansion $X = \Xi_1^2 + \Xi_2^2$ in terms of two independent standard normal germs. The truncated expansion is exact with $p = 2$. In contrast, any expression in terms of a single germ is likely to be more difficult to derive and to approximate with a finite PCE.*

Example 14 *Another example is the Box-Muller transform for generating standard normal random variables. If Ξ_1 and Ξ_2 are independent uniform germs on $[0, 1]$ then $X = \sqrt{-2 \ln \Xi_1} \cos(2\pi\Xi_2)$ is standard normal. This uses a 2D germ to generate a 1D X, but the full Box-Muller uses the same 2D germ to generate a 2D pair of independent standard normals, using $Y = \sqrt{-2 \ln \Xi_1} \sin(2\pi\Xi_2)$ for the other.*

Remark 15 *The independence of X and Y in the last example arises from the fact that the sine and cosine are respectively odd and even functions of Ξ_2. Since the even-order Hermite polynomials are even and the odd-order ones are odd functions, this means that the PCEs for X and Y are supported on disjoint subsets of the basis functions. I imagine this may be a useful device in other contexts.*

Example 16 *In principle, a vector X could be expanded in terms of a scalar Ξ. I remarked already that two functions of a single germ could be uncorrelated or even independent. To see the general case, first look at a discrete analogue. If X is a discrete random vector then its joint distribution is just a discrete set of joint probabilities. If Ξ is a continuous germ, even though it is scalar, we can partition it according to the set of discrete probabilities in the joint distribution of X and map each set to the corresponding discrete vector value of X. Passing to the limit, even if it is hard to see how one could really construct such a representation for a continuous vector X it is easy to see how a (finite) PCE in the single germ Ξ could approximate the distribution of the vector X to any desired accuracy.*

Broadly speaking, the interest in good finite PCEs tends to lead to representations in which Ξ has dimension at least as large as, and often larger than, that of X.

2.5 PCEs for random fields

Moving up to even higher dimensions, we can consider a PCE for a random process (or field) $X(t)$, where the argument is conventionally denoted t but could be time, space or anything else. Now there are as many random variables as elements in the space of t values, which is of course usually (uncountably) infinite. The appropriate PCE representation would be

$$X(t) \approx \sum_{j=0}^{p} x_j(t)\psi_j(\Xi) \ . \tag{8}$$

where the mode strengths $x_j(t)$ are now functions of t. Despite my earlier comments that one can in principle construct a PCE for a vector X from a scalar Ξ, it is hard to imagine any practical PCE for a random field based on a finite-dimensional Ξ that could be exact, even as $p \to \infty$. Any representation (8) must in practice be an approximation both because it uses a finite number p of modes and also because it is based on a finite-dimensional germ.

An apparently different kind of expansion, Karhunen-Loève (KL), is reasonably well-known in statistics. It can be written in the form

$$X(t) = E\left\{X(t)\right\} + \sum_{j=1}^{\infty} \sqrt{\lambda_j}\phi_j(t)\Xi_j \ . \tag{9}$$

Here the λ_js ($\lambda_1 \geq \lambda_2 \geq ...$) and the $\phi_j(t)$s are the eigenvalues and eigenfunctions of a spectral decomposition of the covariance function of $X(t)$. This is exact as an infinite expansion, but again becomes approximate when we truncate it. The germs Ξ_j are uncorrelated, zero mean, unit variance random variables. When $X(t)$ is a Gaussian process, the Ξ_js are independent $N(0,1)$.

Remark 17 *Although not obvious initially, KL is a PC expansion. First, it is a PCE because it represents $X(t)$ as being distributed as some function of the germs. That function actually does have the form of (8) if we remember that in any system of orthogonal polynomials the zero order 'polynomial' is constant and the first order is linear. Therefore, (9) is a PCE with tensor product mode functions of the form (7) up to total order 1. If we truncate to p terms (plus the constant $j = 0$ term which in (9) is the mean function) it fits (8) exactly with a p-dimensional Ξ vector. So the KL expansion is a particularly simple form of PCE.*

Remark 18 *KL strictly only holds for t in a bounded interval or region. This may not be a problem in practice.*

Remark 19 *Whilst the Gaussian process representation does seem to me to have value, I am less sure of the usefulness of KL for other cases. In general, it really serves only as an expansion of the covariance kernel of the process, and says nothing about the distributions of the germs Ξ_j. Finding these to match any required properties of $X(t)$ beyond its second-order moments looks like an enormous challenge.*

3 Propagation

The primary objective in the AM UQ community's approach to UP is to create a PCE expansion for the output of a computer model.

