An Applied/Computational Mathematician's View of Uncertainty Quantification for Complex Systems

Max Gunzburger

Abstract Uncertainty quantification (UQ) is defined differently by different disciplines. Here, we first review an applied and computational mathematician's definition of UQ for complex systems, especially in the context of partial differential equations (PDEs) with random inputs. We then discuss the types of stochastic noises that are used as inputs to the PDEs and, for the case of infinite stochastic processes, how those inputs are approximated so that they are amenable to computations. We then review methods that are used to obtain approximations of solutions of PDEs with random inputs, with special emphases given to stochastic Galerkin and stochastic sampling methods, including sparse-grid methods in the latter case. We close with a brief foray into where UQ in the PDE setting is going moving forwards.

1 Introduction

We begin with some general comments that serve to introduce, set up, and focus the material presented in subsequent sections. We do not provide copious citations of the literature; instead, we list [1–5] a few general references in that provide additional details and more comprehensive mathematical and algorithmic expositions of what is written about in this paper.

Uncertainty Is Everywhere Physical, biological, social, economic, financial, etc. systems always involve uncertainties. Certainly, mathematical models of these systems should account for uncertainty. Uncertainties are often classified into two classes. *Epistemic* uncertainty is due to *incomplete knowledge* of the system so that, at least in principle, uncertainty can be reduced by additional measurements, improvements in measuring devices, etc. Although such uncertainties are, again in principle, predictable, it may be too difficult, perhaps impossible, or too costly

M. Gunzburger (✉)
Department of Scientific Computing, Florida State University, Tallahassee, FL, USA
e-mail: mgunzburger@fsu.edu

N. Fillion et al. (eds.), *Algorithms and Complexity in Mathematics, Epistemology, and Science*, Fields Institute Communications 82,
https://doi.org/10.1007/978-1-4939-9051-1_5

to obtain additional measurements. Subsurface media properties in oil reservoirs or aquifers are an example of this type of uncertainty. *Aleatoric* uncertainty is *intrinsic* in the system so that uncertainty cannot be reduced through additional measurements, improvements in measuring devices, etc. Running an experiment twice with exactly the same settings still results in different outcomes. The distinction between the two is certainly fuzzy; one person's aleatoric uncertainty may be another's epistemic uncertainty.

If computations are involved in uncertainty quantification (UQ), there is a third type of uncertainty which we refer to as *computational epistemic* uncertainty. In this case, not everything in the data and/or solutions can or should be resolved because it is too difficult, perhaps impossible, or too costly to do so in a computational simulation; turbulent flows are an example of this situation. Alternately, some scales may not be of interest, e.g., surface roughness, hourly stock prices; in such cases, uncertainty, e.g. randomness, is sometimes *artificially introduced* into the system to model the effects that behaviors at the unresolved scales have on that can be resolved.

Everyone realizes that laboratory experiments are not precisely repeatable which they often (and always should) be reported with error bars. But, are computer experiments repeatable? Running the same code with exactly the same inputs and on exactly the same computer should result in the same outputs. This statement can remain true even if there is randomness in the inputs because on a computer, one uses quasi-random number generators which one can reproduce from one computational run to the next. However, running the same code with exactly the same inputs on different computers or using different software can result in different outputs, not just because the two computers may use different pseudo-random number generators, but also for other hardware and software differences. With regards to the believability of experimentally determined data vs. data determined computationally, one should recall the quote[1]

> Experimental results are believed by everyone,
> except by the person who ran the experiment.
> Computational results are believed by no one,
> except by the person who wrote the code.

Who Does Uncertainty Quantification and Why Does Everyone Now Want to Do It? Certainly statisticians do; some statisticians even aver that statistics = uncertainty quantification. However, below, the way we define UQ is perhaps not the most common way UQ is thought of by statisticians. Perhaps certain philosophers and probabilists feel the same was as do statisticians. Scientists and engineers of all types do UQ as do some mathematicians, especially those involved in algorithmic and mathematical model development and analysis. So, indeed, everyone does UQ, but not all do it well or honestly.

[1]Some attribute this quote as a slightly modification of a quote attributed to Albert Einstein: A theory is something nobody believes, except the person who made it. An experiment is something everybody believes, except the person who made it.

These days "everyone" wants to do UQ. For many it is because there is money in it but, being less mercenary, some may genuinely believe it is interesting and important.

What Is a Complex System? *Complex system* is a terminology introduced to make it look like one is solving a difficult problem. Thus, it is very useful to write, when you are preparing a proposal for funding, that your work deals with a complex system; that will certainly do you more good than if you write that you work on simple systems. Here is a working definition of what constitutes a complex system.

Let us denote the system input by $\boldsymbol{y}$ and the system output by u. If we change the system input we change the system output so that we can think of the output u as being a function of the system input $\boldsymbol{y}$. *A complex system is one for which determining the dependence of the output on the input requires copious computational resources.* For example, the evaluation of a known function, i.e., $u = f(\boldsymbol{y})$ with $f(\boldsymbol{y})$ a known function, or the integral over a domain D of a known function $f(\boldsymbol{y})$, i.e., $u = \int_D f(\boldsymbol{y})\, d\boldsymbol{y}$ with $f(\boldsymbol{y})$ a known function, are examples of what is not a complex system.

Differential equations, especially partial differential equations, and especially nonlinear partial differential equations provide many examples of what qualifies as a complex system according to our definition, For example, the Navier-Stokes equations for fluid flow

$$\begin{cases} \rho\Big(\dfrac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\Big) - \nu\Delta\mathbf{u} + \nabla p = \mathbf{0} & \text{in a domain } D \\ \nabla\cdot\mathbf{u} = 0 & \text{in a domain } D \\ \mathbf{u} = \mathbf{f}(\boldsymbol{y}) & \text{on the boundary of } D \end{cases}$$

is a complex system for which the outputs (the velocity $\mathbf{u}$ and the pressure p) must be solved for.