3.1 The basic output PCE

Suppose that we have a computer model that produces output $y = \eta(x)$ when given input x.

We suppose that the input is uncertain and so is a random variable X. Consequently the output is a random variable $Y = \eta(X)$. In practice, both X and Y are generally vectors and may even be random fields. We suppose that we have a PCE for X of the form (6), and we seek to represent the output with another PCE

$$Y = g(\Xi) = \sum_{j=0}^{\infty} y_j \psi_j(\Xi) \ , \tag{10}$$

or in practice its truncated form

$$Y \approx g_p(\Xi) = \sum_{j=0}^{p} y_j \psi_j(\Xi) \ . \tag{11}$$

A key consideration here is that it is usual to employ the same germ (vector) Ξ and the same set of mode functions ψ_j for the PCE of Y as was used for the PCE of X. Furthermore, I understand that it is also usual to specify the same finite-order truncation level p for both expansions.

Remark 20 *I presume that in principle one might use different germs, although I suspect that this would make the analysis much more complex. One could surely use expansions of different orders.*

The implication is then that

$$\sum_{j=0}^{p} y_j \psi_j(\Xi) \approx \eta \left(\sum_{j=0}^{p} x_j \psi_j(\Xi) \right) \ . \tag{12}$$

Notice that in practice this cannot be other than an approximation. For general η equation (12) cannot hold exactly because any computer model of practical interest is nonlinear. In fact it will not even be a finite-order polynomial, and so we could not achieve equality by allowing more modes in the PCE of Y. As usual, the expansion only becomes exact when $p \to \infty$.

As before, the pragmatic approach is to increase p until mode strengths stabilise and new weights become small, hoping that this will signify that the approximation is adequate.

Remark 21 *There may be numerical-analytic bounds on the approximation error in some instances, but these are useless in practice. I believe it is fair to say that the PCE approach does not quantify the approximation error as a component of uncertainty.*

3.2 Intrusive solutions

In order to complete the PC analysis, we need to find the mode strengths y_j in (12). Methods to do this are classified in the AM community as *intrusive* and *non-intrusive*. I will briefly describe two methods of each type

Intrusive methods start with (12), substituting an equality sign for the approximation sign. On the assumption that equality holds, the strengths y_j are found by a method known as Galerkin projection. The first intrusive method comes from applying the Galerkin approach directly to (12).

From the original equation $y = \eta(x)$, we derive a system of equations

$$\langle y, \psi_k \rangle = \langle \eta(x), \psi_k \rangle \ ,$$

for $k = 0, \ldots, p$. Substituting both sides as in (12), we have

$$\left\langle \sum_{j=0}^{p} y_j \psi_j, \psi_k \right\rangle = \left\langle \eta \left(\sum_{j=0}^{p} x_j \psi_j \right), \psi_k \right\rangle \ . \tag{13}$$

Now because of orthogonality the left hand side is just $y_k \langle \psi_k, \psi_k \rangle$. If the function η is sufficiently simple we can also evaluate the right-hand side. The result is a system of $p+1$ equations ($k = 0, \ldots, p$) in $p+1$ unknowns ($y_0, \ldots, y_p$). We now need to modify the original computer code, or more realistically to write a new program, to solve those equations.

This direct approach is unlikely to be practical in general because of the complexity of η, and because for the great majority of simulators we do not have an explicit equation for $\eta(x)$. The second intrusive method applies in the very common situation where the simulator actually solves a system of differential equations. The output of such a simulator is generally a function of time and/or space, with the differential equations governing the evolution. We write the simulator now in the implicit form $\mathcal{M}(y(t), x) = 0$, where t denotes time and/or space and $\mathcal{M}$ typically involves derivatives with respect to t.

Now Galerkin projection gives the system of equations

$$\langle \mathcal{M}(y(t), x), \psi_k \rangle = 0 \ , \qquad k = 0, \ldots, p \ ,$$

in which we substitute the PCE expressions $y(t) = \sum_{j=0}^{p} y_j(t)\psi_j(\xi)$ and $x = \sum_{j=0}^{p} x_j \psi_j(\xi)$. Because the differential equations themselves are often not particularly complex, the inner product $\langle \mathcal{M}(y(t), x), \psi_k \rangle$ may be derivable in closed form, and orthogonality will typically simplify the expressions. We are left with a system of $p+1$ differential equations in the $y_j(t)$s. Solution will involve writing a new program to solve those equations.

In short, both intrusive methods involve two steps.

1. Substitute the PCEs for both input and output into the model equations, apply Galerkin projection and evaluate the necessary inner products.

2. Write a program to solve the resulting system of equations.

The apparent simplicity of the above formulation can be deceiving. The reality is that intrusive methods are complex to apply and my understanding is that they have been little used in practice. Some of the complexity and limitations are indicated in the following series of remarks.

Remark 22 *Intrusive methods are tailored to the particular simulator. Algebraic manipulation (and perhaps ingenuity) is required to formulate the system of equations, and a new program is required to solve them. This is a task rather like developing an adjoint for a computer code.*

Remark 23 *The set of equations obtained is specific to the number p of modes in the expansion. The solver program could presumably be written for a range of p values, although this will increase the complexity of the task.*

Remark 24 *If we change the distribution of X, then provided we express it in a PCE with the same germs and orthogonal polynomials we can solve this with the same program, just changing the x_ks. But we have seen that different distributions may be best expressed using different germs, in which case the intrusive method would require both the above steps to be repeated.*

3.3 Non-intrusive solutions

We have seen that intrusive methods require one to use knowledge of the equations that are solved in η. In contrast, non-intrusive methods treat the simulator as a black box. They attempt to solve the explicit version of the problem,

$$\sum_{j=0}^{p} y_j \psi_j(\xi) = \eta(f_p(\xi)) \ ,$$

where $f_p(\xi) = \sum_{j=0}^{p} x_j \psi_j(\xi)$, using runs of the simulator at various values of ξ.

The first non-intrusive method discussed here uses numerical integration and the mode strength equation (5), which now becomes

$$y_k = \langle \eta \circ f_p, \psi_k \rangle \ / \ \langle \psi_k, \psi_k \rangle \ , \qquad k = 0, 1, \ldots, p \tag{14}$$

The denominator $\langle \psi_k, \psi_k \rangle$ is in practice known exactly, and we use numerical integration to evaluate the numerator

$$\langle \eta \circ f_p, \psi_k \rangle = \int \eta(f_p(\xi)) \psi_k(\xi) p_\xi(\xi) d\xi \ .$$

One approach is Monte Carlo, in which values of ξ are sampled randomly from the germ distribution $p_\xi(\xi)$, If there are N such values, $\xi^{(1)}, \ldots, \xi^{(N)}$, then the integral is evaluated as $N^{-1} \sum_{s=1}^{N} \eta(f_p(\xi^{(s)})) \psi_k(\xi^{(s)})$. Alternatively, we can use a quadrature rule. For instance, for a $N(0,1)$ germ, a Gauss-Hermite quadrature rule is an obvious choice.

Remark 25 *Monte Carlo typically requires a large number of simulator runs, which is impractical for computationally-intensive models, but on the other hand in higher dimensions, tensor products of quadrature rules demand even more simulator runs. Although sparse quadrature rules exist and are more efficient, still quadrature seems impractical when the germ vector is high-dimensional.*

The second non-intrusive method also takes a sample of ξ values, but now treats the values $\eta(f_p(\xi^{(s)}))$ as observations, say z_s, in a regression model

$$z_s = \sum_{j=0}^{p} y_j \psi_j(\xi^{(s)}) + \varepsilon_s \ ,$$

in which the unknown regression coefficients $y_0, y_1, \ldots, y_p$ are estimated by least squares (or some other inferential method, e.g. Bayesian inference). The design points $\xi^{(s)}$ would now be chosen according to regression design principles, or simply to cover the input space. In general, random sampling from the germ distribution would not be efficient.