Thus, throughout this paper, it is assumed that *function evaluations*, i.e., determining outputs from the inputs, is an expensive proposition.

A Word About Simulations *Simulation*, a word we have already used without giving its definition, is a noun derived from the verb simulate. In turn, the Oxford English Dictionary definitions of simulate include:

1. pretend to be, have, or feel
2. imitate or counterfeit.

In other words, to simulate is to cheat. However, as a fourth definition, that dictionary has

4. produce a computer model of (a process)

Thus, for us, a simulation is an approximation of the solution, i.e., the output, of a mathematical model that is obtained using computers, especially in situations in which the mathematical model itself is not exactly solvable.

How Do Uncertainties Enter into the Mathematical Descriptions of Systems? Before answering this question, it is useful to keep in mind what the great statistician George Box said about models:

> All models are wrong but some models are useful.

Less well known but equally powerful is another quote of George Box:

> Since all models are wrong the scientist cannot obtain a "correct" one by excessive elaboration. On the contrary, following William of Occam he should seek an economical description of natural phenomena. Just as the ability to devise simple but evocative models is the signature of the great scientist, so overelaboration and overparameterization is often the mark of mediocrity.

Now, back to the question of how uncertainties enter into mathematical models.

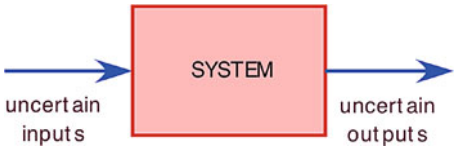
First, *the structure of model itself may not be known precisely.* A trivial example would be if all we knew about a function is that it is continuous and odd, so we could model it as $u = \sin y$, or $u = \tan y$, or $u = y^{2n+1}$ for some positive integer n, or as any of the other countless possibilities. Clearly, we need more information about the function to narrow down the choices. In reality, things are more subtle than that, especially in the context of phenomenological models, i.e., models that cannot be derived with mathematical precision from more basic or more generally accepted models. We also do not often know if simplified models that are cheaper to compute with provide good enough answers.

Even if the model form is agreed upon, it may contain *parameters whose values are not precisely known*, e.g., we know we have a function of the form $u = x^{\alpha}$ but α is a number whose value is not precisely known or we know our model form is $-\alpha \frac{d^2u}{dx^2} = x^2$ but again α is not precisely known.

Beyond input parameters, the model may contain *inputs functions that are uncertain* at every point in their domains, e.g., the coefficient and right-hand side of the differential equation $-\frac{d}{dx}\left(\alpha(x)\frac{du}{dx}\right) = f(x)$.

2 Uncertainty Quantification (UQ)

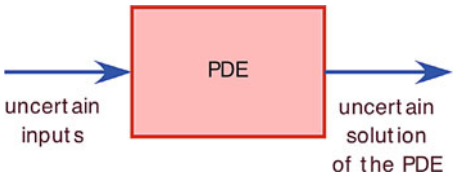
In this section, we provide definitions and discussions about UQ in its many guises as viewed by many applied and computational mathematicians. For that community, a general definition that is illustrated in the figure below is that



UQ is the task of determining information about the uncertainty in the outputs of a system, given information about the uncertainty in its inputs

Of course, a system may have additional inputs that are known with certainty. Let's narrow down this definition a little. We consider systems governed by *partial differential equations* (PDEs) so that now

UQ is the task of determining information about the uncertainty in the solution of a PDE, given information about the uncertainty in its inputs



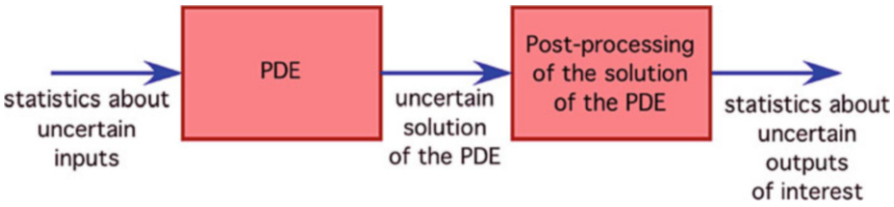
The solution of the PDE defines the mapping from the input variables to the output variables; as already mentioned, very often, PDEs do indeed model complex systems. Determining approximate solutions of a PDE usually requires costly computations. In fact, solving PDES was the reason modern computers were invented in the 1940s and even today, remain a major driving force behind the development new supercomputers.

Often, the solution of the PDE is not the primary output of interest. Of more interest are quantities obtained by post-processing solutions of the PDE to determine outputs of interest. Of course, one still has to obtain a solution of the PDE to determine the output of interest. Thus, now

UQ is the task of determining information about the uncertainty in an output of interest that depends on the solution of a PDE, given information about the uncertainty in its inputs

The desired information about the output is referred to as a *quantity of interest* (QoI).

There are several approaches towards UQ, including but not limited to, fuzzy sets and possibility theory, interval arithmetic, probabilistic approaches, evidence theory (e.g., Dempster-Shafer theory), etc. We consider *probabilistic approaches*, i.e., the uncertainty in the inputs of the PDE are described in terms of statistical quantities, i.e., probability density functions (PDFs), expected values, variances, covariance functions, higher moments, etc. Thus, now



UQ is the task of determining statistical information about the uncertainty in an output of interest that depends on the solution of a PDE, given statistical information about the uncertainty in its inputs

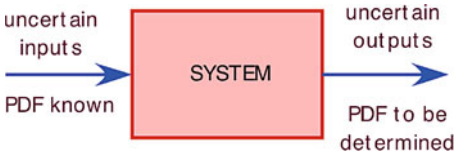
For example, an input parameter α could take the form of a given value plus noise, e.g. $\alpha = \alpha_0 + \eta$, where α_0 is a deterministic number and η is a random number whose value is selected by sampling a given PDF.

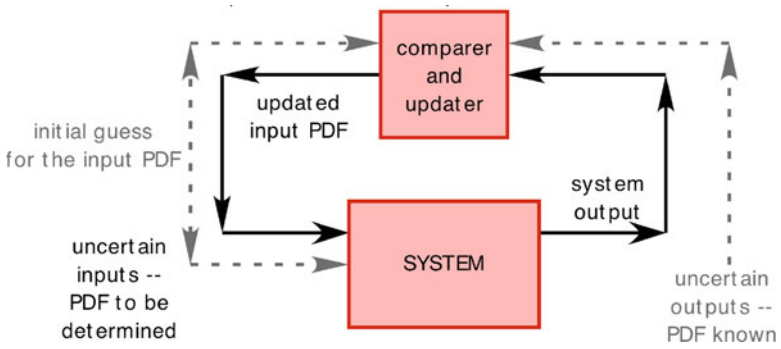
It seems we are in business: someone from on high gives us a mathematical model and statistical information about the inputs to the model. Then, to obtain statistical information about an output of interest depending on the solution of the model, all we have to do is devise a means to (perhaps approximately) solve the model equations. There is small problem however: often that someone on high disappoints us, i.e., often we do now know the needed statistical information about the inputs. What can one do? The most common approach is to make an educated (but sometimes an out-of-the-blue) guess as to what is that information. One can try to do something better such as use field or laboratory observations to make a more informed guess, but one should keep in mind that such observations also come with uncertainty (a troublesome fact that is often ignored),

Model Calibration/Parameter Identification Our discussion so far has been about the forward (or direct) problem of determining information about model outputs, given information about model inputs. Model calibration is about the reverse path, i.e., it is the task of determining statistical information about the inputs of a system, given statistical information about the outputs. On could, e.g., use experimental or field observations to determine the statistical information about the outputs. In particular, one would like to identify the PDF of the input variables. In case the inputs take the form of parameters appearing in the model that needs calibration, model calibration is often referred to as *parameter identification.*

Of course, the system still maps the inputs to the outputs so that determining the input PDF is an *inverse problem* whose solution usually requires multiple forward simulations of the system equations.

UQ: the direct (or forward) problem





Model calibration/parameter identification—inverse problem

Model calibration problems are a particular case of more general stochastic inverse, stochastic control, stochastic optimization, or stochastic design problems.

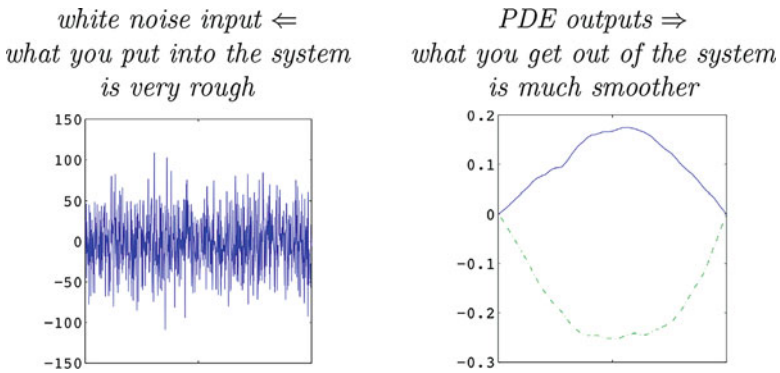
3 Types of Input Noises

We differentiate between the three types of noises that can be used as uncertain inputs that appear in mathematical models such as PDEs.

Random Parameters The input data could depend on a *finite number of random parameters*. Examples are flow rates in an HVAC system, voltages in an electric circuit, load on a beam, and the like. One can think of the parameters as begin "knobs" in an experiment which one has to set before running the experiment, but which in practice cannot be set at the exact value one wants. Each parameter may vary independently according to its own given PDF. Alternately, the parameters may vary according to a given joint PDF or through conditional probabilities.

White Noise Random Fields The value of the input data varies randomly and *independently* from one point of the physical domain to another and/or from one time instant to another. Thus, a white noise random field can be viewed as a function $\eta(\mathbf{x}, t)$ whose value at a point $\mathbf{x}$ and/or at a time t is sampled independently according to a single given PDF from any other point and any other time. Because the values at different times and at different times are independent of each other and are determined by drawing from the same PDF, such fields are referred as being independently and identically distributed, or i.i.d., for short.

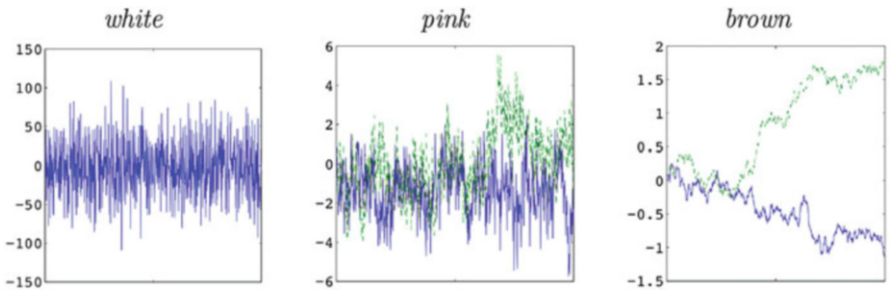
In practice, white noise is by far the most used means for introducing randomness into a system. However, white noise has infinite energy so that it cannot naturally exist. It continuous to be in ubiquitous use because discretizations and truncations of white noise have finite energy and are very easy to implement on a computer. Furthermore, in many settings, the solution of the system equations driven by white noise has finite energy and is much smoother than the input white noise, as is illustrated in the figure below, so that the potential problems with white noise inputs are glossed over.