The fact that in this approach we need an error term ε_s is a clear result of the PCE for Y being only an approximation. The residual error variance from fitting this regression will indicate just how good the approximation is.

Remark 26 *Regression is a common way of building a surrogate model and polynomials are the commonest regressor variables for this purpose. Using orthogonal polynomials helps with the numerical stability of the computations, but orthogonality then needs to be with respect to the empirical distribution of the design points, rather than with respect to the germ distribution. Using PC offers no advantages that I can see.*

3.4 Propagation with the output PCE

Suppose that, by whatever method we use (and there are undoubtedly others not mentioned here), we have obtained values for the strengths $y_0, \ldots, y_p$. (In the case where the simulator output is a vector, these strengths are vectors, and when the output is a function the strengths are functions.) We now have the PCE for the output in the form $y = \sum_{j=0}^{p} y_j \psi_j(\xi)$. UP entails characterising the distribution for the computer model output Y, as induced by the distribution of the input X. This distribution, often called the *uncertainty distribution*, is (approximately) the distribution induced by the germ distribution and the (truncated) PCE (11), which can be seen as a surrogate. The CDF can be found by Monte Carlo sampling from the germ distribution (and of course this is now fast because we are evaluating the surrogate, not the simulator).

Other properties can also be found by Monte Carlo, but it is worth noting that the first two moments of Y can be obtained exactly using properties of the orthogonal polynomials. Thus,

$$E(Y) = y_0 \ , \tag{15}$$

$$E(Y^2) = \sum_{j=0}^{p} y_j \left\langle \psi_j, \psi_j \right\rangle \ , \tag{16}$$

from which the variance of Y follows trivially. For some purposes, it may be that having only the mean and standard deviation of Y is an adequate characterisation of the output uncertainty.

Expressions for higher moments may be obtained similarly, but for instance the third moment requires the evaluation of integrals of the form

$$\int \psi_j(\xi)\psi_k(\xi)\psi_\ell(\xi)p_\xi(\xi)d\xi \ .$$

For the most widely used germ distributions, these can no doubt be obtained in closed form, but for third and higher moments there are rapidly increasing numbers of such integrals involved. If higher moments are required it may be easier to evaluate them by Monte Carlo.

4 Discussion

The preceding sections have set out the basic theory of polynomial chaos expansions and their use for uncertainty propagation, to the extent that I understand them. In this final section I will present a number of personal reflections on these methods from the perspective of a statistician who is much more familiar with the UQ methods of the Stats community. If the overall tone of these reflections is negative, this may be due to misunderstandings on my part and I would very much welcome comments from the AM community.

4.1 Practicalities

In my various remarks in the preceding sections, I have made a number of comments on the practicalities of building PC expansions in general, and in particular for the output of a computer code. It may be helpful to reiterate my concerns here.

In practice PC expansions must be truncated. It seems to me intrinsically difficult to build expansions for the inputs X that are sufficiently accurate (in the sense that the distribution of the expansion is close to that of X) without using a large number p of terms. This would appear to be particularly true when (as will always be the case in practice) X is a vector, unless the elements of X are independent and each is represented in a PCE with its own germ. When X is a random field, only the Karhunen-Loève expansion (a very special and simple form of PCE) seems to be usable, and then only for a Gaussian random field. See Remarks 11, 12 and 19.

Developing a PCE for the model output Y requires the mode strengths y_j to be evaluated. Methods termed intrusive and non-intrusive are considered in the AM UQ community. Intrusive methods are mathematically appealing, and the second method has the particularly attractive feature that it can result in a surrogate for an output that is a function of time and/or space. I am not aware of any other surrogate method that could do that except by the laborious process of building surrogates for numerous individual points on the function.

However, intrusive methods entail much more work (Remarks 22 and 23) and I understand that they are rarely used in practice.

Non-intrusive methods do not make use of knowledge about the function η, and instead operate by evaluating a sample of outputs $\eta(f_p(\xi^{(s)}))$ for a (random or designed) sample of points $\xi^{(s)}$ in the space of the germ(s). As such, they do not seem essentially different to me from other kinds of surrogate methods which build an approximation to η using a sample of outputs $\eta(x^{(s)})$, except that the latter are surely intrinsically more accurate by virtue of the fact that they do not additionally approximate x by the truncated expansion $f_p(\xi)$. The regression method in particular seems to be basically the same as response-surface methods for building surrogates, but with the restriction that the regressor functions must be the orthogonal polynomials ψ_j. See Remark 26.