Colored or Correlated Random Fields The value of the input data varies randomly from one point of the physical domain to another and/or from one time instant to another and is identically distributed but is *not independent* from the values at other points and other time instants. Instead, the values obey a given (spatial/temporal) *correlation structure.*

Colored noises are ubiquitously present. Three well-known colored noises are Brownian or brown noise that are continuous random processes and are related to diffusion; Lèvy noises that are jump processes and are related to anomalous diffusion; and Ornstein-Uhlenbeck or mean-reverting processes.

The figure below contains plots two realizations from each of three one-dimensional random fields, one is white and the other two are colored,[2] with increasing correlation going from left to right. We see that increasing correlation results in increasing smoothness of the field.



Approximate realizations of one-dimensional random fields

[2]Pink noise is "half-way" between white and brown noise; it is referred to as pink because in some circles brown noise is referred to as red noise.

4 Discretization of Stochastic Processes

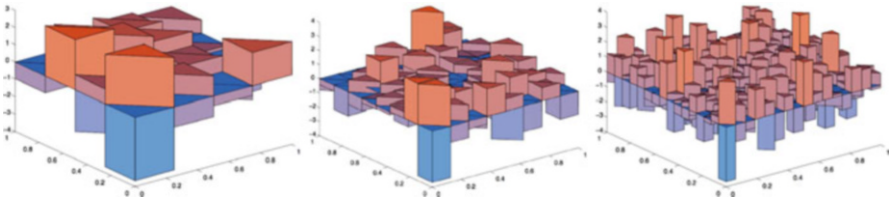
We started with the random parameter case in which the problem inputs involve a *finite* number of random numbers. If we choose values for these parameters, e.g., by sampling them according to their PDFs, we can then solve for the corresponding solution of the system equations, i.e., the PDE.

White noise fields are defined by a given PDF (usually a Gaussian PDF) and are easy to evaluate at a given point and/or at a given instant in time because the values may be chosen independently from the values previously sampled at other points and time instants; one merely samples from a given PDF.

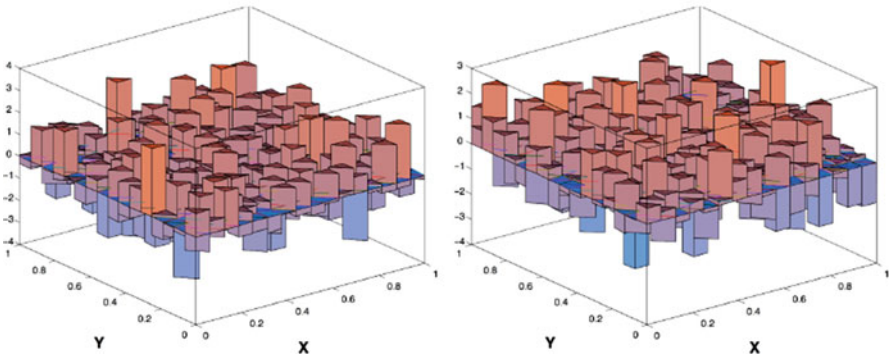
Colored noise (correlated random fields) are usually defined in terms of a PDF and correlation function. One is usually not given a "formula" that allows one to evaluate the random field at a given point and/or at a given instant in time. However, spatially and/or time-dependent random fields, whether the are correlated or white, can also be described in terms of parameters. But, because they are *infinite* stochastic processes, it requires an infinite number of random parameters to describe them.

Of course, on a computer one can only solve problems involving a finite number of random parameters. In the white and colored noise cases, one discretizes the noise so that the noise is approximated in terms of a finite number of parameters.

Discretizing White Noise The most common means for discretizing white noise is to draw independent samples from its PDF and use those samples to assign values of the field at each grid point (or in each grid cell) and/or each time interval used in discretizing the PDE. In this case, if J denotes the number of grid points (or grid cells) and K denotes the number of time intervals used to solve the PDEs, then the number of independent samples drawn is J for steady-state problems and JK for time-dependent problems. Thus, if one refines the spatial or temporal grids, the number of random samples increases. For example, in three dimensions, it one halves the spatial grid size, one would increase the number of parameters by a factor of 8. The following figures illustrate realizations of discretized white noise. Note that discretized white noise is a piecewise constant function in space and time; a piecewise constant function is much smoother than the white noise random field it approximates.



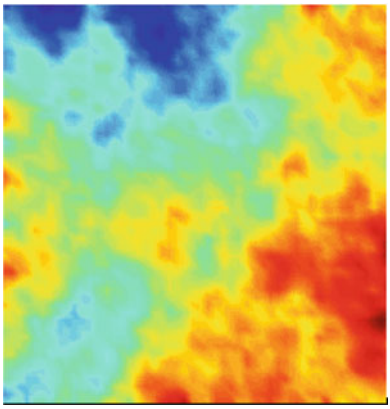
Realizations of discretized white noise at a same time interval in a square subdivided into 32, 128, and 512 triangles



Realizations of discretized white noise at two different time intervals in a square subdivided into the same number of triangles

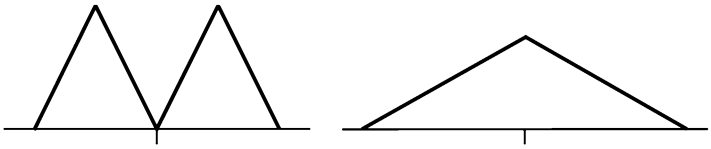
Discretizing Colored Noise In the colored noise case, one can use grid-based discretizations as well, but more often one takes advantage of the fact that one can express the random field in terms of infinite expansions in terms of orthogonal functions. There is more one way to do this, the most commonly used being *Karhunen-Loève expansions* that use the eigenvalues and eigenfunctions of the correlation function. As a result, the random fields are approximated by truncating the infinite expansions so that the approximate field then involves a finite number of parameters. In the case of colored noise, the number of parameters has a weaker dependence, compared to the white noise case, on the spatial/temporal grid sizes. The figure below provides an illustration of a realization of an approximated colored random field.

Realization of an approximated colored random field



A Little Sweeping-Under-the-Rug Part 1 We have tacitly assumed, as is often done, that we know the PDFs or other statistical information about the input

parameters. Actually, in practice, one usually does not know much about the statistics of the input variables. One is lucky if one knows a range of values, e.g., maximum and minimum values, for an input parameter in which case one often assumes that the parameter is uniformly distributed over that range. If one is luckier, one knows the mean and variance for the input parameter in which case one often assumes that the parameter is normally distributed. Of course, one may be completely wrong in assuming such simple probability distributions for a parameter as the figure below illustrates. This, as we have already noted, leads to the need to solve stochastic model calibration problems.



Two PDFs with the same mean and variance

A Little Sweeping-Under-the-Rug Part 2 Input variables could be distributed independently or jointly and could be correlated or uncorrelated. Without proper justification and sometimes incorrectly, it is often assumed that the parameters are independent. Based on empirical evidence, sometimes this is a justifiable assumption in the parameters-are-"knobs" case. But often, independence is a simplifying assumption that is invoked for the sake of convenience, e.g., because of a lack of knowledge.

Let us consider the case of correlated random fields. The Karhunen-Loève expansion does two wonderful things. First, it gives us a *formula*, albeit one with an infinite number of terms, that enables us to evaluate the random field at any point. Second, it expresses a correlated random field in terms of *uncorrelated parameters.* Unfortunately, what KL does not necessarily do is give us a formula involving *independent* parameters because although independence implies uncorrelated, uncorrelated does not necessarily imply independence as the following well-known example shows.

Let y_1 be uniformly distributed on $[-1, 1]$ so that the expected values $\mathbb{E}(y_1) = \int_{-1}^{1} y_1 \, dy_1 = 0$ and $\mathbb{E}(y_1^3) = \int_{-1}^{1} y_1^3 \, dy_1 = 0$. Now, let $y_2 = \frac{3}{2} y_1^2$. We have that the correlation $\mathbb{C}_{12} = \mathbb{E}(\boldsymbol{y}_1 \boldsymbol{y}_2) - \mathbb{E}(\boldsymbol{y}_1)\mathbb{E}(\boldsymbol{y}_2) = \frac{3}{2}\mathbb{E}(\boldsymbol{y}_1^3) = 0$ so that $\{\boldsymbol{y}_1, \boldsymbol{y}_2\}$ are uncorrelated. However, clearly $\{\boldsymbol{y}_1, \boldsymbol{y}_2\}$ are not independent. In fact, uncorrelated guarantees independence if and only if the variables follow a multivariate Gaussian distribution.

Revisiting Quantities of Interest We now can assume that we are given a random input parameterized in terms of a finite number of random parameters $\{y_1, y_2, \ldots, y_N\}$; we use the abbreviation $\boldsymbol{y} = \{y_1, y_2, \ldots, y_N\}$. These parameters

can be actual parameters appearing in the specification of the PDE or could arise from approximations of random field inputs.

The solution of the PDE is not only a function of the spatial variable $\mathbf{x}$ and perhaps also time t, it also a function of the random parameters $\boldsymbol{y}$. A *realization* is a solution $u(\mathbf{x}, t; \boldsymbol{y})$ of a PDE for a specific choice $\boldsymbol{y} = \{y_n\}_{n=1}^N$ of the random parameters.

There almost never is any interest in individual realizations. Instead, one is interested in *statistical information*, e.g., expected values, variances, standard deviations, higher moments, PDFs, CDFs. Most often one is not interested in such statistical information about the solution of the PDE itself. Instead, one is interested in statistical information about outputs depending on the solution of the PDE. The outputs of interest are often functionals of the solution of the PDE. For example, one is not interested in the velocity of a flow around a wing at points in the flow field but rather one is interested in things like the drag and lift on the wing. Again, one is almost never interested in individual realizations of such outputs but are interested in statistical information about such outputs which are referred to as *quantities of interest*.

The Curse of Dimensionality Applied and computational mathematicians live in one or two or three dimensions, with very rare excursions to higher dimensions. They think three dimensions is hard enough; even using today's supercomputers, some three dimensional problems cannot be addressed, e.g., the direct numerical simulations of practical turbulent flows. But, they are not well prepared to deal with the shock of having to find approximations of solutions $u(\mathbf{x}, t, \boldsymbol{y})$ of parameterized PDEs. Discretization has to be effected with respect to all three arguments so that one has to discretize in parameter space as well as in physical space and time and, moreover, the dimension of the parameter space, i.e., the number of parameters, may be large and certainly often much larger than three.

Statisticians are familiar with high-dimensional problems but they are not so used to dealing with problems for which realizations are very costly, as is the case when a realization involves the solution of a discretized PDE.