Having developed an approximate, truncated PCE for the model output, the fact that it is based on orthogonal polynomials with respect to the germ distribution(s) allows the mean and standard deviation of Y to be obtained very simply (but it is important to remember the expansion is only approximate and so equations (15) and (16) are not really exact). However, to characterise the distribution of Y more fully or in any other ways requires numerical computation, for instance using Monte Carlo sampling from the germ distribution(s). In this sense, it is no different from any other surrogate method which creates a computationally quick approximation to the simulator and computes UP by Monte Carlo sampling from the distribution of X. If the distribution of X is not uniform, the Monte Carlo sampling will require an algorithm to draw from that distribution using uniform pseudo-random numbers, but this is completely analogous to building a PCE for X. In fact it is more general because sampling algorithms do not need to be explicit functions like the PC function f. So unless the mean and standard deviation would represent an adequate UP characterisation, I do not see any distinctive advantage of PCEs over other surrogates.

Another general issue is what happens if we wish to change the distribution of X. The original computer model is designed to work for any x in the relevant domain, and this versatility is an important feature of simulators generally, allowing them to be used in a variety of situations. To achieve the same versatility for UP, we would wish to be able to handle any reasonable distribution for X. However, this is not readily achieved with PC methods. In principle, changing the distribution of X and retaining the same germ distribution(s) just means using different mode strengths x_j. However, any change in these strengths will entail recomputing the output strengths y_j (Remark 24). Furthermore, a truncation level p that works for one distribution of X may yield a very poor approximation with another distribution. Other surrogate methods in my experience do not have this problem.

4.2 The nature of the output PCE

I have presented some intrusive and non-intrusive methods for computing the strengths y_j to create what I have called the (truncated) output PCE $Y \approx g_p(\Xi) = \sum_{j=0}^{p} y_j \psi_j(\Xi)$. The implication here is that as $p \to \infty$ it tends to a

valid PC expansion (10) for the model output Y. That is, the limit of g_p is the function $\eta \circ f$, where f is a valid non-truncated PC representation for the distribution of X. However, this is clearly not the case. In all of the methods we have examined, the construction of the y_js uses the finite PCE for X, and so the limit of g_p can at best be the function $\eta \circ f_p$. At best, the output PCE is a (truncated) PCE for an approximation to the computer model. All the other surrogate modelling techniques that I am aware of build surrogates for η, and so do not entail this additional approximation.

The next question I want to consider is whether the limit of g_p is indeed the function $\eta \circ f_p$. This will be true if the mode strengths y_j satisfy the equation (5), which in this case becomes

$$y_j = \langle \eta(f_p), \psi_j \rangle \,/\, \langle \psi_j, \psi_j \rangle \ . \tag{17}$$

Consider each of the four construction methods considered in Section 3 — the explicit and implicit intrusive methods and the quadrature and regression non-intrusive methods.

The explicit approach clearly does satisfy (17) by inspection of (13). However, it is not clear to me whether the implicit method does. I would appreciate some clarification and a formal proof of this. Moving to the non-intrusive methods, again it is obvious from (14) that the quadrature approach does solve (17), but again it is by no means clear that the regression method does; indeed it seems to me that it does not.

It is clear that we can treat the output PCE as a surrogate for the computer model, because this only implies that it is some kind of quick approximation. However, the nature of the approximation is complex. First, a truncated PCE is an approximation. Second, it is at best an approximation to an approximation, the second approximation resulting from replacing $\eta \circ f$ by $\eta \circ f_p$. Third, it is not clear that the truncated PCE actually is the truncation of a valid PCE for $\eta \circ f_p$.

4.3 The output PCE as a surrogate

As a surrogate model, the output PCE will generally be much faster to compute than the simulator $\eta \circ f$ that it approximates. We should be able to use it for all the purposes to which surrogates are typically put. However, its argument is not x but ξ; it is not explicitly a surrogate for η. If I want a surrogate with argument x, I can use the PC surrogate provided I can convert from ξ to x. Suppose that f is invertible, so that $\xi = f^{-1}(x)$. Then (but only then) I argue that it is reasonable to treat $\sum_{j=0}^{p} y_j \psi_j(f^{-1}(x))$ as a surrogate for $\eta(x)$. I believe that in the case of univariate X and Ξ it would not be difficult to show that there is a unique monotone increasing f. If so, then it would have to be the inverse-CDF transformation (2), whereupon $\xi = f^{-1}(x) = F_{\xi}^{-1}(F_x(x))$.

Remark 27 *Note that I would not suggest inverting the truncated form f_p to give the surrogate $\sum_{j=0}^{p} y_j \psi_j(f_p^{-1}(x))$. Not only does it entail another approximation but even if f is monotone there is no guarantee that f_p will be. In some*

applications, f may not be known, with f_p constructed from partial information about the distribution of X. It is not immediately obvious to me that even if f_p is monotone it will be unique.

In the more important multivariate case we can do this inversion for each element separately when each element of X has a separate PC representation in terms of the corresponding element of Ξ. I suppose it would not be feasible otherwise. The following are common uses of surrogates.

1. *Uncertainty analysis (UA).* This is what we have been calling UP: X is uncertain, and we seek to characterise the distribution of $Y = \eta(X)$. As we have discussed in Section 3.4, the output PCE can be used for this purpose. If X is random, approximating $\eta(f(\Xi))$ when Ξ has the specified germ distribution is obviously equivalent to approximating $\eta(X)$ when X has its original distribution since this is also the distribution of $f(\Xi)$.

2. *Sensitivity analysis (SA).* SA seeks to identify which elements (or groups of elements) of the input vector X are most influential on the model output. Local SA involves examining derivatives of y with respect to the elements of the input vector x. Because it is trivial to differentiate both y and x with respect to the elements of ξ, using their respective PCEs, I presume we can readily compute local SA (and it does not depend on the distribution of X). However, unless f is invertible I suspect the answer will depend on the choice of f. Variance-based SA again treats X as random, and I presume the output PCE can be used for this in the same way as for UA.

3. *Prediction.* The simplest use of a surrogate is simply to approximate the model output $\eta(x)$ at some specified x (when the computer time required to run the simulator and compute $\eta(x)$ directly is not available). This may only make sense when f is invertible.

4. *Optimisation.* Another common use of surrogates is to search for the x that minimises $\eta(x)$. I foresee this also being a problem unless f^{-1} is well defined.

4.4 Accounting for all uncertainties

Uncertainty is everywhere around us, and comes in many forms with many sources. This is why Uncertainty Quantification sounds to me as though it ought to address much more than UP. I acknowledge that the emerging field that has been given the name UQ has hitherto really only concerned itself with UP, but I will take this opportunity to mention just two aspects of uncertainty that I believe have to be addressed as part of any exercise that deserves the name of Uncertainty Quantification. These aspects are very much addressed within the Stats UQ community and I hope that by raising them here I can persuade the AM community to give them serious consideration, and thereby to widen the remit of UQ itself.

The first is simply quantifying the uncertainty in the input vector X. In this article I have assumed, as is the case in UQ generally, that the distribution of X is given, but where does it come from? This is a serious problem in its own right and techniques for input uncertainty quantification are every bit as important as tools to propagate the uncertainty in X through the model. Those techniques include analysis of source data and elicitation of expert judgement. A starting reference is O'Hagan (2012).

The second area is uncertainty concerning the real world system that is modelled by $\eta(x)$. All models are imperfect, either in their form, in the numerical values of parameters in equations or in the solution of those equations. UP characterises uncertainty about $Y = \eta(X)$, but it does not address the real question of uncertainty about the underlying true system. If we denote this as Z, then Z differs from Y due to model discrepancy and uncertainty about model discrepancy means that there is additional uncertainty about Z. Quantifying this should also be part of UQ, and again it is something that is seriously addressed in the Stats UQ community. A starting reference is Brynjarsdóttir and O'Hagan (2013).