Let us leave PDEs alone for a minute and consider interpolation in N dimensions using polynomials of total degree at most p, e.g., for $N = 2$ and $p = 1$ we have the linear polynomial $a + by_1 + cy_2$ and also consider interpolation in N dimensions using tensor product polynomials of degree at most p in each direction, e.g., for $N = 2$, $p = 1$ we have the bilinear polynomial $a + by_1 + cy_2 + dy_1y_2$. For the same p, both types have same the approximation properties, i.e., for interpolating sufficiently smooth functions, the rate of convergence is the same.

What about the complexity? In the next table, one sees the *explosive* growth in the number degrees of freedom one needs for both types of approximations as the number N of parameters and the degree p of the polynomials increase. Note that, for the same rate of convergence, total degree interpolation is relatively much more economical compared to total tensor product interpolation but that even total degree interpolation suffers the explosive growth in the number of degrees of freedom, i.e., suffers from the curse of dimensionality.

Returning to the PDE setting, M in the table is the number of PDE solves one needs to determine the polynomial approximation. *The curse of dimensionality* refers to the explosive growth in the number of parameter degrees of freedom and therefore in the number of (costly) PDE solves needed for a certain accuracy as the number of parameters N and the degree p of the polynomials increases.

5 Approximation of Solutions of PDEs with Random Inputs

Stochastic finite element methods refer to the use of finite element methods (FEMs) to *spatially* discretize a PDE with random inputs. Of course, spatial discretization may be effected by any of a number of other methods, e.g., finite difference, finite volume, spectral, radial basis function, etc., so that, e.g., when using the first of these, one can also use the terminology "stochastic finite difference methods."

With respect to discretizing the parameter dependence of the problem, we discuss two approaches.

Global polynomial approximation in parameter space

$N =$ number of variables	$p =$ maximal degree of polynomials	$M =$ number of degrees of freedom	
		Using total degree polynomial basis	Using tensor product basis
3	3	20	64
	5	56	216
5	3	56	1024
	5	252	7776
10	3	286	1,048,576
	5	3003	60,046,176
20	3	1771	$>1\times10^{12}$
	5	53,130	$>3\times10^{15}$
100	3	176,851	$>1\times10^{60}$
	5	96,560,646	$>6\times10^{77}$
		$\Uparrow$ $\frac{(N+p)!}{N!p!}$	$\Uparrow$ $(p+1)^N$

Stochastic Galerkin Methods (SGMs) refer to effecting the discretization with respect to the random parameters using a Galerkin method. Galerkin methods are variational or projection methods which are commonly used for FEM and spectral method spatial discretization of the PDE.

The good news about SGMs is that only a single discrete system has to be solved to determine both the spatial/temporal and parameter dependencies of the

approximate solution. In particular, no parameter sampling is needed. As a result, one obtains an approximation that can be evaluated at any point in the parameter domain, e.g., to determine a quantity of interest.

The bad news is that the physical and parameter degrees of freedom are coupled. Thus if J denotes the number of finite element degrees of freedom and if M denotes the number of degrees of freedom used for parameter approximation, then the size of the discrete system that has to be solved is $JM \times JM$. Given than in a practical calculations J could be in the millions and we already saw that M could be gazillions, one can quickly get to systems of formidable size when using SGMs.

Another disadvantage of SGMs are that their implementation requires extensive recoding of a deterministic PDE solver because the discretizations of the spatial and parameter dependences are tightly coupled. For this reason, such methods are referred to as being *intrusive.*

The usual choice for effecting parameter approximations are orthogonal polynomials; in such cases, in the mathematical UQ community, the method unfortunately[3] goes by the name polynomial chaos. To take advantage of the high-accuracy of orthogonal polynomial approximations, whatever is approximated has be smooth with respect to the parameters.

Stochastic Sampling Methods (SSMs) refer to methods in which the PDE is solved at each member of a set of sample points in the parameter domain. Here, any spatial discretization of the PDE can be used because that discretization is uncoupled from how the parameter dependence is treated. For this reason, a deterministic PDE solver can be used as a black box for determining approximate solution at the selected parameter points. For this reason, SSMs are referred as being *non-intrusive.*

Stochastic sampling methods proceed as follows:

- sample M points $\{\boldsymbol{y}_m\}_{m=1}^M$ in the parameter domain Γ;
- for $m = 1, \ldots, M$, solve the spatial/temporal discretized PDE for each of the sample points;
- then use the M solutions so obtained to build ensemble averages or other statistical information about the approximate output of interest that depends on the approximate solution of the PDE.

Thus, to determine statistical information in the sampling setting one solves M discrete systems, each of size $J \times J$, compared to the stochastic Galerkin case for which one solves a single discrete system of size $JM \times JM$.

[3] *Polynomial chaos* was a term coined by Norbert Weiner when he studied PDEs driven by white noise and whose solution displayed chaotic behaviors. He expressed white noise random fields by truncated expansions in terms of orthogonal polynomials. In the mathematical and engineering UQ communities, polynomial chaos is used as a substitute name for orthogonal polynomial approximation, even though very few of the problems addressed by those communities have solutions that display any chaotic tendencies. We ourselves eschew the use of "polynomial chaos" and instead call it by what it actually is, namely, orthogonal polynomial approximation.

The list of possible sampling schemes is, of course, very, very, very long.[4] Obfuscating the situation is that what makes a set of points in parameter space "good or bad" depends on whether the points are used for quadrature, interpolation, regression, or some other purpose.

Sampling + Simple Averaging Methods for Quadrature By far, the most used sampling method for parameter domain quadrature is the *Monte Carlo* (MC) method for which one randomly chooses the set of sample points and simply averages the values of the integrand over those points. There is a lot of good news about MC. Perhaps uniquely among sampling strategies (and other approaches to UQ), the convergence rate of MC is independent of the number N of parameters so in this sense it does not suffer from the curse of dimensionality. Also, MC does not care about the smoothness of the integrand in the sense that the convergence rate is unaffected by smoothness. In addition, MC does not care much about the shape of the domain of integration. However, there is also bad news about MC. Convergence (which is in expectation) is very slow, with a rate $1/\sqrt{M}$, where M denotes the number of sample points. Furthermore, that remains the convergence rate regardless of how smooth is the integrand, i.e., MC cannot take advantage of any smoothness the integrand possesses.

The slow convergence of MC has spawned a huge industry aimed at devising alternative quadrature schemes that are "better" than MC, i.e., that converge faster, but which still are simple sampling and averaging schemes. Both deterministic and probabilistic, some sequential and some not, alternative sampling strategies have been invented. A non-exhaustive list includes variance reduction techniques, quasi-Monte Carlo sequences (e.g., Halton, Sobol, Faure, ... ad infinitum), Hammersley, Latin hypercube, importance sampling, stratified sampling, lattice sampling, orthogonal arrays, multilevel Monte Carlo, etc. For several of these methods, the error is proportional to $\frac{(\ln M)^{N+s}}{M}$ for some $s > 0$. For a "small" number of parameters N, one can ignore the logarithmic term and obtain close to linear convergence, i.e., $1/M$, which is a decided improvement over the $1/\sqrt{M}$ convergence rate of MC. However, for "large" N, the logarithmic term dominates so that the curse of dimensionality bites us again.

Interpolation and "better" Quadrature Rules It is well known that in one dimension and for smooth integrands one can do "better" compared to simple averaging rules $\frac{1}{M}\sum_{m=1}^{M} f(\boldsymbol{y}_m)$ by using weighted quadrature rules $\sum_{m=1}^{M} w_m f(\boldsymbol{y}_m)$, where the quadrature points $\{\boldsymbol{y}_m\}_{m=1}^{M}$ and weights $\{w_m\}_{m=1}^{M}$ are judiciously chosen. In one dimension, Gauss rules are beautiful examples of how to define "better" rules. This is why for a (very!) small number N of parameters, tensor products of Gauss rules have proven to be very useful. But, as we have seen, tensor products should be avoided like the plague, even for moderate N.

[4] There is even a non-intrusive version of SGMs, but that version is better viewed as a sampling method.

We consider *interpolatory quadrature rules* which are constructed by first constructing an interpolation method and the approximating an integral of a function by the integral of the interpolant of the function. Thus, the discussion here applies to both interpolation and to quadrature. In particular, we focus on polynomial interpolation and the quadrature rules that they engender.

Interpolants are built by requiring that they match the value of a function at sample points that, in this context, are referred to as interpolation points. When interpolants are used to define quadrature rules, the interpolation points become the quadrature points. Thus, we first must define "good" interpolants. Without any knowledge about the function being interpolated other than its smoothness, total degree interpolation requires the least number of interpolation points (i.e., in our context, the least number of PDE solves) to achieve the best rate of convergence possible by polynomial interpolation. Unfortunately, "good" interpolation points for total degree interpolation in as simple domain as a hypercube are not known, even in three dimensions, and there is some controversy about them even in two dimensions.[5] Fortunately, the along came Sergey Smolyak.

Stochastic Collocation or Sparse Grid Methods Smolyak (or sparse)[6] grids[7] are a judiciously chosen subset of tensor product grids. For the same precision, i.e., for integrating the same polynomial space exactly, sparse-grid quadrature rules require more points that do interpolatory quadrature rules based on total degree interpolation but require substantially fewer points than does tensor product interpolation or quadrature; see the table below. Note that as N and/or p increase, the gap between the number of total degree and sparse grid points grows quickly, but still at a much slower pace compared to the gap between sparse grids and tensor product grids. Note also that because total degree quadrature rules suffer from the curse of dimensionality, so do sparse grids rules.

[5]It is well known that in one dimension, evenly spaced points are "bad" interpolation points for general smooth function, bad because the interpolation error can get can get worse as the degree of the polynomial increase and the point spacing decreases. On the other hand, the unevenly spaced Chebyshev points are known to be ideal for the interpolation of smooth function in one dimension.

[6]In truth, any sampling method can be referred to as being a stochastic collocation methods because "stochastic" simply refers to the fact that we are dealing with random variables within some domain in parameter space and "collocation" simply means that we are evaluating the function, in our case the solution of the PDE, at points in the domain, ergo, we are sampling the solution at points in the parameter domain. However, stochastic collocation methods is now thought of as referring to a class of methods for which deterministic sampling is done on a structured set of points that are much fewer, e.g., much sparser, than, e.g., a tensor product of points, ergo, the synonymous moniker "sparse grids."

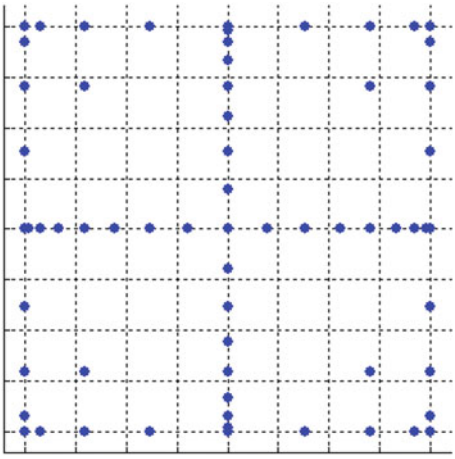
[7]The quadrature rules that use Smolyak of sparse grids as the quadrature points are not always interpolatory quadrature rules, but they are all built using combinations such rules.