However, even if we ignore these additional sources of uncertainty, there is more uncertainty in Y than is addressed in PCE uncertainty propagation. We have seen that the resulting characterisation of output uncertainty is necessarily approximate because the output PCE is an approximation to Y itself. This is true of all surrogate methods, and all should acknowledge this extra component of uncertainty regarding Y. When Monte Carlo is used for UP with many runs of the simulation model itself, rather than via a surrogate, this uncertainty is explicitly quantified in the Monte Carlo sampling precision. In the case of the output PCE, formal accounting for that additional uncertainty will be difficult because as discussed in Section 4.2 it has three distinct components (plus even more uncertainty due to error in estimates of the y_js!). This does not, however, justify ignoring the additional uncertainty.

4.5 PC versus GP

I will end with some personal comments on the relative merits of PC expansion and Gaussian process (GP) emulation for propagating and analysing uncertainty in computer models, i.e. for UQ.

- *Surrogate bases.* Both the PC and GP methods build surrogates, and there are other surrogate methods which are advocated by various authors. A PC expansion has an orthogonal polynomial basis, and as soon as we consider using polynomials of order three or more they can be poor interpolators prone to unstable swings (as is the experience with corresponding forms of spline). For the PCE, the polynomials are in ξ rather than x, but I do not believe this will improve their performance. GP emulators effectively use radial basis functions instead of polynomials and so (in common with some other forms of surrogate including wavelets) tend to perform better in capturing local structure.

- *Efficiency.* The PCE approach has some advantages over some of the surrogate methods in terms of efficiency. The fact that the mean and variance of Y are available (approximately) in closed form is an obvious benefit. Also, when the output is a function, and when the implicit intrusive method is feasible, the PCE approach is impressive. GP emulators also deliver closed form expressions for the mean and variance of Y in some practically important situations, but to build a GP emulator for a function requires many compromises. However, it seems in practice the implicit intrusive method is not used; the effort involved in intrusive methods presumably outweighs any benefits for practitioners. Another aspect of efficiency is the number of simulator runs required to build the surrogate, and here my experience is that GP emulation is substantially more efficient than other surrogate approaches.

- *Quantifying uncertainty.* An emulator is not just a surrogate. I use the term 'surrogate' to describe a fast approximation to a simulator. In contrast, an emulator is a stochastic process providing a full probability distribution for the simulator output as a function of its inputs. Its mean function is a surrogate, but the GP is much more than its mean function. The GP probability distribution for predictions is itself a quantification of uncertainty (and is quantified in UA or SA as 'code uncertainty'). The PC approach involves approximations via the finiteness of the PC expansions of both input and output. To me, this means additional uncertainty in every use (e.g. UA, SA or prediction) of that surrogate, and this uncertainty is not quantified. (The only exception is the non-intrusive regression method which quantifies uncertainty via the residuals, but this quantification is wrong because the regression model is wrong for a deterministic simulator.) The GP leaves no such components of uncertainty unquantified.

- *Flexibility.* The distribution of X is fixed in the PC surrogate. If we change the distribution of X, even with the same germ, the surrogate must change, and will generally require a new effort to create. This is not the case with the GP emulator (or other kinds of surrogate). Note, however, that the GP approach could borrow the idea of expressing X as a PCE, and thereby create a surrogate whose argument is also ξ rather than x. There might be value in expressing a non-normal X in terms of normal Ξ, since one common way of constructing a GP allows UA/UP and SA to be done in closed form for normally-distributed inputs. The drawbacks of this would be (a) the fact that the PCE entails an approximation, and for the reasons given in the previous comment I would be reluctant to do this, and (b) the fact that the GP would now also require the building of a new emulator if the distribution of X changed.

I look forward to seeing more discussion of the relative merits of these approaches, with a view to identifying the strengths and weaknesses of each and the kinds of situations where one might be preferred over the other.

References

Brynjarsdóttir, J. and O'Hagan, A. (2013). Learning about physical parameters: The importance of model discrepancy. Submitted to *SIAM/ASA Journal of Uncertainty Quantification*.

O'Hagan, A. (2012). Probabilistic uncertainty specification: Overview, elaboration techniques and their application to a mechanistic model of carbon flux. *Environmental Modelling and Software* **36**, 35–48.