For three types of grids in N-dimensional hypercubes, the degrees of freedom for quadrature rules having the same convergence behavior as total degree quadrature using polynomials of degree at most p

	Total degree		Sparse grid		Tensor product
Degrees of freedom	$\frac{(N+p)!}{N!p!}$	$<$	$O\left(p(\ln p)^{N-1}\right)$	$\ll$	$(p+1)^N$

The figure below is an illustration of a sparses grid. Note the big holes in the grid, i.e., the large areas containing no points. If the function being interpolated or integrated is very smooth, the big holes do not matter. For moderate parameter dimension and polynomial degree, sparse grids beat Monte Carlo, quasi-Monte Carlo, etc. but, of course, the curse of dimensionality rather quickly kicks in so that for quadrature,[8] sparse grids start losing to MC sampling because, as we have already mentions, MC does not suffer from the curse.

A 65 point sparse grid



The need for smoothness in sparse grid quadrature and the lack of such need for Monte Carlo quadrature is illustrated in the table below.

Comparison of sparse grid and Monte Carlo approximations of the integral of a discontinuous function

M	SG estimate	SG error	MC estimate	MC error
1	4.000	1.167	0.00000	5.16771
13	64.000	58.832	0.00000	5.16771
85	−42.667	47.834	3.01176	2.15595
389	−118.519	123.686	4.77121	0.39650
1457	148.250	143.082	5.15216	0.01555
Exact	5.16771	–	5.16771	–

[8] MC and QMC points are useless for interpolation.

6 Quo Vadis Uncertainty Quantification?

It is clear the what we have written above does not completely exorcize the curse of dimensionality, although in some situations, e.g., for smooth dependences on the random parameters, progress has been made in reducing the cost of UQ in the PDE setting, at least for a moderate number of parameters. However, there are efforts out there devoted to make further progress towards defeating the curse. In addition, there are other important aspects of UQ that we have not touched upon. We close by giving some very brief comments about some of these topics, with the comments ordered in decreasing length but not necessarily in decreasing importance.

Informed Sampling In the algorithms discussed so far, the sample points can be pre-selected to yield useful, i.e., efficient and accurate, approximations of function belonging to certain classes of functions such as functions with a certain number of continuous derivatives. However, they do not take into account any features of the specific function, e.g., an approximation of the solution of the PDE, that one wishes to integrate or approximate.

If one has some information about that function, as one often does, one can take advantage of that information to lessen the cost of integration or interpolation. For example, if one knows that some parameters are more influential than others, one can do anisotropic sampling so that there is a lower density of points in directions in parameter space that correspond to less important parameters compared to that for more important parameters.

Even before that, there are techniques available, e.g., screening methods, sensitivity analyses, etc., that can be used to determine the relative importance of the parameters.

Adaptive point selection is another promising approach. Here, one sequentially samples the point, with the position of a new point in parameter space selected so that it minimizes some function that can be computed from the current set of points and which in some (approximate) way represents the error that the use of the augmented set of points would induce.

Surrogates Our focus has been on becoming more efficient in approximating in parameter space. However, the large cost of approximate PDE solves, e.g., function evaluations in our setting, also contributes to the curse of dimensionality. The idea here is to use a relatively *few* approximate solutions of the PDE to construct a cheap-to-evaluate surrogate that can be used, instead of additional PDE solves, to provide the *huge* number of PDE solutions needed to do UQ. Actually, it is even more efficient to directly build surrogates for output of interest. Many types of surrogates approaches have been studied, including interpolation, least-squares approximation (regression), greedy algorithms, reduced-basis methods, proper orthogonal decomposition, etc.

Risk Assessment As has already been mentioned more than once, with the possible exception of simple sampling and averaging methods, what we have talked about, including orthogonal polynomial and sparse grid collocation methods, requires

smooth dependences on the parameter. As a result, these approaches have limited usefulness for quantifying risk assessment because that often involves integration of discontinuous functions. This includes, e.g., most approaches for determining probabilities of failure. In the UQ for PDE setting, little has been successfully done to move away from sampling and simple averaging approaches.

Compressed Sensing Compressed sensing is one of several approaches that try to determine the least number of terms that are needed in a polynomial approximation to achieve a certain accuracy. Of course, this has to be done without first having to compute all the terms in the polynomial and then throwing out those that are insignificant. Instead, a priori estimates for the coefficients are used to determine which terms to not include. The idea behind approaches such as compressed sensing is to reduce the number of terms below that needed for total degree interpolation without sacrificing accuracy.

Inverse Problems We have already mentioned stochastic optimization, identification, calibration, and control problems, all of which are obviously of great interest. Often these problems suffer from an even worse curse because of the possible need to compute the quantities of interest several times during a control or optimization process. Still, inverse problems constrained by PDEs with random inputs is a quickly growing industry. Baysean approaches are quite seductive and enjoy widespread popularity, but direct optimization approaches are also used.

Rare Events Quantifying rare events is another task for which not much progress has been made in the UQ for PDEs setting, a setting in which function evaluations are very expensive. The techniques being used are mostly standard, well-known ones in the statistics community.

Acknowledgements The author "Max Gunzburger" was supported in part by the US Department of Energy Office of Advanced Scientific Computing Research and the US Air Force Office of Scientific Research.

References

1. Ghanem R, Higdon D, Owhadi H (eds) (2017) Handbook of uncertainty quantification. Springer, Berlin
2. Gunzburger M, Webster C, Zhang G (2014) Stochastic finite element methods for partial differential equations with random input data. Acta Numer 23:521–650
3. Smith R (2014) Uncertainty quantification: theory, implementation, and applications. SIAM, Philadelphia
4. Soize C, (2017) Uncertainty quantification: an accelerated course with advanced applications in computational engineering. Springer, Berlin
5. Sullivan R (2015) Introduction to uncertainty quantification. Springer, Berlin